Applying a ThinkLet- and ThinXel-based Group Process Modeling Language: A Prototype of a Universal Group Support System

Stefan Werner Knoll
Faculty of Computer Science
University of Magdeburg
Germany
sknoll@sim-md.de

Martin Hörning
Igosys IT-Service GmbH
Germany
martin.hoerning@igosys.de

Graham Horton
Faculty of Computer Science
University of Magdeburg
Germany
graham@sim-md.de

Abstract

Group Support Systems (GSS) can improve the productivity of Group Work by offering a variety of tools to assist a virtual group across geographical distances. Experience shows that the value of a GSS depends on how purposefully and skillfully it is used. We present a framework for a universal GSS based on a thinkLet- and thinXel-based Group Process Modeling Language (GPML). Our framework approach uses the GPML to describe different kinds of group processes in an unambiguous and compact representation and to guide the participants automatically through these processes. We assume that a GSS based on this GPML can provide the following advantages: to support the user by designing and executing a collaboration process and to increase the applicability of GSSs for different kinds of group processes. We will present a prototype and use different kinds of group processes to illustrate the application of a GPML for a universal GSS.

1. Introduction

A Group Support Systems (GSS) is an information technology-based group meeting environment, which can be used to manage group work with virtual teams across geographical distances. Under certain circumstances a GSS can increase the productivity of group work by offering a variety of tools that link the participants via computers and assist them in structuring of activities and improving group communication [7] [11] [14]. But research shows that the value of the technology depends on how purposefully and skillfully it is used [2]. Experience is necessary for the adaption of the GSS to a designed group process and for the facilitation during the group work.

We assume that a GSS should assist practitioners in the adaption of the GSS to different kinds of group processes and in the facilitation during the group work. We hope that this will support group work involving GSS. The challenge is to find a way to develop a universal GSS that:

- increases the applicability of a GSS for different kinds of group processes;
- supports practitioners in designing and executing a group process;
- adapts the system automatically to a designed group process to make facilitation easier;

Our design approach for a universal GSS uses an object-oriented approach for group processes. We assume 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 Group Process Modeling Language (GPML) that describes the order of activities of the participants during the executing of a group process and can be used to design a universal GSS. This universal GSS uses a group process model to execute different kinds of group processes and to adapt the system automatically.

In this paper we will introduce a framework based on GPML that establishes the basis for a universal GSS. We will show the functionality of this framework in a first prototype that can execute different kinds of group processes.

2. State of the Art

In this section we will give a short introduction to our object-oriented approach for group processes and the resulting GPML.

2.1. An Object-Oriented Approach for Group Processes Using ThinkLets and ThinXels

Our object-oriented approach for group processes uses the concept of thinkLets which is developed from Collaboration Engineering (CE). The purpose of CE is to create collaboration processes for recurring high value tasks in companies that can be executed by practitioners without ongoing support by professional facilitators [4]. CE classifies these collaboration processes into six key patterns of collaboration (for further information see Briggs et al. [5]). Researchers use this classification to collect, create, document and test collaboration activities, called thinkLets, which together form a pattern language for group collaboration [4]. 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 [5]. The specification of a thinkLet comprises the following components: identification, script and selection guide (for further information see de Vreede et al. [16]). 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 [15].

Building on the concepts of Vreede et al. [16], Kolfschoten et al. [10] developed an object-oriented approach for a collaboration process with the thinkLet class diagram. This is based on the object-oriented modeling approach for systems engineering [6] and uses the unified modeling language (UML) notation to illustrate the key concepts and relations of a thinkLet (for further information see Kolfschoten et al. [10]). The approach defines thinkLets as reusable logical design elements based on the object-oriented paradigm and organized in patterns of collaboration. Practitioners use these elements during the logical design phase to design a group process which is abstracted from the technology upon which it could be implemented. This logical design of a group process, which can be flexibly applied in many contexts and with many technologies, represents an interesting and suitable approach for designing a modeling language for group processes. We assume that this modeling language can be used to develop a GSS which supports practitioners in designing and executing a group process. This GSS needs different data from a group process model to adapt the system automatically to a designed group process. This data can be described as a combination of the following process information:

- **Participant flow -** describes the order and types of activities of the participants in a group process. A GSS can use this information to compute the next activity for a participant.
- **Data flow** - describes the connections between data storage elements and the activities of the group process. A GSS can use this information to provide or store data for the activities of the participants flow.
- **Signal flow** - describes the connection between events and the activities of the group process. A GSS can use this information to change the participant or data flow for a specific event during the group process.

We propose that a modeling language for group processes should describe this information. This implies that the activity of a participant must be known for each time step during the group process. A thinkLet is a logical design element which includes abstract rules for the participant to achieve a collaboration pattern. These rules describe actions that participants must execute using the capabilities provided to them under some set of constraints [10]. During the physical design phase the practitioner adapts the rules of the implemented thinkLets to a selected technology. We assume that a GSS can also use these rules to provide an interface for the activities of the participants during a group process. But little research exists to specify the physical design of a thinkLet-based work practice. In this case the rules leave open the question, which facilitation instruction or interface should be used to achieve these actions. Therefore it depends on the experience of the practitioner or software engineer which instruction or interface he uses. In order to compensate for this weakness we introduce the concept of thinXels as an atomic facilitator instruction leading to a response which has a well-defined function in the context of the group goal [8]. ThinXels were invented as physical design elements and are based on the Shannon-Weaver Model of a communication process [12]. To limit the cognitive load of the participant, we assume that a facilitator instruction may contain only those pieces of information that must be conveyed to the participant to perform the activity intended by the rule. An example is the instruction “Please write down on a sheet of paper a comment, which you associate with the issue” for the rule “create a comment for the issue”. The ThinXel concept uses this property and defines that each instruction should only lead to one basic activity like “add”, “select” and “move”. This makes a thinXel an atomic instruction. As a result, each rule in the logical design of a group process should only lead to one action of a participant. This condition leads to the situation that we need to redesign some existing thinkLet rules. For example, the thinkLet “Mood Ring” [10] contains the rule “Discuss the issue”, which should create the activity “discuss”. In our opinion this rule can lead to a combination of different activities: explain the issue, collect comments and collect proposals for solutions. In this case the rule still leaves open the question, which facilitation instructions should be used to achieve the activity "discuss". Therefore it is not possible to define a thinXel for this rule. We assume that a thinXel can be categorized into context and data-oriented thinXels [9]:

- A **context-oriented thinXel** - represents a facilitation instruction with the intention to create a working environment for the group process. These instructions explain the constraints, goals and the working process and lead to acceptance among the participant. Examples are:
    - *"Please use the next steps to (goal)"* [goal: e.g. collect comments for the existing issue];
    - *"Please read the (element)"* [element: e.g. the issue, the information];
    - *"Please take the (work equipment) to (goal)"* [work equipment: e.g. a pen and some sheets of paper], [goal: e.g. collect your comments].
- A **data-oriented thinXel** - represents 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). We think that each other data activity can be represented by a combination of these three activities. An example is the activity "delete", which can be represent by the activity "organize" leading to a relation between a contribution and a contribution named "wastebasket". Examples are:
    - **create**: "*Please write down on (work equipment) a (element), that (condition)*" [work equipment: e.g. a sheet of paper], [element: e.g. comment, idea] [condition: e.g. solves the problem]
    - **grow**: *"Please write down on (work equipment) a (element) for the selected (element), that (condition)"* [work equipment: e.g. a sheet of paper], [element: e.g. comment, idea] [condition: e.g. explain the element in detail, describe the implementation];
    - **organize**: *"Please put the (element) into the (cluster)* [element: e.g. comment, idea] [cluster: cluster comments];

Collaboration Engineers can use the thinXel-concept to develop facilitation concepts for the basic activities of a rule. Practitioners can use these thinXels to design a facilitation instruction or an interface for these rules in a given context by adjusting the thinXel to the capabilities provided to them. We propose that thinkLets and thinXels can be used to design a modeling language for group processes that can be used to develop a universal GSS.

2.2. A ThinkLet- and ThinXel-Based Group Process Modeling Language

In this section we will give a short introduction to a thinkLet- and thinXel-based group process modeling language (GPML). The GPML is explained in [9] by Knoll et al.. Please refer to this paper for a more detailed explanation. The design of our modeling approach follows graphical representations of existing elements of process models like Event Driven Process Chains (EPCs), Petri Nets and UML activity diagrams. The GPML includes the following elements [9] (shown in Figure 1):

- **Participant flow** - describes the way of the participants through the group process. The element discriminates between an individual participant and a group of participants and allows us to illustrate concurrent processes of participants with different roles;
- **Decision** - guides the participant flow and allows us to react to internal and external stimuli. 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;
- **Transition** - represents the places in the model where individual participants can be added to a group or where a group can be split into individual participants under a set of conditions. This element allows us to describe group processes with concurrent processes including parallelization and synchronization;
- **Sender-Receiver** - describes internal and external signals during the group process. The graphical elements are connect by a *signal path*;
- **Sender-Response** - represents the influence of a signal on the activities of the participants. It interrupts the participants in their current activities and changes the participant flow;
- **ThinXel** - is a physical design element which represents a formal instruction or interface that leads to one activity of a participant in the participant flow. The intended activity can be abort or executed by the participant. In a logical design phase, the formal instruction is represented by the rule for the intended activity. Depending on the group process, the element *thinXel* can be connected via a *data path* to a *storage place* which is able to store particular types of data;

- **ThinkLet** - represents a process model described using the elements of the GPML, which can be integrated into another group process. The detailed process model is represented by a *thinkLet construction plan*. The interface elements *start* and *end* represent the connection of the participant flow between the thinkLet and other group processes. Data and signals can be received with the elements *data parameter* and *signal parameter* and can be forwarded by *data forwarding* and *signal forwarding*;
- **Signal-Data generator** - describes an abstract element that produces data or signals. An example of this is a timer which sends a signal after a certain period of time.

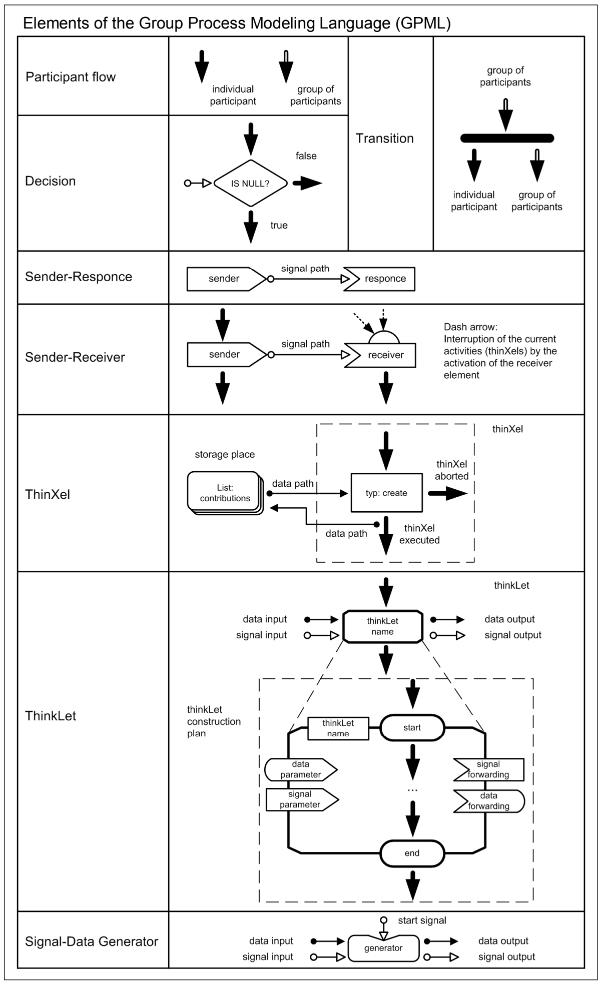


Figure 1. Elements of the GPML

An example of a group process model which uses this GPML describes the thinkLet construction plan of the thinkLet *Brainstorming* [3] (shown in Figure 2.). *Brainstorming* represents a logical design of a group process with the goal to create contributions for a given topic. The thinkLet construction plan can be connected with other group processes by the input element *parameter: topic* which represents the topic that will be used to create contributions. The process flow of the thinkLet guides each participant individually to the thinXel element *create: contributions*. This data-oriented thinXel represents a formal instruction or interface that leads to the activity *to create a contribution for the given topic*. This can be the instruction “Please write down on a sheet of paper a contributions for the active topic” for a face-to-face workshop or an interface which includes a text area for new contributions for a GSS. The participant has access to all existing contributions through the data path between the thinXel and the storage place element *list: contributions.* This storage place can be implemented by a flipchart or a database. If the participant will execute the intended activity of the thinXel, a created contribution will be stored in the storage place element. In this case the process flow will guide the participant back to the thinXel element *create: contributions*. If the participant aborts the intended activity of the thinXel he or she will be guided to the exit of the thinkLet element and finish the group process.

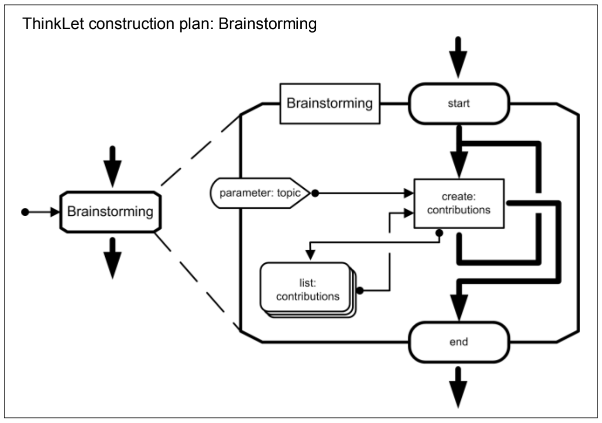


Figure 2. ThinkLet: Brainstorming

We assume that the presented elements of the GPML form a modeling language for group processes. They describe the activities of the participants, the data and the events of a group process in a logical design. Each activity of a participant can be represented by the binary element *thinXel* which represents a formal instruction or interface for the intended activity. An object-oriented approach is given by the element *thinkLet* that represents a described process model which can be connected with other group processes. We assume that these properties of the GPML can be used to develop a participant flow algorithm, which can be implemented in a GSS to adapt the interface for a participant automatically to the active position of the participant and the attributes of the active thinXel element.

3. Applying the GPML for a Participant Flow Algorithm

We will now present an application of the GPML for a participant flow algorithm that establishes the basis for a universal GSS. This algorithm uses a described group process model to guide the participant automatically through the process. This implies that for each time step during the group process the algorithm must compute the position and activity of a participant. We assume that this property can be used to design a framework for GSS that uses this information to adapt the interface of the GSS individually for each participant. This could make facilitation easier for a group process especially for concurrent processes.
Our design approach for a participant flow algorithm is based on the elements of the GPML that have an influence on the participant flow of the group process. These elements are: *decision*, *transition*, *sender, response*, *thinXel* and *thinkLet*. One property of the GPML is to enable the practitioner to design two kinds of participant flows:

- **Individual flow** - represents the sequence of activities of an individual participant. A practitioner can use this flow to describe group processes with concurrent activities for participants with different roles;
- **Group flow** - represents the sequence of activities of a group of participants. The participants activate the elements of the GPML simultaneous and will only be guided as a group to the next element.

Both participant flows can be concurrently used in a group process model but apply different elements of the GPML. The elements *decision*, *sender* and *thinkLet* can be activated from both participant flows. The element *thinXel* can only be used in an individual flow because of the property that a thinXel represents an activity of one participant. The same situation exists for the element *response* that sends an interruption signal to all thinXel elements of the participant flow and collects each participant individually. For this reason our approach for a participant flow algorithm distinguishes between an individual and a group flow. A practitioner can change the state of flow by the element *transition* which represents the places in the model where individual participants can be added to a group or where a group can split into individual participants. A *transition* will be activated as soon as all participants have arrived at the transition or all conditions are fulfilled. The participant flow algorithm must support this property of a *transition* by a synchronization algorithm. This algorithm must monitor the state of the group process and activate the *transitions* if all conditions are fulfilled. Summarized our participant flow algorithm includes the following components:

- A **participant algorithm** - computes the next step for a participants as an individual participant based on the active position and activity of a participant in the described group process model;
- A **group algorithm** - computes the next step for a participant in a group of participants based on the active position and activity of the group in the described group process model;
- A **synchronization algorithm -** monitors the state of the group process and switches between the participant and group algorithm in relation to the kind of the participant flow (individual or group).

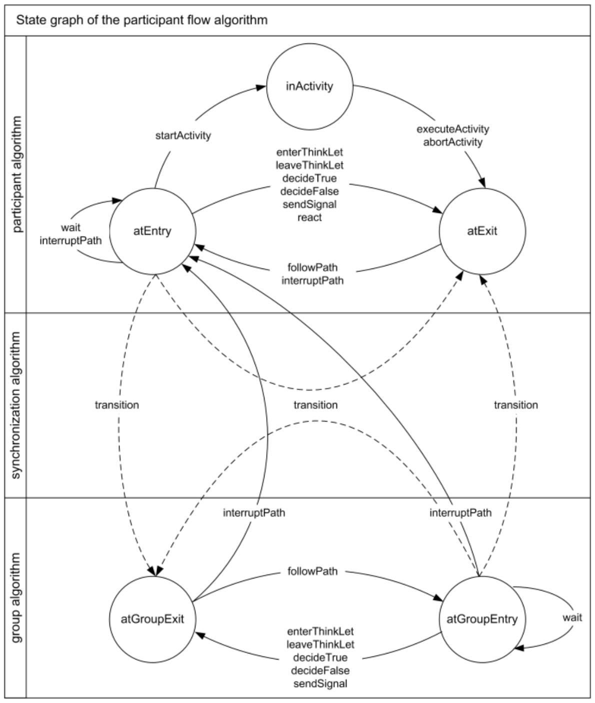


Figure 3. State graph of the participant flow algorithm

These algorithms are illustrated in a state graph shown in Figure 3. It represents the possible states of a participant and a group during a participant flow for each of the supported elements of the GPML that have an influence on the participant flow. The group algorithm uses the following group states:

- **atGroupEntry** - the group of participants is at the entry of the element;
- **atGroupExit** - the group of participants is at the exit of the element.

Table 1. Steps of the participant and group algorithm

Step	Element of the GPML	Description	Input state	Output state
enterThinkLet	*thinkLet (start element)*	the activity to enter a *thinkLet*	atEntry / atGroupEntry	atExit / atGroupExit
leaveThinkLet	*thinkLet (end element)*	the activity to leave a *thinkLet*	atEntry / atGroupEntry	atExit / atGroupExit
decideTrue	*decision*	a positive decision making for the element *decision*	atEntry / atGroupEntry	atExit / atGroupExit
decideFalse	*decision*	a negative decision making for the element *decision*	atEntry / atGroupEntry	atExit / atGroupExit
sendSignal	*signal*	the activation of a signal by the element *signal*	atEntry / atGroupEntry	atExit / atGroupExit
followPath	*participant flow*	the step between two elements that are connected by a *participant flow*	atExit / atGroupExit	atEntry / atGroupEntry
wait	*transition*	the activity to wait for the activation of a *transition*	atExit / atGroupExit	atEntry / atGroupEntry
interruptPath	*response*	interrupt the active activity and guide the participant to the element *response*	atExit / atGroupExit	atEntry
react	*response*	the activity to guided an participant to a new *participant path* by the element *response*	atEntry	atExit
startActivity	*thinXel*	represents the activity to enter a *thinXel*	atEntry	inActivity
executeActivity	*thinXel*	represents the execution of the intended activity of a *thinXel*	inActivity	atExit
abortActivity	*thinXel*	represents the abortion of the intended activity of a *thinXel*	inActivity	atExit

These states are similar to the states of an individual flow but must be extended by the properties of the elements *thinXel*. The element *thinXel* represents a formal instruction for an intended activity of one participant in the participant flow that can be aborted or executed. It requires input data from the participant to resume the participant flow to the next element. As the result we define the following states of an individual participant for the participant algorithm:

- **atEntry** - the participant is at the entry of the element;
- **atExit** - the participant is at the exit of the element;
- **inActivity** - the participant is in a thinXel;

A change of state results from the steps of the participants in the participant flow which are defined by the function of elements of the GPML. These steps for the participant and group algorithm are shown in Table 1 and represent all kinds of activities of the participants for a described group process based on the GPML. An exception is the element *transition* which cannot be activated by the entry of a participant or a group. This activation is provided by the synchronization algorithm that monitors each activity of the group process. After each change of the group process, it verifies if the conditions of a transition are fulfilled. In that case the algorithm will activate the transition. This transition can lead to a switch between the participant and group algorithm.

Summarized, the presented states and steps of the participants and the synchronization of the element *transition* form a participant flow algorithm for a group process model based on the GPML. This algorithm allows us to compute the next position of a participant in a group process against their active position and activity in the group process model. We assume that this property will allow us to generate a framework for a universal GSS that will guide the participants automatically through the group process and adapt the interface of the GSS individually for each participant.

4. Applying the Participant Flow Algorithm to Design a Framework for a universal GSS

In this section we will present a framework for a universal GSS that assists the participants in structuring their activities. We define a universal GSS as a system that supports the practitioner to design and execute different kinds of group processes. As a result our framework for a universal GSS is based on the GPML and the participant flow algorithm. The framework is divided into different modules, which provide data structures and functions for the design and execution of a group process. This division is object-oriented and provides a module for design, execution and data management of a group process. The framework consists of the modules:

- **ElementLib** - contains the data structures to describe a group process model based on the GPML. It provides the functions of loading and saving a group process model in a data file. To support the flexibility of the GPML, the module allows the practitioner to define and use new thinXels and data types without any adaption of the module.

- **ProcessLib -** implements the participant flow algorithm and provides a data structure for the description of the participants, the groups and the process states. The module computes the next position of a participant against their active position and activity in the group process model. These active activities of the participants are stored in the interface *ActivityMap* which provides the information to the GSS. An internet based GSS can use this interface to compensate the access lag which can result from the client-server communication.
- **DataLib -** contains the data structures to save or provide process data and contributions of the participants during a group process. The module includes database connectivity and functions to analyze the dataset.

The module *ElementLib* provides a data structure, which enables us to represent the elements of the GPML as objects of this structure. We use the extensible markup language (XML) to import a described group process model. This extensible language allows us to define our own elements for each element of the GPML. The module *ElementLib* uses this XML syntax to compute objects of the GPML against the specific attributes which are defined by the practitioner in the XML data file. These objects of the group process represent the input data for the module *ProcessLib* which computes the active position of the participants in this model. Each activity of a participant will be stored individually in the interface *ActivityMap*. This *ActivityMap* will be updated if a participant executes or aborts an intended activity represented by a thinXel. In this case the *ProcessLib* will synchronize each transition of the group process and will compute the new activity for the participants. The process data for a thinXel like a task or a topic for contributions will be provided by the module *DataLib*. This module is connected to a database and will also store all contributions of the participants during the group process.

The presented framework provides the data structures and functions to design and execute different kinds of group processes based on the GPML. It allows us to compute the activities of the participants, the data and events of these group processes. We assume that these properties build the base for the development of a universal GSS.

5. A Prototype of a universal GSS

We will now present an application of the framework for a prototype of a universal GSS. This prototype represents a web-based GSS that links the participants via the internet and was developed as a server application which provides the activities for the participants via a website. The participants can use this website to execute the intended activities of the group process. The prototype is built on the framework for a universal GSS and uses its data structures und functions to create and execute a group process (shown in Figure 4). This framework uses the group process model to compute the active activity for each participant of the group process and allows us to store or recall data from a database. As a result, the prototype provides the interface for the practitioner to use the functions of these software modules. We design an interface for the practitioners of the GSS, which allows them to upload a group process model based on the XML syntax and the GPML. The practitioners can configure the group process by applying participants to a selected group process or adapting the thinXel to the given context.

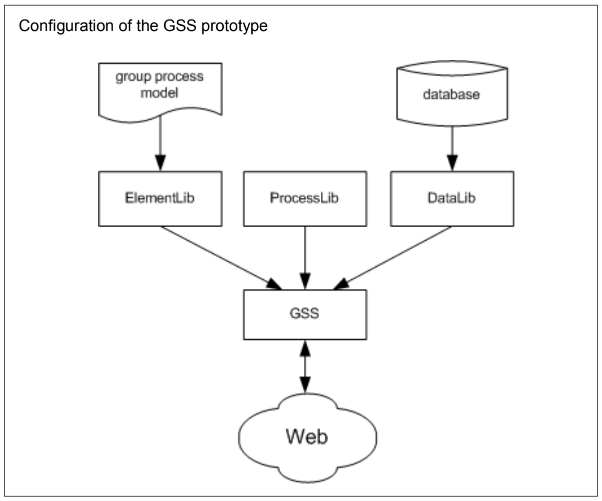
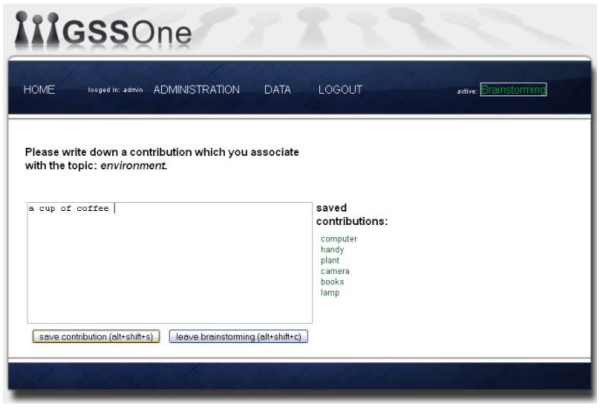


Figure 4. Configuration of the GSS prototype

The interface for an activity of a participant during the group process is defined by the element thinXel. As a result, our prototype of a GSS uses the information about the type of the active thinXel of a participant and the data of his configuration to design a webpage which allows the participant to executed or abort the intended activity. An example of this configuration is the webpage for the thinXel *create a contribution* as an element of the thinkLet *Brainstorming* (see Figures 2 and 5). This webpage includes a text area for a new contribution and the two buttons: *save contribution* and *leave brainstorming*. The button *save contribution* represents the execution of the indented activity. The participant can abort this activity by the button *leave brainstorming*. The prototype provides different interfaces for data- and context-oriented thinXels like the data-oriented thinXels*: to create a new contribution, to enhance an existing contribution with new attributes* and *to create a relation between two contributions*. We assume that these thinXels represent a basis foundation for different group processes and allow us to evaluate the functionality of our prototype.

We define an experiment to evaluate the requirement that the prototype should allow us to load and execute different group process models based on the GPML. In this context the prototype should adapt the interface of the GSS automatically to a designed group process and should allow the participants to collaborate with each other. It was not our goal to evaluate the efficiency of the prototype against any existing GSS; this is a topic for future work.



Please write down a contribution which you associate with the topic: *environment*.

a cup of coffee

saved contributions:
computer
handy
plant
camera
books
lamp

save contribution (alt+shift+s)
leave brainstorming (alt+shift+c)

Figure 5. Interface of the thinXel: create a contribution

To verify our assumption that the prototype allows us to design and execute different group process models, we used the following collaboration techniques: the thinkLet *LeafHopper* [3], the creativity technique *Analogies* [13] and the discussion technique *Six Hats* [1]. Each of these collaboration techniques was described in a logical group process model based on the GPML. An example of these group process models is shown in Figure 6 by the thinkLet *LeafHopper*, which represents a diverge thinkLet with the goal to create concepts that have not yet been considered by using different topics for a brainstorming process. The thinkLet starts with a verification process, which explains the topics to the group and the kind of ideas the group is expected to contribute. This verification is represented by the thinkLet element *thinkLet: verify*. A group of participants will be guided to this thinkLet element and verifies the input elements *list: topics* and *info: contribution*. The output of this verification is a list of topics that all participants agree on; it will be stored via a data path in the storage element *list: topics*. The process flow uses a transition element to collect all users from the thinkLet element *thinkLet: verify* and guides them individually to the thinXel element *select: topic*. This context-oriented thinXel element provides an interface that allows the participants to select a topic from the list of topics. A selected topic will be stored in the storage place element *topic*. The participant will be guided to the signal element *signal: start* that automatically sends a signal to the data-generator element *filter*. This generator filters all existing contributions for the selected topic and stores them in the storage place element *list: contribution*. The decision element *exist contributions* checks if the selected topic includes any contributions and – if any contributions exist – guides the participant to the context-oriented thinXel *select: contribution*.

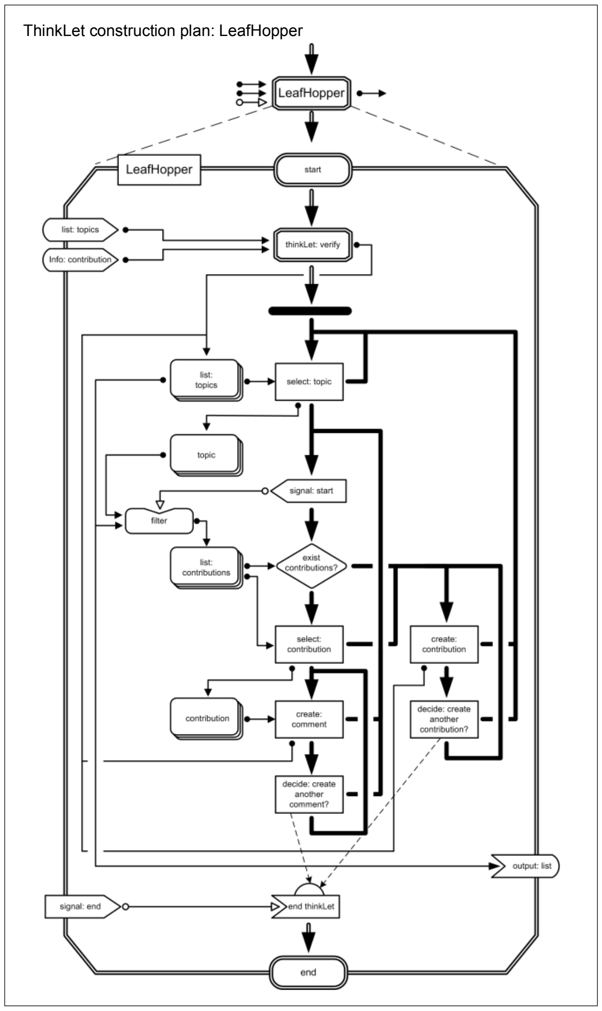


Figure 6. ThinkLet: LeafHopper

The participant is now invited to select one contribution from the provided contributions. If the participant selects a contribution, the process flow guides the participant to the data-oriented thinXel element *create: comment* which provides a text area to create a comment for the selected contribution. A created comment is stored in the storage place element *list topic* and will be available for each participant. The context-oriented thinXel *decide: create another comment* allows the participant using two buttons to decide if he or she wants to create another comment for the selected contribution or wants to be guided back to the signal element *signal start* to update the contributions of the selected topic. Another participant flow is described for the situation that no contributions exist for a selected topic. In this case the participant will be guided by the decision element to the data-oriented thinXel *create: contribution* that provides a text area to create a new contribution for the selected topic. A created contribution will also be stored in the global storage place element *list: topics*. The participant will be guided to the context-oriented thinXel *decide: create another contribution*. This thinXel guides the participant back to the thinXel element *create: contribution* or to the thinXel element *select: topic* to select another topic. The group process is stopped by a signal from the sender element *signal: stop*. The receiver element *end thinkLet* will interrupt each activity of the participants and a transition will guide the group to the exit of the thinkLet element.

The described group process models were transformed into XML syntax and loaded into the prototype and adapted to a given task. We defined an experiment which used these group processes to verify the functionality of the prototype. The participants in this evaluation were 23 students located all over Germany. Upon the login to the prototype, the participants were informed via a text chat about the task of the experiment. They received an introduction to the functionality of the prototype and were told to follow the facilitator instructions which the GSS provided. We analyzed the data from the participants against our requirements and recorded all contributions of the participants during the experiment. A questionnaire was used to document the impressions of the participants of the group processes.

In the first phase of the experiment we used the thinkLet *LeafHopper* to verify our assumption that a group process can be executed by a group of participants via the prototype. Each of the participants was guided automatically by the prototype through the defined participant flow. The evaluation of the questionnaire and created data shows that each participant could create contributions and comments for each of the given thinXels. The prototype stored the created contributions and allowed the participants to read each other contributions. This supports our assumption that the prototype allows us to execute a group process.

In the second phase of the experiment the assumption to use a group process with different parameters was evaluated. For this evaluation we divided the participants into two groups. Each group executed the creativity technique *Analogies* with different tasks and break conditions. One group was guided automatically to the next phase of the technique if a specific number of contributions were created and the other group after a specific time period. An analysis of the created contributions and the questionnaires showed that the prototype guided the participants automatically after the conditions were fulfilled.

In the last phase of the experiment we used the discussion technique *Six Hats* to evaluate the assumption that a facilitator can influence the participants and data during a group process. In this experiment the facilitator selected categories for contributions to support the participants in their discussion. Our observation showed that the facilitator could define and select the working category for the participants and change the participant flow by selecting a new category. The prototype guided the participants automatically to the next thinXel in relation to this participant flow.

To sum up, the experiment showed that different group processes can be described as group process models based on the GPML. The prototype allowed us to load and execute these group process models with different parameters. Each participant could be guided automatically by the described participant flow and a facilitator can influence this participant flow. The participants could use the contributions of other participants to collaborate with each other. We assume that these properties of the prototype represent the same basic properties for a universal GSS which could support group work by using GSS.

6. Discussion and Further Research

We defined a universal GSS as a software tool that:

- increases the applicability of a GSS for different kinds of group processes;
- supports practitioners in designing and executing a group process;
- adapts the system automatically to a designed group process to make facilitation easier;

These requirements resulted from the existing situation that experience is necessary for the adaption of a GSS to a designed group process and the facilitation during the group work. We think that a thinkLet- and thinXel-based GPML can establish the basis for this universal GSS. The GPML describes the activities of the participants, the data and events of a group process and allows us to describe different kinds of group processes in an unambiguous and compact representation. We presented a participant flow algorithm as an application of this GPML, which allowed us to compute the next position of a participant in a group process against their active position and activity in the group process. We further presented a framework for a universal GSS and developed a first prototype which allowed us to load and execute different kinds of group processes based on the GPML.

In summary, our prototype represents the first step on the path towards a universal GSS. The prototype offers a new possibility to design and execute different kinds of group processes. However, 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 GPML as well as including existing thinkLets from Collaboration Engineering;
- Developing a software tool to design a group process model with the presented GPML and XML syntax;
- Developing a user interface that supports practitioners in designing and executing a group process;
- Evaluating the efficiency of a universal GSS against existing GSSs.

Further research in this area will allow us to develop a universal GSS which could support group work by using GSS.

7. References

[1] de Bono, E., Six Thinking Hats, Penguin Books, London, 2000.

[2] R.O. Briggs, M. Adkins, D.D. Mittleman, J. Kruse, S. Miller, J.F. Nunamaker, „A technology transition model derived from field investigation of GSS use aboard the USS Coronado", Journal of Management Information Systems 15, M.E. Sharpe Inc., Winter 1999, pp. 151-195.

[3] R.O. Briggs, G.J. de Vreede, ThinkLets: Building Blocks for Concerted Collaboration, GroupSystems.com, 2003.

[4] R.O. Briggs, G.J. de Vreede, J.F. Nunamaker, "Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems", Journal of Management Information Systems 19, M.E. Sharpe Inc., Spring 2003, pp. 31-63.

[5] R.O. Briggs, G.L. Kolfschoten, G.J. de Vreede, D.L. Dean, "Defining key concepts for collaboration engineering", Proceedings of the 12th Americas Conference on Information Systems, Garcia I and Trejo R (eds), Acapulco, Mexico, 2006, pp. 121–128.

[6] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Pattern: Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing Company, Amsterdam, 1995.

[7] R. Grohowski, C. McGoff, D.R. Vogel, B. Martz, J.F. Nunamaker, "Implementing electronic meeting systems at IBM: lessons learned and success factors", MIS Quarterly 14, Management Information Systems Research Center, University of Minnesota, Dec. 1990, pp. 369-383.

[8] S.W. Knoll, R. Chelvier, "Formalized Online Creativity using ThinXels", Proceedings of the 10th European Conference on Creativity and Innovation, Copenhagen, 2007.

[9] S.W. Knoll, M. Hörning, G. Horton, "A Design Approach for a Universal Group Support System using ThinkLets and ThinXels. Proceedings of the Group Decision and Negotiation 2008, INESC Coimbra, Coimbra, 2008.

[10] G.L. Kolfschoten, R.O. Briggs, G.J. de Vreede, P.H.M. Jacobs, J.H. Appelman, "A conceptual foundation of the thinkLet concept for Collaboration Engineering", International Journal of Human-Computer Studies 64, Academic Press Inc., Duluth, MN, USA, 2006, p.p. 611-621.

[11] J.F. Nunamaker, A. Dennis, J. Valacich, D.R. Vogel, J.F. George, "Electronic Meeting Systems to Support Group Work", Communication of the ACM 34, pp. 40-61.

[12] C.E.A. Shannon, "Mathematical Theory of Communication", Bell System Technical Journal 27, 1948, pp. 379-423.

[13] VanGundy, A. B., Techniques of Structured Problem Solving, Van Nostrand Reinhold Company Incorporated, 1981

[14] G.J. de Vreede, D.R. Vogel, G.L. Kolfschoten, „Fifteen years of GSS in the field: A comparison across time and national boundaries", Proceedings of the 36th Annual Hawaii International Conference on System Science, IEEE Computer Society Press, Los Alamitos, Calif. 2003.

[15] G.J. de Vreede, R.O. Briggs, "Collaboration engineering: designing repeatable processes for high-value collaborative tasks", Proceedings of the 38th Annual Hawaii International Conference on System Sciences, IEEE Computer Society Press, Los Alamitos, Calif. 2005.

[16] G.J. de Vreede, G.L. Kolfschoten, R.O. Briggs, "ThinkLets: A Collaboration Engineering Pattern Language", International Journal of Computer Applications in Technology 25, Inderscience Publishers, 2006, pp. 140-154.