View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

A Structured Approach to Designing Interleaved Workflow and Groupware Tasks

Amit Deokar
University of Arizona

Therani Madhusudan
The University of Arizona

Robert Briggs
GROUPSYSTEMS.COM

Jay Nunamaker, Jr.
The University of Arizona

Follow this and additional works at: http://aisel.aisnet.org/amcis2004

Recommended Citation
Deokar, Amit; Madhusudan, Therani; Briggs, Robert; and Nunamaker, Jr., Jay, "A Structured Approach to Designing Interleaved Workflow and Groupware Tasks" (2004). *AMCIS 2004 Proceedings*. 509.
http://aisel.aisnet.org/amcis2004/509

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2004 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

A Structured Approach to Designing Interleaved Workflow and Groupware Tasks

Amit V. Deokar
Department of MIS
The University of Arizona
Tucson, AZ 85721
amitd@eller.arizona.edu

Madhusudan, T.
Department of MIS
The University of Arizona
Tucson, AZ 85721
madhu@eller.arizona.edu

Robert O. Briggs
GROUPSYSTEMS.COM
1430 E. Fort Lowell Rd. #301
Tucson, AZ 85719
bbriggs@groupsystems.com

Jay F. Nunamaker, Jr.
Department of MIS
The University of Arizona
Tucson, AZ 85721
nunamaker@eller.arizona.edu

ABSTRACT

Modern organizations are investing in organizational process coordination tools such as groupware and workflow systems for better decision-making and business process execution. However, these systems are deployed in a standalone manner with the result that business processes that span these systems are highly fragmented. Processes are designed and managed independently for each system, resulting in large administration overheads to coordinate activities in business processes that use these systems. This paper reports on the development of a unified framework to design distributed organizational processes that interleave both individual and group activities for a seamless flow of information, promote efficient information sharing and effective task execution and improve the quality of decision making. Though process modeling for workflow and groupware systems has been treated as separate tasks in the extant literature, we provide a declarative AI planning based framework for enabling organizational process design based on recent developments of *thinkLets* for collaborative engineering and case-based reasoning for workflow design. We illustrate our framework in the context of designing processes to support new product development and describe a prototype system currently under development.

Keywords

Group systems, workflow systems, process definition, AI planning, thinkLets

INTRODUCTION

Organizations have adopted large scale deployment of IT systems to support a variety of information management and decision-making activities across the enterprise. However, the widespread use of IT systems within organizations has led to major changes in routine business process execution such as 1) introduction of complex flows of information between systems and knowledge workers, 2) the opacity of information flows across systems leading to mismanagement, 3) fragmentation of business processes into uncoordinated individual and group activities, and 4) increasing costs of process coordination and management. To ameliorate such issues, organizations are adopting process coordination tools such as groupware and workflow management systems to support seamless process execution and support both individual and group knowledge worker activities. Such process support systems are being deployed in organizations in an ad hoc manner without any overall guiding process design principles leading to additional costly overheads without the requisite benefits. The advent of flexible process delivery technologies such as web services and increasing complexity of knowledge worker tasks requires a structured approach to organizational process design, providing effective task and information management.

This paper reports on the development of a conceptual framework to support design of organizational processes considering both individual and group activities in a unified manner. A business process is modeled as a problem solving mechanism consisting of a series of steps (also called a plan), each of which may be an individual or group activity. The task of designing business processes is the development of an effective plan to solve a problem by searching the process design space (which may consist of many possible plans to solve an individual business problem). Our proposed model builds on recent advances

in the field of AI planning for dynamic systems (Russell and Norvig, 2003), wherein declarative process representations enable the search for effective process definitions to solve a problem, given well-defined criteria (Madhusudan, Zhao, and Marshall, 2004).

The need for structured approaches to organizational process design has been suggested in (Basu and Blanning, 2003). The existence of structured routines in organizations at multiple levels of abstraction has been discussed in (Pentland, 1995; Yoshioka, Herman, Yates, and Orlikowski, 2001). However, extant research in process modeling has addressed the design of workflow models primarily supporting individual tasks (Basu and Kumar, 2002; Stohr and Zhao, 2001) in an independent manner from the design of group activities (Dourish, Holmes, MacLean, Marqvarsden and Zbyslaw, 1996); Briggs, Vreede and Nunamaker, 2003). The design of groupware systems has been addressed in (Dourish and Edwards, 2000). However, process design to support deployment of workflow and groupware tools in organization requires the explicit interleaving of individual and group activities, allowing for effective information flow and process coordination. A generic model for enabling team activity in workflow systems is outlined in (Aalst and Kumar, 2001).

Main contributions of this paper include: 1) a declarative AI planning based formalism to describe both individual and group activities, and 2) a structured design process including design of group activities. We illustrate the benefits of the formalism to support the development of intelligent build-time tools for process design. We describe an architecture for developing intelligent buildtime design tools, discuss the benefits of the formalism, and outline a variety of avenues for further research.

The remainder of the paper is organized as follows: Section 2 provides an overview of the structured approach to process design which interleaves individual and group activities. Section 3 discusses the basic computational models supporting the different design stages. We illustrate the framework in designing a process for new product development. Section 4 describes our preliminary efforts at the implementation and evaluation of the framework. Concluding remarks are given in Section 5.

OVERVIEW OF APPROACH

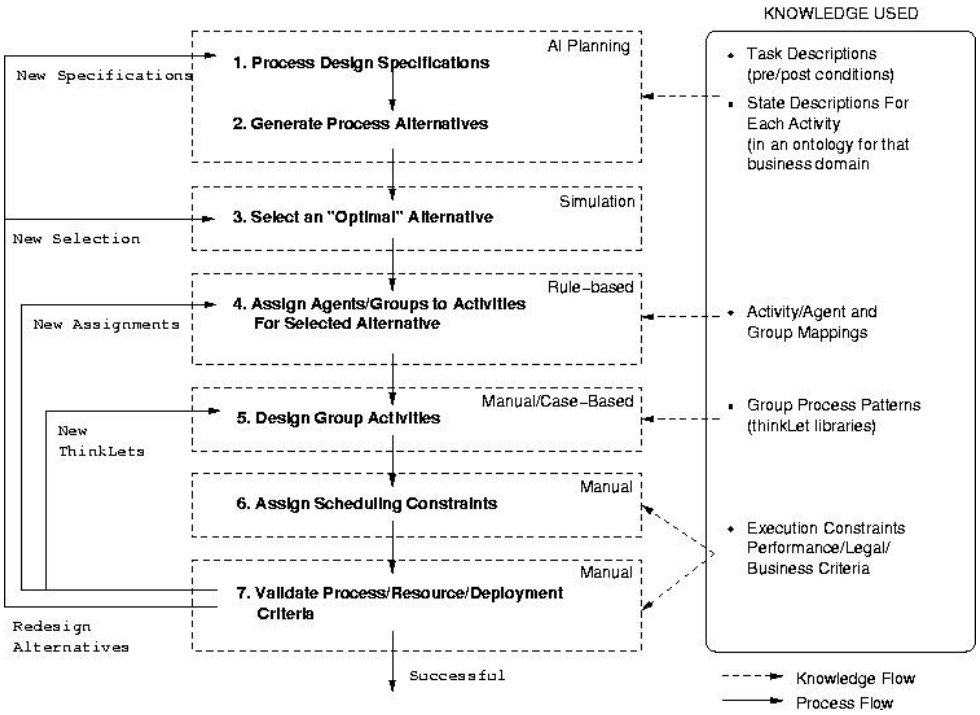

Figure 1. A Structured Process Design Cycle

Our overall approach to organizational process design is shown in Figure 1. The structured design cycle is supported by a well-defined knowledge-base from which appropriate knowledge is used at different stages of the cycle. Further, the figure also illustrates how each of the steps in the lifecycle is supported. For example, step 3 is currently supported via manual simulation and animation of a process. In the following paragraph, we provide an overview of the various steps.

In steps 1 and 2, process specifications for a given business domain are defined in a well defined ontology (as shown on the right-hand side of the figure). Process specifications may only specify the initial conditions, final conditions of the process

and some intermediary constraints. The ontology includes well-defined task descriptions which manipulate the information state of the process (further discussed in Section 3). Given such specifications, a variety of process alternatives may be generated using AI planning algorithms. Each process alternative may consist of different task combinations. In step 3, a process alternative may be selected (possibly based on results of interactive simulation). In step 4, for each task in the chosen process alternative, an individual agent or a group of agents (based on the organizational chart) may be assigned to execute all the roles associated with that task. Individual tasks can be accomplished usually by a particular organizational member who provides the requisite skills. Group tasks require interaction between and knowledge from different areas of expertise to successfully accomplish the task. An individual may not provide the most effective execution of the task.

Group tasks (involving multiple individuals) usually needs to be organized further to allow for efficient information flow and successful meetings (Ellis, Gibbs and Rein, 1991). Therefore in step 5, each group activity is further decomposed into basic collaborative process patterns as outlined in (Briggs et al., 2003). Repeatable collaborative process patterns called *thinkLets* may be used to coordinate the group activity. Details of thinkLets are discussed below. During step 5, a process designer has to select appropriate thinkLets and compose them into an effective sequence. Our current approach is based on selecting appropriate thinkLets for the current group task from a library of such thinkLets that have been successfully deployed in a real world context. Finally in step 6, specific organizational roles are filled with appropriate individuals, resources assigned and tasks are scheduled appropriately. Further validation may be performed in step 7 to consider other criteria as listed in the figure. Unsuccessful process models may be appropriately revised at many stages of the cycle as shown in the figure.

Current approaches to deploying workflow and groupware systems do not follow such a structured approach leading to major complexity and costly maintenance. In order to support such a structured process, our current research is focused on developing appropriate buildtime tools to support steps 1,2, 4 and 5 of the process design cycle. Towards this end, we have developed a declarative process formalism to model processes at the task-level of granularity focusing on the information dependencies between tasks and an explicit state representation. Further, to support design of group tasks based on thinkLets, we have developed a formal representation of thinkLets to represent, store and retrieve thinkLet patterns to support process design. We describe these formalisms in the following section.

COMPUTATIONAL MODELS FOR PROCESS DESIGN

Current workflow modeling systems are based on a variety of process modeling formalisms including: 1) Petri Net based process models (Aalst and van Hee, 2002), 2) Event-Condition-Action formalism (Kumar and Zhao, 1999), 3) UML activity diagrams (Booch, Rumbaugh and Jacobson, 1998), 4) Metagraphs (Basu and Blanning, 2000) and 5) Process algebraic models such as FSP (Magee and Kramer, 1999). These formalisms support a manual approach to modeling wherein DAG (directed acyclic graph) models of processes are constructed in a graphical interface. Tools based on a Petri Net formalism support modeling of loops within a block structure. The buildtime model thus created is primarily procedural in nature. The construction of such models depends on the experience of the process designer to construct robust process models that can cope with various runtime contingencies. Support for exploring the space of alternative process models is minimal. Further, the tasks in these process models do not discriminate between group and individual activties. Recent efforts have focused on developing interoperable representation between different formalisms such as XML Process definition Language (XPDL) (XML Process Definition Language, 2003).

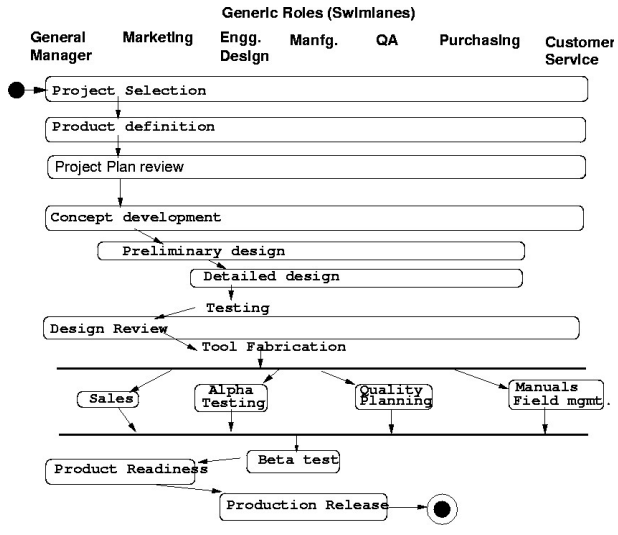

Figure 2. UML Activity Diagram for New Product Development

```
<Activities>
              ..............
               <Activity Id="NPD_PKG01_Wor1_Act2" Name="Project Selection">
                   <Implementation>
                       <Tool Id="ProjectSelection" Type="APPLICATION">
                           <ActualParameters>
                               <ActualParameter>SelectedProject</ActualParameter>
                           </ActualParameters>
                       </Tool>
                   </Implementation>
                   <Performer>Par1</Performer>
                   <ExtendedAttributes>
                       <ExtendedAttribute Name="ParticipantID" Value="Par1"/>
                       <ExtendedAttribute Name="XOffset" Value="210"/>
                       <ExtendedAttribute Name="YOffset" Value="40"/>
                       <ExtendedAttribute Name="SystemActivity" Value="Groupware Service"/>
                   </ExtendedAttributes>
               </Activity>
              ..................
```

Figure 3: Snippet of XPDL Representation

Shown in Figure 2 is the UML-activity diagram for a New Product Development process (NPD) (outlined in (Ulrich and Eppinger, 1995)). The figure illustrates different tasks of the NPD process spread across multiple swimlanes (*i.e.* organizational roles that need to participate in each task). For sake of clarity, we have not included all the decision and branching nodes. The key point to note is that early stages involve group activities whereas later stages involve individual activities. Information needs to flow seamlessly between these activities. Further, the process may iterate if needed. Also shown in Figure 3 is a snippet of an XPDL fragment of the above process description (generated from a graphical workflow tool wherein the workflow model is created). Analysis of the XPDL representation from an integrated design perspective shows the following deficiencies: a) XPDL does not provide the ability to identify that a task spans multiple swimlane, distinguish between individual and group activities (It provides a singleton tag called `Participant`), b) information flow between individual and group activities is not explicitly represented and thus cannot be effectively traced or managed during process execution, c) graphical tools provide no support for modeling the internals of group activities in an explicit manner, and d) the lack of an explicit state representation prevents dynamic task scheduling and coping with interleaved group and individual tasks. For example, one may need to coordinate a design review meeting to refine a design feature during `Alpha testing` task as shown in Figure 2. Our recent research has focused on the use of AI planning formalisms to support process modeling (Madhusudan and Uttamsingh 2003; Madhusudan, Zhao and Marshall, 2004). We describe the basic ideas of the formalism in the following sections.

Declarative Process Representations

Our declarative process model is based on the (first-order logic based) action-state formalism for modeling dynamic systems (Russell and Norvig, 2003; Reiter, 2001). Action-state formalisms support two fundamental notions, a) a means for describing states, and b) a means to describe transitions between states. A state is a snapshot of the world being modeled at an instant of time. State descriptions are composed of atomic propositions (in some formal logic, usually first-order). This is to ensure that at a requisite level of granularity of the world, the effects of an action can be modeled adequately. Propositions represent properties or in general relationships among entities in the domain. The truth value of propositions may change in the course of time as a consequence of actions. Thus a state is characterized by defining which propositions are true and which are false. The second notion is that of actions. Actions cause state transitions when performed. A key assumption is that actions on execution affect just a small fraction of an entire state. Describing the effects of an action thus amounts to specifying which propositions change their truth value when an action is executed. Action descriptions describe the truth value of propositions in a state that enables that action for execution (pre-conditions) and also the truth value of propositions in the ensuing state after action (post-conditions). Thus many actions may be eligible for execution in a given state. Further, actions may have deterministic or non-deterministic effects *i.e.* transitions from a state may be to just one state or any one of a set of states.

A variety of action-state formalisms exist in the literature. Recent research has proposed the use of AI planning techniques for workflow modeling (for dynamic systems such as web services) (Srivastava and Koehler, 2003) and also for design of group tasks such as collaborative writing (Babaian, Grosz, and Shieber, 2002). We have developed our framework on the Hierarchical Task Network (HTN) formalisms for AI planning (Nau, Cao, Lotem and Munoz-Avilla, 1999). In these formalisms, states are represented as collections of first order predicates. Tasks are of two types, 1) Compound tasks called

methods (which can be decomposed further), and 2) Primitive tasks. Both types of tasks are defined by a well-defined set of pre- and post-conditions which define when a task is applicable to a current state and what effects the task may have on the state, when it is successfully executed. Shown below in Figure 4 is the representation of a state, a primitive task and a compound task (also called a method). The figure illustrates the state of the design (defined by First-order logic predicates) and the definition of a compound task (for detailing a design as a set of components) and the primitive task sources appropriate components from suppliers and updates the current product state. Such a description is also called a domain description. Different business domains may require different descriptions.

```
;;STATE
(design_status INITIAL)
(design_stage REVIEW)
(design_func ((CONVERT SINUSOIDS) (STORE MIN ENERGY)))
(design_specs ((.....................))
(design_components (( C1 C2.........))
(Design_geom_definition ((.........
(Design_customer..
(Project_begin_date........
.....................................
;;COMPOUND TASK
 (:method (detaileddesign)
;;PRECONDITIONS
  (
  (currproductstate ?y)
  (valid-curr-state ?y '(C23 C27))
  (goal (goalproductstate ?g))
  (assign ?r_components (find_remaining_comps `?y `?g))
  (valid-remaining ?r_components '(C1 C2 C45 C49))
  (assign ?currdesign (check_design `?ws `?r_components)))
;;POSTCONDITIONS
 (:ordered
 (!Obtain ?r_components)
 (!Check_new_assembly ?r_components ?y)
 (!Test_assembly ?r_components ?y)
 (!Review_assembly ?r_components ?y)
 (!Finalize_assembly ?r_components ?y))
--------------------------------------
;;PRIMITIVE TASK
 (:operator (!obtain ?comps)
;;PRECONDITIONS
 ((currproductstate ?y))
 ((newproductstate ?z))
;;POSTCONDITIONS
  (include-components ?comps)
  (source-comps ?comps)
```

Figure 4. Planning Representation for Tasks and States

AI Planning For Generating Process Alternatives

Given such domain descriptions, the task of process modeling is to generate a suitable set of action sequences to solve a business process problem. The technique of AI planning involves generating a network of actions (or operators), *i.e*., a plan, that transforms a world state from an initial state to a goal state. An operator is a parameterized function that transforms a given state into a new state if the operator is applicable in the given state. The parameters to the operator provide access to the various entities in the current state and the new values that their current values may need to be changed to, in order to evolve into the new state. More specifically, a plan is composed of (a) a sequence of operators with a temporal ordering constraint, *i.e*., $O_i < O_j$ implies that operator O_i must occur sometime before operator O_j; and (b) for each parameterized operator O_i a set of parameter values that can be applied to the current state to transform it into a new state.

A simple formulation of the classical planning problem defines three inputs also called the domain description (Russell and Norvig, 2003):

- a description of the initial state of the world in some formal language, usually first-order predicate logic and its variants
- a description of the goal *i.e*. the final state of the world in the same formal language as above
- a description of the possible operators that can be performed

A domain-independent planner's output is a sequence of operators, a plan, which, when executed in a world satisfying the initial state description, will achieve the goal. A plan thus defines a project network, with ordering constraints between the different operators (or tasks) in the plan. This project network may be transformed into the process graph representation of a workflow.

HTN planning (Nau et al., 1999) is a search technique that creates plans by task decomposition. The planning problem is specified by an initial task network, which is a collection of tasks that need to be performed under a specified set of constraints. The planning process decomposes tasks in the initial task network into progressively smaller subtasks until the task network contains only primitive tasks or operators. The decomposition of a task into subtasks is performed using a method from a domain description. A method specifies how to decompose the task into a set of subtasks. Each method is associated with various constraints that limit the applicability of the method to certain conditions and define the relations between the subtasks of the method. HTN planning performs recursive search of the planning state space via task decomposition and constraint satisfaction. HTN planning techniques have been implemented in a variety of real-world systems (Russell and Norvig). Due to lack of space, we do not detail the algorithm. However, we note that the algorithm produces multiple alternative plans (each plan being a path through the state space from the initial state to the goal state). Further, the plan also identifies the input and output of each action in the plan. Plans may consist of both sequential and concurrent tasks. A key feature to note is that a plan corresponds to a customized workflow instance for the current problem at hand. Thus, the planning based approach to process modeling allows for considering unique features of the current business problem.

Assigning Task Types

Once a process plan has been developed, it may need to be further embellished with information regarding the organizational roles involved in the task. Individual and group task may need to be defined. In our current prototype, we make these assignments based on a rule-base we have developed for each business domain. Rules define which tasks are group activities and which organizational roles are defined. These rules were developed based on analysis of real world cases (that were gathered in the context of prototype development in (Madhusudan et al., 2004).

Designing Group Tasks

After task types have been defined, group tasks need to be elaborated further. Our approach to designing group tasks is based on the development of collaborative process patterns.

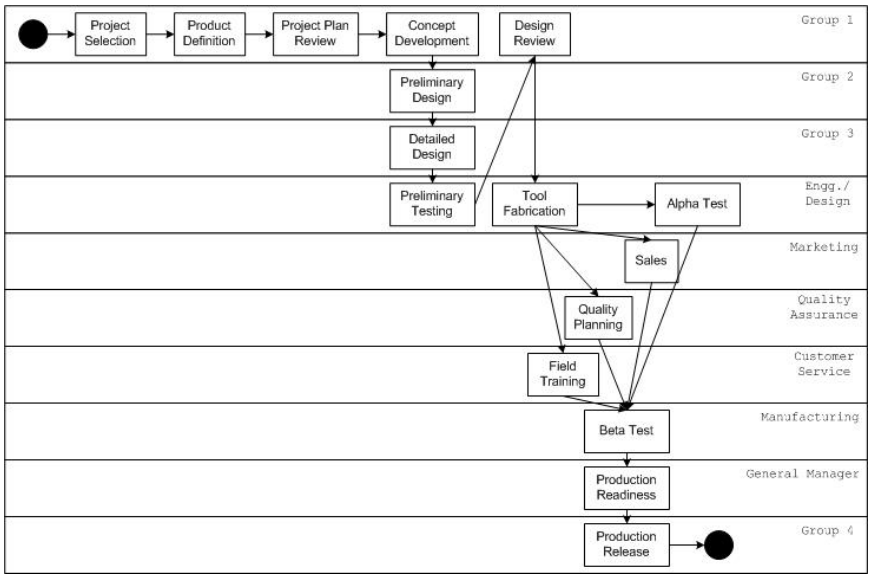

Figure 5: Extending the Workflow Model with Group Activity Information

Figure 5 above provides an alternative view of the new product development process shown in Figure 2 from the perspective of group tasks. For example, Group 1 (spanning multiple swimlanes in Figure 2) executes the tasks of project selection, definition, plan review and design review. Next, preliminary design and detailed design are also group activities, but, they are performed by groups with different role compositions. Furthermore, preliminary testing activity is solely performed by the

engineering and design personnel and do not involve interaction with other roles. Thus, a single agent scenario can be considered as a special case of the group agent representation.

Consider a group activity such as concept development shown in Figure 5. The rationale behind executing the same as a group activity is that inputs from multiple perspectives are needed from different stakeholders, each role involved therein can bring different perspectives to the table, and no single role can have all the required knowledge. Thus, by including available knowledge early, pricey and time-consuming design iterations at later stages of the product development process can be avoided. Similar rationales apply to other group activities. Studies on deployments of groupware systems show that each group activity is goal-oriented and follows a certain underlying structure or patterns. These patterns have been termed as *thinkLets* (Briggs et al., 2003). A thinkLet is a sequence of group interactions, representing *one* of the following basic patterns of thinking (by each group participant and thus the group): divergent, convergent, organizational, elaboration, abstraction, evaluation, and or consensus making. For example, the product concept development group activity, may exhibit the following thinkLets (in order): 1) *FreeBrainstorm* (to diverge in search of new ideas), 2) *FastFocus* (to organize ideas), 3) *StrawPoll* (to evaluate ideas), and 4) *Crowbar* (to build group consensus). It is also possible to obtain similar results by following another sequence of thinkLets such as 1) *LeafHopper* (to diverge, given a set of different possible ideas), 2) *FastFocus* (to organize ideas), 3) *MultiCriteria* (to evaluate against multiple criteria), and 4) *Red-Light-Green-Light* (to build group consensus). Currently, tools exist to explicitly model thinkLets in groupware applications. We have used Cognito for our research developed by GroupSystems.com (GroupSystems Cognito, 2003).

Figure 6. Structuring Group Activity with ThinkLets

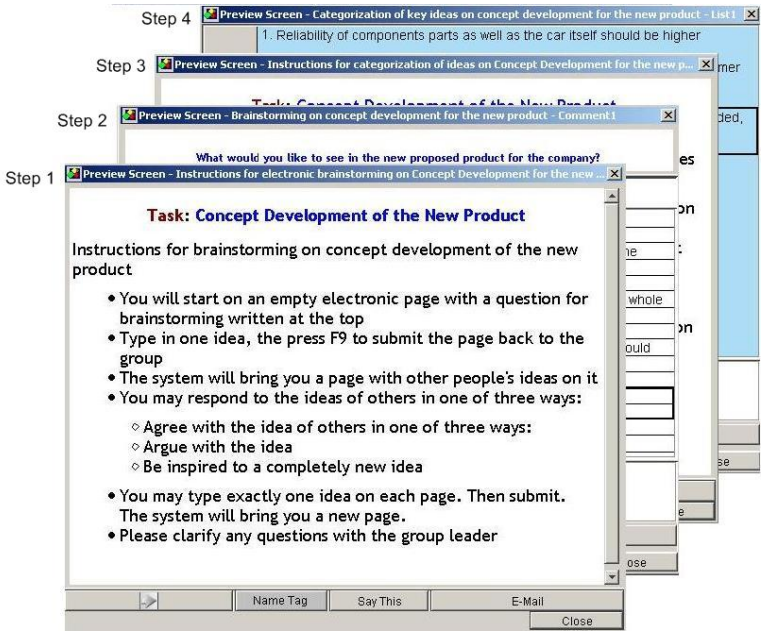

Figure 7. Overlay of Screenshots to Represent Group Interaction Steps in ThinkLets

Figure 6 shows an illustration of structuring group activities with thinkLets in the context of the concept development activity. On the left-hand side, the different organizational representatives involved in the activity are listed. Across the top (from left-to-right), the different thinkLets are listed. The figure shows only two thinkLets (from the overall sequence for

illustrative purposes), namely, *FreeBrainstorm* and *FastFocus*. Also, each of these thinkLets, in turn, is decomposed further into a series of interactions with each individual role. Usually, the same series is adopted across all roles participating the meeting. In this example, the *FreeBrainstorm* thinkLet consists of step 1, which is the instructions step, and step 2, which is the brainstorming step. Similarly, the *FastFocus* thinkLet consists of step 3, which is the instructions step, and step 4, which is the categorization step. Figure 7 shows an overlay of screenshots for these steps in Cognito.

```
<Roles>
        <Role Id="Role1" Name="General Manager"/>
        <Role Id="Role2" Name="Marketing"/>
        <Role Id="Role3" Name="Engineering and Design"/>
        <Role Id="Role4" Name="Manufacturing"/>
        <Role Id="Role5" Name="QA"/>
        <Role Id="Role6" Name="Purchasing"/>
        <Role Id="Role7" Name="Customer Service"/>
</Roles>

<ThinkletSequence Id="Seq1" Name="NPD Concept Development Activity">
        <ThinkLet Id="TL1" Name="FreeBrainstorm" Desc="For generating new ideas by brainstorming">
                <Action Id="Step1" Name="Instructions for Free Brainstorming">
                        <ActionInput> Init Thinklet</ActionInput>
                        <ActionOutput>Instructions Reviewed/Accepted</ActionOutput>
                        <ActionDescription>
                           (1) You will start on an empty electronic page...
                        </ActionDescription>
                </Action>
                <Action Id="Step2" Name="Free Brainstorming">
                        <ActionInput>
                           <QuestionList>
                             <Q1>
                                 What would you like to see in the new proposed product for the company?
                             </Q1>
                           </QuestionList>
                        </ActionInput>
                        <ActionOutput>
                           <BrainstormingIdeasList>
                             <Idea1>I would like the product to more reliable</Idea1>
                             <Idea2>The product should have a better interface</Idea2>
                             ...........
                           </BrainstormingIdeasList>
                        </ActionOutput>
                        <ActionDescription>
                                This activity is useful when you want to follow a divergence pattern and
                                generate a large group of ideas
                        </ActionDescription>
                </Action>
        </ThinkLet>
        <ThinkLet Id="TL2" Name="FastFocus" Desc="To converge from a large set of ideas to fewer ones">
                <Action Id="Step1" Name="Instructions for Idea Categorization">
                        <ActionInput>List of ideas from a divergence thinkLet</ActionInput>
                        <ActionOutput>A clean, non-redundant list of key issues</ActionOutput>
                        ..........
                </Action>
                ............
        </ThinkLet>
        ............
</ThinkletSequence>
```

Figure 8: XML Representation of ThinkLet Sequences

The process of designing sequences of thinkLets is currently manual and based on experience. To support this process, we are currently developing a library of such patterns for different group activities in the context of different domains (e.g. software development, engineering design). Current XML representation of such a pattern is shown in Figure 8 (for a sequence of *FreeBrainstorm* and *FastFocus* thinkLets in the example under discussion). We are currently including additional tags in XPDL for integrating representation of group activities based on the above framework.

WORK-IN-PROGRESS, DISCUSSION AND RELATED WORK

Our current efforts are focused on implementing an interactive design tool based on the process model defined in Figure 1. We have completed prototyping the basic planning cycle. However, work is currently underway to complete domain

descriptions and develop the thinkLet library. Further, we have also collected a set of test cases to validate our approach. During our research effort thus far, some valuable lessons have been learnt. Firstly, though the declarative formalism provides the requisite ability to flexibly generate custom workflows, the knowledge has to be gathered, tested and validated. Developing such declarative descriptions is a knowledge-intensive task. However, we believe such effort needs to be pursued with the advances in the development of the Semantic Web and web services. Secondly, much of the process modeling effort views the process design activity as a programming activity rather than a design activity at a much higher level of abstraction. Given this view, techniques for converting process model knowledge available in Data Flow Diagrams (DFD) and activity diagrams need to be developed. The recent advent of Model-Driven Architecture (MDA) approaches to systems development reflects this trend. From an operational level, standardizing on predicate descriptions, developing appropriate ontologies for any business domain and developing process selection criteria require further research.

Our proposed framework unifies both group and individual tasks based on defining them as activities that cause state transitions in a dynamic domain. Thus information may flow between both types of activities in a standardized manner. However, how such information may be processed internal to a group task (or individual task) is highly implementation dependent. Our focus on structuring group tasks further based on the notion of collaborative process patterns is motivated by the need to lower the operational costs of group activities (requiring facilitators) and the lack of information management between multiple group activity sessions. However, though group activities are inherently asynchronous (during execution), an overall structure provides for a focused result-oriented activity. Our current research is focused on further developing the notion of *interactions* as a unit of group activity and also to develop guidelines for an intelligent facilitation agent. Additionally, current group systems do not have any explicit semantics of the content generated during the interaction. Our state-based representation supports modeling of such content transformations explicitly.

We note that the process design cycle in Figure 1 is similar to the workflow development process outlined in (Weske et al, 2001) at a high level of abstraction. However, the focus in the proposed approach is on developing automated tools to support decision-making at the various stages of process design in contrast to a manual approach. The need for improved tools integrating workflow systems and teamware has been discussed recently in (Georgakopoulos, 2004) including the need for developing declarative task coordination policies. Team oriented tasks in the context of new product development are well-described in (Subrahmaniam et al, 1997).

CONCLUSION

In this paper, we have presented a structured approach to design business process consisting of interleaved individual and group activities. The design process is based on a declarative formalism to model a process as a path through a sequence of states, wherein the state representation captures the domain semantics in an explicit manner. The approach provides an effective means to model information flow between group and individual tasks in a seamless manner. Further, the paper also outlines basic computational approaches to support the different stages of the design cycle building on developments in AI planning.

We plan to continue with our current prototyping efforts and perform an evaluative study in the short-term to validate the design approach. Additionally, we plan to extend the XPDL representation to account for group activities and develop execution management techniques based on interleaving process design and process execution to cope with organizational dynamics.

REFERENCES

1. GroupSystems Cognito. http://www.groupsystems.com/products/cognito.htm, 2003.
2. XML process definition language. http://www.wfmc.org/standards/XPDL.htm, 2003.
3. Aalst, W. M. P. and Kumar, A. (2001) A Reference Model for Team-Enabled Workflow Management Systems, *Data Knowledge Engineering*, 38, 3, 335-363.
4. Aalst, W. M. P. and van Hee, K. (2002) Workflow Management, MIT Press.
5. Babaian, T., Grosz, B. J. and Shieber, S. M. (2002) A Writer's Collaborative Assistant, *Proceedings of the Intelligent User Interfaces Conference*, CA, USA.
6. Basu, A. and Blanning, R. W. (2003) Synthesis and decomposition of processes in organizations, Information Systems Research, 14, 4, 337–355.
7. Basu, A. and Kumar, A. (2002) Research Commentary: Workflow Management Issues in e-Business, *Information Systems Research*, 13, 1, 1–14.

8. Basu, A. and Blanning, R. W. (2000) A formal Approach to Workflow Analysis, *Information Systems Research*, 11, 1, 17–36.
9. Booch, G., Rumbaugh, J. and Jacobson, I. (1998) The Unified Modeling Language User Guide, Addison-Wesley.
10. Briggs, R. O., Vreede, G. J. and Nunamaker, J. F. Jr. (2003) Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems, *Journal of Management Information Systems*, 19, 4.
11. Dourish, P. and Edwards, W. K. (2000) A tale of Two Toolkits: Relating Infrastructure and Use in Flexible CSCW Toolkits, *Journal of Computer Supported Cooperative Work*, 9, 33–51.
12. Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P. and Zbyslaw, A. (1996) Freeflow: Mediating between Representation and Action in Workflow Systems, In *Proceedings of CSCW'96 (ACM)*, 190–198.
13. Ellis, C., Gibbs, S. J. and Rein, G. (1991) Groupware: Some Issues and Experiences, *Communications of the ACM*, 34, 1, 38–58.
14. Georgakopoulos, D. (2004) Teamware: An Evaluation of Key Technologies and Open Problems, *Distributed and Parallel Databases*, 15, 9-44.
15. Kumar, A. and Zhao, L. (1999) Dynamic Routing and Operational Controls in Workflow Management Systems, *Management Science*, 45, 2, 253–272, 1999.
16. Madhusudan, T. and Uttamsingh, N. (2003) A Declarative Approach for Composition of Web Services in Dynamic Environments, *Decision Support Systems*, Conditionally Accepted.
17. Madhusudan, T., Zhao, L. and Marshall, B. (2004) A Case-based Reasoning Framework for Work-flow Model Management, *Data and Knowledge Engineering*, To appear.
18. Magee, J. and Kramer, J. (1999) Concurrency, Wiley.
19. Nau, D. S., Cao, Y., Lotem, A. and Munoz-Avilla, H. (1999) SHOP: Simple Hierarchical Ordered Planner, In *Proceedings of IJCAI-99*, 968–973.
20. Pentland, B. T. (1995), Grammatical Models of Organizational Processes, Organization Science, 6, 5, 541–556.
21. Reiter, R. (2001) Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, MIT Press.
22. Russell, S. and Norvig, P. (2003) Artificial Intelligence: A Modern Approach, Prentice-Hall, second edition.
23. Srivastava, B. and Koehler, J. (2003) Web Service Composition - Current Solutions and Open Problems, In *Proceedings of ICAPS'03 Workshop on Planning for Web Services*, Trento, Italy.
24. Stohr, E. A. and Zhao, L. (2001) Workflow Automation: Overview and Research Issues, *Information System Frontiers*, 3, 3, 281–286.
25. Subrahmanian, E., Reich, S. L., Dutoit, A., Cunningham, D., Patrick, R., Thomas, M., and Westerberg, A. W. (1997) The n-dim Approach to Creating Design Support Systems, In *Proceedings of ASME Design Engineering Technical Conference '97*, Sacramento, CA, USA.
26. Ulrich, K. T. and Eppinger, S. D. (1995) Product design and development, McGraw-Hill, NY.
27. Weske, M., Goesmann, T., Holten, R., and Striemer, R. (1999) A Reference Model for Workflow Application Development Processes, In *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration*, San Francisco, CA, USA, 1-10.
28. Yoshioka, T., Herman, G., Yates, J. and Orlikowski, W. (2001) Genre Taxonomy: A Knowledge Repository of Communicative Actions, *ACM Transactions on Information Systems*, 19, 4, 431–456.