Association for Information Systems

AIS Electronic Library (AISeL)

AMCIS 1997 Proceedings

Americas Conference on Information Systems (AMCIS)

8-15-1997

Object Oriented Role Modeling and Group Support Systems

Sangeetha Narayanan
sxn0165@homer.njit.edu

Ajaz R. Rana
New Jersey Institute of Technology, rana@cis.njit.edu

Object Oriented Role Modeling and Group Support Systems

Sangeetha Narayanan
(Email: sxn0165@homer.njit.edu;
phone: (201) 642-4795)
Ajaz R. Rana
(Email: rana@cis.njit.edu;
phone: (201) 596-2661)
New Jersey Center for Multimedia Research
Department of Computer & Information Science
New Jersey Institute of Technology,
University Heights, Newark, NJ 07102

Abstract

The design and development of Group Support Systems (GSS) is a complex endeavor. The long term usefulness of these systems can only be as good as their design. Sound design principles demand the adoption of a systematic methodology that matches the system design objectives. The focus of this paper is the use of Object-Orientation as a modeling technology for designing GSSs. We present a brief overview of popular Object-Oriented (OO) analysis and design methodologies, and argue in favor of a methodology that allows the description of phenomenon of interest in the form of role models [10] - an abstraction well suited to capturing the essence of collaborative work.

Introduction

Group Support Systems (GSS) can be viewed as a collection of electronic and non-electronic technological artifacts brought to bear on the individual or joint activities of co-located or dispersed (across time or space) members of a group. Real life groups are "complex intact social systems that engage in multiple, interdependent functions, on multiple, concurrent projects, while partially nested within, and loosely coupled to, surrounding systems" [6, pp. 151]. GSSs are social technologies. The characteristics of individual group members (personality, cognitive styles, etc.); the group itself (norms, structure, etc.); and the larger social unit in which the group is embedded (organizational culture), all play an important role in design choices. It is agreed that the availability of technology that matches the needs of the task at hand, if not sufficient, is however necessary for the expected benefits of technology to be realized [2, 7, 3]. Designing a system that provides tools which match the needs of group tasks is complicated by the fact that the definitions of group tasks are tied to the nature of the group and hence vary on several dimensions. For instance, tasks vary in their level of complexity, validation, and coordination mechanism [8, 9]. Successful completion of a task may involve performance of diverse activities by group members that are critical for the fulfillment of certain functions. Groups are multifunctional - their activities make contributions to the goals of embedding system, to the well being of its members, and the well being of the group itself [6]. At any given time, individual members of a group maintain membership in various groups, and the whole or part of a group may itself be a member of other groups in a larger social unit (e.g., organization). Group members accomplish much of their work when group is not in session, hence necessitating a diverse set of tools to facilitate individual activities, group process activities, and coordination related activities to be undertaken in either synchronous or asynchronous manner.

The benefits of object orientation for designing and implementing computer based systems are well documented. The use of object orientation to design a system involves the identification of objects of interest and relationships among those objects. However, frequently, even simple systems involve more objects than can be grasped without some abstraction that highlights needed aspects of the system. A common approach is to adopt the "class" abstraction to represent a set of objects. We agree with Reenskaug and colleagues [10] that "class" is a powerful implementation mechanism, but its usefulness for modeling complex systems, such as GSSs, has its limitations. This is due to the fact that classes provide an isolated view of objects. We believe that the modeling of a system that is intended to support multifunction and interdependent activities of partially nested group members (where members play different roles depending upon their group membership and the context of collaborative activity) must take a process based view of the collaborative phenomenon. As a result, we argue that the concept of a "class: as a modeling abstraction, which is static by definition, is not the most suitable abstraction for designing (or modeling) a GSS. What is needed, is a methodology that provides modeling constructs and techniques for capturing the flow of activities in a collaborative work or decision making process.

In order to support our argument, we present a brief overview of popular object modeling methodologies (Section 2), and highlight their limitations. We elaborate on a methodology (Section 3) that we believe would serve well for modeling an object oriented GSS. We conclude this paper suggesting directions for future research.

Popular Design Methodologies

The popular methodologies available for analyzing collaborative activities and design a system to support such activities include: Object Oriented Design and Analysis by Booch [1], Object Modeling Technique (OMT) by Rambaugh [11], Use-case driven analysis method by Jacobson [5], and the OOram software engineering method by Reenskaug [10]. Each of these methodologies are described in books written by respective authors [1, 11, 5, 10].

- The Booch method has its basis in object-oriented design from Ada and is the most comprehensive method for modeling language oriented design features such as parameterized classes and inheritance. The method consists of logical and physical model. The basic concepts of Booch method are the object, class, and inheritance. The logical model consists of class diagrams (used to depict classes and their relationships), object diagrams (used to show instances of classes and their relationships), interaction diagrams (used to trace the execution of a scenario), and state diagrams (used to develop a dynamic model of a single class or the whole system in terms of events that cause a transition in the state of a class or the system). The physical model consists of modules (to map classes and objects to modules in the physical design of the system), and process diagrams (used to show physical devices. processors, and their connections).
- The Object Modeling Technique (OMT) was initially employed to develop real time systems and systems software (compiler, graphical user interfaces and database products). The OMT method consists of object model, dynamic model and functional model. The object model is based on entity-relationship modeling with extensions to modeling classes, inheritance and behavior. The dynamic model is based on state charts and events while the functional model is based on data-flow diagrams.
- The use-case model concentrates on actors and use cases. The actors represent things that exchange information and which interact with the system. The role of an actor is non-deterministic and is very different from that of an object. Each instance of an actor is unique and performs various functionalities of the system which is referred as a use-case.[3]
- The main modeling concepts of the OOram methodology are Roles and Role Models. It promotes the description of a phenomenon (or sub-phenomenon) of interest as a set of cooperating objects. Objects involved in a phenomenon of focus are organized in patterns of objects, called the role models, and each object in a pattern is called a role (that the object plays). Through the notion of separation of concern during the modeling process it allows building complex role models by synthesizing simpler models. The OOram supports a number of different views (or diagrams) for the role model (total 10). At the implementation stage it maps a role into a class. A class constructed from a role captures its relationship with other classes in the role model.

Role Modeling and GSS

No single methodology can fit all purposes. The choice of methodology should be based upon the nature of the phenomenon of interest. We believe, as noted earlier (Section 1), that a methodology that provides modeling constructs and techniques for capturing the flow of activities in a collaborative work or decision making process should be the methodology of choice.

In object oriented terminology, a process based view of collaborative work implies a structure of objects that interact to achieve some purpose. Conceptualization and development of a process based view involves identification of: the activities needed to complete a task; the actors involved in the performance of those activities and the roles that the actors play; the functions that the activities serve in the completion of a task process; and the resources or the data that the actors use to complete the process.

Except for OOram, the basic abstraction construct in all popular modeling methodologies discussed above is the "class" abstraction [10]. We concur that the "class" is a powerful implementation construct. However, it is not well suited at the modeling stage when the interest is in the modeling of collaborative process. The representation of collaborative activities (or process) as role models models and implementation of "roles" as a "class", as suggested by the OOram methodology, offers the advantage of building GSSs that can support a diverse and evolving set of collaborative processes by instantiating appropriate combinations of role objects.

In most popular object-oriented methodologies, objects are portrayed as individual entities (or classes), and during the implementation stages they are mapped to classes in object oriented programming languages. In most object oriented methodologies do support interaction views (e.g., interaction and object diagrams of Booch Methodology and Rambaugh's event trace diagrams), but they are used to model the execution of scenarios (or message passing among objects), and hence providing a snap shot of a transitory stream of events among a particular configuration of objects. This is not to say that such views are not useful. An application intended to support the collaborative work processes of groups that engage in multiple interdependent functions on multiple projects while maintaining their memberships and performing their expected roles in other social systems or groups can be modeled very well by adopting the role model abstraction. We believe this is to be the case due to the following reasons.

- The notion of roles is central to collaborative technologies [12]. An individual in a GSS supported group communication process may assume various roles: owner of a conference or discussion group, monitor, editor, appender, indexer, administrator, masquerader, contributor, observer, etc. [12]. The development of role models and later mapping of various roles into classes (for implementation purposes) provides a natural way of encapsulating collaborative processes as a set of cooperating objects.
- The process based nature of group work requires modeling of interactions between objects. Under such circumstances, we agree with Reenskaug that the idea of role can be very useful to depict object interaction. In OOram, role models act as a foundation for implementation. Objects that have similar external properties can be considered to be a part of a role. These roles in object-oriented programming act like classes except that we could have a single class, which at different instances could be performing more than one role. Classes comprising of roles share the feature of both single and multiple inheritance depending on the implementation language. Since roles are not programming language dependent in OOram, most popular programming languages can be used to implement them.
- Another strength of OOram is that it merges many concepts provided by earlier methodologies, and at the same time creates a frame of reference for the family of object-oriented methodologies. It promotes separation of concern through focusing on a subset of the phenomenon of interest by starting with models of simple objects, and then creating a complex structure from them. This provides opportunity for developing applications systematically through reusable components.

For any application, a given role model may be observed from different perspectives. It is left for the analyst to create a mental model to capture the underlying phenomenon, and to be able to express views on object interaction. Reenskaug and colleagues have proposed different views that would help the analyst to manipulate and to perceive object interaction in their methodology.

In our view, the Object-Oriented Role Analysis and Modeling (OOram) [10] is best suited for applications involving flow of information and work processes in human organizations.

Conclusions

Due to the popularity of object orientation and World wide Web along with the availability of standards (e.g., CORBA and OLE), it is expected that, the new generation of universally accessible collaborative environments will be built by developing and extending robust frameworks. An example of such a framework is the Habanero Framework implemented by researchers at National Center of Supercomputing Applications.

However, the fact remains that, a methodology by itself, no matter how well suited to an application it may be, cannot ensure a perfect product. A sound understanding of the phenomenon to be modeled is a pre-requisite. Unfortunately, the nature of the phenomenon studied by GSS researchers is not so well understood. We believe, that a clear understanding of the nature of group tasks and the context is essential for applying the role modeling methodology. At present, we are at the early stages of exploring the suitability of various group tasks classification schemes for applying the OOram methodology [6, 9] to design frameworks for building a multi-mode (synchronous and asynchronous) collaborative environment over the World Wide Web.

Acknowledgments

This research was partially supported by grants from New Jersey Commission on Science and Technology, the National Science Foundation (NSFIRI9015236 and NSFIRI9408805), and the New Jersey Institute of Technology (SBR42140). The opinions expressed do not necessarily represent those of the funding agencies. We would like to thank Robert Czech, James Whitescarver and Ananth Ganesh for their valuable insights.

References

Available upon request from the first author.

[1] Booch, Grady (1994) Object-Oriented Analysis and Design with Applications, 2nd edition, Addison-Wesley Publishing Company.

[2] DeSanctis, G. L. & Gallupe, R. B. (1987) A Foundation for the Study of Group Decision Support Systems, Management Science 33:589-609.

[3] Dennis, A. R., George, J. F., Jessup, L. M., Nunamaker, J. F. & Vogel, D.R. (1988) Information technology to support electronic meetings. MIS Quarterly 12:591--624.

[4] Hutt, A.T.F (1994) Object Analysis and Design - Description of methods, John Wiley and Sons, New York.

[5] Jacobson, I.(1992) Object-Oriented Software Engineering- A Use-Case Driven Approach Revised Fourth Printing, Addison-Wesley Publishing Company,California.

[6] McGrath, J. E. (1991) Time, Interaction and Performance (TIP): A theory of groups. Small Group Research 22:2:147-174.

[7] Poole, M. S. & DeSanctis, G. (1990) Understanding the Use of Group Decision Support Systems: The Theory of Adaptive Structuration. In: Organizations and Communication Technology, Ed. J. Fulk & C. Steinfeld, pp. 173-193. Sage Publications, Beverly Hills.

[8] Rana, A. R. (1995) Peer Review Process and Group Support Systems: Theory Development and Experimental Validation. Doctoral Dissertation, Rutgers University.

[9] Rana, A. R., Turoff, M. & Hiltz, S. R. (1997) Task and Technology Interaction: A Theory of Technological Support for Group Tasks. HICSS-30

[10] Reenskaug, T.(1996) Working with Objects - The OOram Software Engineering Method Manning Publications, Greenwich, CT.

[11] Rumbaugh, J. (1991) Object-Oriented Modeling and Design, Prentice Hall, NJ

[12] Turoff, M. (1991) Computer-Mediated Communication Requirements for Group Support Journal of Organization and Computing, Vol. 1, pp. 85-113.