Rule-Mitigated Collaboration Technology

Carl K. Chang, Alexei Vorontsov, Jia Zhang, Francis Quek
University of Illinois at Chicago
{ckchang, avoronts, jzhang}@eecs.uic.edu

Abstract

The design of an appropriate paradigm for collaboration ultimately stands or falls on the question of whether human users are able to cooperate effectively with it. In this work, we begin with a paradigm of interaction in which human collaborators have shown themselves facile. This paradigm is based on the formal meeting protocol commonly known as parliamentary procedure or Robert's Rules of Order (RRO). These rules are at the same time descriptive and prescriptive of effective meeting behavior. Electronic meeting systems, likewise, have to manage time and communicative resources, maintain logs, and produce artifacts that constitute the fruit of the collaboration. The technology disclosed here facilitates the generation of co-authored artifacts (documents, designs, project plans, etc.) as the direct outcome of the collaborative process. The efficacy of rule-mitigated collaboration technology is based on four major components: an extended parliamentary procedure rule set, a scoping policy and set of application programming interfaces, an object-based client-server architecture, and an M-Net synchronous meeting environment.

Keywords: CSCW, RRO, Scope, Discussion threads, Synchronous/Asynchronous Collaboration

1. Introduction

In the broadest sense, Computer Supported Cooperative Work (CSCW) is the application of computing and networking technologies to facilitate cooperation and collaboration among people. CSCW embraces such computer science and engineering disciplines as human-computer interaction, networks, multimedia, communications, database management, distributed system, object oriented concepts, virtual reality, software engineering and artificial intelligence [4].

The technology disclosed here facilitates the generation of co-authored artifacts (documents, designs, project plans, etc.) as the direct outcome of the collaborative process. Three key process-related challenges to CSCW have to be considered: communication, coordination, and collaboration [5]. Our Rule-Mitigated Collaboration addresses these challenges by implementing an extension of *parliamentary procedure* within a consistent cogent framework. We embody this framework in the facilitating technologies that realizes the collaboration process. We accomplish this by our object-oriented client-server architecture that provides the services and interaction components. By using a super-subset of the common "Robert's Rules of Order" [13], we leverage the familiar and proven to produce a collaboration environment that exploits the power of modern computing and networking.

Beside the "collaborative process" challenges, we address the challenge of data and version integrity. Since we envision collaborative groups co-authoring textual, graphical and numerical documents with multiple simultaneous modifications to each document, our technology has to be application independent and has to maintain multiple disparate versions of each document. We accomplish this by exploiting the inherently recursive nature of the amendment-decision cycle, and defining a set of *application program interfaces* (APIs) that implement a "scoping strategy". We shall describe each of these technologies in turn.

Since CSCW was first introduced in 1984, there are already hundreds of collaborative environments created to support different collaboration purposes. Of these, co-authoring systems are an important subset. Co-authoring systems may be divided

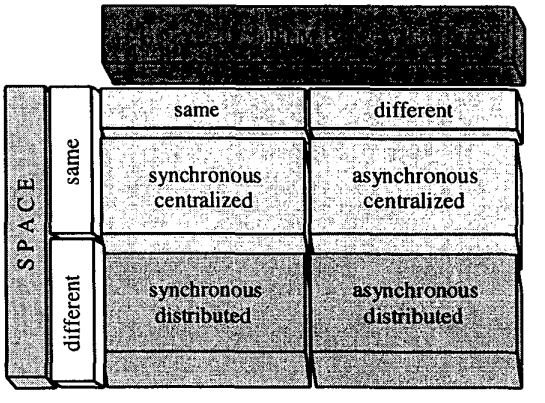

Figure 1 Time –Space Dimensions

0-7695-0468-X/99 $10.00 © 1999 IEEE

into two categories: synchronous co-editing systems that allow users to work on a same document at the same time such as DistEdit [8], SASE [1], and Shared Books [10], asynchronous co-editing environments such as Quilt [6] and PREP [11]. IRIS [9] and Lotus Notes which support both asynchronous and synchronous collaborations [12].

Each environment introduced above has its own advantages and disadvantages [3]. Lotus Notes is not really designed to be a co-authoring tool, but its features allow it to be used as a write-review-comment environment. DistEdit allows users to use their favorite editors, but it requires those editors to be modified. PREP and Quilt are based on the writing, reviewing, and commenting process. However, they do not allow equal access to the document for all users because they define the roles of authors as writers and reviewers. IRIS is a powerful multi-user editing environment that allows the integration of specifically designed applications for IRIS and supports both synchronous and asynchronous editing. However, IRIS is not easily extendible to integrate other applications without changing IRIS itself.

CSCW systems may be classified along the dimensions of time and space as shown in Figure 1 [7] . The technology here disclosed permits general *asynchronous distributed* (Different Time, Different Place) operation. The technology also facilitates *synchronous distributed* (Different Place, Same Time) operation by implementing *M-Net* places where collaborators may 'meet' for time-sensitive functions such as real-time discussions and time-critical decision making. One may think of a distributed time space collaborative discussion as an extended distributed graph in space-time space. *M-Net* places are special synchronous nodes in such an extended graph.

2. Extended RRO for Rule-Mitigated Collaboration

In the following section we give a brief overview of Robert's Rules of Order as well as the set of modifications to adapt these rules for the domain of electronic meeting support.

2.1 Robert's Rules of Order: Traditional

Robert's Rules of Order (RRO) were meant to be used to help control meetings of various types of assemblies. These rules, known also as Parliamentary Procedures, have been tested by time and proved to be an efficient and reliable mechanism for meeting control. Capturing essential features of human interaction, these rules give a solid framework for building logically complete model for computer-supported collaboration system.

An assembly, or collaboration group, will gather regularly at a *meeting* to transact their business and make decisions. The meeting is an essential entity in this model and is used as an instance of formal collaborative activity.

RRO introduce the notion of a motion. Most of the meeting related activity takes the form of motions. Rules categorize motions into four classes: main motions, subsidiary motions, accidental motions, privileged motions. Each motion has been attached with priority. Another important element of this meeting model is the concept of the floor. The floor is a communication channel that is shared by the members of the assembly and is used for controlled interactions. Meanwhile, each meeting must have a well-defined purpose or plan that is formalized as an agenda. This plan must be established before meeting may be called and is usually scheduled as the last business of the previous meeting of the assembly, approved by members and adopted for further consideration.

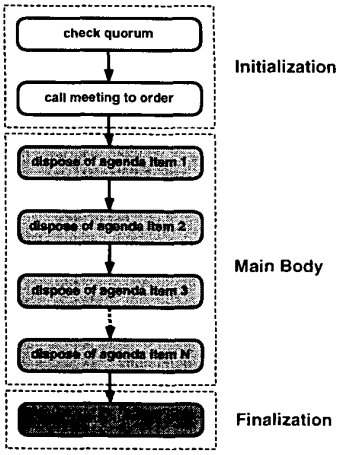

Figure 2 Meeting

Having considered essential pieces of traditional RRO and having introduced the necessary terminology, we may now give a general picture of how a regular meeting may proceed shown in Figure 2. Figure 3 illustrates the disposal of agenda items. This process of consideration of an agenda item, or the agenda item life cycle, has a regular structure which may be described by the following production system:

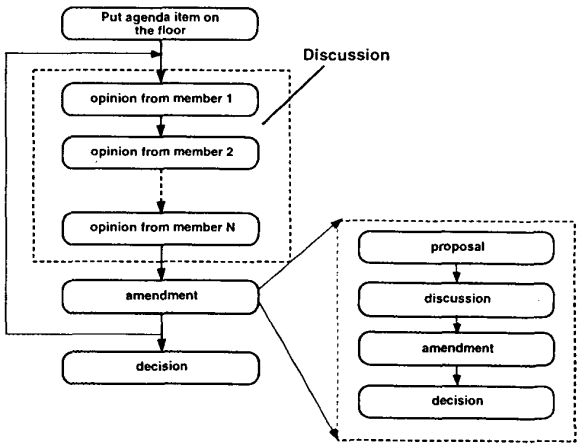

Figure 3 Proposal – Discussion – Decision Cycle

Agenda item consideration := Proposal, {Discussion}, Decision;
Discussion := Opinion | Amendment, {Discussion}, Decision;

Using the same approach we may describe the meeting as follows:

Meeting := Preliminary part, {Agenda item consideration}+, Adjournment.

2.2 Robert's Rules of Order: Extensions

Since RRO was designed to regulate formal collaboration on a single communication channel, and because it has to account for the limitations of human short-term memory and attention, it incorporated certain limitations on the flow of the meeting. Some of these impediments do not exist, or are ameliorated in network-based collaboration. Furthermore, some of the limitations may be relaxed by employing a well-designed user interface. However, one essential purpose of rules is to regulate or limit behavior. It is essential, therefore, that we understand which RRO rules address the accident of physical meeting room impediments, which address the limitations of human cognition, and which are regulatory for the semantic flow of meetings. We need, also, to know what new rules and concepts are necessary to regulate the unique nature of electronic collaborations.

One obvious limitation that may be eliminated from the traditional RRO is the requirement for the uniqueness of the floor. This is a limitation of the first category, and needs to be amended for electronic collaboration to take advantage of the capacity of electronic networks to handle multiple simultaneous communication channels. Another limitation of traditional RRO is the restriction of amendment nesting to a single level. This is a limitation of the second category because it is difficult for members of an assembly to track multiple levels of detail and complexity. Such confusion may be lessened by good user interface design. By generating running minutes of the meeting and making the tree-like structure of the meeting explicit, the system can help users track the course of a meeting. There are, however, still limitations for such nesting. Infinite nesting will ultimately derail the course of a discussion, and the limitation of screen real-estate to represent a too-complex tree is real.

Since traditional meetings requiring the physical presence of assembly through the course of a meeting, RRO provides for some form of physical interruptions so that member of the assembly could take a break. There is no such need in the asynchronous electronic meeting and, thus, motion to "Recess" may be abolished. We also considered avoiding meeting adjournments. Considering the persistence property of collaborative environment we believe that electronic meeting may be adjourned only once - when the collaboration group reached the point of logical conclusion of their intended work.

In traditional RRO, an agenda for the meeting must be fixed before the meeting may begin and after the commencement of the meeting agenda cannot be changed. Each meeting must follow its agenda and no questions may be considered during the meeting which are not scheduled on the agenda. This rule of the immutability of the agenda proves too restrictive for a persistent meeting environment that may continue for weeks and months. Over time, the relevance and efficacy of any agenda will eventually become outdated. We provide a mechanism by which the agenda may be modified dynamically during the collaboration process.

We also add motions to modify the makeup of the membership of the assembly (such as the addition and removal of members) as the people with legitimate interests in the collaboration may change over time. These membership-related motions are also handled in the management.

RRO were not designed for inherently persistent environments. Typical assembly meetings would last for several hours and results of the cooperative efforts would normally get externalized in the form of some meetings minutes. The introduction of a persistent meeting environment precipitates other challenges that require modification to traditional RRO.

3. Discussion Thread: Concept

We introduce the idea of discussion threads to deal with the 'multiple-floors' situation. One may think of a traditional meeting with a single physical floor as a thread of proposals, amendments, discussions and decisions. This thread may be nested one level deep when amendments are considered. We extend this concept to that of a general discussion thread to represent a single semantically cogent path through the 'meeting space'. A key requirement of such threads is that it must be relatively independent of each other. A discussion thread is very much like an execution thread in a multi-threaded computer

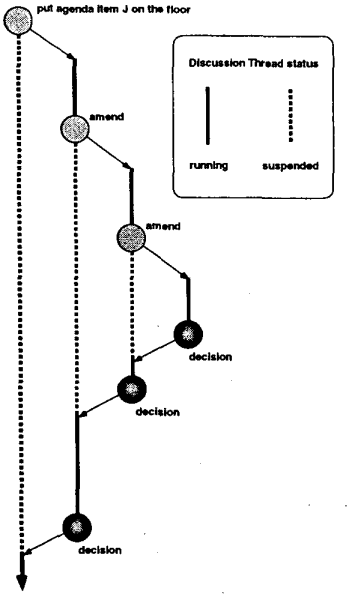

Figure 4 Discussion Thread

program. And it is a powerful tool, in general, for handling program concurrency. We may further organize threads into parent-child hierarchies. This parent-child thread relationship will be particularly helpful for dealing with our other extension to traditional RRO *vis a vis* multiple amendment nesting. We may consider the whole meeting as a collection of concurrently running discussion threads, each with its own objective, and all threads are coordinated by some main thread and with their parent threads using synchronization mechanisms yet to be established.

Given the concurrency of discussion threads, synchronization mechanisms have to be embedded directly into the extended RRO definition. Since our model uses discussion threads to embody such meeting-related action, we use motions as triggers to launch discussion threads. This is the basic synchronization mechanism that also ties the discussion thread directly to the meeting semantics prescribed by RRO. Formally, we define a discussion thread as an independent execution context for meeting-related action, and it is used to organize cooperative activity into logically related clusters with well-defined boundaries. Hence, there is a one-to-one mapping of motions to discussion threads, and the lifetimes of corresponding motions and discussion threads are simultaneous.

Given the recursive nested nature of an RRO-based discussion hierarchy, we may represent them as parent-child threads as shown in Figure 4. Every meeting has a main discussion thread which represents the whole meeting and serves as a parent to all other discussion threads. This main thread does not have any other goal but to serve as a top-level synchronization-capable container for its children. This main discussion thread has the same life cycle as the meeting. Each subsequent discussion thread has a parent thread and it must coordinate its execution with its parent thread. A parent discussion thread may not terminate while it has non-terminated child discussion threads. This requirement is imposed by the fact that child thread is nested inside its parent thread just like amendments are nested within proposals. This is the place we may find parent-child relationship handy while dealing with multiple levels of amendmnt nesting. When we allow amendments to be amended to an arbitrarily deep level, discussion threads nested ewithin each other will give the required structure to represent this amendment nesting.

Having several sibling discussion threads running concurrently in the meeting requires synchronization. This synchronization would be required when several discussion threads are initiated concurrently, each dealing with its own agenda item, and there is an interdependency among them. We may use these RRO motions as special of synchronization motions. Since the latter kind of interdependency is semantic to the content of the meeting, and not structural or syntactic to the mechanism of meeting flow, it is reasonable to leave such synchronization within the control of the human collaborators using these special motions.

Motions and decisions serve an important synchronization function in the discussion life cycle. Apart from the synchronization function, decision threads delineated by motions and decisions serve the important purpose of *control of flow*. It ensures that the flow of the collaborative work is goal directed, and makes explicit the structure of this flow. As discussed earlier, this implicit synchronization and control of flow is augmented by the pair of RRO motions that postpones and resumes discussion threads to resolve interdependency deadlocks.

Interdependent discussion threads, combing all, consist a persistent meeting environment which may span multiple days, weeks, or even months. During this persistent meeting, the meeting environment is maintained, and users may participate in different active discussion threads and review the flow of the meeting by browsing the minutes of the threads.

4. Delta Documents (δ-Documents)

Our rule-mitigated collaboration technology is designed to function as a distributed collaborative production environment for multiple applications and media. The philosophy of our paradigm is that the outcome of the collaboration is the product, the cooperative document is one of our central concern. We want a persistent, secure, concurrency-control-enabled document that would be generic enough to serve as a template for representation of a growing variety of different types of documents. To achieve this goal we introduce the concepts of scope and δ-document.

A δ-document is a document (parent δ-document) which may

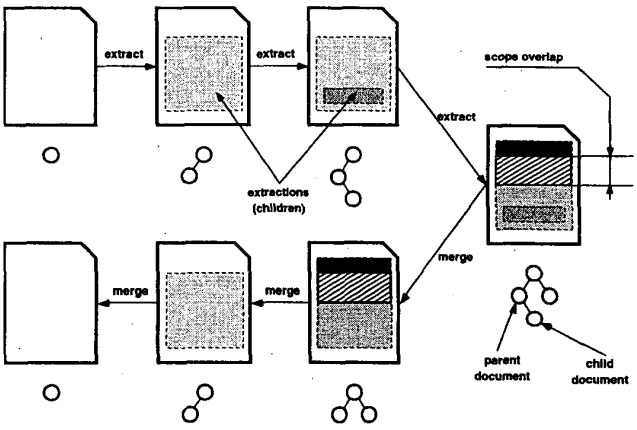

Figure 5 Delta Document

have other δ-documents as parts of it (children) and the extent

(content) of these δ-documents are defined by their respective scopes. Figure 5 can be an example to show how δ-document works.

Scope may be defined in the context of some δ-document. Each δ-document has a parent δ-document – the δ-document from which this one is extracted (or NULL parent if this δ-document is at the top-level in the parent-child hierarchy) and a set of children – δ-documents which are extracted from this document. Each parent-child association has scope attribute which describes the nesting constrains.

We have defined scope and δ-documents as an extension to the operations applicable to typical documents that will allow sharing these documents in a disciplined manner. Possible implementation of this extension may take the form of plug-ins which provide the necessary functionality and may be used to add new types of documents to the distributed collaborative environment defined by our technology.

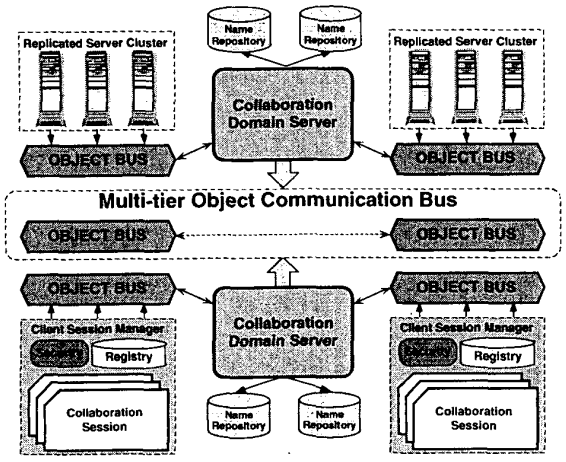

Figure 6: General Client-Server Architecture

The approach of scope/δ-document is well suited for integration into our rule-based environment. The actual points of interaction – extraction, merge – map directly into the activity of amendment preparation and decision-making (voting).

5. Object-based Client/Server Architecture

In our implementation, we selected the CORBA (Common Object Request Broker Architecture) standard to serve as our middleware infrastructure. This design decision was driven by two factors. First, CORBA is one of the dominant standards in the marketplace and is largely system/vendor independent. Second, CORBA development tools were available to the UNIX platforms on which we developed our prototype. We could have selected DCOM which is a proprietary Microsoft 'standard', but that would have required us to develop solely on Windows platforms. Figure 6 is a diagram of the general system architecture for our rule-mitigated collaboration system. This client-server architecture is made up of four major components:

Collaboration Server that maintains all the system databases for a particular collaboration group. It handles all client requests for data and manages all collaborative events (e.g. submission of motions, discussions, etc.), all notification services (e.g. signaling a client that a new proposal is on the table), and the general flow of the meeting (i.e. maintaining the discussion database in accordance to the extended RRO). In our prototype, each collaboration group will have a separate collaboration server.

Collaboration Client that provides user access to the collaboration environment. Each collaborator will have a *Client Session Manager* that serves as the front-end application. It allows the collaborator to visualize the state of the multi-threaded discussion, tender her discussion comments, make motions, access the server databases, and communicate with other members in the assembly. This session manager permits a user to participate in multiple collaboration groups simultaneously.

Collaboration Domain Server that serves as an active directory of collaboration groups in which a user may participate. This is a sort of name server (there are several replicated instances of it in the system) that functions as a repository for the names of available collaboration servers/groups.

Middleware specific components that support interoperation among the other architectural components. In CORBA parlance, this is the **Object Bus**. This is an 'active data bus' that receives object-based messages, locates the appropriate method (object-specific function), and activates the method to respond to the message. It also provides event, time and security invocation services.

6. The M-Net Synchronous Meeting Place

Synchrony is often useful for instantaneous decision making in time-critical situations. By requiring all participants to be simultaneously on-line, a synchronous system can maximize real-time response and reduce the lag-time inherent to asynchronous meetings. The trade-off is that all members HAVE to be 'present' virtually, making meetings difficult to schedule. Also, since the time dimension is 'locked', all members have to attend to a shared communication stream. Hence, the power of parallel activity by meeting participants in an asynchronous system is sacrificed. Therefore both synchronous and asynchronous meetings are required to create an effective collaboration environment.

As for its original purpose, M-Net is a multimedia-based software product for usual Web meetings [2]. M-Net manages a

thread of discussion and does not address concurrency. Hence, an M-Net meeting may generate nested discussion threads reflecting the RRO amendment hierarchy. M-Net uses a subset of the extended-RRO rule base. Since M-Net is synchronous, and the discussions are typically based on a *What-you-see-is-what-I-see* (WYSIWIS) paradigm, floor control is important to regulate what is presented to the collaborators. The δ-document database must be kept synchronized Net subsystem shares the same members and groups database as the *Collaboration Server*. Hence, it simply through the course of the M-Net session. In an M-Net meeting, the *M-Net Server* must communicate with the *Collaboration Server* to check the validity of the scopes of δ-document being proposed. The M- queries the *Collaboration Server* for membership services.

M-Net is now acting as a component to collaboration net. When members of an asynchronous meeting decide that a synchronous meeting place is needed, they may schedule an M-Net meeting using the usual proposal and discussion process. The Collaboration Server will maintain the new M-Net thread in its own environment but leave the M-Net session alone until the M-Net users comes with a certain resolution on the meeting material. Once checked out, there is a copy created for the checked-out item and it is under the control of M-Net. The M-Net synchronous tools ensure the integrity of an ultimate version of the checked-out material ready for check-in. For any temporary meeting information created on the fly out of an M-Net session, it will be discarded upon exit of M-Net and promoted to a persistent data object for archive. In the latter case, the Collaboration Server interface must be invoked again.

Generally, most of the RRO rules we adopted in the Collaboration Net are also suitable to M-Net. However, since synchronous mode is a special case of the asynchronous mode, and real-time properties must be observed in M-Net, we will have to restrict and extend the RROs in the following two manners. First, there is a restriction to the asynchronous mode of RROs as discussed above. Second, we need to include some of the meeting coordination rules in common RRO. These include the usual privileged motions for meeting recess, adjournment, and reconvening.

7. Claims of Innovation and Conclusion

Our Rule-mitigated Collaboration Technology, augmented with the synchronous electronic meeting environment M-Net, provides the following innovations: an extended parliamentary procedure rule set adapted for network and parallel operation; a scoping policy and set of application programming interfaces; an object-based client-server architecture; object databases of discussion and shared documents that are compatible with the concept of scoping and δ-document update; an architecture for user communications; replication strategy; an M-Net synchronous meeting environment that permits users to gather in real-time virtual meetings for time and synchronization critical discussion and decision making.

8. References

[1] R. M. Baecker, D. Nastos, et al., "The User-Centered Iterative Design of Collaborative Writing Software", in Proceedings of ACM Conference on Human Factors in Computing Systems, pp. 399-405, 1993.

[2] C. K. Chang, L. Cai and F. Quek, "On Formal Meetings for Net-centric Web-based Computing", in Proceedings of the Internet ISIT'98, pp.199-204.

[3] B. Cronin, "Progress in Documentation", Journal of Documentation, 38(3):212-236, September, 1982.

[4] W. K. Edwards, "Policies and Roles in Collaborative Applications", ACM CSCW'96.

[5] C. A. Ellis, S. J. Gibbs and G. L. Rein, "Groupware: Some Issues and Experiences", Communications of the ACM, 34(1):38-58, January 1991.

[6] R. S. Fish, R. E. Kraut, et al., "Quilt: a Collaborative Tool for Cooperative Writing", In Proceedings of the Conference on Office Information Systems, pp. 30-37, 1988.

[7] J. Grudin, "Computer-Supported Cooperative Work: Its History and Participation", IEEE Transactions on Computers, pp. 19-26, vol.27, no.5, 1994.

[8] M. Knister and A. Prakash, "Issues in the Design of a Toolkit for Supporting Multiple Group Editors", Computing Systems – The Journal of the Usenix Association, 6(2): 135-166, 1993.

[9] M. Koch, "Design Issues and Model for a Distributed Multi-user Editor", ACM CSCW'95, 3:359-378, 1995.

[10] B. T. Lewisand and J. D. Hodges, "Shared Books: Collaborative Publication Management for an Office Information System", In Proceedings of the Conference on Office Information Systems, pp. 197-204, 1988.

[11] C. M. Neuwirth, D. S. Kaufer, et al., "Issues in the Design of Computer Support for Co-Authoring and Commenting", in Proceedings of the Conference on Computer Supported Cooperative Work, pp. 183-195, 1990.

[12] J. P. Popows, "Notes for Lotus and the World: Sighting the Goal", In Groupware Technology and Applications, eds. D. Coleman and R. Khanna, pp. 201-223. Upper Saddle River, New Jersey, Prentice Hall, 1995.

[13] H. M. Robert III and W. J. Evans, "Robert's Rules of Order", Newly Revised, 9th Edition, SCOTT, FORESMAN AND COMPANY, 1990.