Modular software to support parliamentary procedure

Dana Dahlstrom
Computer Science and Engineering
University of California, San Diego
9500 Gilman Dr
La Jolla, CA 92093-0114
dana at cs.ucsd.edu

Bayle Shanks
Computational Neurobiology
University of California, San Diego
9500 Gilman Dr
La Jolla, CA 92093-0346
bshanks at ucsd.edu

ABSTRACT

Parliamentary procedure is a widely used system of rules for group decision making. Software can help people engage formal group deliberation in familiar face-to-face settings, and can also enable computer-mediated online deliberation. This paper presents a reusable software module, *Parliament*, that implements the logic and bookkeeping of parliamentary procedure using a separately modifiable specification of the rules. A prototype application for supporting face-to-face meetings is described. Together with a working partial specification of Robert's Rules of Order, the application provides a demonstration and testbed for the module. Future directions are proposed for both the module and the prototype application, and some related work is mentioned.

1. INTRODUCTION

Parliamentary procedure is a system of rules and customs used in formal group decision making. It aims for thorough, fair deliberation, allowing the majority to take action while safeguarding the rights of the minority and of individual members, even in the face of intense disagreement. As a tradition, parliamentary law has evolved in bodies such as the UK Parliament and the US Congress, and there is considerable variation among its forms. The most widely used codification is called *Robert's Rules of Order*, after Henry M. Robert, who wrote the first popular manual on parliamentary law in the late-19th-century United States.

Robert's Rules of Order govern numerous public and private organizations from local school boards to the boards of trans-national corporations. Order is vital to democratic participation, and effective participation in democratic processes depends on the ability to engage them according to rules. Software can assist participants in face-to-face meetings using parliamentary procedure, and could facilitate similarly conducted deliberation online. These applications have in common the need to track and indicate the state of formal group decision-making processes.

2005 ISR Graduate Student Research Forum Irvine, California USA

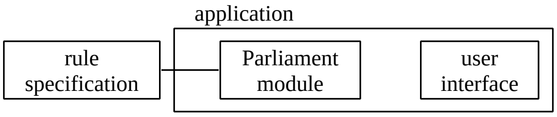


Figure 1: The Parliament module is embedded in an application, and uses an external rule specification.

Parliament is an open-source software module[1] written in Python[2] for use in programs that conform to or moderate parliamentary procedure. Parliament encapsulates logic and bookkeeping functions, and is intended for applications supporting face-to-face as well as synchronous or asynchronous computer-mediated deliberation.

The Parliament module does not incorporate the details of parliamentary procedure, such as the motions and customs described in Robert's Rules of Order. Instead, Parliament requires an external rule specification, allowing the user or developer to modify the rules independently and even to develop whole new rule systems.

A prototype application for supporting face-to-face meetings is presented. For use with it, a partial specification of Robert's Rules of Order has been developed. Together, these components demonstrate Parliament's capabilities and facilitate testing the module.

This short paper summarizes work in two longer papers presented at the Second Conference on Online Deliberation.[3] One paper focuses on the Parliament module, its API and rule-specification language [7]; the other focuses on the design of software to support parliamentary procedure in face-to-face meetings [2].

2. A REUSABLE MODULE

Many conceivable applications could share a common implementation of parliamentary procedure. Programs to support face-to-face meetings can assist the entire group, individual participants, or specific roles such as chair, secretary, or parliamentarian. Away from meetings, applications can train people in the rules.[4] Meetings conducted synchronously online could use parliamentary procedure, and some tasks of the chair could easily be done by software, such as counting votes and recognizing members to speak.

[1] `http://parliament.sourceforge.net/`
[2] `http://python.org/`
[3] `http://www.online-deliberation.net/`
[4] Non-modular training software for parliamentary procedure is available at `http://imovethat.com/`

Software will play an even more significant role in asynchronous collaborative decision making. Applying traditional parliamentary procedure to asynchronous communication is not straightforward; modified rules will be needed. Groups already make decisions online through collaboration by e-mail, newsgroups, message boards, and wikis; formalized decision-making processes could be of value in online communities.

With so many applications, it would be inefficient to re-implement the logic and bookkeeping of parliamentary procedure for each. A reusable module will simplify implementation efforts and promote uniform, high-quality handling of the rules.

The Parliament module keeps track of meeting state, such as pending questions, the relationships among them, and business already transacted. The Parliament API provides functions to manipulate and query meeting state; the program using Parliament informs the module about events as they occur, such as "member X made motion Y" or "motion Y was rejected." Parliament answers queries such as "Which motions are currently in order?" or "Which motions have been adopted?" Parliament is also capable of answering hypothetical questions such as "Which motion will be pending if this one is adopted?"

3. MODULAR RULE SPECIFICATIONS

Rather than choosing a specific set of parliamentary rules and "hard-coding" them into the module, we opted for a separate rule specification in order to accommodate many different rule sets. Not only can an application built using Parliament use interchangeable rules, but multiple applications built using Parliament can share rule specifications in a common format.

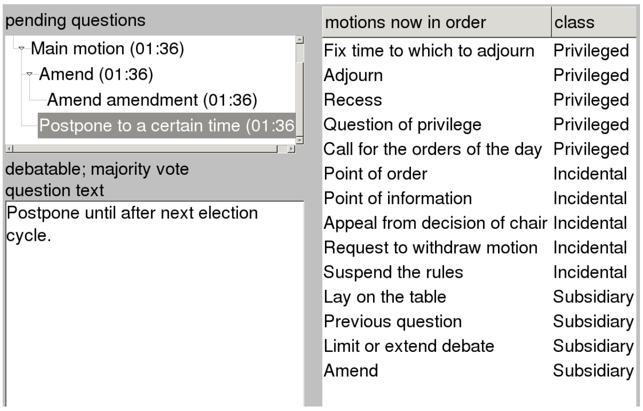
The user or developer of an application specifies the rules in a simple language, which is later compiled into Python code that Parliament can load at runtime. In cases where the simple language cannot produce the desired behavior, arbitrary Python code can be embedded in the rule specification.

Robert's Rules of Order Newly Revised is the most common parliamentary authority, but there are others. Some might prefer to use the 1915 edition of *Robert's Rules of Order Revised*, now in the public domain and freely available on the World Wide Web. Alice Sturgis has written a newer, slightly different authority called *The Standard Code of Parliamentary Procedure*, which has achieved some notoriety as an alternative to Robert's.

Many governmental assemblies and legislatures—such as the houses of UK Parliament, US Congress, and the United Nations—have their own rules. It is also common for an organization to adopt a standard parliamentary authority but supersede certain parts in its bylaws. Parliament's separate rule specification accommodates such customization, and makes it much easier to make minor modifications by allowing users to edit the specification.

As already mentioned, new settings for parliamentary procedure, particularly online, will demand significant modification to traditional rules. Configurable parliamentary rule specifications provide a convenient platform for experimentation, and could prove useful to researchers studying group decision making. For example, a study could compare control and experimental groups between which the only difference is a small modification to the rules.

4. FACE-TO-FACE MEETING SUPPORT



pending questions

- Main motion (01:36)
  - Amend (01:36)
    - Amend amendment (01:36)
    - Postpone to a certain time (01:36

debatable; majority vote
question text

Postpone until after next election cycle.

motions now in order	class
Fix time to which to adjourn	Privileged
Adjourn	Privileged
Recess	Privileged
Question of privilege	Privileged
Call for the orders of the day	Privileged
Point of order	Incidental
Point of information	Incidental
Appeal from decision of chair	Incidental
Request to withdraw motion	Incidental
Suspend the rules	Incidental
Lay on the table	Subsidiary
Previous question	Subsidiary
Limit or extend debate	Subsidiary
Amend	Subsidiary

Figure 2: The group display. The tree of *currently pending motions* is in the upper-left corner, with the *rules* and *text* of the immediately pending motion below. The list of *motions currently in order* is on the right.

Using the Parliament module, a prototype application has been built to support face-to-face meetings conducted according to Robert's Rules of Order. The application can run on a portable computer connected to a large display such as a digital projector. A single user enters events, such as motions and votes, as they transpire in the meeting. Based on this input, the software keeps track of the meeting state and publicly displays information about items under consideration and about actions available under the rules. These features are intended to facilitate shared context among the participants, encourage adherence to the rules, and help novices engage and learn the process.

The interface of the prototype application, shown in Figures 2 and 3, is built using PythonCard;[5] the specification of Robert's Rules of Order used with it is based on Henry Prakken's formalization [6].

4.1 The Group Display

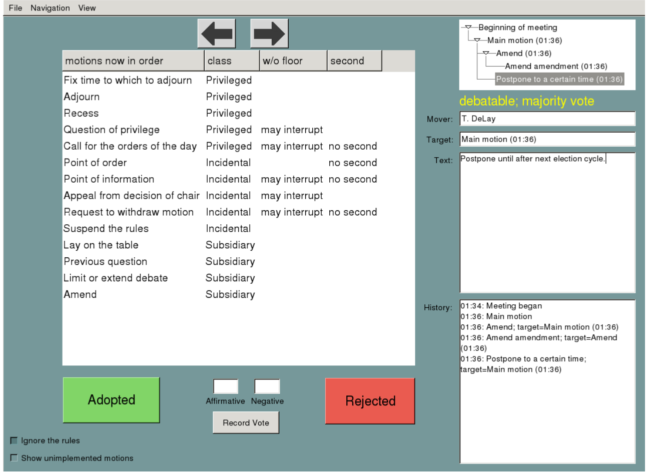
Figure 2 depicts the group display. It provides a list of *motions currently in order*, which depend on the meeting state—primarily on which motions are currently pending. A tree diagram displays the *currently pending motions* and how they are related. The immediately pending motion is always at the bottom of the diagram, and the *rules* relating to it and its *text* are below.

4.2 The User Interface

Figure 3 depicts the interactive user interface, which drives the group display. The user interface provides all the elements of the group interface, but allows interaction with them, and provides other interactive elements as well.

The user may activate any of the *motions currently in order* to indicate a motion has been made in the meeting. The *vote-entry components* comprise text fields for the number of affirmative and negative votes and a button that, when pressed, records the counts and automatically determines if the motion is adopted. *Adopted* and *rejected* buttons allow the user to indicate the fate of the immediately pending motion directly when votes are not counted. *Back* and *forward* buttons navigate through meeting history, providing a multiple undo/redo mechanism critical for usability.

[5] http://pythoncard.sourceforge.net/



motions now in order	class	w/o floor	second
Fix time to which to adjourn	Privileged		
Adjourn	Privileged		
Recess	Privileged		
Question of privilege	Privileged	may interrupt	
Call for the orders of the day	Privileged	may interrupt	no second
Point of order	Incidental		no second
Point of information	Incidental	may interrupt	no second
Appeal from decision of chair	Incidental	may interrupt	
Request to withdraw motion	Incidental	may interrupt	no second
Suspend the rules	Incidental		
Lay on the table	Subsidiary		
Previous question	Subsidiary		
Limit or extend debate	Subsidiary		
Amend	Subsidiary		

```
Beginning of meeting
  Main motion (01:36)
    Amend (01:36)
      Amend amendment (01:36)
    Postpone to a certain time (01:36)
```

debatable; majority vote

Mover: T. DeLay

Target: Main motion (01:36)

Text: Postpone until after next election cycle.

History:
```
01:34: Meeting began
01:36: Main motion
01:36: Amend; target=Main motion (01:36)
01:36: Amend amendment; target=Amend
(01:36)
01:36: Postpone to a certain time;
target=Main motion (01:36)
```

Figure 3: The user interface. The list of *motions currently in order* is on the left. The *back* and *forward* buttons are above. The *adopted* and *rejected* buttons and the *vote-entry components* are below along with the *ignore rules* checkbox. The tree of *currently pending motions* is in the upper-right corner; below it are the *rules* and the motion-detail fields: *mover*, *target*, and *text*. The *event log* is in the bottom-right corner.

The *currently pending motions* in the tree diagram can be selected, populating several other fields with information about the selected motion. In addition to the *text* of the motion, these also include its *mover* and its *target*. All these fields are editable by the user.

The interface also provides an *event log* of each motion in the order it was moved, and whether it was adopted or rejected.

Real-world assemblies sometimes deviate from the rules, either by means of a motion to suspend the rules or by mistake. The software must not break down under these circumstances, but rather must continue to track meeting state. Hence the interface provides an *ignore rules* checkbox that allows the user to record events not normally allowed by to the module's interpretation of the rules.

An organization's secretary is a natural choice to run the software because it is the secretary's duty to prepare the official minutes, and the software records most of the information required for this task. Ideally whoever runs the software should sit next to the chair so the chair can monitor the software and suggest corrections when necessary.

5. FUTURE WORK

There are many possible directions to pursue. The prospect of using Parliament for other applications such as online deliberation is especially promising. Applications for face-to-face and online deliberation built with Parliament could inter-operate. Also appealing is the exercise of writing rule specifications for different parliamentary authorities and possibly other, more radically different formal group decision-making processes such as consensus models.

For the prototype face-to-face support application, a flexible report tool that can automatically produce a draft of the minutes would be immensely useful. Other possible features include built-in support for agendas and floor control. The prototype application has been tested in one real meeting; a longitudinal evaluation could address questions such as whether the system improves conformance to the rules, whether it increases participation, and whether it leads to greater satisfaction with the outcomes and with meetings generally.

6. RELATED WORK

Other researchers have investigated implementing Robert's Rules of Order in software, and adapting the rules for use in the context of computer-mediated communication. Group Decision Support Systems (GDSS) to apply parliamentary procedure were envisioned at least as early as 1987 by DeSanctis and Gallupe [4]. One group designed a document-based collaboration system using an "agenda item life cycle" inspired by Robert's Rules of Order, and based on this, built a client-server architecture for spatially distributed collaboration, both synchronous and asynchronous [1, 8]. Horan and Benington [5] describe a protocol for conducting electronic deliberations by e-mail in academic committees that use Robert's Rules; they assume users will implement the protocol, but software could automate some of what they recommend. Davies *et al.* [3] have built an online-deliberation environment, *Deme*,[6] primarily to supplement the activities of groups that already meet face-to-face. Another project, *e-Liberate*,[7] aims to build an online-deliberation system based on Robert's Rules of Order.

7. REFERENCES

[1] C. K. Chang, A. Vorontsov, J. Zhang, and F. Quek. Rule-mitigated collaboration technology. In *Proceedings of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 1999)*, page 137, Washington, DC, USA, 1999. IEEE Computer Society.

[2] D. B. Dahlstrom and B. Shanks. Software support for face-to-face parliamentary procedure. Presented at the *Second Conference on Online Deliberation / DIAC-2005*.

[3] T. Davies, B. O'Connor, A. A. Cochran, and J. J. Effrat. An online environment for democratic deliberation: Motivations, principles, and design. Working paper, March 2004.

[4] G. DeSanctis and R. B. Gallupe. A foundation for the study of group decision support systems. *Management Science*, 33(5):589–609, May 1987.

[5] S. M. Horan and J. H. Benington. A protocol for using electronic messaging to facilitate academic committee deliberations. *Journal of Higher Education Policy and Management*, 22(2):187–197, November 2000.

[6] H. Prakken. Formalizing robert's rules of order: An experiment in automating mediation of group decision making. Technical Report REP-FIT-1998-12, GMD, April 1998.

[7] B. Shanks and D. B. Dahlstrom. Parliament: a module for parliamentary procedure software. Presented at the *Second Conference on Online Deliberation / DIAC-2005*.

[8] J. Zhang, C. K. Chang, K. H. Chang, and F. K. H. Quek. Rule-mitigated collaboration framework. In *Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC 2003)*, volume 1, pages 614–619, Washington, DC, USA, June 2003. IEEE Computer Society.

[6] `http://groupspace.org/`

[7] `http://trout.cpsr.org/program/sphere/e-liberate/about.php`