INSTITUTE OF MATHEMATICAL ECONOMICS

WORKING PAPERS

No. 298

ARTUS: The Adaptable Round Table with a User-specific Surface

by

Matthias G. Raith and Helge Wilker

May 1998

University of Bielefeld

33501 Bielefeld, Germany

ARTUS: the **A**daptable **R**ound **T**able with a **U**ser-specific **S**urface

May 1998

Matthias G. Raith **Helge Wilker**

Institute of Mathematical Economics
University of Bielefeld
P.O. Box 100131
D-33501 Bielefeld
Germany

Abstract

ARTUS is a process control system that provides an infrastructure for group decision making over the Internet in the form of controllable "Virtual Tables." The system includes options for defining the method and structure of interaction, and records the complete process for documentation or evaluation. ARTUS creates an environment which can serve as a variable platform for actual group interactions, as a laboratory for experiments, or as a classroom for educational purposes. The common basis for these different applications is conceived to facilitate the development, testing, and deployment of group support systems. In this paper we describe design and implementation of the system, and explain the technologies used.

Keywords: Process Control System, Group Support Systems, Virtual Tables

The development of ARTUS is part of a research project on "Multilateral Policy Decisions in the European Union." Financial support by the 'Ministerium für Wissenschaft und Forschung, Nordrhein-Westfalen' is gratefully acknowledged. We would like to thank Uli Middelberg and Peter Naeve for critical comments and helpful suggestions.

E-Mail: MRaith@wiwi.uni-bielefeld.de – HWilker@rz.uni-osnabrueck.de

1. Introduction

A characteristic feature of research in negotiation analysis is its prescriptive, problem-solving approach in dealing with conflicts. This has lead to a variety of concepts and procedures aimed at increasing the efficiency or fairness of negotiated outcomes. Combined with modern information technology, the analytical foundation has spurred the development of *negotiation* or, more generally, *group support systems*, allowing interacting parties to deal with otherwise too complex problems.

Despite the vast number of innovative software products, the reactions to these developments are mixed. On the one hand, SHELL (1995), for example, offers a clearly optimistic vision of future developments, while, on the other hand, WHEELER (1995) points mainly to the drawbacks of computer assisted negotiation, and thus finds the progress in software development questionable. Although a few years can make a big difference in this field, the spectrum of views has not narrowed significantly since then. However, the deficits of existing concepts now seem to be clearer, and the research agenda has become more detailed (cf. STEVENS AND FINLAY (1996)).

The development of a group decision often involves several phases of negotiation, many of which precede the actual decision. In political negotiations, for example, this is manifested in institutionalized procedures, which are known to influence the final outcome. The agenda, the type of negotiation (bi- or multilateral), or the form of communication are all important strategic factors. CHRISTENSEN AND FJERMESTAD (1997) argue that strategic decision making affects the usefulness of group support systems, but that research has not focused on this aspect sufficiently. What is missing, in their view, is a greater knowledge of group processes. From an empirical perspective, SPECTOR (1997) also finds that quantitative analytical support to negotiations is not all too popular with practical negotiators, indicating that we have not yet fully understood which type of support can be made operational in a negotiation process.

In order to study support systems in actual negotiations, one must have an infrastructure where procedures can easily be controlled, modified, and institutionalized, and which is rich enough to allow a variety of analytical approaches. This dynamic interaction of group process and support is emphasized by SHAKUN (1995,1996) in his characterization of 'evolutionary systems design,' which he sees as the appropriate framework for designing negotiation support systems (cf. BUI AND SHAKUN (1996)).

In this paper we introduce a new *process control system* which we have labelled ARTUS: the **A**daptable **R**ound **T**able with a **U**ser-specific **S**urface. The motivation behind ARTUS is to create an environment which can serve as a variable platform for actual group interactions, as a laboratory for experiments, or as a classroom for educational purposes. The common basis for these different applications is conceived to facilitate the development, testing, and implementation of group decision support systems. In combination with a data base, this allows both the documentation of the interaction and an analysis during or after the process. Due to the diversity of its tasks, the system is designed to ensure a maximum degree of flexibility.

The infrastructure of ARTUS is based solely on open Internet technologies, allowing remote use from possibly all over the world and a large number of computer platforms. Some of these technologies can be labeled as 'leading edge' at this time, meaning that they are not yet in general use, but in time it can be expected that they will be as widespread as other Internet technologies today.

In order to cope with the heterogeneity of its potential users, the platform offers easy accessibility. Negotiators can participate directly from their own computer via an Internet browser. This not only reduces communication costs, but also makes it easier to include professional negotiators. Trainings and experiments can be conducted with participants in different locations. Activities within organizations, such as in-house negotiations, training programs, or experiments do not require the prior installation of a complete software system in their local network.

An important feature is that any kind of service offered by the system is 'virtual,' implying that the participants only have access to it as long as the table is activated. There are no programs to be downloaded and installed; everything happens in the browser. This combination of flexibility and security is made possible through the system's three-stage hierarchy of users.

In section 2, we characterize the three different types of ARTUS users and how they are related. Section 3 then describes design and implementation of the system, and the technologies used. In section 4, we discuss limitations of the system and problems that may be encountered during deployment. In section 5 we conclude with an invitation to visit the Website of ARTUS. All technical terms are collected in a Glossary that we have included at the end of the paper.

2. The Users of ARTUS

Users fall into three categories. The ARTUS system is managed and administered by one or more persons known as *Owner.* They play a role similar to the system administrator in the UNIX system, who has absolute control over the system. On the next level are *Controllers*, who are responsible for a single negotiation table. The *Participants* of a negotiation are just that, participants.

Using traditional media like the telephone or email, a would-be Controller approaches an Owner about an 'appointment' for using a virtual table. The Owner uses a few web forms to enter the Controller's wishes into the database: The type of table and user interface, the expected number of Participants, the date and time. A password is entered and given to the Controller. This uncomplicated registration process allows the Controller to enter the restricted area via the ARTUS main page. Using the password to access another set of web forms, the Controller then prepares her own table. She enters the names of the Participants and some other data. As before, passwords must be provided, this time for each Participant, and there is the option of automatically sending the data out via email, if the addresses are known.

At the specified time, the Controller visits a web page on the ARTUS server containing a Java applet. This applet, called the *Controller Client*, allows her to "switch on" her virtual table, which will have the requested functions. From this moment, Participants can log in to the table using another Java applet, especially made for the type of table in question. There may be more than one type of *User Client* for a single virtual table, depending on the structure of the virtual table (e. g., one may wish to specify a table with a designated "chairman" who has more rights than the other Participants).

Participants enter through the ARTUS main page and access their client applet on a web page created specially for their table. This activates the user-specific surface chosen by the Controller.[1]

When everybody is (virtually) present, the Participants interact, the system faithfully keeps minutes about everything that crosses the table, and the Controller can monitor what happens using her Controller applet. For real-life negotiations, in which Participants may wish to communicate privately, Controller surveillance can be shut off. Another possibility is increased Controller participation, e. g. for educational purposes. For this, out-of-band communication is provided, along with a "Freeze" mechanism allowing the Controller to suspend the session. If the Controller actively wants to take part in the negotiation, she must use a separate user applet, e. g. in another browser window on her computer screen. The Controller is primarily a "table technician," and if she wishes to act as a Participant or Mediator (in principle just another type of Participant), she must do so explicitly.

If somebody inadvertently crashes their computer, exits their browser program or stumbles upon one of the many possible ways modern computers can be made to fail, the system provides a way for this unfortunate one to reestablish the connection to the virtual table and the status of the negotiation.

After the session concludes, the Controller, now using web forms once more, can review the minutes of the negotiation and either discard them (perhaps for reasons of confidentiality) or offer them to the Owners for inclusion into the long-term negotiation archive. When she is finished with this, all traces of her table, along with the user data, are purged from the system. It should be pointed out here that even if the minutes are included into the archive, no private user data is kept.

[1]During this step the Controller may ask her Participants to fill out a questionnaire, which can also be provided by ARTUS. For example, experimenters may wish to know the socio-economic background of their subjects. Information collected in this way takes the same path as negotiation minutes for a table, so that also in this case, privacy is guaranteed.

The records are simply identified by the ID everybody is given by the system, and after deleting the table-specific data from the system, there is no way of matching names to IDs.

3. Technical Design

The ARTUS system consists of the following components:

- A relational database management system (RDBMS, or DB for short),
- an HTTP (or web) server,
- a Java server program, providing core negotiation facilities, and
- multiple Java client applets, making up the user interface for Participants.

All components are integrated and interdependent. Their relationship is illustrated in Figure 1.

The user interface is made up of several web forms posing as a frontend for the database together with a set of Java applets enabling the user to participate in a negotiation. The advantage of this combination is that both parts of the user interface are easily accessible from a Java-capable web browser, thus removing many potential obstacles in the form of client-side platform dependencies, client software distribution and remote access.

The web server, using information from the database, dynamically creates appropriate pages for each visitor – Owners, Controllers, and Participants get pages for their special Java applets and access to the forms mentioned above, casual visitors can view general information and try out a demo table for a limited time.

The core functions of the system are provided by the Java server program, which makes use of information from the database and stores negotiation data in the database. The use of a fully-grown RDBMS with its built-in safety and backup features should ensure that no valuable data about negotiations is lost, as well as giving some protection against errors on the client side by enabling clients to retrieve status after crashes, usage mistakes etc. It also makes it easy to administrate users and system use. In addition, the database design helps provide security and privacy,

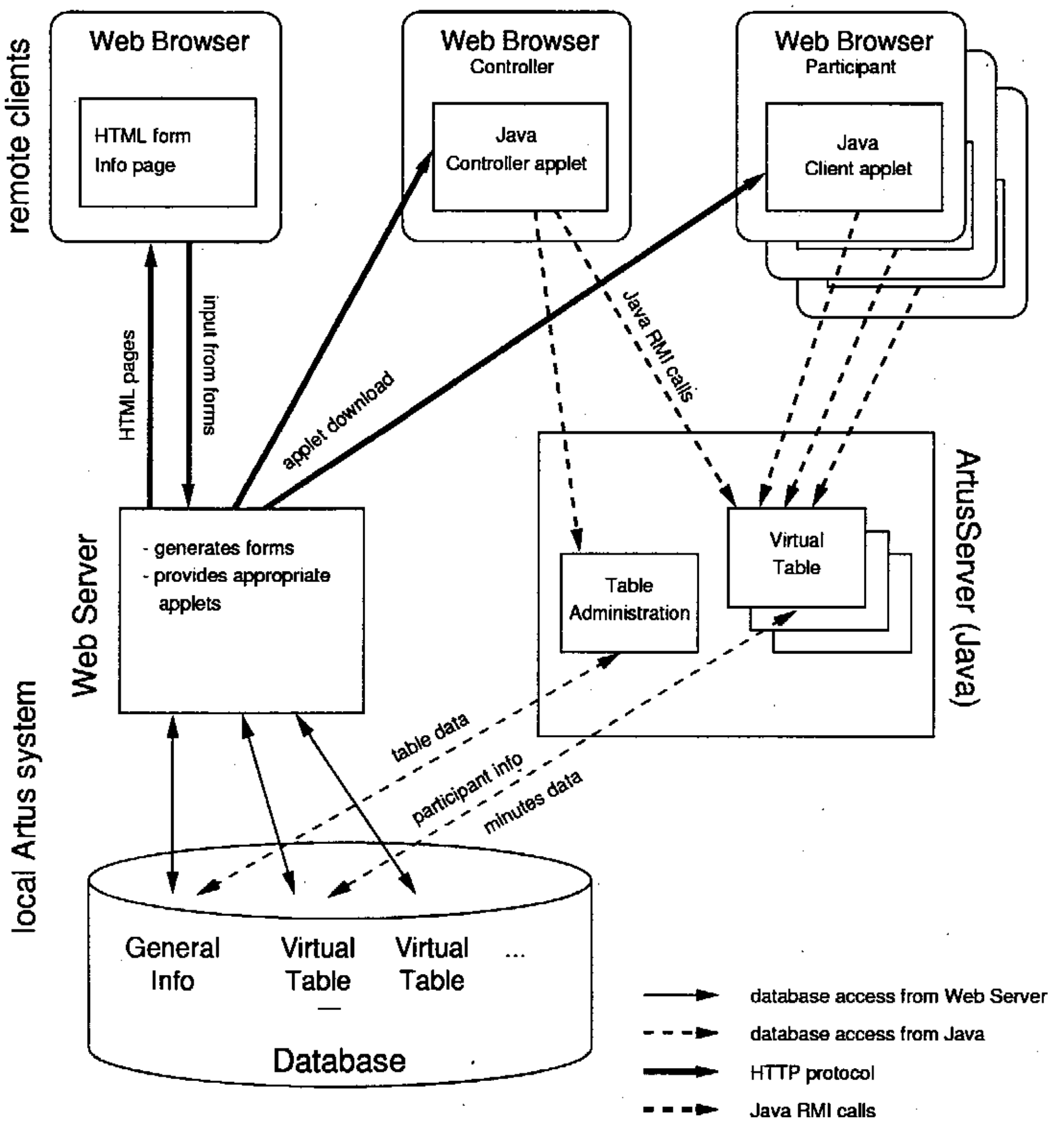

Figure 1: The architecture of ARTUS

since data about each virtual negotiation table is owned by a separate database user (the Controller, actually), making it possible for the Controller to delete it for reasons of privacy or otherwise utilize it further. This split also circumvents problems with concurrent database access when multiple negotiation tables are active at the same time, as every table works with its own set of database tables.

The choice of a relational database management system (RDBMS) may seem illogical at first. Since Java is an object-oriented language, an object-oriented database management system (ODBMS) should naturally be first choice. In fact, this may be a point for future extension of the system. At this stage, however, an

RDBMS seems the "safer" choice, as experience with this is available, and most ODBMS are, relative to relational systems, still in their infant age.

Java offers a simple way to create and use remote software objects called *Remote Method Invocation*, or RMI. This is the reason why the server program is written in Java and not just the client applets: communication between the two parts of the system is handled completely by RMI, removing another especially murky source of errors.[2] Communication between client applets and the server program is restricted to the exchange of so-called *ArtusEvents*. These are software objects used to send information in both directions. They carry information about their own type, the sender, the recipients and provide space for arbitrary data. "Type" indicates that there are many kinds of ArtusEvents: some for indicating status, some for sending messages between Participants, some for login/logout etc. This architecture is designed to be extensible, allowing the addition of new ArtusEvent types for new applications. Clients send ArtusEvents away using one Java RMI method call, and receive them using another. These two method calls are the whole communications setup, ultimately enabling clients to negotiate from remote locations using only standard web browsers. Other provisions for extensibility are given by the feature that the software is designed for easy extension of the system's functionality. It provides two Java classes, one for the client applet and another one for the server side, able to handle the basic work of login/logout, status monitoring and error recovery. The only thing lacking in these so-called *abstract base classes* is the message handling itself, i. e. the types of message content and how messages are generated or acted upon.

Using this software substrate, it is now possible to specify an actual negotiation table. Basically, the designer of the user specific surface (not to be confused with a Controller who later chooses a designed surface) defines a "method of negotiation," say, the simple sending of text messages from Participant to Participant. He then

[2]This is not to mean that using RMI (or Java) guarantees bug-free programs! It is only much simpler to use than, for instance, Remote Procedure Calls (RPCs) or the implementation of a proprietary low level protocol using TCP.

implements a subclass of the Java *Table* class defined in the ARTUS system, which offers the basic services mentioned above for the server side. He extends this class by a mechanism that takes ArtusEvents of the type "message", looks at the recipient to check whether the sender is allowed to talk to him and sends it along to the recipient's client. On the client side, he designs a user-specific surface, i.e. a user interface – in this case probably some sort of text window and a list of Participants to select from. This user interface is built by extending the other abstract base class, *Client*, which is already a subclass of the Java *Applet* class.[3]

Leaving the case of simply passing along text messages, it is possible to implement *Table* subclasses that contain more elaborate mechanisms of message processing, such as automatic mediators, voting procedures, auctioning systems, and algorithms for implementing solutions. At this point, ARTUS can become a laboratory for experiments, a bargaining table with or without analytical support (cf. RANGASWAMY AND SHELL (1997)), or an arsenal of group support techniques. ArtusEvents for messages must then contain other data than just simple text, and it may be necessary to create new types of ArtusEvents. The complexity of Clients increases accordingly. The table and its surface can thus be adapted to the specific needs of the users.

Despite the simple way in which the system can be extended, the programming must be done "in-house", as the definition of new Java subclasses or ArtusEvent types requires recompilation. This may sound inflexible, and there may indeed be a way to further extend the openness of the system to a point where Controllers can bring their own table and client classes and simply "plug" them into the system. However, the all-important question of system and user security must be answered before.

[3]If the capability of the *Client* class is provided in a way that does not require using an applet, then applications can be built independent of a browser, thus enhancing the flexibility of the system.

4. Limitations for ARTUS

One of the main obstacles for easy deployment of the system described above is the sluggish implementation of the Java standard by browser vendors, such as Netscape. Netscape's Communicator browser started its career with a real Java logo indicating full compatibility with the Java standard as defined by SunSoft. However, in one of the subsequent releases of the browser, this logo was quietly removed. It had become clear that, indeed, the Java standard had changed and what the browser supported was not complete, and therefore not compatible, anymore. This is important for the ARTUS system, because one of the elements not supported by widely used versions of the Communicator (before version 4.05) is Remote Method Invocation (RMI), the base upon which the whole software design of ARTUS rests.

Other, less important, but nonetheless annoying bugs appear in the Graphical User Interface parts of the Netscape implementation. However, Netscape has made the program source code for the Communicator software available for free, and at the same time has announced that future releases of Communicator will provide a way of integrating third-party Java Virtual Machines with the browser, thus enabling users to use the best Java implementation for their platform.

More problematic is Microsoft's Internet Explorer, the other "big browser." It does come with a Java Virtual Machine, but one that, like Netscape's, does not implement the full Java standard. Unlike Netscape, though, Microsoft is promoting its own standard for distributed objects based on the Windows operating system and expects developers to use this proprietary standard. Of course, this constrains Java programs written by this standard to computers running Windows, thus negating the alleged Java advantage of "write once, run anywhere." So for our intent of building what SunSoft calls "pure Java" programs, the Microsoft browser in its current form cannot be used.

The Java standard continues to be a moving target, and at this time the only web browser that can be relied upon to fully implement "pure Java" is SunSoft's own HotJava browser. This browser is available for download ready-to-run in packages for Windows 95/NT and SunSoft Solaris. The browser can be made to run under other environments where a suitable Java runtime implementation is available, e. g. Linux (definitely) and other Unixes (almost certainly).

It is to be expected that the problem with lacking RMI support will go away in time as browser vendors update their Java implementations. Hence, time seems to be working for ARTUS. Other ways to circumvent this obstacle would be to create standalone, compiled client applications for certain platforms when necessary. This, however, would sacrifice the advantage of the simple, standard user interface in the form of the browser.

Another possible source of problems could be the performance of the RMI layer of Java. It remains to be seen whether the architecture described above survives in the wilderness of the Internet jungle, or if delays and data loss make communication by this method impossible over long distances. At least in local networks, though, this should be less of a problem.

5. Summary

The ARTUS system provides an infrastructure for group decision making over the Internet in the form of "Virtual Tables." The system includes options for defining the method and structure of interaction, e. g. through negotiations, voting, or auctions, and keeps "minutes" of everything that happens during a session for later examination. An important feature is that every Virtual Table has its own Controller.

Through its user-specific surface, ARTUS is adaptable to different environments. This variability is necessary to satisfy the wide research framework for group support systems.

ARTUS allows support tools to be integrated, thus creating possibilities for a variety of group support systems. However, due to the fact that the virtual negotiation tables run inside a browser window, individual support systems can be used separately as well.

ARTUS can be contacted under the following address:

http://artus.wiwi.uni-bielefeld.de/

Visitors who are not registered Participants or Controllers are invited to test a simple 'Demo Table' for a limited amount of time, in order to gain an impression of the systems capabilities.

Glossary

applet	A type of Java program made to be downloaded from a web server and run inside a web browser. Special security considerations apply: An applet normally cannot access files on the computer the web browser is running on or create network connections to other computers.
ArtusEvent	A class in the ARTUS system, responsible for communication between the different components (User and Controller clients, and the Table server program). There are many types of ArtusEvents, and new ones may be defined to implement new system features. ArtusEvents are stored in the system's database for archival and safety purposes (in case a user's client crashes, it can retrieve all communication after restart).
class	A "blueprint" for a software entity, defining data structures and methods to manipulate them. Classes can be extended by subclassing. A subclass is class that has all the features of another class, called the superclass, but may change some of them or add other features. A subclass is said to *inherit* its superclass's features.
client	A program meant to be used by a human. It accesses some server to retrieve information, perform computation etc., and display the results in a convenient form. In ARTUS, clients come in the form of Java applets. There are User and Controller clients for access to a Virtual Table, as well as an Owner client for monitoring the whole system.
HTML	Hypertext Markup Language: the *lingua franca* of the World Wide Web. A document structuring language. The author of a document specifies categories such as "title," "heading," "text" etc., not caring about how they are eventually displayed. A reader generally uses a web browser to view an HTML document, which transforms the structure information given by the author to visual forms, like large, bold type for headings or italic type for emphasis.
HTTP	Hypertext Transfer Protocol: a network protocol used mainly to transfer HTML pages and associated data (images, sound, video) between a web server and a web browser.

Java	An object-oriented programming language designed by SunSoft. Once written, Java programs run on many hardware platforms. Java has features simplifying Internet applications: A well-defined security model, easy inter-program communications over the Internet, and integration with web browsers.
Java Virtual Machine	A piece of software that actually runs a Java program. In most cases, Java programs are translated into *bytecode*, a transitional stage more efficient than human-readable Java code and less hardware-dependent than machine code for a specific computer. This bytecode is interpreted by the Virtual Machine.
object	A software entity encapsulating data and methods for working on this. An object is an *instance* of a class. There may be many objects of the same class.
RDBMS	Relational Data Base Management System: The software that runs a database. "Relational" means that everything is stored in two-dimensional tables (called relations). An RDBMS provides efficient storage and retrieval of data, as well as safety features like backup.
Virtual Table	A part of the ARTUS system. A Virtual Table has special properties, e. g. defining a negotiation method, and it is the server to the Participant and Controller clients.
web browser	A program enabling users to request HTML pages from remote web servers. Many widely-used web browsers also contain a Java Virtual Machine, enabling users to run Java applets.
web server	A program that serves requests made by web browsers for HTML documents. These documents may be static, containing text, images or other non- or slowly-changing information, or dynamic, created on the fly by the web server, e. g. by extracting data from a database. Documents also can contain Java applets.

References

BUI, T.X., M.F. SHAKUN (1996): "Negotiation Processes, Evolutionary Systems Design, and NEGOTIATOR," *Group Decision and Negotiation*, 5, 339-353

CHRISTENSEN, E.W., J. FJERMESTAD (1997): "Challenging Group Support Systems Research: The Case for Strategic Decision Making," *Group Decision and Negotiation*, 6, 351-372

RANGASWAMY, A., G.R. SHELL (1997): "Using Computers to Realize Joint Gains in Negotiations: Toward an "Electronic Bargaining Table"," *Management Science*, 43, 1147-1163

SHAKUN, M.F. (1995): "Restructuring a Negotiation with Evolutionary Systems Design," *Negotiation Journal*, 11, 145-150

SHAKUN, M.F. (1996): "Modeling and Supporting Task-Oriented Group Processes," *Group Decision and Negotiation*, 5, 305-317

SHELL, G.R. (1995): "Computer-Assisted Negotiation and Mediation: Where We Are and Where We Are Going," *Negotiation Journal*, 11, 117-121

SPECTOR, B.I. (1997): "Analytical Support to Negotiations: An Empirical Assessment," *Group Decision and Negotiation*, 6, 421-436

STEVENS, C.A., P.N. FINLAY (1996): "A Research Framework for Group Support Systems," *Group Decision and Negotiation*, 5, 521-543

WHEELER, M. (1995): "Computers and Negotiation: Backing into the Future," *Negotiation Journal*, 11, 169-176