An Architecture for Intelligent CSCL Argumentation Systems

Frank Loll, Niels Pinkwart, Clausthal University of Technology, Clausthal-Zellerfeld, Germany
frank.loll@tu-clausthal.de, niels.pinkwart@tu-clausthal.de
Oliver Scheuer, Bruce M. McLaren, German Research Center for Artificial Intelligence, Saarbrücken, Germany
oliver.scheuer@dfki.de, bmclaren@dfki.de

Abstract: Argumentation is a key research area within CSCL. Yet, while many empirical studies investigating the educational benefits of various forms of collaborative argumentation have been conducted, there has not been much work done towards developing generic and reusable software architectures for collaborative argumentation that have the potential to reduce the development time for argumentation learning systems. This paper proposes a general architecture for intelligently supported collaborative argumentation systems.

Introduction

Argumentation skills are critical for humans in many aspects of life. Some researchers have characterized argumentation skills as central to thinking itself, supporting people in coming to rational conclusions about various issues in life (Kuhn, 1991). Consequently, teaching argumentation skills is a central goal of education, both on a general level – for instance, classroom dialog is a form of group argumentation – and also in specific application areas such as the physical sciences or the law. In addition, argumentation and learning are intertwined: argumentation involves elaboration, critical reasoning, and reflection, all of which support deeper learning. Often, human teachers provide instruction on argumentation through face-to-face dialog and direct interaction with students.

However, while classroom learning is an excellent way for students to learn to argue, it is difficult to "scale up" this approach, teaching large numbers of students effectively, due to limitations in teacher time and availability. Educational technology and especially CSCL systems can help realize instruction on argumentation at a large scale. Indeed, there has been considerable effort in developing and assessing educational technology to support argumentation (e.g., Andriessen, 2006; Weinberger et al. 2005). Many of these efforts have been shown to be effective for various argumentation domains. At the same time, there has not been the same amount of research towards generic, flexible and reusable software architectures for building educational collaborative argumentation systems. Being able to build upon a well-designed software architecture has the potential effectively to reduce the development time required for constructing collaborative argumentation learning systems as compared to a typical "from scratch" development approach. Since many non-trivial development, design, testing and deployment aspects can be dealt with at an abstract architecture level, such a general architecture holds the promise that we, as a research community, won't have to keep reinventing the wheel and building throw-away prototypes. But what are the essential design features and requirements for a flexible software architecture that facilitates the implementation of a rich variety of potentially differently targeted educational argumentation systems for research purposes and practical classroom usage? Based on an extensive literature review of approximately 50 argumentation systems (covering both general-purpose and educationally targeted tools, e.g., Suthers et al., 2001; Reed and Rowe 2006; McLaren et al, 2007), this paper summarizes the key requirements and – based on these – proposes an architecture for a CSCL argumentation system with intelligent support functions.

Architecture Requirements

The architecture requirements for a CSCL argumentation system can be classified into several categories. On the *overall* level, the architecture must be flexible (enabling different variants of argumentation tools to be built on top of it) and at the same time platform independent at least with respect to the clients. The latter requirement is especially important for classroom usage, where multiple machine types have to be supported. It should also be programming language independent and provide a loose coupling between system parts: these two characteristics facilitate the addition of new components (such as specific add-ons for research purposes) to the system. Next, there should be support for both *synchronous* and *asynchronous* collaboration. Here, it is important that the system works well for small groups (≤5) in synchronous usage and scales up well in asynchronous usage for larger, at least classroom size, groups. Furthermore, there should be a detailed role and rights management for controlling the access of users to system functions (important for comparative studies and for classroom usage) based on roles such as moderator, administrator, teacher, or "student group member." These roles are also important for using pre-defined learning processes (collaborative learning scripts) that have shown to be educationally effective (e.g., Weinberger et al. 2005) and should thus be foreseen in the architecture.

Awareness mechanisms are a further key feature required for CSCL argumentation systems. For synchronous usage, this means highlighting parts of the argument space to provide users with information about the current contributions of other users. For asynchronous usage, notification messages (e.g., what has changed since the last login?) are an important means for creating awareness. *Communication* is another important factor for productive collaboration: The architecture has to allow for different forms of user communication such as text-chat as well as audio or even video chat (it should be easy to turn these options on and off for research purposes), and there must be support for special communicative roles like a moderator or an intelligent feedback agent with special rights such as interrupting a user group to give hints or to ask questions. The group mode of communication has to be highly flexible, enabling different groups (small vs. large) to communicate about different aspects of an argument and to jointly edit/create arguments. The communication protocol should be easy to understand, so that future developments and additions are supported. Concerning the *application data*, an important requirement is *persistent storage*: neither in research contexts nor in classrooms is a data loss acceptable. Also, there should be support for logging single user operations to replay the argumentation process or to restore an argumentation state at any time. These functions are important for automated or manual data analysis. Especially for mobile usage contexts, the possibility of working with local data without accessing a network can be helpful.

A next class of requirements comes from the fact that different areas of argumentation (such as scientific argumentation, argumentation in law or argumentation in philosophy) involve different types of arguments and different styles and rules of argumentation. This has implications both related to system *interaction* and *visualization* (e.g. threaded discussions, graphical style, text, matrix-views, etc. – here it should be possible to use different views for the same data) and to the underlying argument ontology. Thus, the architecture must support different *argument ontologies* and provide the option of extending or manipulating these (which is especially interesting in research contexts) as it is, for instance, possible in Digalo (Kochan, 2006). Another important aspect for some argumentation domains is the ability to create links from argument elements to *external resources* (or parts of these) such as texts, web pages or videos. LARGO (Pinkwart et al., 2006), Belvedere (Suthers et al., 2001) and Araucaria (Reed and Rowe 2006) are examples of argument systems that employ such links to external resources. Finally, the ability to automatically analyze student group argumentation and to provide the learners with feedback on their progress is a key requirement that has been shown to be educationally beneficial in a variety of studies. As such, *analysis and feedback components* should be included in the system architecture. Here, flexibility is a key requirement: especially for research purposes, it should be straightforward to develop and include custom analysis and feedback agents in the architecture.

System Design Approach

Currently, there are no argumentation systems (CSCL or other) that satisfy all of the above requirements. In the following, we propose a system architecture that addresses all the listed challenges. Overall, this architecture (shown in Figure 1) is structured as a three-tier system with the following building blocks:

1. The data layer, which manages all of the data of the application. This layer stores the registered users with their models, including roles and rights. It logs all actions (for use in replay and analysis) and contains the argument ontologies as well as the collaborative learning scripts.
2. The server layer, which provides functionalities to control the user actions (who is allowed to do what?), bring the concurrent user actions into a consistent state, and manage the different scripted collaborative argument sessions with their multiple participants.
3. The client layer, which contains the user applications as well as the intelligent agents. Components in this layer can manipulate the current state of an argument and can communicate with others.

These three tiers are loosely coupled: they communicate only via platform independent XML messages. This design facilitates system extensions, since new components – such as new feedback agents or new client user interfaces – do only have to "understand" the XML data format and can then be included in the system regardless of the programming language they are built with. XML as a data exchange format also has the advantage that it is – at least partially – human readable, and some relevant XML based standards (e.g., IMS-LD for learning scripts and AML (Reed & Rowe, 2006) for argument models) already exist.

The architecture diagram also shows that there is a separate argument model in each layer (called "SessionObject"). These models all are kept consistent with the "central" argument model stored in the data layer. This approach, which is similar to the "hybrid CSCL architecture" proposed by Suthers (2001), means that all client applications have the argument data stored on the local machine. This design makes it possible to work locally without losing work even when the network connection to the server is temporarily unavailable. The server ensures that all the data models are synchronized (e.g., when one machine that was temporarily disconnected rejoins the network). Most of the communication and application logic is stored in the server. This is important for two reasons: first, to achieve a loose coupling of components (required to allow for an exchange of components: not too much should depend on a specific client) and, on the other hand, to allow the distribution of client applications via the web, where large downloads are usually not appreciated.

To make system extensions as easy as possible, the technical interfaces between server/client and server/AI engine are identical. This means that generally, intelligent agents have the same action options that human users have. Possible differences between human users and artificial intelligence (AI) agents concerning access or action rights can be defined in the data layer: e.g., a feedback agent might be allowed to interrupt any group discussion, while a student will not. An AI client might take the role of a moderator, an ordinary participant, a (domain- specific) advisor or even an "awareness agent" which provides other users with awareness information. The flexible architecture even allows using several AI clients at the same time, for instance, a team of advisors / coaches each of which having one specific expertise (e.g., a collaboration coach and a domain-knowledge coach).

The client layer contains a "portal" component (which allows users to choose the argument they want to join) and supports multiple views on argument data. This is useful to allow for devices with special interface requirements (such as mobile devices), and to visualize the same argument in different forms (which may be desired in research contexts). Finally, to get easily maintainable code and exchangeable components, general software design patterns like observer (see client layer) or proxy (see server layer) (Gamma et al, 1995) are employed.

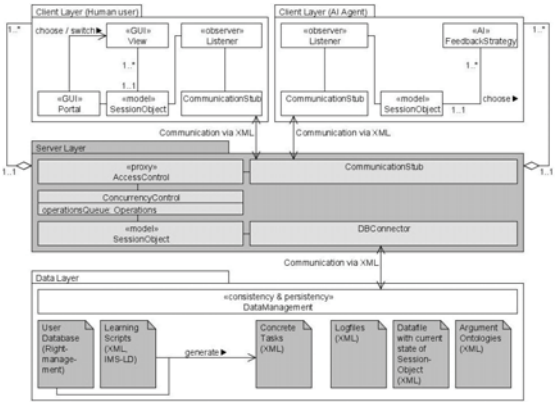

Figure 1. UML diagram of proposed system architecture

Outlook

In our current work we are implementing an argumentation system based on the described architecture. The aim is to create a generalized framework and methodology for the construction of argumentation support systems to help students learn argumentation in different domains.

References

Andriessen, J. (2006). Arguing to Learn. In: Sawyer, R.K. (Ed.), *The Cambridge Handbook of the Learning Sciences*, Cambridge University Press, New York, pp. 443-460.

Gamma, E., Helm, R., Johnson, R. E., & Vlissides, J. (1995). Design Patterns. *Elements of Reusable Object-Oriented Software*. Addison-Weslay Longman, Amsterdam.

Kochan, E. L. (2006). Analysing Graphic-Based Electronic Discussions: Evaluation of Student's Activity in Digalo. In: *Proceedings of the European Conference on Technology Enhanced Learning* (EC-TEL 2006), Crete, Greece, Lecture Notes in Computer Science, Springer, pp 652–659.

Kuhn, D. (1991). *The Skills of Argument*. Cambridge University Press.

McLaren, B.M., Scheuer, O., De Laat, M., Hever, R., De Groot, R., & Rosé, C.P. (2007). Using Machine Learning Techniques to Analyze and Support Mediation of Student E-Discussions. In: *Proceedings of the 13th International Conference on Artificial Intelligence in Education* (AIED 2007), IOS Press, (p. 141-147).

Pinkwart, N., Aleven, V., Ashley, K., & Lynch, C. (2006). Toward Legal Argument Instruction with Graph Grammars and Collaborative Filtering Techniques. In: *Proceedings of ITS2006*, pp. 227-236. Springer.

Reed, C. & Rowe, G. (2006). Araucaria 3.1 User Manual (Including: Argument Markup Language (AML)), http://araucaria.computing.dundee.ac.uk/usermanual3_1.pdf, last visited 2008-12-19

Suthers, D. D. (2001). Architectures for Computer Supported Collaborative Learning. In: Proceedings of the IEEE *International Conference on Advanced Learning Technologies* (ICALT 2001), Madison, Wisconsin

Suthers, D. D., Connelly, J., Lesgold, A., Paolucci, M., Toth, E. E., Toth, J., Weiner, A. (2001). Representational and Advisory Guidance for Students Learning Scientific Inquiry. In: Forbus, K.D. & Feltovich, P.J. (Eds.), *Smart Machines in Education: The Coming Revolution in Educational Technology*, pp.7-35

Weinberger, A., Fischer, F. & Stegmann, K. (2005). Computer-Supported Collaborative Learning in Higher Education: Scripts for Argumentative Knowledge Construction in Distributed Groups. In: *Proceedings of Computer-Supported Collaborative Learning* (CSCL-05).

Acknowledgments

This work is supported by the German Research Foundation (DFG) under the grant "Learning to Argue: Generalized Support Across Domains" (LASAD), PIs: Bruce M. McLaren, Niels Pinkwart.