HOW TOUGH SHOULD IT BE? SIMPLIFYING THE DEVELOPMENT OF ARGUMENTATION SYSTEMS USING A CONFIGURABLE PLATTFORM

Frank Loll[1], Niels Pinkwart[1], Oliver Scheuer[2], Bruce. M. McLaren[2]

[1]Department of Informatics, Clausthal University of Technology, Clausthal-Zellerfeld, Germany
[2]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany

Abstract: Teaching to argue is challenging. Classic face-to-face approaches do not scale up for large groups due to resource limitations (teacher time), but have shown to be effective. As a consequence, there have been attempts to convey argumentation skills via educational software. Even though some of these systems have shown their suitability in their original domains of application, the systems typically do not generalize – there has been little carry over to other domains. This chapter reviews existing approaches, their technological strengths and weaknesses, and proposes a generic architecture to overcome the latter. Based on this architecture, the LASAD (Learning to Argue – Generalized Support Across Domains) framework has been developed. The goal of this framework is to simplify the development of argumentation systems based on some well-defined configurations. In this chapter, we describe the flexibility of the LASAD framework and demonstrate how it can be configured to emulate the existing argumentation systems Belvedere and LARGO.

INTRODUCTION

Argumentation skills are essential in various aspects of life. On the one hand, there are domain-dependent argumentation skills. Examples can be found, for instance, in the law, where a lawyer tries to win a case by convincing a judge or jury, or in science, where a researcher supports his or her hypothesis with data gathered from experiments or observations. On the other hand, argumentation skills are also important in everyday life – imagine a child trying to persuade his or her parents to increase her weekly pocket money. Thus, it is important to learn how to argue. Some researchers characterize argumentation even as central to thinking itself (Kuhn, 1991).

Although argumentation and the underlying principles of what makes up good (or bad) arguments differ across domains, there are similar ways to teach argumentation in many domains, typically following a face-to-face approach. Here, one teacher instructs a small group of learners or even just one learner. These approaches have been shown to be highly effective teaching methods (Bloom, 1984; Kulik and Kulik, 1991).

Although this approach is effective and convincing, it lacks scalability, i.e. it is not possible to apply the same teaching method to larger groups since time and person resources are naturally limited. Thus, there have been attempts to support the acquisition of argumentation skills via software tools. These tools differ in the way they support the development of argumentation skills as well: Some systems serve as pure visualization tools to reach a common understanding via different visualization techniques (Kirschner, Buckingham Shum and Carr, 2003; Van Gelder, 2003). These visualizations could be graphs (as used, for instance, in *Belvedere* (Suthers, Weiner, Conelly and Paolucci, 1995), *Convince Me* (Schank and Ranney, 1995; Siegel, 1999), *Araucaria* (Reed and Rowe, 2004) and *Athena* (Rolf and Magnusson, 2002), matrices (as used in *Belvedere*), containers (as used in *SenseMaker* (Bell, 1997; Bell and Linn, 2000) or linear and threaded texts (as used in *Academic Talk* (McAlister, Ravenscroft and Scanlon, 2004) and HERMES (Karacapilidis and Papadias, 2001). Other educational argumentation systems and frameworks try to analyze the arguments created by the learners to find

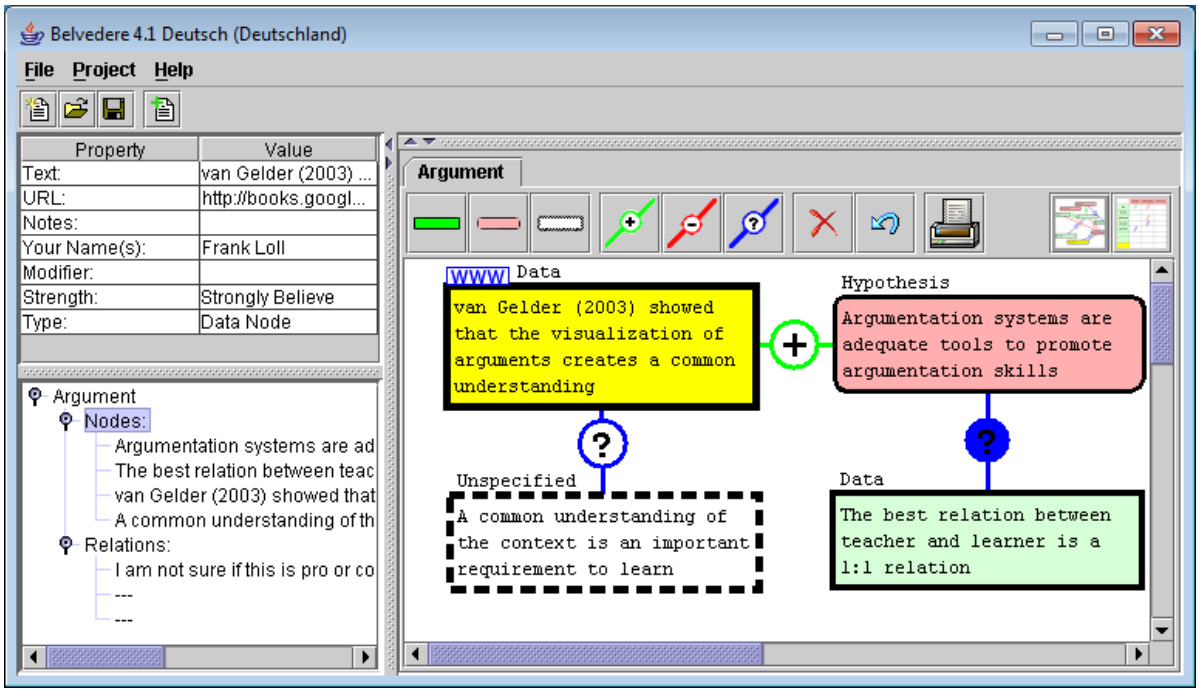

Figure 1: Belvedere (version 4.1) in "Evidence Mode"

possible weaknesses and give hints on how to improve argumentation, as is done, for example in *LARGO (Legal Argument Graph Observer)* (Pinkwart, Aleven, Ashley and Lynch 2006) and *ARGUNAUT* (De Groot *et al.* (2007); McLaren, Scheuer and Mikšátko, 2010).

ARGUMENTATION IN DIFFERENT DOMAINS

To clarify the question why there are so many different approaches and tools to support the acquisition of argumentation skills, one must have in mind that the domains in which argumentation takes place differ considerably. In the legal domain, for instance, argumentation is a structured process involving two parties, the defendant and the plaintiff. The lawyers of both parties try to "win" the case for their respective clients by convincing the judge or a jury with arguments. The ground rules for arguing in the courtroom even differ between countries. In contrast to the Civil law premise (applied in many countries in continental Europe) in which laws are encoded as statutes, in the Common Law used in England and the U.S. the law is highly reliant on "precedent cases", i.e. new cases should be decided in accordance with prior similar cases. Apparently, decisions in such cases are also based on laws and statutes. The difficulty in using these for argumentation is based on their open textured nature (Gardner 1987), meaning that their conditions for application are abstract, must be interpreted in the context of specific cases, and are thus prone to subjectivity. Unlike many other types of argumentation, legal argumentation features a moderator (the judge) present at all time, who has to assure that protocol and legal ground rules are correctly applied so that either the judge himself or a jury can decide the case.

Compared to legal argumentation, argumentation in ethics is different in many respects: Here, there is no authoritative or established and structured approach to resolve ethical problems, i.e. there is no judge who decides which argument is strongest and no institutional use of *stare decisis* (the legal principle by which judges are obliged to obey the precedents established by prior decisions). Thus, ethical arguments are typically more free-form in style and structure. Another key distinction is that the decision-making process

in ethics does not always (or even typically) involve a pre-defined number of parties: even a single ethicist may present both pro and con positions or there may be more than two parties debating. Additionally, ethics cases are not constrained to binary conclusions as compared to legal argumentation. Finally, the goal in arguing and evaluation ethical problems is (typically) not to "solve a case" but rather to learn about the ethical ramifications of various actions.

A third example is scientific argumentation. Here, the number of parties involved is also not restricted to two opponents. Instead, there can be multiple parties who agree on a common standpoint but differ in details, and there can also be multiple (more than two) standpoints. The facts and theories that can be used to argue can be revised based upon observations and conclusions drawn from new insights or experiments. Since the relevant knowledge and information are subject to change (whenever new observations are made), there is not always a definite decision about a specific point. This is different compared to law where a judge or jury can (and has to) finally close the discussion with a decision applied to a single case (which will not be changed). To illustrate how these domain-dependent differences influence the design of argumentation systems, we will look at two prominent examples of argumentation systems in more detail: *Belvedere* and *LARGO*.

Belvedere, on the one hand, is a multi-user, graph-based diagramming tool especially designed for scientific argumentation. In *Belvedere*, one states hypotheses that can be supported or rebutted by means of facts. An example of an argument created in *Belvedere* is found in Figure 1. *LARGO*, on the other hand, is an argumentation system designed to support individual law students in the acquisition of argumentation skills. Here, a transcript of a trial is given to the students who are asked to extract the arguments from both sides. An example of an argument

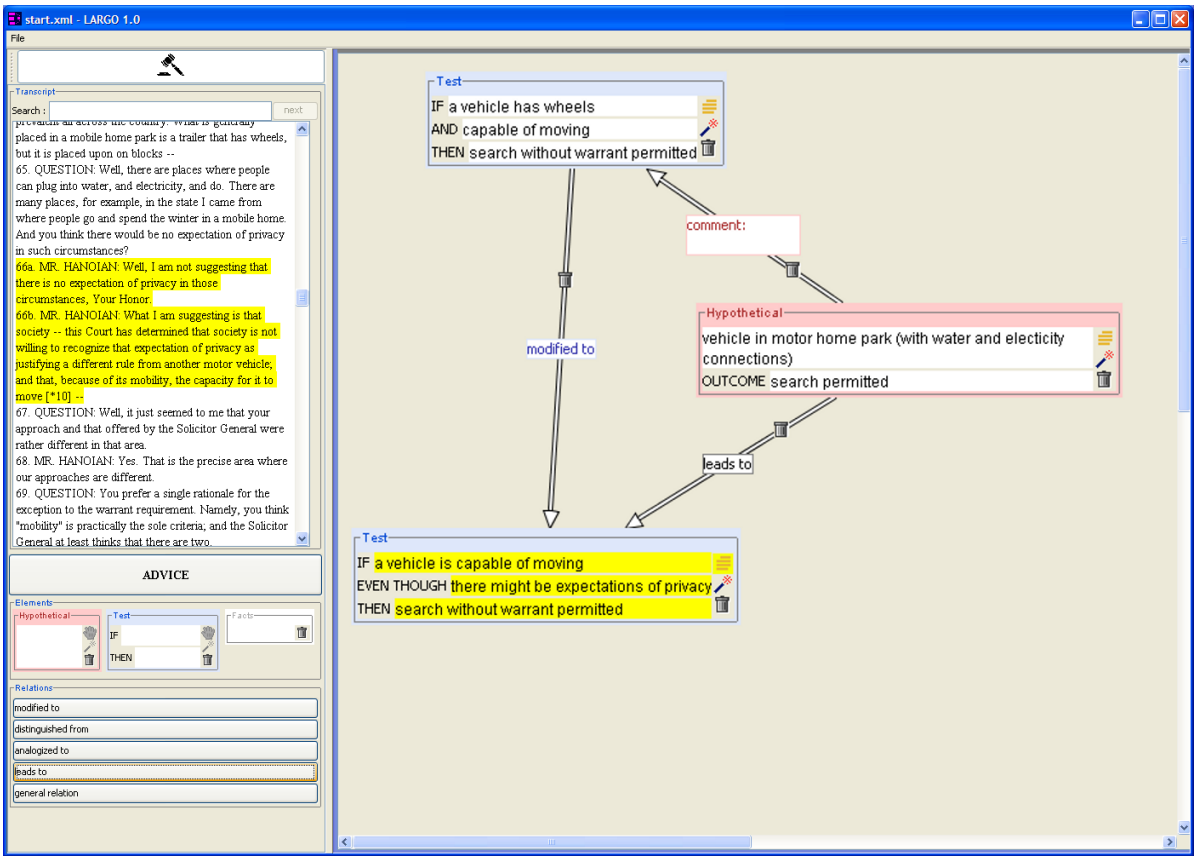

Figure 2: Screenshot from LARGO tutorial

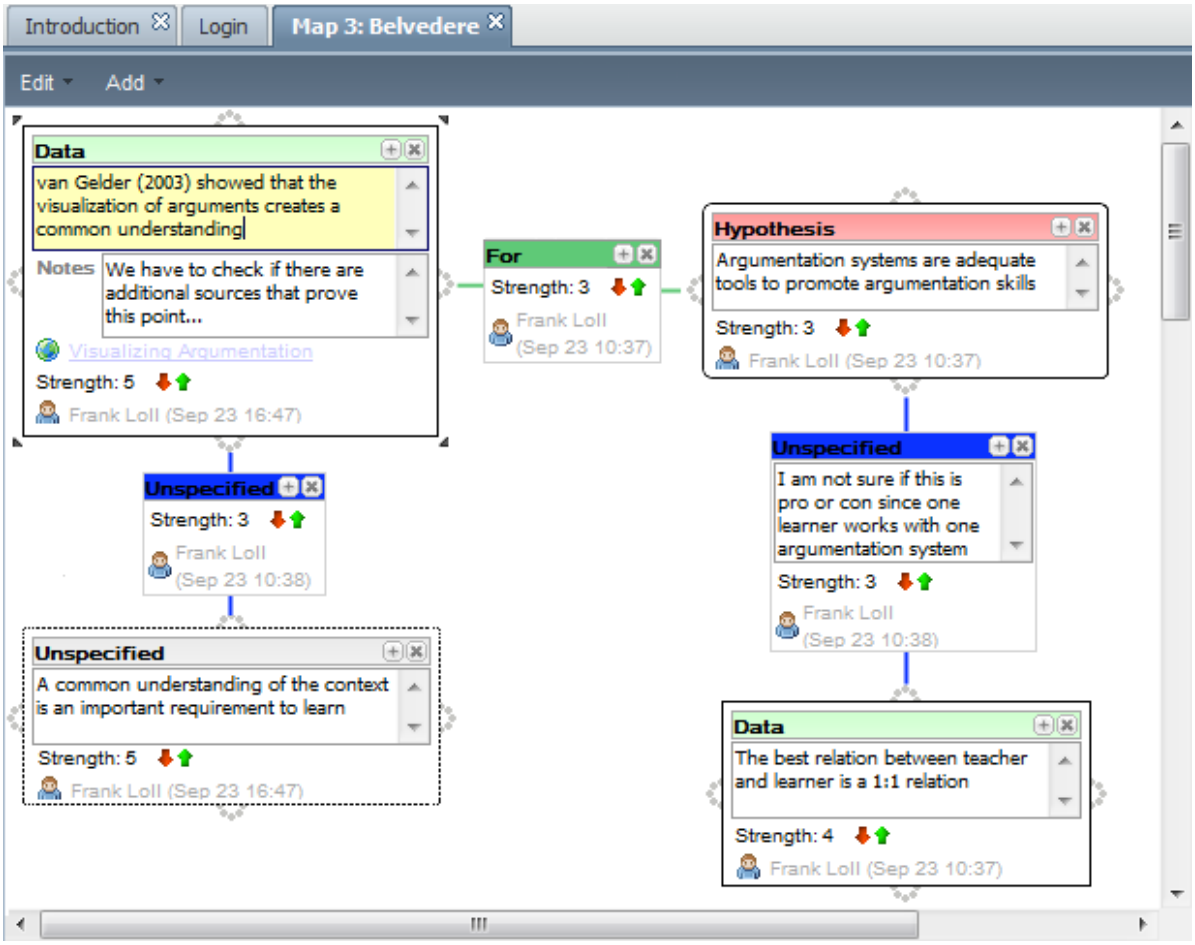

Figure 3: LASAD in graph-style visualization with Belvedere ontology

created in *LARGO* is shown in Figure 2.

On the basis of these two short descriptions, you will notice at first glance that both tools, even though they aim at the same goal (the acquisition of argumentation skills in their specific domain), differ in their core principles and approaches. Whereas *Belvedere* guides students to make external references to back up stated facts, *LARGO* requires the integration of an internal text (an argument transcript) to have users link nodes in the graph to parts of this transcript. Furthermore, the available modeling elements differ: In *LARGO* there are only three types of nodes (hypothetical, test, fact), but five types of relations (modified to, distinguished from, analogized to, leads to, generalization). In *Belvedere* one can choose between three node types as well, but the available types (data, hypothesis, and unspecified) differ. On the relations' side, there are only three types available (pro, con, unspecified). Apart from the available elements, the systems also follow different user strategies. *Belvedere* provides multi-user functionalities, allowing users to create arguments together with other arguers, whereas *LARGO* is designed to be used by students on their own (an exception here is the assessment of the quality of an argument which is done by peer reviews, see (Pinkwart *et al.* 2006; Loll and Pinkwart, 2009) for details). These are the most obvious differences, but they led to costly and time-expensive development of independent systems aiming for the same purpose: Training students to argue.

THE LASAD FRAMEWORK

To minimize the efforts of developing tools that fit domain specific needs, we developed the *LASAD (Learning to Argue: Generalized*

Support Across Domains) framework. It was particularly designed to facilitate the creation of argumentation systems by means of a flexible configuration mechanism. Its primary goal is to avoid excessive development time and costs in future development of argumentation systems. Whereas the development of *Belvedere* and *LARGO* took several years, it is our goal to create argumentation systems that offer similar possibilities to their users by means of a configuration mechanism in only a fraction of the time of past developments. That is, it is possible to "create" most parts of the system like the available elements, the graphical user interface, the collaboration support etc, by means of configurations, eliminating the need of coding as far as possible. Examples of how different configurations of LASAD that emulate *Belvedere* and *LARGO* look are shown in Figures 3 and 4.

These figures also show the different ontologies (i.e., node and edge types) used in the two emulated systems, and how these can be represented in the LASAD framework. However, argumentation systems do not differ only in their domain-dependent ontology. To identify open issues in the development of argumentation systems that should be solved by a generic framework, we conducted a detailed review of existing argumentation systems (Scheuer, Loll, Pinkwart and McLaren, 2010). In the review we covered a broad range of topics including general information (e.g., system purpose/intended usage), argumentation related criteria (e.g., domain and ontology), main system functions, degrees of system flexibility, collaboration options, intelligent argument analysis and system feedback, user-interface design and interaction techniques, technological criteria (e.g., adopted technology standards, software architecture) as well as evaluation related criteria. The results can be summarized as following:

First, most argumentation systems are either specially designed for a single domain, e.g. the law, ethics or science, or are too general to serve as appropriate e-learning tools in specific domains. While the former case often involves a limited and too specific ontology, i.e. in a domain-specific set of elements to create an argument (e.g. hypothetical, test and fact as node types in the law as used in *LARGO*), the latter usually entails an ontology that is too general to fit domain-specific needs (e.g. only general nodes as used in Athena). A happy medium between these two approaches will, on the one hand, provide domain-specific tools to create adequate arguments, but, on the other hand, will be flexible enough to be used in multiple domains. Such a system development tool is not yet available. System configurability that would allow a system to be that flexible would be beneficial (Dimitracopoulou, 2005; Lonchamp, 2006; Slagter, Biemans and Ter Hofte, 2001). First attempts along these lines can be found, for instance, in *Digalo* (Schwarz and Glassner, 2007), where it is possible to define the available elements to model argumentation with respect to their number and appearance as well as to define user roles and rights to fit domain-specific needs. Nevertheless, the configuration mechanisms of *Digalo* are restricted to the appearance of the elements. Thus, it is not possible to add domain-specific elements such as, for instance, a transcript, as is used in LARGO.

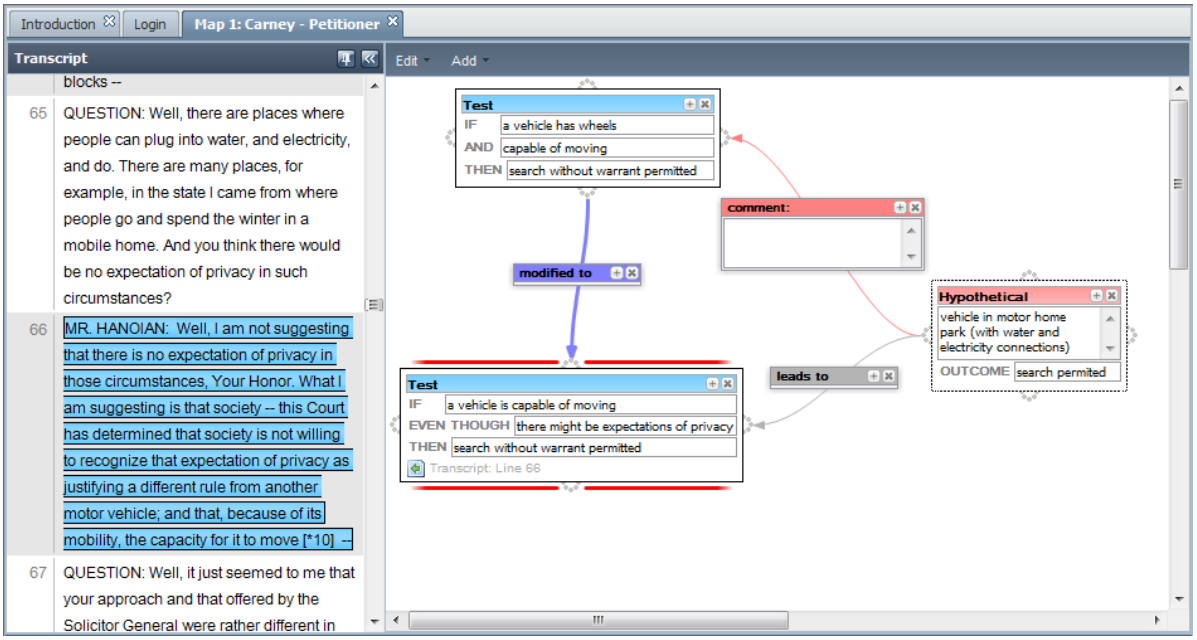

Figure 4: LASAD in graph-style visualization with LARGO ontology

Second, lots of available argumentation systems (e.g. *Athena* (Rolf and Magnusson, 2002), *Araucaria* (Reed and Rowe, 2004) are designed solely for single users. This is somewhat surprising, since the question whether argumentation skills are typically best practiced in learner groups – i.e., through students interacting with peers (and/or the teacher) – or in individual work is still open. Thus, an approach supporting both settings would be beneficial. Some systems attempt to bypass this problem by providing means for argument data import and export to at least support asynchronous collaboration. Athena, for instance, uses a report generator to prepare arguments for later group discussion, while *Araucaria* provides a central database (AraucariaDB) to make arguments exchangeable via the web. Nevertheless, it would be beneficial to provide adequate collaboration support (also for synchronous collaboration) during the whole argumentation process like, e.g., *Belvedere* does.

Third, most systems are isolated from other systems and technology, i.e. they do not offer public interfaces to communicate with other systems that may provide superior tools for some tasks. An example for the usefulness of this kind of interoperability between systems is *ARGUNAUT* (De Groot et al., 2007), an analysis framework designed to support teachers and moderators in identifying possible problems in students' interactions independent of the underlying system so that it is possible to intervene. Another example is *CoFFEE* (see (De Chiara, Manno, and Scarano, 2010) in this book), which is an expandable framework in which new functionalities can be added as autonomous and configurable components. Some components are designed especially for argumentation (e.g., the graphical and the threaded discussion tool). Together, these components use a shared workspace and can be used in groups to define sessions to adapt the system for use in courses with different requirements, e.g. in schools or universities.

Finally, until now there is no common and established methodology on how to create argumentation systems. Consequently, the wheel is constantly reinvented. While in general software engineering developers are aware of the importance of documenting and reusing typical recurring problem solutions for future system designs, there are only few comparable approaches in e-learning and especially in the argumentation domain.

Suthers (2001), for instance, evaluated the usefulness of varying Model-View-Controller (MVC) concepts for data distribution and coupling in different versions of *Belvedere*. Comparing the centralized architecture (one server holds the model and all clients are tightly coupled to it) used in Belvedere v1 and a mixed replicated / duplicated architecture (a copy of the model is held on all clients and must be kept in sync at all time) used in Belvedere v2, he finally proposed a hybrid architecture, i.e. a model which is stored on the server as well as (in form of a duplicate) on the clients. This way, users are able to choose a view on the data which fits best on their needs without losing the possibility for collaboration with others that use a different view, i.e. a model-level coupling is used. Other existing software design approaches applicable to argumentation are either general software design patterns or mainly focused on *ITS* (Intelligent Tutoring System) design. Wenger (1987), for instance, proposed an architecture based on four software modules (expert, student, tutor and communication), and Harrer, Pinkwart, McLaren and Scheuer (2008) as well as Harrer and Devedzic (2002; Devedzic and Harrer 2005) identified recurring patterns in ITS. Examples for the latter are the *KnowledgeModel-View* pattern which manages multiple models and views (similar to the *MVC* pattern for one model and view) or the *ICSCL* pattern, which allows adapting learning materials separately for individuals and groups at the same time. Even though primarily designed for general-purpose ITSs, these patterns can be used for the specific task of developing argumentation (ITS) systems as well. Nevertheless there are – to our knowledge – no design patterns especially designed for argumentation systems.

REQUIREMENTS

In addition to the open challenges listed above, there are a couple of successfully applied practices in existing argumentation systems. In this section we present practices and propose a software architecture that is capable to support them on the technology level. All identified requirements are summarized in Table 1.

General

On the general side, a generic framework should be easily maintainable (→ Req. 1) to simplify application use in educational school

General	(1) Maintainability (2) Avoid installation and firewall problems on the client side (3) Flexibility & extensibility (4) Must scale up for a fair amount of users
Collaboration	(5) Support for synchronous and asynchronous collaboration (6) Users have to be aware of other users‘ actions (7) Communication via different channels: text, audio or video chats (8) Concurrency control to avoid the loss of data (9) Scripting support to define collaboration and learning settings (10) Definition of roles and rights
Analysis & Feedback	(11) Multiple analysis and feedback engines must be supported (12) Highlighting of elements to give feedback
Ontology	(13) Underlying ontology should be flexible, i.e. an ontology can be defined for each argumentation separately (14) Support to embed external resources (15) Micro-references to parts of resources should be supported
Visualization	(16) Multiple views on the data set, e.g. graphs or matrices
Logging	(17) Action-based logging (18) State-based logging (19) Support for replays

Table 1: Requirements of a general argumentation system

settings with no professional admin present. Especially on the client side, there should be no installation required to avoid conflicts with access rights or firewalls (→ Req. 2) as reported, for instance, in (Ravenscroft, McAlister and Sagar, 2009). Flexibility with respect to the integration of additional tools to model arguments or to analyze arguments should be supported (→ Req. 3). One way to do this could be plug-ins, i.e. the core module of the framework will be extended by external components which make use of a pre-defined interface to the core system (a similar approach is described for instance in (De Chiara *et al.*, 2010, in this book). This would result in a loose coupling of system components, i.e. all components can be added or removed on-the-fly. To allow for a fluent collaboration, the system must scale up also for a larger number of users (→ Req. 4).

Collaboration

As mentioned before, argumentation (and especially argumentation learning) often benefits from group discussions. Due to this fact, we classified the existing systems with respect to their support for collaboration. Here we found out that the support functions present in existing systems (which have been shown to be effective in different settings) vary. It may thus be beneficial to be able to switch between various collaboration settings (→ Req. 5) to fit the needs of the respective application scenario. Examples for different successful collaboration strategies are – on the one hand - *Academic Talk* and its successor *Interloc* (Ravenscroft et al., 2009) which have been used in a synchronous fashion in classroom, and – on the other hand – *HERMES* (Karacapilidis and Papadias, 2001), an asynchronous forum-like system that has been used to decide medical cases. Also, adequate awareness and communication support are required, i.e. each user must be made aware of the actions of others (→ Req. 6), and there should be communication facilities like text, audio or video chat (→ Req. 7), especially in settings where the participants are in different places and cannot talk to their partners directly. Connected to this point is a sophisticated concurrency control, i.e. parallel actions from different users must be processed avoiding data loss ensuring consistency. An acceptable solution should also avoid locks, which could cause frustration among learners which are not able to work on argument parts when another one is working on the same part (→ Req. 8). To improve the learning effects, it should be possible to construct typical argumentation scenarios - e.g., simulated dialectic arguments in a courtroom setting may be more effective than argumentation exercises without this simulated setting. These scenarios could be specified by means of scripts (Suthers, Toth and Weiner, 1997; Kobbe *et al.*, 2007); written for instance in IMS-LD[1] (→ Req. 9) as done for instance in CoFFEE (Belgiorno, De Chiara, Manno and Scarano, 2008). To implement these scripts, one should be able to assign roles and rights (→ Req. 10) to different groups of users as is possible for instance in *Digalo*. To extend the trial example: Imagine one group acting as plaintiff, while another group acts as defendant. These roles could be emphasized by means of different rights, e.g. each group is only able to manipulate their own arguments.

Analysis & Feedback

On the ITS side, a general framework should provide support for integrating multiple analysis techniques, including machine learning techniques as well as rule or grammar based approaches and peer-to-peer reviewing approaches, to face the *ill-definedness* (Lynch, Ashley, Aleven and Pinkwart, 2006) of argumentation which may require advanced techniques to analyze arguments and give feedback. Machine learning techniques can try to identify possible lacks in argumentation based on pattern learned from earlier experiences (De Groot et al., 2007; McLaren

[1] http://www.imsglobal.org/learningdesign/

et al., 2010). Grammar based approaches are able to analyze and compare the structure of the argument to pre-defined rules (Suthers *et al.*, 1997; Pinkwart et al., 2006). An example here may be a circular argument that should be avoided. In peer-to-peer reviewing approaches, the quality of a part of the argument is evaluated by other users working on a similar part of the argument (Pinkwart *et al.*, 2006; Loll and Pinkwart, 2009). While these methods have been shown to be effective on their own, a combination of multiple techniques may be even more effective (→ Req. 11). Of course, the results of these methods must be shown to the users in an adequate way, e.g. by highlighting the elements under critique (→ Req. 12) (De Groot et al., 2007; McLaren et al., 2010).

Ontology

To avoid a restriction of the framework to pre-defined domains, the underlying ontology must be flexible, i.e. the framework must allow for different configurations (pre-defined like Toulmin (1958) or Wigmore (1931) as well as customized ones) for multiple argumentation domains (→ Req. 13) as possible, e.g., in *Digalo*. This approach should be beneficial compared to other approaches that try to achieve universality or expressiveness by a large set of elements since it avoids overwhelming the user with a "plethora of choices"(Suthers, 2003, p. 8) as Suthers noted during the iterative refinement of the *Belvedere* system, which comes with a more detailed ontology in the first versions than present in the latest one.

As part of the ontology, embedding links to external resources into arguments (→ Req. 14) (done for example in *Belvedere*) should be allowed, including micro-references to parts of it (→ Req. 15). An example here is an article on the web or an inline transcript of a trial which could be linked line-wise to argument elements (as used in *LARGO*, for example). Based on these ontologies, multiple visualizations like graphs, matrices, frames or linear and threaded text are imaginable and should be supported (→ Req. 16). These different visualizations may be beneficial in different situations to improve the argumentation. Suthers (2003) for instance, showed that the use of different visualizations would scaffold different actions. While a graph-style visualization could be beneficial to get a common understanding of the problem, a matrix, for instance, highlights missing relation.

Logging

Another important factor – for researchers as well as for teachers and tutors – is the support of adequate logging mechanisms. Here action-based (→ Req. 17) and state-based logging (→ Req. 18) should be supported. While the former is beneficial for replay functions, e.g., when a tutor tries to reconstruct an argumentation step-wise, (→ Req. 19) to give feedback to the learners (as done for instance by means of the Common Format in *Digalo* and *ARGUNAUT*), the latter is important for performance reasons: when a new user joins an ongoing argumentation, it is beneficial for the overall system performance not to provide him with all single actions – instead, he or she should receive the current document state immediately to avoid unnecessary processing steps on the client.

THE LASAD ARCHITECTURE

Based on the challenges and requirements identified above, we propose the architecture shown in Figure 5 as the foundation of our LASAD framework. It uses a classic layered architecture, i.e. the software is structured into layers where each layer is only capable of communicating with its neighbor layers. The main advantage is that each layer works more or less independently from the others. The communication takes place via interfaces that enable a transparent use of the whole layer. Hence, the internal structure of the layers can be easily exchanged. That is, the whole system is loosely coupled. The (exchangeable) technologies currently used in the framework are marked with a star *.

CLIENT LAYER

On the client side, different types of applications are possible. On the one hand, there is the user client (UC). It provides a graphical interface for each user to create and manipulate arguments as well as communication tools (→ Req. 7: Communication via different channels: text, audio or video chats). The graphical interface comprises different views (→ Req. 16: Multiple views on the data set, e.g. graphs or matrices), for instance, a graph, forum or matrix visualization, presenting the same underlying data. Thus, the user client is the main tool to interact with (a) other users and (b) the system.

On the other hand, there are analysis & feedback clients (AFCs). Their main purpose is to automatically analyze the arguments created by the learners. The analysis can be done by multiple clients (→ Req. 11: Multiple analysis and feedback engines must be supported) with different methods (cf. (Scheuer, McLaren, Loll and Pinkwart, 2010) in this book) at the same time. Based on this

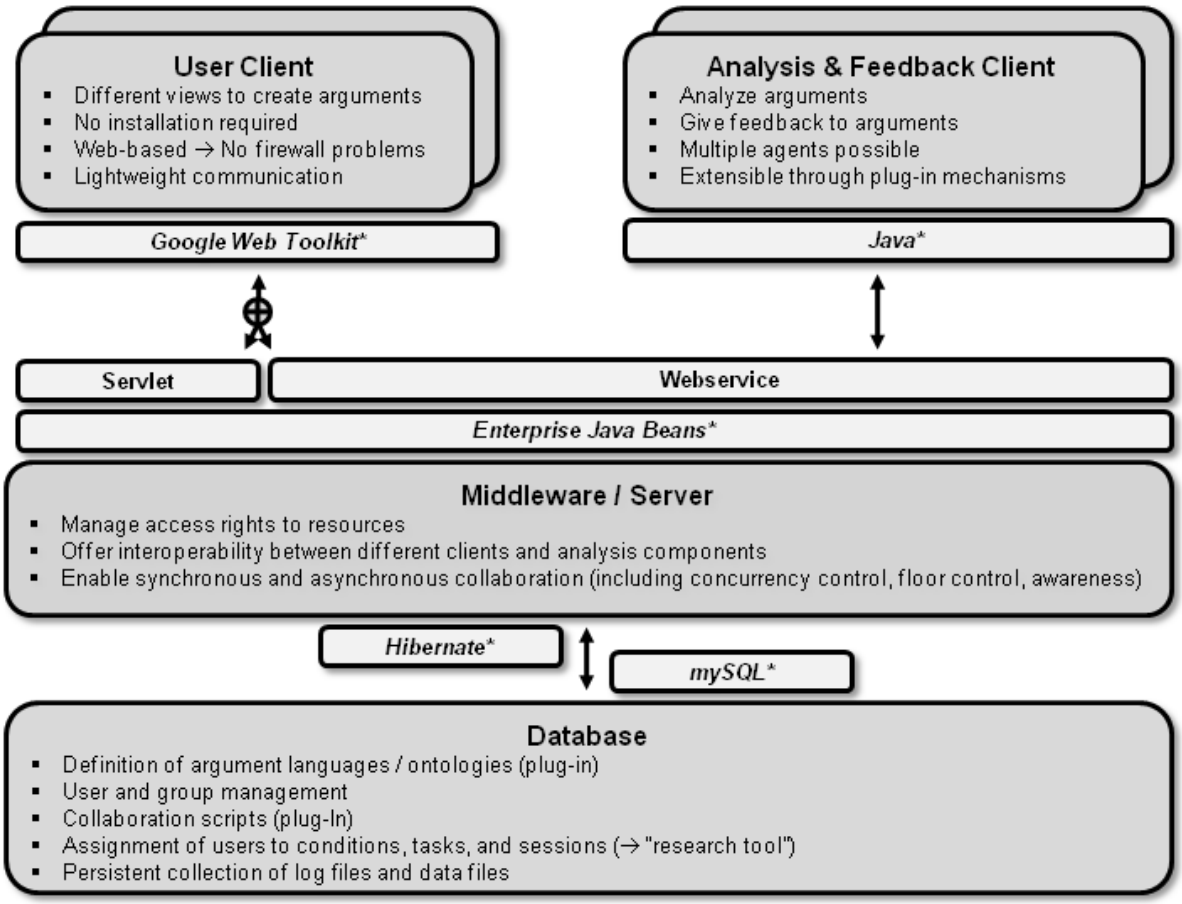

Figure 5: The LASAD Framework - Architecture

analysis, the AFCs give feedback to the learners or to a teacher or moderator to either highlight possible weaknesses of the created argument (→ Req. 12: Highlighting of elements to give feedback) or to assist the tutor to help learners. To communicate with the UCs, the AFCs are provided with the same technical interface as the UCs. The server differentiated between AFCs and UCs via different roles and rights for different clients (→ Req. 10: Definition of roles and rights, see below).

To avoid possible firewall and installation problems (→ Req. 2: Avoid installation and firewall problems on the client side), the clients can be web-based. Our prototype client for instance uses Google Web Toolkit (GWT) which provides a Java-to-JavaScript compiler. Thus, it is possible to use a high-level programming language (Java in our case) including their established development tools that accelerates the whole argumentation process and, at the same time, to benefit from the possibilities of a scripting language like JavaScript. By means of JavaScript it is possible to run the whole application in a completely platform independent way in a web browser. This eliminates installation requirements, since all modern web browsers support JavaScript.

SERVER LAYER

Following the established layer architecture design pattern, all data processed by the clients is sent to the server layer (Figure 6, step 1). Here, multiple checks are performed before the client gets notified whether the action is allowed or not, and the data is processed to the data layer as well as distributed to all other client with adequate awareness information (→ Req. 6: Users have to be aware of other users' actions) to enable collaboration (→ Req. 5: Support for synchronous and asynchronous collaboration). The checks comprise (a) the concurrency control (→ Req. 8: Concurrency control to avoid the loss of data) and (b) the access control (→ Req. 10: Definition of roles and rights). During the

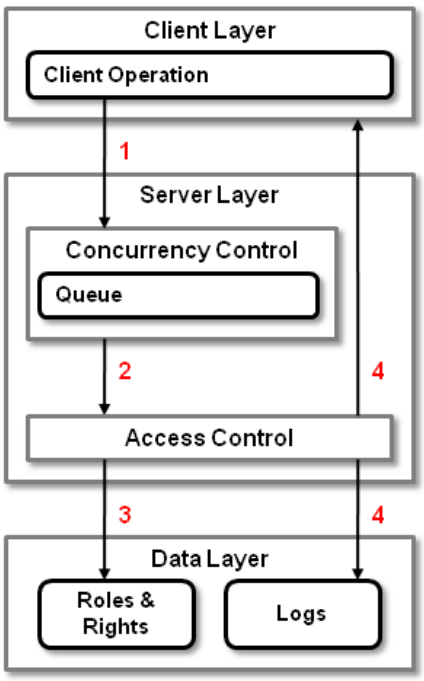

Figure 6: Server processing

concurrency control, the action is stored in a queue with all other incoming actions to guarantee the processing of actions in a consistent manner without data loss.

Once an element of the list passes through to the next step (Figure 6, step 2), a check is done whether the user is allowed to do the action, i.e. the access control takes place by verifying the user rights with help of the data layer (Figure 6, step 3). For instance, a user of group A may want to delete an argument stated by group B. This may be forbidden in the rights management of the corresponding group. Since the application logic is located on the server side, the client can send a command requesting to delete the box, but this request will then be denied by the server. Otherwise, if the action is allowed, it will be confirmed and stored persistently in the data layer (Figure 6, step 4).

In addition to these control mechanisms, the server acts as mediator between the data and the client layer, i.e. all information stored in the data layer is only accessible through the server.

DATA LAYER

The key to achieving flexibility is the data layer. Here, one is able to configure the whole platform to fulfill domain-specific needs. The configuration consists of three parts: (1) The definition of roles and rights (→ Req. 10), (2) the definition of collaboration and learning scripts (→ Req. 9: Scripting support to define collaboration and learning settings), and (3) the definition of the underlying argumentation ontology (→ Req. 13: Underlying ontology should be flexible, i.e. an ontology can be defined for each argumentation separately). In the first step, different user roles will be specified. Typical roles in general educational argumentation are learner, teacher or moderator. In more specific argumentation scenarios like the law, other roles are possible, e.g. defendant and plaintiff. After defining different roles, there is the possibility of assigning different rights to different user groups: While learners are able to create and manipulate the argument structure, it might be beneficial if a moderator is also allowed to highlight parts of the argument to guide further argumentation (De Groot *et al.*, 2007). A similar situation is possible for the AFCs (each AFC belongs to a user group as well). This way it is possible to define more or less complex scenarios. On the one hand, there may be a scenario with the roles student and teacher. Here all students will have the same role and rights, i.e. each participant is able to add elements to the argument, while only the teacher is able to highlight elements to scaffold the discussion. On the other hand, even more complex scenarios are possible: There may be two parties, one pro and one con for the discussed question (for instance: "Should taxes be reduced to increase the economic growth?"). Here one could define two different roles so that each party is only able to edit its own contributions, and not the contributions of the other party.

Also multiple AFCs may be used, each of them with different rights: Whereas one automatic analyzer may only be able to give hints via highlighting of elements, another may have the additional right to delete rebutted points. To define these scenarios, in more detail, scripts can be specified. Via scripts it is possible to define and guide the whole argumentation process. For instance, it is possible to define different phases, like brainstorming, argument building and argument discussion. These different phases can be supported with different ontologies (see below) and collaboration settings. As shown in Figure 7, an early brainstorming phase, for instance, may use synchronous collaboration in connection with a graph-based visualization of only one node and one relation type. In the next argumentation phase, there might be asynchronous collaboration with an ontology that supports different types of nodes and relations to structure the argumentation in more detail, based on the results from the brainstorming session. Even though the support of different collaboration

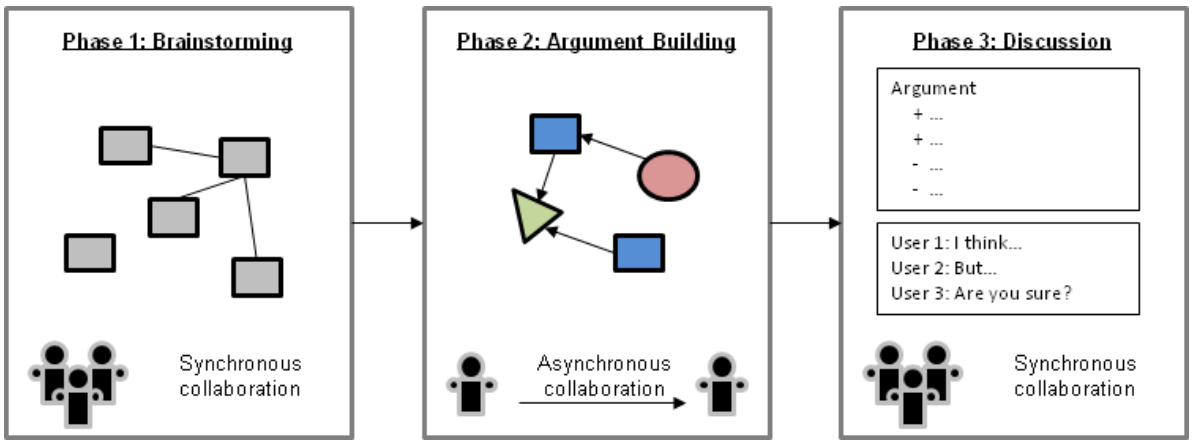

Figure 7: Possible scenario defined by scripts

styles is independent from the graphical representation, there will be different manners of support needed for synchronous and asynchronous collaborative system usage. While in asynchronous collaboration it will be enough information to know who created which element and when he or she did it, it may be beneficial for synchronous collaboration to provide additional information, for instance, who is currently working at which part of the argument. Finally, in the discussion phase, the arguments may be shown as list and a chat window will serve as primary communication channel. Together with the roles and rights specified before, a variety of other settings are possible. For instance there can be two parties which are arguing against each other, while each user has access to different information that is designed to help him or her argue. Thus, it will be possible, e.g., to simulate a trial in the legal domain or scientific argumentation, for instance when an observable phenomenon can be explained by different theories.

Achieving flexibility is largely a function of the underlying argumentation ontology. An ontology, i.e. the explicit specification of a conceptualization (Gruber, 1993), provides the foundation of an argumentation system. Here one may differentiate between systems that make their ontology explicit and others which provide an implicit ontology. It describes the available elements, including their contributions, relations, possible modifiers and other components such as given texts etc. to create an argument. Typical examples of contributions are *hypotheses* or *evidences*. Those can be connected by means of relations like *pro* or *con*. In addition, modifiers like *believability* or *relevance scores* can be added to both contributions and relations - these are used to analyze the conclusiveness of an argument (for instance by an AI engine, cf. (Scheuer *et al.*, 2010) in this book). An important point is that different argumentation domains require different ontologies to create meaningful arguments. At the same time, the ontology's aim is to make the users of the argument system aware of the conceptual components of the task domain, i.e., an ontology may guide users (Suthers, 2003). Another part of the ontological specification is the possibility to add external resources such as text on web pages or external applets (→ Req. 14: Support to embed external resources). These external references may be used in an argumentation process, either by having learners point to the whole resource or by just referencing a part of it (→ Req. 15: Micro-references to parts of resources should be supported). We will provide an example of how to configure an ontology in the *LASAD* framework in the next section.

Apart from the definition of different settings, the data layer is responsible for the consistent and persistent storage of the whole data resulting from the argumentation, including user actions, the argument structure and additional meta-data like creation date or a user assessment. Here, two types of logging are done in parallel: (1) state-based logging (→ Req. 18) and (2) action-based logging (→ Req. 17). During the state-based logging, all incoming actions are applied to the current revision of the argument. Once a new client connects to the argumentation, only the current state needs to be transferred. Compared to action-based logging, this results in improved network performance. The action-based logging, however, stores all single actions separately. This is beneficial because one may want to undo a step or to replay an entire argument stepwise (→ Req. 19: Support for replays), which is especially important for teachers and researchers who want to understand how and why the argument evolves over time. If one used state-based logging here, it could result in poor performance since the whole argument would have to be sent to all participants every time an action occurs. Together with the concept of the layered architecture and open interfaces to plug in new components (→ Req. 3: Flexibility & extensibility), the framework

scales up well (→ Req. 4: Must scale up for a fair amount of users) and at the same time is easily maintainable (→ Req. 1: Maintainability), because all components are independent from each other.

ONTOLOGY CONFIGURATION

After the underlying architecture of the framework has been described, we will now discuss how the configuration mechanisms of the *LASAD* framework work in detail. In this section we focus on the configuration of the ontology. For this purpose, we rebuilt the argument modeling part of the *Belvedere* system (or, more specifically, the evidence mode of this tool) as well as the argument modeling part of *LARGO* system by means of ontology configurations of the *LASAD* system. Illustrative parts of the configuration are shown in Listing 1 (see Appendix A) and Listing 2 (see Appendix B). An overview of all (currently) available XML tags to define the ontology is given in Table 2. Please note that we did not rebuilt the analysis and feedback parts of the systems yet, even though it would be possible based on the *LASAD* architecture.

As mentioned before, the *Belvedere* ontology, on the one hand, consists of three contribution types (*data*, *hypothesis*, and *unspecified*) and three relations (*pro*, *con*, and *unspecified*). Each contribution and relation comprises as child elements a text, a URL, notes, the author's name, the name of the modifier, and a strength modifier.

The *LARGO* ontology, on the other hand, consists of three contributions (*hypothetical*, *test*, and *fact*) and five relations (*modified to*, *distinguished from*, *analogized to*, *leads to*, and *general*). Compared to *Belvedere*, the contributions comprise different child elements. While a *fact* only has a simple text area, a *hypothetical* has an optional labeled text field (*outcome*, see Figures 2 and 4) as well. Even more detailed is the *test*, since it comprises at least the labeled *if* and *then* text fields, but may also include a set of other labeled fields, e.g. *and* or *even though*. The relations, however, do not have any child-elements or at most comprise a text area (see *comment* relation in Figure 1 and 3).

To map these ontologies into an XML configuration of the *LASAD* framework, the following structure is used: Each ontology has a root tag <*ontology*> which defines the name (*type*) of the ontology, i.e. in these cases "*Belvedere*" and "*LARGO*". Inside of this, the elements (contributions, relations, and - in the *LARGO* case – a transcript) are defined (<*element*>). Each element has got a *type* and an *id*. While the former defines via keywords (e.g., *contribution, relation, transcript, tutorial*) what the client is expected to show, the later uses keywords to tell an AFC to which category this may belong. For instance, a set of contributions is defined. To differentiate between multiple contributions, each contribution has a unique name. This way an AFC is able to differentiate between them. Typically, the *id* would be the name that can be found in the element's label, but other names are possible too.

Each element has additional options (<*options*>) and style information (<*uisettings*>) defined by different attributes. Inside of the element options, additional information is given, including the name of the element (*heading*). Within the user interface settings *(<uisettings>),* preferred style settings are defined including the element's colors (*background-color, font-color*), the element's size (*width, height*) and whether it should be resizable (*resizable*), and its border *(border,* possible properties here are, e.g. standard, dashed, round, etc.). The definition of the relations is similar to the definition of the contribution. In addition to the contributions, a relation has further style information attributes such as *line-color, line-width* and *directed*, which define the appearance of the relation in more detail. All user interface attributes are optional, i.e. if there is no attribute specified, the framework will use a standard setting. Finally, each top level element, i.e. relation, contribution and transcript, may have child elements. Typical children are textboxes with and without labels, hyperlinks to external resources, awareness

Tag	Properties	Parent	Function
<ontology>	type	-	The root element
<element>	id contribution quantity min-quantity max-quantity	<ontology>	Defines an ontology element. Examples are: contribution, relation, and transcript
<childelements>	-	<element>	Container to store elements that belong to one parent element
<uisettings>	width height min-height max-height resizable border background-color font-color line-width line-color	<element>	Provides additional information for the clients' visualization; optional
<options>	label texttype score min-score max-score	<element>	Provides additional information for an element like, for instance, the value which is set on start up (for a rating element) or if the text-container has multiple lines or not

Table 2: Overview of XML tags to define an ontology in LASAD

information panels, or rating elements. In addition to the top-level elements, each child has a *quantity*, which defines how many instances of the element are present when its parent is created. This *quantity* can be changed during runtime so that the overall number of instances is between the *min-quantity* and the *max-quantity*. An example for these quantities is the *test* contribution in the LARGO ontology. Here, there is only an *if* and a *then* text field (defined via *quantity="1"* in the ontology, cf. Listing 2 in Appendix B). If one would like to extend the box with an *and* or an *even though* text field, this could be done during runtime (by clicking on the plus button in the header of the contribution). Then it is checked whether the *max-quantity* is already obtained and the *quantity* gets updated. Compared to the *Belvedere* ontology, the *LARGO* ontology does have an additional element: the transcript, which is defined in analogy to the other relation and contribution elements. The transcript is specified analogously to the other elements and its concrete content, i.e. the lines which are readable in the transcript can be defined in a concrete instance of a map using this ontology (see Figure 6 and 7, left side). To allow the linking between parts of the transcript with a contribution, each contribution must have another child-element of the type *transcript-link* (see Listing 2 in Appendix B for details). An example for such a link can be found in Figures 6 and 7 in the lower left *test* contribution.

Based on these definitions each client is able to work on the data. On the side of the AFCs, the ontology data can be analyzed to reveal possible weaknesses in learner's arguments using pre-defined rules or machine learning techniques. For example, an AFC may know that an unconnected item is not helpful for the argumentation process or that a hypothesis object must be supported or rebutted by a data object. This is analog to one of the Belvedere coaches (Suthers, 2003), which examines the structure of the argument based on general rules like "*multiple lines of evidence converging on a hypothesis is better than one consistent datum*"(Suthers, 2003, p. 4). The second Belvedere coach, which compares the argument created by a learner to an argument

of an expert, could also be implemented by means of another AFC. On the side of the human user client, however, the data is used as basis for the visualization. Here it is important to know that each client that works on the data is allowed to have its own visualization or even multiple visualizations to choose from at runtime. While our Google Web Toolkit client makes use of a graph-style visualization (see Figures 5 and 7), another client may use a threaded discussion visualization. Here, it is important to know that the use of multiple visualizations may result in pieces of information which are hidden. An example for such a case is the presentation of a cyclic argument structure (which is easy to represent in a graph) in a threaded discussion (where cycles cannot be expressed): the threaded discussion visualization should make the users aware of the fact that there is additional information available which could not be shown in their current visualization. The key for exchangeability and cooperation is the common ontology, i.e. all connected clients provide their users with the same elements available but may differ in their visualization. Our ontology definition contains visualization information to some degree, but this can be ignored by clients which use a visualization that does not support this style information.

CONCLUSION

This chapter highlights the importance of tools to assists teachers in teaching argumentation. We summarized the basic approaches of modern argumentation systems, including their strengths and weaknesses. Based on a review of 49 existing argumentation systems and methods we collected technology requirements for a generic argumentation system and proposed an architecture which is able to deal with all the identified requirements. As a next step, we described the *LASAD* framework, which is based on the proposed architecture. The framework can be used to simplify the development of new argumentation tools by means of detailed and flexible configuration mechanisms. We exemplified this point by configuring the *LASAD* tool to emulate parts of the argumentation systems *Belvedere* and *LARGO*.

In future work, we plan to develop support for additional visualization styles on the client layer and improve the XML configuration mechanisms by separating more clearly between ontology and visualization parts that can be reused in different contexts. To simplify the creation of the XML ontology, we also plan to develop an authoring tool, which guides the system's users through the creation of an ontology.

ACKNOWLEDGEMENTS

This work is supported by the German Research Foundation (DFG) under the grant "Learning to Argue: Generalized Support Across Domains" (*LASAD*).

REFERENCES

Belgiorno, F., De Chiara, R, Manno, I. and Scarano, V. (2008). A flexible and tailorable architecture for scripts in F2F collaboration. *Times of Convergence. Technologies Across Learning Contexts, LNCS, Proceedings of the 3rd European Conference on Technology Enhanced Learning*.

Bell, P. (1997). Using argument representations to make thinking visible for individuals and groups. In R. Hall, N. Miyake, N. Enyedy (Eds.), *Proceedings of the 2nd International Conference on Computer Support for Collaborative Learning* (pp. 10-19).

Bell, P. and Linn, M. C. (2000). Scientific arguments as learning artifacts: Designing for learning from the web with KIE. *International Journal of Science Education*, *22*(8), 797-817.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, *13*(6), 3-16.

De Chiara, R., Manno, I. and Scarano, V. (2010). CoFFEE: an expandable and rich platform for computer-mediated, face-to-face argumentation in classroom. In N. Pinkwart, B. M. McLaren (Eds.) *Educational Technologies for Teaching Argumentation Skills*. Bentham.

De Groot, R., Drachman, R., Hever, R., Schwarz, B., Hoppe, U., Harrer, A., De Laat, M., Wegerif, R., McLaren, B. M. and Baurens, B. (2007). Computer supported moderation of e-discussions: the ARGUNAUT approach. C. Chinn , G. Erkens, S. Puntambekar (Eds.), *Proceedings of the 8th International Conference on Computer-Supported Collaborative Learning* (pp. 165-167).

Devedzic, V. and Harrer, A. (2005). Software patterns in ITS architectures. *Intl. Journal of AI in Education*, *15*(2), 63-95.

Dimitracopoulou, A. (2005). Designing collaborative learning systems: current trends & future research agenda. *Proceedings of the 2005 Conference on Computer Support for Collaborative Learning* (pp. 115-124).

Gardner, A. (1987). *An Artificial Intelligence Approach to Legal Reasoning*. Cambridge, MA: MIT Press.

Gruber, T. R. (1993). A translation approach to portable ontology specification. *Knowledge Acquisition, 5*(2), 199-220.

Harrer A and Devedzic V (2002). Design and analysis patterns in ITS architectures. *Proceedings of the Intl. Conf. on Computers in Education*, (pp. 523-527).

Harrer, A., Pinkwart, N., McLaren, B.M. and Scheuer, O. (2008). The scalable adapter design pattern: Enabling interoperability between educational software tools. *IEEE Transactions on Learning Technologies, 1*(2), 131-143.

Karacapilidis, N. and Papadias, D. (2001). Computer supported argumentation and collaborative decision making: the Hermes system. *Information Systems, 26(*4), 259-277.

Kirschner P. A., Buckingham Shum S. J. and Carr C. S. (2003). *Visualizing argumentation. Software tools for collaborative and educational sense-making*. London: Springer.

Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämäläinen, R., Häkkinen, P. and Fischer, F. (2007). Specifying computer-supported collaboration scripts. *International Journal of Computer-Supported Collaborative Learning, 2*(2). 211-224.

Kuhn, D. (1991). *The skills of argument*. Cambridge, Cambridge

Kulik C. C. and Kulik J. A. (1991). Effectiveness of computer-based instruction: An updated analysis. *Computers in Human Behavior, 7*, 75-95.

Loll, F. and Pinkwart, N. (2009). Using collaborative filtering Algorithms as eLearning Tools. In R. H. Sprague (Ed.), *Proceedings of the 42nd Hawaii International Conference on System Sciences*. Los Alamitos, CA: IEEE Computer Society Press.

Lonchamp, J. (2006). Supporting synchronous collaborative learning: a generic, multi-dimensional model. *International Journal of Computer-Supported Collaborative Learning, 2*(1), 247-276, Springer.

Lynch, C., Ashley, K. D., Aleven, V. and Pinkwart, N. (2006). Defining ill-defined domains: A literature survey. In V. Aleven, K. D. Ashley, C. Lynch, N. Pinkwart (Eds.) *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 8th Intl. Conf. on Intelligent Tutoring Systems* (pp. 1-10). Jhongli, Taiwan: National Central University.

McAlister, S., Ravenscroft, A. and Scanlon, E. (2004). Combining interaction and context design to support collaborative argumentation using a tool for synchronous CMC. *Journal of Computer Assisted Learning: Special Issue: Developing Dialogue for Learning, 20*(3), 194-204.

McLaren, B. M., Scheuer, O. and Mikšátko, J. (2010). Supporting collaborative learning and e-discussions using artificial intelligence techniques. *Intl. Journal of Artificial Intelligence in Education, 20*(1), 1-46.

Pinkwart, N., Aleven, V., Ashley, K. D. and Lynch, C. (2006). Toward legal argument instruction with graph grammars and collaborative filtering techniques. *Lecture Notes in Computer Science Vol. 4053* (pp. 227-236). Berlin: Springer.

Ravenscroft, A., McAlister, S. and Sagar, M. (2009). Digital dialogue games: *JISC Final Project Report London Metropolitan University, Learning Technology Research Institute.*

Reed, C. and Rowe, G. (2004). Araucaria: Software for argument analysis, diagramming and representation. *International Journal of AI Tools, 14*(3-4), 961-980.

Rolf, B. and Magnusson, C. (2002). Developing the art of argumentation. A software approach. *Proceedings of the 5th Intl. Conf. on Argumentation. Intl. Soc. for the Study of Argumentation.*

Schank, P. and Ranney, M. (1995). Improved reasoning with Convince Me. In *Human Factors in Computing Systems CHI'95 Conf. Companion* (pp. 276-277). New York: Association for Computing Machinery.

Scheuer, O., Loll, F., Pinkwart, N. and McLaren, B. M. (2010). Computer-supported argumentation: A review of the state-of-the-art. *International Journal on Computer Supported Collaborative Learning, 5*(1), 43-102. Springer.

Scheuer, O., McLaren, B. M., Loll, F and Pinkwart, N. (2010). Automated analysis and feedback techniques to support argumentation: a survey. In N. Pinkwart, B. M. McLaren (Eds.), *Educational Technologies for Teaching Argumentation Skills*. Bentram.

Schwarz, B. B. and Glassner, A. (2007). The role of floor control and of ontology in argumentative activities with discussion-based tools. *International Journal of Computer-Supported Collaborative Learning, 2*(4), 449–478. Springer.

Siegel, M. A. (1999). Changes in student decisions with Convince Me: Using evidence and making tradeoffs. *Proceedings of the 21st Annual Conf. of the Cognitive Science Soc.* (pp. 671-676). Mahwah: Erlbaum.

Slagter, R., Biemans, M. and Ter Hofte, H. (2001). Evolution in use of groupware: Facilitating tailoring to the extreme. *Proceedings of the 7th International Workshop on Groupware*.

Suthers, D. D. (2001). Architectures for computer supported collaborative learning. *Proceedings of the IEEE Intl. Conf. on Advanced Learning Technologies*.

Suthers, D. D. (2003). Representational guidance for collaborative inquiry. In J. Andriessen, M. Baker, D. D. Suthers (Eds.), *Arguing to Learn, Computer-Support Collaborative Learning Series, Vol. 1 (*pp. 27-46). Springer.

Suthers, D. D., Toth, E. E. and Weiner, A. (1997). An integrated approach to implementing collaborative inquiry in the classroom. *Proceedings of the 2nd International Conference on Computer Support for Collaborative Learning* (pp. 272-279).

Suthers, D. D., Weiner, A., Connelly, J. and Paolucci, M. (1995). Belvedere: Engaging students in critical discussion of science and public policy issues. *Proceedings of the 7th World Conference on Artificial Intelligence in Education* (pp. 266-273).

Toulmin, S. E. (1958). *The Uses of Argument*.

Van Gelder, T. (2003). Enhancing deliberation through computer supported argument mapping. In P. A. Kirschner, S. J. Buckingham Shum, C. S. Carr (Eds.) *Visualizing argumentation. Software tools for collaborative and educational sense-making* (pp. 97-115). London: Springer.

Weinberger, A., Fischer, F. and Stegmann, K. (2005). Computer-supported collaborative learning in higher education: Scripts for argumentative knowledge construction in distributed groups. *Proceedings of CSCL 2005.*

Wenger, E. (1987). Artificial intelligence and tutoring systems. Los Altos: Morgan Kaufmann.

Wigmore, J. H. (1931). *The Principles of Judicial Proof* (2nd Edition). Little, Brown & Co.

APPENDIX A

```
<ontology type="Belvedere">

  <element id="data" type="contribution" quantity="" min-quantity="" max-quantity="">
    <options heading="Data" />
    <uisettings width="200" height="250" resizable="true" border="standard" background-
      color="#C4FFC4" font-color="#000000" />
    <childelements>
      <element id="text" type="text" quantity="1" min-quantity="1" max-quantity="1">
        <options texttype="textarea" />
        <uisettings background-color="#FFFFFF" font-color="#000000" min-height="40"/>
      </element>
      <element id="notes" type="text" quantity="1" min-quantity="1" max-quantity="1">
        <options texttype="textarea" label="Notes" />
        <uisettings background-color="#FFFFFF" font-color="#000000" min-height="40"/>
      </element>
      <element id="externalreference" type="url" quantity="0" min-quantity="0" max-
          quantity="1">
        <options />
        <uisettings background-color="#FFFFFF" font-color="#CCCCFF" min-height="16"/>
      </element>
      <element id="strength" type="rating" quantity="1" min-quantity="1" max-quantity="1">
        <options score="3" min-score="1" max-score="5" />
        <uisettings background-color="#FFFFFF" font-color="#000000" min-height="16"/>
      </element>
      <element id="awareness" type="awareness" quantity="1" min-quantity="1" max-quantity="1">
        <options />
        <uisettings background-color="#FFFFFF" font-color="#A9A9A9" min-height="16"/>
      </element>
    </childelements>
  </element>

…

<element id="for" type="relation" quantity="" min-quantity="" max-quantity="">
  <options heading="For" endings="true" />
  <uisettings width="100" height="120" resizable="false" border="" background-color="#5FC977"
     font-color="#000000" line-width="2px" line-color="#5FC977" />
  <childelements>
     <element id="strength" type="rating" quantity="1" min-quantity="1" max-quantity="1">
       <options score="3" min-score="1" max-score="5" />
       <uisettings background-color="#FFFFFF" font-color="#000000" min-height="16"/>
     </element>
     <element id="awareness" type="awareness" quantity="1" min-quantity="1" max-quantity="1">
       <options />
       <uisettings background-color="#FFFFFF" font-color="#A9A9A9" min-height="16"/>
     </element>
  </childelements>
</element>

…

</ontology>
```

Listing 1: Parts of the XML definition of the Belvedere ontology in LASAD

APPENDIX B

```
<ontology type="LARGO">
…
  <element id="test" type="contribution" quantity="" min-quantity="" max-quantity="">
    <options heading="Test" />
    <uisettings width="180" height="160" resizable="true" border="standard" background-
          color="#55C3FF" font-color="#000000" />
    <childelements>
      <element id="if" type="text" quantity="1" min-quantity="1" max-quantity="1">
        <options texttype="textfield" label="IF" />
        <uisettings background-color="#FFFFFF" font-color="#000000" min-height="16"/>
      </element>
      <element id="and" type="text" quantity="0" min-quantity="0" max-quantity="5">
        <options texttype="textfield" label="AND" />
        <uisettings background-color="#FFFFFF" font-color="#000000" min-height="16"/>
      </element>
      <element id="eventhough" type="text" quantity="0" min-quantity="0" max-quantity="5">
        <options texttype="textfield" label="EVEN THOUGH" />
        <uisettings background-color="#FFFFFF" font-color="#000000" min-height="16"/>
      </element>
      <element id="then" type="text" quantity="1" min-quantity="1" max-quantity="1">
        <options texttype="textfield" label="THEN" />
        <uisettings background-color="#FFFFFF" font-color="#000000" min-height="16"/>
      </element>
      <element id="transcriptlink" type="transcript-link" quantity="0" min-quantity="0" max-
            quantity="1">
        <options />
        <uisettings  min-height="16" max-height="16"/>
      </element>
    </childelements>
  </element>

  …

  <element id="transcript" type="transcript" quantity="1" min-quantity="1" max-quantity="1">
    <options />
    <uisettings width="" height="" resizable="true" border="" background-color="#FFFFFF" font-
          color="#000000" />
  </element>

…
</ontology>

<maptemplate ontology="LARGO" title="Carney - Petitioner">
    <mapdetails>
       <description> … </description>
       <options> … </options>
       <transcript>
         <lines>
            <line number="3" text="CALIFORNIA, Petitioner, v. CHARLES B. CARNEY, RESPONDENT">
                …
         </lines>
       </transcript>
    </mapdetails>
</maptemplate>
```

Listing 2: Parts of the XML definition of the LARGO ontology and parts of the concrete map in LASAD