Ein generisches Framework zur Erstellung von argumentationsunterstützenden Systemen

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

[1]Institut für Informatik, Technische Universität Clausthal
[2]Deutsches Forschungszentrum für Künstliche Intelligenz, Universität des Saarlands

1 Einleitung

Argumentationsfähigkeiten spielen in vielen beruflichen Kontexten eine zentrale Rolle – z. B. bei der Arbeit und Entscheidungsfindung in Teams, der Leitung von Arbeitsgruppen, der Kommunikation von Entscheidungen oder bei Verhandlungen mit Geschäftspartnern. In speziellen Anwendungsgebieten, wie etwa vor Gericht, in der Wissenschaft oder in der Politik, gehören sie zu den beruflichen Kernkompetenzen. Einige Forscher gehen sogar soweit, Argumentationsfertigkeiten als entscheidend für das eigentliche *Denken* anzusehen, das Menschen darin unterstützt, zu rationalen Entscheidungen zu kommen. (Kuhn 1991).

Leider sind die Argumentationsfähigkeiten vieler Menschen im Vergleich zur Praxisrelevanz dieser Fähigkeiten nur recht schwach ausgeprägt. Studien zeigen, dass ein Großteil der Bevölkung (auch aus gebildeten Schichten) Probleme dabei hat, Behauptungen von Argumenten zu unterscheiden, Angriffe auf ihre eigenen Argumente zu erkennen und angemessen darauf zu reagieren (Tannen 1998, Easterday et al. 2009). Folgerichtig nimmt die Ausbildung von Argumentationsfähigkeiten eine zentrale Rolle in der Lehre ein (Andriessen 2006) – sowohl im schulischen Klassenzimmer und der Berufsbildung als auch bei der professionellen Weiterbildung, z. B. im Bereich der Verhandlungsführung und Teamleitung.

Beim klassischen Lehransatz zur Schulung dieser Fähigkeiten unterrichtet ein Dozent eine Gruppe durch einen direkten persönlichen Dialog. Manchmal besteht sogar ein 1:1 Verhältnis zwischen Lehrendem und Lernendem, was sich als effektivste Form herauskristallisiert hat (Bloom 1984; Kulik & Kulik 1991). Wenngleich dieser Ansatz im Allgemeinen überzeugt, so birgt er Probleme: er ist nicht ohne weiteres auf große Gruppen übertragbar, da entsprechendes Personal und deren Zeit begrenzt ist.

2 Argumentationssysteme

An dieser Stelle setzen computergestützte Argumentationssysteme an, die versuchen, die schlechte Skalierbarkeit des *face-to-face* Ansatzes zu umgehen, indem sie versuchen, Argumentationsfähigkeiten durch Softwareunterstützung auszubilden. Hierbei wurden in den letzten Jahren verschiedene Ansätze verfolgt: Einerseits wurden Programme entwickelt, die lediglich dazu dienen, Argumentationsstrukturen in unterschiedlichen Visualisierungen (z. B. als Graph, Matrix oder als strukturierte Diskussion ähnlich einem Forum) abzubilden (Kirschner et al. 2003). Andererseits wurden Applikationen entwickelt, die durch intelligente Analysemethoden (z. B. Graph-Grammatiken) Lernende durch geeignetes Feedback in der Akquisition von Argumentationsfähigkeiten unterstützen. Dies ist kein leichtes Unterfangen: Argumentation im Allgemeinen ist *ill-defined* (Lynch et al. 2006), d. h. es existiert typischerweise kein eindeutiges richtiges Argument, sondern es sind oft mehrere Argumentationsstrategien denkbar, sodass eine vollständig automatisierte rechnergestützte Bewertung von Argumenten nicht ohne weiteres möglich ist.

Ein Blick auf aktuell bestehende Systeme und Methoden zur Unterstützung der Vermittlung von Argumentationsfähigkeiten offenbart rasch die folgenden Schwächen und Beschränkungen: Einerseits sind viele Argumentationssysteme *entweder explizit für ein spezielles Einsatzgebiet*, z. B. juristische Argumentation (bspw. LARGO, Pinkwart et al. 2006) oder wissenschaftliche Argumentation (z. B. Convince Me, Ranney & Schank 1998, Siegel 1999), konzipiert, *oder zu allgemein* (bspw. Athena, Rolf & Magnusson 2002), um als angemessenes eLearning-Werkzeug zu dienen. Ein goldener Mittelweg, der etwa eine Vielzahl konfigurierbarer, domänenspezifischer Werkzeuge im Rahmen einer flexiblen, erweiterbaren Systeminfrastruktur anbietet, existiert derzeit nicht.

Weiterhin sind (überraschend) viele existierende Argumentationssysteme (z. B. Convince Me, Athena) *ausschließlich für Einzelnutzer konzipiert* und damit zu weit entfernt von praktischen Anwendungen, in denen Argumentation von Gruppendiskussionen lebt. Einige Systeme versuchen, dieses Problem durch den Einsatz von Austauschmedien, welche die Resultate von Einzelnutzertätigkeit an andere Nutzer exportierbar machen, zu umgehen. Beispiele hierfür sind etwa der Report Generator aus Athena oder die AraucariaDB aus Araucaria (Reed & Rowe 2004). Nichtsdestotrotz wäre es vorteilhaft, adäquate Kooperationsunterstützung für den ganzen Argumentationsprozess zu bieten wie etwa in Belvedere (Suthers 2003).

Außerdem sind bestehende Systeme *isoliert*, d. h. sie bieten keinerlei öffentliche Interfaces an, um *Interoperabilität* mit anderen Systemen, die evtl. bessere Werkzeuge für gewisse Teilaufgaben bereithalten, zu ermöglichen. Ein Beispiel hierfür sind etwa Analyseframeworks wie in ARGUNAUT (de Groot et al. 2007, McLaren, Scheuer & Mikšátko im Druck), die es dem Dozenten ermöglichen, schneller auf Konflikte oder Fehler zu reagieren.

Zusätzlich dazu ist die Mehrheit der bestehenden Argumentationssysteme *unflexibel*, d. h. es gibt keine Möglichkeit, sie zu erweitern oder zu modifizieren (z. B. via Plug-Ins), um domänenspezifische Anforderungen zu erfüllen. Eine beispielhafte Anforderung ist etwa die Fähigkeit, externe Ressourcen, welche als Verankerung von Argumenten dienen könnten (beispielsweise Naturgesetze in der Physik oder Gesetzestexte im Bereich der juristischen Argumentation), einzubinden. Ein anderes Beispiel mangelnder Flexibilität ist die größtenteils fehlende Möglichkeit zur Konfigurierung der Argumentationsontologie, d. h. der zur Erstellung von Argumenten nutzbaren Elemente und Beziehungen – ein Gegenbeispiel ist hier z. B. Digalo (Schwarz & Glassner 2007).

Abschließend existiert keine allgemeingültige, formale *Methodik*, um Argumentationssysteme zu erstellen. Folglich wird das Rad jedes Mal zeitraubend neu erfunden, ohne dabei auf bewährte Konzepte von bestehenden Ansätzen zurückzugreifen und diese weiterzuführen. Während die Bedeutung von Entwurfsmustern und Entwicklungsmodellen, welche die Implementierung von typischen, wiederkehrenden Problemen durch Standardvorgehensweisen erleichtern und eine gemeinsame Sprache (in Form von bekannten Mustern) schaffen, im allgemeinen Software-Engineering im letzten Jahrzehnt rapide zunahm, existieren nur wenige vergleichbare Ansätze für eLearning-Technologien im Bereich der Argumentation (z. B. Suthers 2001). Die wenigen veröffentlichten Ansätze konzentrieren sich hauptsächlich auf Intelligente Tutorensysteme (Wenger 1987, Harrer & Devedzic 2002, Devedzic & Harrer 2005).

Das DFG-Projekt LASAD (Learning to Argue: Generalized Support Across Domains[1]) beschäftigt sich mit der Lösung von genau diesen angesprochenen Problemen. Im Rahmen dieses Projektes haben wir Mechanismen zur Flexibilisierung von eLearning-Argumentationssystemen entwickelt. Im Verlauf dieses Papers werden einige dieser Verfahren näher dargestellt. Ein Schwerpunkt liegt dabei auf der Vorstellung einer Systemarchitektur, die es ermöglicht, mit minimalem Konfigurationsaufwand neue Argumentationssysteme zu erstellen, die domänenspezifischen Anforderungen genügen. Anschließend wird eine beispielhafte Konfiguration näher dargestellt, um das Konzept zu erläutern.

3 Anforderungsanalyse

Ausgehend von einem detaillierten Review existierender Argumentationssysteme (Scheuer et al. 2010), können folgende (hier knapp zusammengefassten) Anforderungen an ein generisches Argumentationsframework herausgefiltert werden: Auf dem allgemeinen Level muss das Framework *flexibel* sein, d. h. es muss möglich sein, eine Systeminstanz, die auf dem Framework basiert, so zu konfigurieren bzw. zu erweitern, dass sie den Anforderungen unterschiedlichster Domänen genügt.

[1] Weitere Details online: http://lasad.dfki.de

Die Systemkomponenten sollten ferner *lose gekoppelt* sein, um die *Wartbarkeit* des Systems zu gewährleisten. Beispielsweise muss es möglich sein, einzelne Komponenten des Systems kurzfristig auf andere Rechner auszulagern oder durch neue Versionen auszutauschen. Zur gleichen Zeit ist es jedoch erforderlich, dass das System trotz dieser losen Kopplung *robust* genug ist, um auch bei großen Gruppen von Nutzern entsprechend zu *skalieren*. Weiterhin sollte das Framework jeder Instanz eine *öffentliche Schnittstelle* bereitstellen, um einen Austausch von Daten mit anderen Systemen zu ermöglichen. Diese *Daten* müssen zur gleichen Zeit *konsistent und persistent* gespeichert werden. Insbesondere in Mehrbenutzersystemen ist die Konsistenz ein wichtiger Faktor. Es muss zu jeder Zeit gewährleistet sein, dass alle Nutzer auf der gleichen Datenbasis arbeiten, da sonst Missverständnisse oder gar Datenverluste auftreten können. Die Persistenz ist insbesondere für asynchrone Zusammenarbeit und die Forschung wichtig, um die Verfügbarkeit der Daten auch zu einem späteren Zeitpunkt zu ermöglichen.

Auf Seite der funktionalen Anforderungen müssen unterschiedliche Mechanismen, die ihre Tauglichkeit in früheren Systemen unter Beweis gestellt haben, unterstützt werden. Zu diesen Mechanismen gehört die Unterstützung für verschiedene Arten von Kooperation: Während synchrone Kooperation für Brainstorming-Aktivitäten vorteilhaft sein kann, ist es beispielsweise möglich, dass eine reflektierte Diskussion über formulierte Argumente von asynchroner Kooperation profitiert. Folglich müssen beide Arten möglich und von entsprechenden Awarenessmechanismen unterstützt werden, sodass auch ein fliegender Wechsel zwischen beiden Arten jederzeit möglich ist. Besonders für fortgeschrittene Prozessspezifikationstechniken wie etwas *collaboration* bzw. *learning scripts* (Weinberger et al. 2005, Kollar et al. 2006) wird zudem ein feines Rollen- und Rechtemanagement (denkbare Rollen wären z. B. Moderator, Dozent, Lernender oder Feedbackagent) benötigt. Unterstützende Kommunikationswerkzeuge wie Text-, Audio- oder Videochats sind zudem essentiell für eine kollaborative Nutzung des Systems.

Zusätzlich muss das Framework in der Lage sein, unterschiedliche Argumentvisualisierungen (s. o.) zu unterstützen, da Untersuchungen zeigen, dass unterschiedliche Visualisierungen unterschiedliche Argumentationsprozesse fördern können. Beispielsweise dienen Graphstrukturen eher zum Schaffen einer Übersicht bzw. eines gemeinsamen Verständnisses (van Gelder 2003), wohingegen Matrizen das Fehlen von Beziehungen zwischen Argumentationselementen aufzeigen (Suthers 2003).

Indirekt verbunden mit der Frage nach der Visualisierung von Argumenten ist die Flexibilität hinsichtlich der Argumentationsontologie (s. o.). Denkbar ist hier die Unterstützung von vorgefertigten Ontologien wie etwa Toulmin (1958) oder Wigmore (1931) ebenso wie die Möglichkeit, selbstständig Ontologien zu definieren, wie es beispielsweise in Digalo (Schwarz & Glassner 2007) möglich ist.

Eine weitere Möglichkeit, domänenspezifische Anforderungen zu erfüllen, ist die Integration von externen Ressourcen (z. B. Texte, interaktive Webseiten, Bilder, Audio- oder Videofiles). Hierbei spielen auch Mikroreferenzen, d. h. Referen-

zen zu *Teilen* von externen Ressourcen eine wichtige Rolle in einigen Anwendungsbereichen. Denkbare Anwendungen sind hier beispielsweise textuelle Fassungen anerkannter Theorien in der Wissenschaft, auf die über Mikroreferenzen in Argumenten leichter Bezug genommen werden kann, oder Gesetzestexte, auf deren einzelne Paragraphen in der juristischen Argumentation verwiesen werden könnte.

Insbesondere für die Forschung sind flexible Loggingmechanismen von essentieller Bedeutung, um Argumentationsprozesse analysieren und Beobachtungen belegen zu können. Hier sind zwei Ansätze zu unterscheiden, die beide unterstützenswert sind: Einerseits das Aktionen-basierte Logging, bei dem jede einzelne Aktion in einer Argumentation (z. B. das Erstellen einer Beziehung zwischen zwei Teilen oder das Verschieben eines Knotens in einer Graph-basierten Darstellung) aufgezeichnet wird. Dies ist hilfreich, um einen Argumentationsprozess nachvollziehen zu können bzw. nachträglich zu simulieren, um darüber reflektieren und daraus Schlüsse ziehen zu können. Andererseits gibt es jedoch auch Zustands-basiertes Logging, bei dem der aktuell gültige Argumentationszustand ohne Performanceeinbußen (die beim Aktionen-basierten Logging durch ein Durchgehen aller Schritte bis zum aktuellen Zeitpunkt entstehen würden) an neue Teilnehmer gesendet werden kann. Letzteres ist wichtig, um die Skalierbarkeit des Systems auch für größere Gruppen zu gewährleisten.

4 Architektur

Ausgehend von den identifizierten Anforderungen wurde die in Abbildung 1 dargestellte Systemarchitektur als Basis des generischen Argumentations-Frameworks entwickelt. Hierbei handelt es sich um eine klassische Schichtenarchitektur. Die (austauschbaren) Technologien, die dazu von uns gegenwärtig eingesetzt werden, sind mit einem * markiert.

4.1 Client-Layer

Auf dem obersten Level gibt es zwei denkbare Arten von Clients: Benutzer-Clients (BCs, s. Abb. 1 oben links) zur Erstellung von Argumentationen durch menschliche Benutzer und Analyse-und-Feedback-Clients (AFCs, s. Abb. 1 oben rechts) zur automatischen Untersuchung von Argumentationsstrukturen, z. B. um auf mögliche Schwächen oder Inkonsistenzen hinzuweisen und so den Lerneffekt zu stärken oder einzelne Nutzer (beispielsweise den Dozenten) mit zusätzlichen Überblicksinformationen zu versorgen (etwa ob ein Nutzer sich nicht beteiligt). Beide Clients nutzen dasselbe Server-Interface. So ist ein AFC in der Lage, die gleichen Aktionen wie ein menschlicher Nutzer auszuführen, z. B. das Bewegen von Elementen in einer graphischen Repräsentation oder das Ändern von Textabschnitten in einem Argument. Zusätzlich dazu könnten AFC durch das Rollen-

und Rechtemanagement weitere Rechte erhalten, die normalen Nutzern nicht zustehen, um beispielsweise besonders auffälliges Feedback zu geben. Denkbar wäre hier etwa das Hervorheben von ganzen Argumentationssträngen, um auf Inkonsistenzen hinzuweisen.

Auf Seite der BCs steht die einfache Nutzbarkeit im Vordergrund. Dazu basiert der entwickelte Web-Client auf dem Google Web Toolkit (GWT), welches

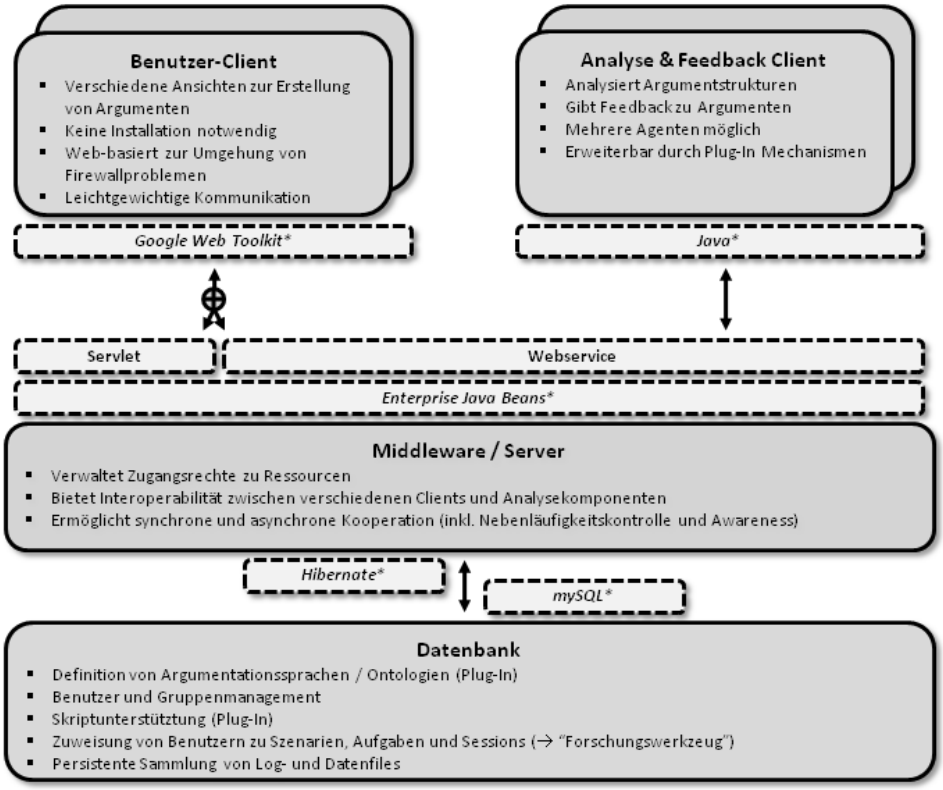

Abbildung 1: Systemarchitektur

seinerseits einen Java-to-JavaScript Compiler bereitstellt. Folglich ist der Client komplett plattformunabhängig, bedarf keiner Installation oder Firewallkonfiguration und läuft in jedem gängigen Browser, sodass typische Probleme beim praktischen Einsatz des Systems umgangen werden. Der BC ist zugleich das Hauptwerkzeug für Benutzer, um Argumente zu erstellen, zu bearbeiten und darüber zu diskutieren. Dazu stellt er eine graphische Oberfläche bereit, die unterschiedliche Visualisierungsformen unterstützt. Auf der Seite der AFCs sind unterschiedliche Typen von Analyse-Engines denkbar, beispielsweise Graph-Grammatiken (Pinkwart et al. 2006) oder Moderatoren-Assistenzsysteme wie ARGUNAUT (de Groot et al. 2007, McLaren, Scheuer & Mikšátko im Druck).

Diese Clients können in jeder beliebigen Programmiersprache geschrieben werden, sofern diese Web-Services unterstützt.

4.2 Server-Layer

Um die Clients zu entlasten, werden die eher komplexen Aufgaben wie Nebenläufigkeitskontrolle oder das Rollen- und Rechtemanagement vom Server übernommen. Hierbei handelt es sich um typische Serveraufgaben. Der Server kommuniziert mittels Servlets und Web-Services mit den angeschlossenen Clients. Während ersteres für GWT benötigt wird, stellt letzteres die Hauptschnittstelle für alle anderen Clients dar. Jede Aktion (z. B. das Hinzufügen eines Arguments oder das Schicken einer Chatnachricht), die auf einem Client ausführt wird, wird an den Server gesendet und dort verarbeitet, eher dieser sie an alle anderen Clients weiterleitet. Dies geschieht mittels eines Server-Pushs, d. h. die angeschlossenen Clients müssen ihrerseits einen Web-Service bereitstellen, den der Server aufrufen kann, um sie über neue Nachrichten zu informieren. In der aktuellen Fassung nutzen wir Enterprise Java Beans, um diese Aufgaben zu erledigen.

4.3 Daten-Layer

Der Schlüssel zur Flexibilität des Frameworks liegt auf der untersten Ebene des Systems, dem Daten-Layer. Hier ist es möglich, die verwendeten Ontologien via XML zu definieren sowie die Rollen- und Rechteverteilung genau zu spezifizieren. Alle Log-Daten werden hier zentral in beiden Varianten, d. h. sowohl Aktions-basiert als auch Zustands-basiert, persistent gespeichert. Weitere Konfigurationsmöglichkeiten wie Prozessbeschreibungen über *collaboration* und *learning scripts* werden ebenfalls hier gespeichert und entsprechend an den Server (und damit an die Clients) verteilt. Aktuell verwendet unser Prototyp ein Hibernate Mapping zu einer mySQL Datenbank.

5 Konfiguration der Ontologie

Nachdem im letzten Abschnitt die Grundstruktur des Systems und damit die Flexibilisierung des Systems „im Großen“ beschrieben wurde, wird in diesem Abschnitt gezeigt, wie genau die Definition der Ontologie – und damit einer der Schlüsselpunkte der Flexibilität „im Kleinen“ – geschieht. Aufgabe der Ontologie ist es, die zur Gestaltung der Argumentation verfügbaren Elemente, deren Beziehungen untereinander sowie mögliche Modifikatoren (z. B. Scores oder Glaubwürdigkeitsbewertungen) zu definieren. Die Ontologie ist damit vergleichbar mit einer Grammatik in der Sprache, die ein Rahmenwerk definiert, mit dem kommuniziert wird. Bestehende Argumentationssysteme unterscheiden sich sehr hinsichtlich der verwendeten Ontologien: so bietet z. B. Athena nur einen Beitragstyp

(Node) und einen Relationstyp (Link) an, während Convince Me zur wissenschaftlichen Argumentation zwei Beitragstypen (Hypothesis & Evidence) sowie zwei Beziehungstypen (Contradiction & Explanation) anbietet. Andere existierende System haben noch weit umfangreichere Ontologien.

In Abbildung 2 ist ein kleiner Auszug aus einem Argumentsgraphen dargestellt, bestehend aus zwei Beiträgen (den größeren gerahmten Boxen) sowie einer gerichteten, gelabelten Relation (das Label ist dargestellt als kleinere Box).

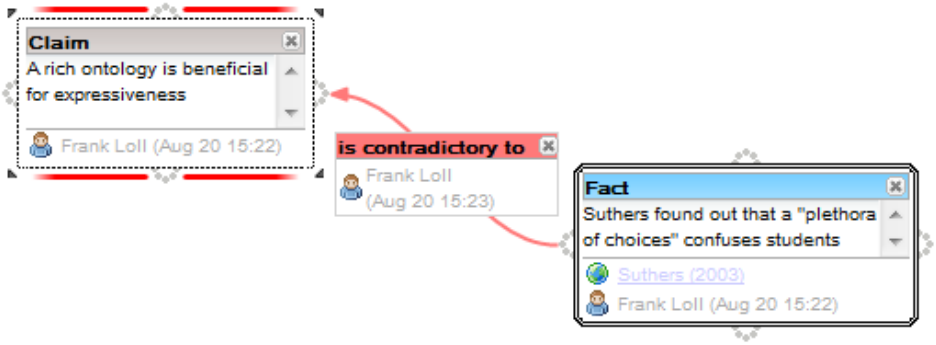

Abbildung 2: Visualisierung der Ontologie in einer Instanz des Frameworks

Alle drei Modellierungselemente in diesem Beispiel haben einen Typ, dargestellt im Titel (*claim*, *is contradictory to*, *fact*) sowie einem Awareness-Element, welches den Namen des Autors und das Erstellungsdatum zeigt. Zusätzlich dazu besitzt das *fact*-Modellierungselement noch ein Unterelement, welches auf eine externe Ressource im Web verlinkt. Weitere Unterscheidungen sind die Farben der einzelnen Modellierungselemente (claim = grau, is contradictory to = rot, fact = blau) sowie im Fall der Knoten unterschiedliche Rahmen, um diese besser auseinander zu halten (claim = gepunkteter Rahmen, fact = doppelter Rahmen). Die Beziehung zwischen den Knoten besteht aus zwei Teilen: einer gerichteten Kante und einem Panel. Im Gegensatz zu den beiden Knoten enthält das Panel in diesem Fall kein Text-Unterelement. Als aktives Modellierungselement ist das claim-Modellierungselement gekennzeichnet (durch die roten Hervorhebungen über und unter der Box sowie durch die sichtbaren Ecken zum Ändern der Größe der Box). Die Ontologie, die dieser Struktur zu Grunde liegt, wird in unserem System hierbei in XML definiert. Tabelle 1 zeigt ein (vereinfachtes) Beispiel einer solchen Ontologiedefinition. Im Folgenden beschreiben wir anhand dieses Beispiels die Prinzipien und Ausdrucksmöglichkeiten der Spezifikationstechnik.

Die Definition der Beispiel-Ontologie besteht insgesamt aus vier XML Tags. Das Wurzelelement <*ontology*> definiert dabei den Namen (*type*) der Ontologie. Anschließend werden die einzelnen Modellierungselemente definiert (<*contribution*> für Elemente und <*relation*> für Relationen).

Eine Box wird als <*contribution*> definiert und besitzt unterschiedliche Eigenschaften (*width*, *height*, *resizable*, *border*, *background-color*, *font-color*, *type*), die die Darstel-

lung beeinflussen. Analog verhält es sich mit den <*relation*> Tags, die die verfügbaren Kantentypen beschreiben. Hierbei entfallen die Rahmentypen, da ein Beziehungspanel keinen Rahmen besitzt. Zusätzlich zu den Eigenschaften einer <*contribution*> sind die Attribute *line-color*, *line-width* und *directed* verfügbar, die die Kante näher definieren. Die einzelnen Attribute sind hierbei allesamt optional, d. h. wenn sie nicht angegeben werden arbeitet der Client mit entsprechenden Standardwerten. Dies ist wichtig, da diese Eigenschaften lediglich Zusatzinformationen für eine mögliche Visualisierung bieten. Da die eigentliche Ontologie jedoch unabhängig von der Visualisierung ist, muss eine neutrale Definition ebenfalls möglich sein.

Tabelle 1: XML Definition der Ontologie

```
<ontology type="Example">

  <contribution width="180" height="100" resizable="true" border="double"
      background-color="#55C3FF" font-color="#000000" type="Fact">

      <element type="text" required="true" background-color="#FFFFFF"
          font-color="#000000" />

      <element type="url" required="true" />

      <element type="awareness" required="true" />

  </contribution>

  <contribution width="180" height="100" resizable="true" border="dashed"
      background-color="#C0B7B4" font-color="#000000" type="Claim">

      <element type="text" required="true" background-color="#FFFFFF"
          font-color="#000000" />

      <element type="awareness" required="true" />

 </contribution>

  <relation width="120" height="40" line-color="#ff7979" line-width="2px"
      directed="true" background-color="#ff7979" font-color="#000000"
      type="is contradictory to">

      <element type="awareness" required="true" />

  </relation>

</ontology>
```

Innerhalb der <*contribution*> und <*relation*> Tags werden einzelne Unterelemente definiert, die den Modellierungselementen zugeordnet werden und deren Datenmodell (jenseits von oben skizzierten visuellen Parametern) ausmachen. In unserem Beispiel besitzt das *fact*-Modellierungselement drei Unterelemente: ein Text-Unterelement (type="text"), ein URL-Unterelement (type="url") sowie ein Unterelement mit Awarenessinformationen für eine kooperative Nutzung des Systems (type="awareness"). Beim *claim*-Element entfällt das URL-Unterelement. Der einzige dargestellte Kantentyp besitzt weder ein Text- noch ein URL-Unterelement.

Das Panel besteht hier einzig aus dem Label des Modellierungselements ohne eine Möglichkeit, weiteren Text einzugeben.

Wichtig an dieser Stelle ist, dass der jeder Client die jeweiligen Daten auf seine Weise darstellen kann und – sofern einzelne Sichten bestimmte Zusatzinformationen (z. B. *size*, *color*, etc.) nicht unterstützen (denkbar wäre hier z. B. eine Forendarstellung, die die Liniendicke der Beziehungen nur eingeschränkt darstellen könnte) – diese auch weglassen kann. Die Mächtigkeit des Frameworks ist damit indirekt abhängig von der Unterstützung des jeweiligen Clients, der die jeweils definierten Elemente unterstützen bzw. darstellen können muss.

In der aktuellen Version unseres Clients ist es möglich, eine Graph-basierte Visualisierung zu erstellen, die eine zuvor definierte Ontologie als Regelwerk nutzt.

6 Zusammenfassung und Ausblick

Das in diesem Paper beschriebene Framework vereinfacht die Erstellung von eLearning-Argumentationssystemen durch weitreichende Konfigurationsmechanismen, die es ermöglichen, domänenspezifische Anforderungen durch entsprechende Anpassung zu erfüllen. Basierend auf den hier vorgestellten Konfigurationsmechanismen ist es möglich, einen Großteil der bestehenden Argumentationssysteme binnen kürzester Zeit mit minimalem Aufwand nachzubauen sowie neue Systeme zu kreieren. Unser aktueller auf dem Framework aufbauender Prototyp des mehrbenutzerfähigen webbasierten Clients unterstützt dazu eine Graph-basierte Visualisierung. Eine beispielhafte Anwendung zur Nachbildung existierender Systeme findet sich in Loll et al. (2010).

Neben der hier vorgestellten Graph-basierten Ansicht sind weitere Visualisierungen in Arbeit. Parallel dazu entsteht als Proof-of-Concept ein erster Analysis-and-Feedback Client, der die entstehenden Argumente mittels unterschiedlicher Methoden aus dem Bereich der Künstlichen Intelligenz analysiert und den jeweiligen Nutzer mit entsprechendem Feedback versorgt. Im Laufe des nächsten Jahres sind zudem erste empirische Studien geplant, welche die Eignung des Frameworks nachweisen sollen. Ein visueller Editor, der die Erstellung der XML Dateien zur leichten Ontologiedefinition auch für Endbenutzer übernimmt, ist ebenfalls geplant.

Literatur

Andriessen J (2006). Arguing to learn. Sawyer RK (Ed.) The Cambridge Handbook of the Learning Sciences, pp. 443-460. Cambridge.

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

de Groot R, Drachman R, Hever R, Schwarz B, Hoppe U, Harrer A, De Laat M, Wegerif R, McLaren BM, Baurens B (2007). Computer supported moderation of e-discussions: the ARGUNAUT approach. Chinn C, Erkens, G, Puntambekar S (Eds.) Mice, Minds, and Society – The Computer Supported Collaborative Learning Conference, 8:165-167. Intl. Soc. of the Learning Sciences.

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

Easterday MW, Aleven V, Scheines R, Carver SM (2009). Will Google destroy western democracy? Bias in policy problem solving. In Proceedings of the Intl. Conf. on Artificial Intelligence in Education, pp. 249-256. IOS Press, Amsterdam.

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

Kirschner PA, Buckingham Shum SJ, Carr CS (2003). Visualizing argumentation. Software tools for collaborative and educational sense-making. Springer, London.

Kollar I, Fischer F, Hesse FW (2006). Collaboration scripts – a conceptual analysis. Educational Psychology Review 18(2): 159-185.

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

Kulik CC, Kulik JA (1991). Effectiveness of computer-based instruction: An updated analysis. Computers in Human Behavior, 7:75-95.

Loll F, Pinkwart N, Scheuer O, McLaren BM (2010). Simplifying the Development of Argumentation Systems using a Configurable Platform. In: N. Pinkwart, & B. M. McLaren (Eds.), Educational Technologies for Teaching Argumentation Skills. Bentham Science Publishers, im Druck.

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

McLaren BM, Scheuer O, Mikšátko J (im Druck). Supporting collaborative learning and e-discussions using artificial intelligence techniques. Intl. Journal of Artificial Intelligence in Education, im Druck.

Pinkwart N, Aleven V, Ashley KD, Lynch C (2006). Toward legal argument instruction with graph grammars and collaborative filtering techniques. In: Proceedings of the 8th Intl. Conf. on Intelligent Tutoring Systems, Lecture Notes in Computer Science 4053, pp. 227-236. Springer, Berlin.

Ranney M, Schank P (1998). Toward an integration of the social and the scientific: Observing, modeling, and promoting the explanatory coherence of reasoning. In: Read S, Miller L (Eds.) Connectionist models of social reasoning and social behavior, pp. 245-274. Lawrence Erlbaum

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

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

Scheuer O, Loll F, Pinkwart N, McLaren BM (2010). Computer-supported argumentation: A review of the state-of-the-art. Intl. Journal of Computer-Supported Collaborative Learning, im Druck.

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

Siegel MA (1999). Changes in student decisions with convince me: Using evidence and making tradeoffs. In: Proceedings of the 21st Annual Conf. of the Cognitive Science Soc., pp. 671-676. Erlbaum, Mahwah.

Suthers DD (2001). Architectures for computer supported collaborative learning. In: Proceedings of the IEEE Intl. Conf. on Advanced Learning Technologies.

Suthers DD (2003). Representational guidance for collaborative inquiry. Andriessen J, Baker M, Suthers D (Eds.) Confronting Cognitions: Arguing to Learn, pp. 1-17. Kluwer.

Tannen D (1998). The Argument Culture: Moving from Debate to Dialogue. New York: Random House Trade.

Toulmin SE (1958). The uses of argument. Cambridge.

van Gelder T (2003). Enhancing deliberation through computer supported argument visualization. Kirschner PA, Buckingham Shum SJ, Carr CS (Eds.) Visualizing argumentation. Software tools for collaborative and educational sense-making, pp. 97-115. Springer, London.

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.

Wenger E (1987). Artificial Intelligence and Tutoring Systems. Morgan Kaufmann, Los Altos.

Wigmore JH (1931). The principles of judicial proof (2. Ed.). Little Brown & Co.