# Diplomarbeit

Die Nutzung des LASAD-Systems zur Unterstützung juristischer Argumentation

**vorgelegt von**

Jan Brenner
cand. Dipl. Wirt.-Inf.
Matrikelnummer 325606
Clausthal-Zellerfeld, den 04.08.2010

**Erstgutachter**
Prof. Dr. Niels Pinkwart
Institut für Informatik

**Zweitgutachter**
Prof. Dr. Jörg Müller
Institut für Informatik

## Eidesstattliche Erklärung

Hiermit erkläre ich eidesstattlich, dass ich die bei der Fakultät für Mathematik/Informatik und Maschinenbau (Institut für Informatik) eingereichte, hier vorliegende Diplomarbeit mit dem Thema „Die Nutzung des LASAD-Systems zur Unterstützung juristischer Argumentation“ selbstständig und ohne unerlaubte Hilfe verfasst habe. Alle benutzten Hilfsmittel wurden vollständig angegeben.

---

Ort, Datum Unterschrift

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

| **Kürzel** | **Beschreibung** |
|---|---|
| **AJAX** | **A**synchronous **J**avaScript **a**nd **X**ML |
| **API** | **A**pplication **P**rogramming **I**nterface |
| **CLI** | **C**ommon **L**anguage **I**nfrastructure |
| **Cool Modes** | **Co**llaborative **O**pen **L**earning and **Mode**ling **S**ystem |
| **CPL** | **C**ommon **P**ublic **L**icense |
| **CSS** | **C**ascading **S**tyle **S**heets |
| **DFKI** | **D**eutsches **F**orschungszentrum für **K**ünstliche **I**ntelligenz |
| **DHTML** | **D**ynamic **HTML** - **D**ynamic **Hyper**t**ext** **M**arkup **L**anguage |
| **DOM** | **D**ocument **O**bject **M**odel |
| **GPL** | GNU **G**eneral **P**ublic **L**icense |
| **GWT** | **G**oogle **W**eb **T**oolkit |
| **Http** | **H**yper**t**ext **T**ransfer **P**rotocol |
| **IBIS** | **I**ssue-**B**ased **I**nformation **S**ystem |
| **ITS** | **I**ntelligentes **T**utoren **S**ystem |
| **Java SE** | **Java** Platform, **S**tandard **E**dition |
| **JRE** | **J**ava **R**untime **E**nvironment |
| **JSON** | **J**ava**S**cript **O**bject **N**otation |
| **JVM** | **J**AVA **V**irtual **M**aschine |

| **Kürzel** | **Beschreibung** |
|---|---|
| **LARGO** | **L**egal **Ar**gument **G**raph **O**bserver |
| **LASAD** | **L**earning to **A**rgue: Generalized **S**upport **A**cross **D**omains |
| **LGPL** | GNU **L**esser **G**eneral **P**ublic **L**icense |
| **LRDC** | Learning and **R**esource **D**evelopment **C**enter |
| **REST** | **Re**presentational **S**tate **T**ransfer |
| **RIA** | **R**ich **I**nternet **A**pplication |
| **RPC** | **R**emote **P**rocedure **C**all |
| **RTMP** | **R**eal **T**ime **M**essaging **P**rotocol |
| **SQL** | meist **S**tructured **Q**uery **L**anguage, obwohl laut Standard eigenständiger Name. |
| **SUS** | **S**ystem **U**sability **S**cale |
| **WPF** | *W*indows *P*resentation *F*oundation |
| **XAML** | E**x**tensible **A**pplication **M**arkup **L**anguage |
| **XMPP** | E**x**tensible **M**essaging and **P**resence **P**rotocol |

# 1 Einleitung

Argumentationsfertigkeiten spielen in vielen beruflichen wie privaten Kontexten eine wichtige Rolle. Dennoch hat, durch alle Bildungsschichten hinweg, ein Großteil der Bevölkerung, Probleme Behauptungen von Argumenten zu unterscheiden sowie Angriffe auf Ihre eigenen Argumente zu erkennen und adäquat zu reagieren. Folglich nimmt die Ausbildung von Argumentationsfertigkeiten eine zentrale Rolle in der klassischen Lehre ein. Dies gilt sowohl im schulischen Umfeld und der Berufsbildung als auch bei der professionellen Weiterbildung, z.B. im Bereich der Verhandlungsführung und Teamtraining ein.

Computergestützte Argumentationssysteme setzen dort an, wo der klassische Lehransatz zur Schulung dieser Fähigkeiten mit seinem ausgeprägten *face-to-face*-Ansatz an seine Grenzen stößt. Dieser birgt ein großes Problem: Er ist nicht ohne weiteres auf große Gruppen übertragbar, da entsprechendes Personal und deren Zeit beschränkt ist. [LOLL et al., 2010]

Genau hier setzen computergestützte Argumentationssysteme an. Sie versuchen, die schlechte Skalierbarkeit des *face-to-face*-Ansatzes zu umgehen, indem sie versuchen Argumentationsfertigkeiten durch Softwareunterstützung zu schulen.

*LARGO*, ein intelligentes Tutorensystem (*ITS*) für juristische Argumentationsstrategien ist ein solches computergestütztes Argumentationssystem. Es wurde von Beteiligten der *Carnegie Mellon University* (Pittsburgh) sowie der *University of Pittsburgh* entwickelt. Es soll Studierende beim Lernen juristischer Argumenationsstrategien unterstützen. [PINKWART et al., 2006b]

Obwohl Argumente in den Rechtswissenschaften typischerweise rational auf Basis von Gesetzestexten begründet werden, ist der Argumentationsprozess ein sprachlicher Diskurs in dessen Verlauf juristische Prinzipien und Gesetzesartikel im Angesicht spezifischer Faktenlagen interpretiert werden. Die fehlende Lösungseindeutigkeit und

daraus resultierende Unstrukturiertheiten dieser Domäne macht das Erstellen verlässlicher Modelle unmöglich und ist ein Grund warum nur wenig vergleichbare Systeme für den Bereich der Rechtswissenschaften existieren. [PINKWART et al., 2006a]

Zu diesem Zweck bietet *LARGO* den Studierenden die Möglichkeit aus Verlaufsprotokollen von Gerichtsverfahren die Beiträge der einzelnen Akteure in Diagrammform zu beschreiben. Beiträge werden im Diagramm durch Boxen verschiedener Typen repräsentiert und durch verschieden geartete Verbindungen miteinander verknüpft. Zur Wahl stehen *modified to*, *destinguished from*, *analogized to*, *leads to* und eine *general relation*.

Die Diagrammform visualisiert dabei die Zusammenhänge des Protokolls, im Weiteren auch Transcript genannt, und löst sie gleichzeitig von dessen linearem Zeitcharakter. Die Möglichkeit, Elemente des Diagramms mit Textpassagen des Transcripts zu verknüpfen, erleichtert es die Argumentationsstrukturen der einzelnen Akteure mit ihren weitreichenden Verpflechtungen nachzuvollziehen.

Weiterhin kann *LARGO* die Struktur der erstellten Diagramme analysieren und den Studierenden auf strukturelle und inhaltliche Schwachstellen hinweisen. Technisch wird dies durch Graph-Grammatiken und kollaborative Filteralgorithmen realisiert. [PINKWART et al., 2006b]

Sowohl 2006 als auch 2007 wurden Studien mit dem *LARGO*-System an der *University of Pittsburgh* durchgeführt. Dabei wurde die Hypothese überprüft, ob *LARGO*'s grafische Repräsentationen und Feedback-Mechanismen Studierenden helfen würden besser zu lernen als mit der Hilfe eines rein textbasierten Tools.

Bestätigte die erste, mit freiwilligen Teilnehmern, durchgeführte Studie noch die entsprechende Hypothese, konnte das *LARGO*-System in einer zweiten, für die Teilnehmer verpflichtenden, Studie nicht länger überzeugen.

Mögliche Gründe für die Unterschiede zwischen den beiden Studien könnten unter anderem die Motivation der Teilnehmer und deren Umgang mit dem System selbst sein.[PINKWART et al., 2008]

Neben strukturellen Problemen führten weitere Schwierigkeiten mit der technischen Plattform von *LARGO* zu der Entscheidung das *LARGO*-System im Rahmen dieser Arbeit und in Kooperation mit Partnern aus Pittsburgh auf das noch in der Entwicklung befindlichen *LASAD*-Framework zu migrieren.

*LASAD* (Learning to Argue: Generalized Support Across Domains) ist ein *DFG*-Projekt, das in einer Kooperation der *TU Clausthal* und des *DFKI* in Saarbrücken ins Leben gerufen wurde. Es hat sich zum Ziel gesetzt eine allgemeine Systemarchitektur zu entwickeln, die es ermöglicht mit minimalem Konfigurationsaufwand neue, bei Bedarf auch kollaborative, Argumentationssysteme zu erstellen, die domänenspezifischen Anforderungen genügen.

Im Rahmen einer abschließenden, an der *University of Pittsburgh* durchgeführten, Usability-Studie sollte untersucht werden, ob das neue System sowie die erarbeiteten Veränderungen positive Auswirkungen auf die in früheren Studien gezeigten Schwächen von *LARGO* haben und das *LASAD*-Framework sich generell dazu eignet ein domänenspezifisches Tool, wie *LARGO*, abzubilden.

# 2 Computergestützte Argumentationssysteme

Auch wenn sich diese Arbeit hauptsächlich mit dem *LARGO*- und *LASAD*-System beschäftigt, soll in diesem Kapitel ein Blick über den Tellerrand gewagt und einige weitere relevante Vertreter computergestützter Argumentationssysteme näher beschrieben und vorgestellt werden.

Als Ausgangspunkt dient die kürzlich von [SCHEUER et al., 2010] veröffentlichte Arbeit, die einen detaillierten Überblick über den aktuellen Stand der Forschung wgibt.

## 2.1 Auswahlkriterien

[SCHEUER et al., 2010] haben im Appendix ihrer Arbeit eine Übersicht aller 45 von ihnen untersuchten Argumentationssysteme erstellt. Sie enthält neben den konkreten Funktionen jedes einzelnen Systems, eine Referenz einer bedeutenden wissenschaftlichen Veröffentlichung samt der Häufigkeit mit der sie zitiert wurde.

Die Vorstellung aller 45 Systeme würde den Rahmen dieser Arbeit sprengen und nicht alle System sind gleich relevant. Eine gewisse Auswahl war zu treffen. Einige der Veröffentlichungen sind über 20 Jahre alt und spielen heutzutage keine wichtige Rolle mehr. Als Auswahlkriterium dient das Veröffentlichungsdatum der Publikation. Betrachtet werden nur Systeme deren gelistete Publikationen nicht älter als 4 Jahre sind. Die Auswahl reduziert sich somit auf 20 Systeme.

Da sich in den konkreten Zielen dieser Arbeit die kollaborativen Fähigkeiten solcher Systeme wiederfindet, wird die bisherige Auswahl ebenso nach ihren beschriebenen Funktionen gefiltert. Ignoriert man sämtliche Systeme die kein „collaborative“ zu ihren Funktionen zählen, reduziert sich die Menge der relevanten Systeme weiter auf elf.

Aus diesen elf Systemen wurden nach weiteren Recherchen sechs Systeme ausgewählt. Bei den restlichen Systeme waren entweder keine verlässlichen Daten zu finden oder es war keine weitere Relevanz zu erkennen.

*gIBIS* wird, trotz der Nichterfüllung einiger der oben genannten Kriterien, in Hinblick auf dessen Relevanz dennoch mit aufgeführt. 1310 Zitate der referenzierten Veröffentlichung sprechen für sich. Ebenso wird *Rationale* als kommerzielles Produkt, als einziger Vertreter dieser Gattung, mit aufgenommen.

Neben *gIBIS* und *Rationale* werden *ArguNet*, *CoFFEE*, *Compendium* und *CoPe_it!* näher betrachtet.

*LARGO* wird detailliert im Kapitel 3, *LASAD* in Kapitel 6 behandelt.

## 2.2 Übersicht relevanter Argumentationssysteme

### 2.2.1 ArguNet

*ArguNet*[1] ist ein Tool für kollaborative Argumentationsanalyse und - forschung. ArguNet kann dabei lokal auf einem einzelnen Computer wie auch online in einem kollaborativem Modus betrieben werden. Dabei können mehrere Clients in Echtzeit gemeinsam an einer Debatte über das Internet teilnehmen. Es ist ebenfalls möglich Onlineinhalte mit einer lokalen Instanz oder lokal erstellte Inhalte online mit anderen Nutzern weiter zu bearbeiten.

*ArguNet* stellt den Nutzern dazu zwei Editoren zur Seite, siehe Abbildung 2.1, den *Graph Editor* und den *Argument Editor*. Im *Graph Editor* lassen sich die Grundgedanken einer Debatte schnell skizzieren und Behauptungen semantisch miteinander verknüpfen. Im *Argument Editor* können weitere Behauptungen importiert, ihre Position und Details modifiziert sowie mit externen Referenzen angereichert werden.

*ArguNet* basiert auf einem klassischen Client-Server-Paradigma und wurde als Open Source-Software veröffentlicht. Der Client basiert auf der *Eclipse Rich Client Platform*[2], diese realisiert eine flexible und moderne Benutzeroberfläche samt einheitlichem Look&Feel. Er ist für Microsoft Windows, Mac OS sowie Linux verfügbar. Der Server

[1] http://www.argunet.org – Stand: 31. Juli 2010
[2] http://www.eclipse.org – Stand 30. Juli 2010

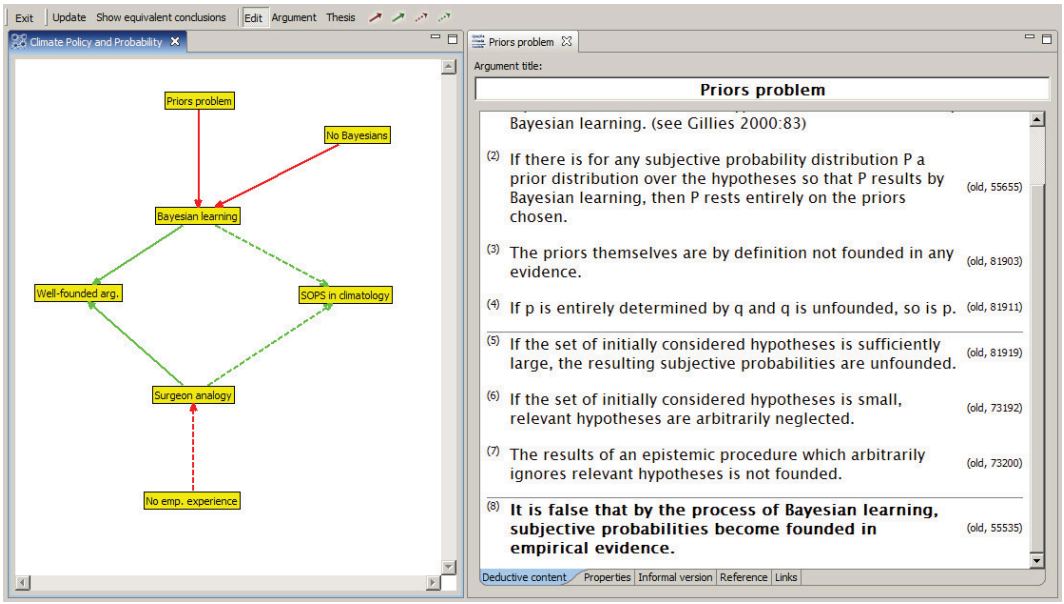


**Abbildung 2.1:** ArguNet: Graph Editor links, Argument Editor rechts.

wurde basierend auf der *J2SE*, in Java geschrieben und sichert so die nötige Plattformunabhängigkeit. Die Persistenz der Daten wird mit *DB4O*[3], einer objektorientierten Datenbank für Java, und dem *.Net Framework* gewährleistet. [SCHNEIDER et al., 2007]

### 2.2.2 CoFFEE

*CoFFEE* wurde zur Unterstützung von *face-to-face*-Gruppendiskussionen im Klassenraum entwickelt. Es stellt eine rollenbasierte Lernumgebung bereit die eine Vielzahl von kollaborativen Tools, siehe Abbildung 2.2, in sich vereint.

*Threaded Discussions* ermöglichen es den Nutzern untereinander in einem Chat zu diskutieren. Nachrichten werden dabei nicht klassisch in einem linearen Verlauf dargestellt. Vielmehr können Nutzer ihre Beiträge in einer Baumstruktur publizieren. Erweiterte Awareness-Mechanismen sorgen dafür, dass die Übersichtlichkeit, insbesondere bei neuen Nachrichten, erhalten bleibt.
*Graphical Discussions* bieten eine generische Umgebung um beispielsweise Mindmaps oder konzeptuelle Diagramme zu erstellen. Beiträge werden als Boxen in einer graphischen Umgebung dargestellt und können mit Pfeilen verbunden werden. Ein

[3] http://db4o.com – Stand 31. Juli 2010

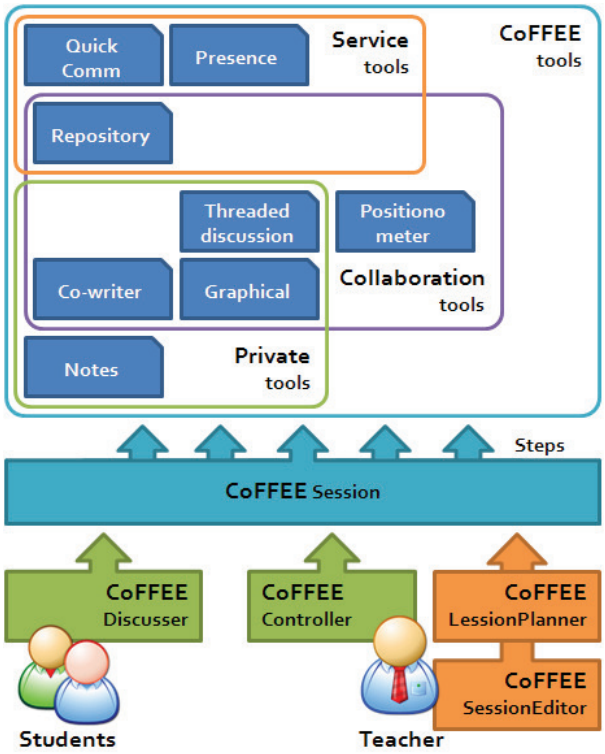


**Abbildung 2.2:** Übersicht der Toolsammlung des CoFFEE Frameworks

konfigurierbares Notationssystem erlaubt das Anpassen der Elemente an eigene Einsatzzwecke.

Das *Co-Writer* Modul stellt einen Shared-Text-Editor dar, indem jeweils nur eine Person das Recht zu Schreiben hat. Dieses Recht kann von der Lehrperson vergeben werden und erlaubt so das kontrollierte Arbeiten einer Gruppe an einem gemeinsamen Text.

Mit Hilfe des *Positionometer* lassen sich Umfragen erstellen und konfigurieren. Dabei besteht die Möglichkeit sowohl anonyme als auch personenbezogene Umfragen zu konfigurieren.

Weiterhin hält *CoFFEE* Möglichkeiten zum privaten und öffentlichen *File-Sharing* bereit. Sämtliche Aufgaben können in *Sessions* zusammengefasst und in einzelne *Steps* unterteilt werden. Der spätere Ablauf wird von der Lehrperson bestimmt.

Die Softwarearchitektur basiert, ebenso wie *ArguNet*, auf der *Eclipse Rich Client Platform*. Die Netzwerkkommunikation zwischen den einzelnen Instanzen wird mit Hilfe des

*Eclipse Communication Framework*[4] realisiert. Es handelt sich dabei um ist ein Projekt der Eclipse Community und bietet ein Framework zur Entwicklung verteilter Eclipse-basierter Programme mit der Unterstützung von asynchronen *point-to-point*- oder *publish-and-subscribe messaging*-Funktionen. Veröffentlicht wurde *CoFFEE* unter der *ECLIPSE EPL* Open Source-Lizenz und ist frei verfügbar[5]. [BELGIORNO et al., 2008]

### 2.2.3 Compendium

*Compendium*[6] beschreibt sich selbst als Management-Tool zur Verbindung von Informationen und Gedanken. Es bietet eine graphische Echtzeitrepräsentation sämtlicher Daten in Form von Diagrammen. Standardmäßig sind bereits alle von *IBIS* bekannten Arten von Kontenpunkten implementiert. Weiterhin stehen Typen zur Integration von externen Ressourcen wie Webseiten und andere Dokumente bereit. Sie lassen sich durch entsprechende Verbindungstypen miteinander verknüpfen.

Es ist ebenfalls möglich seine eigene Modellsprache zu entwickeln Dabei können eigene Konten- und Verbindungstypen definiert und mit entsprechenden Icons versehen werden. Allerdings ist *Compendium* kein vollwertiges Meta-Modeling-Tool, sodass sich keine Einschränkungen bezüglich der Verbindung von Elementen definieren lassen. Elemente können mit jedem anderen Element verbunden werden.

Eine wohl definierte API erlaubt den direkten Zugriff auf die Datenbank durch andere Systeme. Dies ermöglicht das Erstellen von Diagrammen aus anderen Anwendungen heraus. Ebenso ist es möglich in *Compedium* erstellte Diagramme von anderen Programmen interpretieren zu lassen. Die kollaborativen Funktionen werden mit Hilfe des durch Jabber[7] bekannten XMPP[8] Protokoll realisiert. Dies erlaubt synchrones Arbeiten über Rechnergrenzen hinweg. Als Austauschformat wird XML bevorzugt, Diagramme lassen sich aber ebenfalls als JPEG-Bild oder als als HTML-Datei exportieren. [MISTRIK et al., 2006]

[4] http://www.eclipse.org/ecf/ – Stand 31. Juli 2010
[5] http://www.coffee-soft.org/ – Stand 31. Juli 2010
[6] http://compendium.open.ac.uk/institute/index.htm – Stand: 31. Juli 2010
[7] Jabber ist ein Instant Messenger, der auf dem *XMPP*-Protokoll aufbaut und ähnliche Funktionen wie ICQ und Windows Live Messenger bietet. www.jabber.com – Stand 31. Juli 2010
[8] XMPP beschreibt einen Internet-Standard für XML-Routing und wird primär für Instant Messaging eingesetzt, wobei der eigentliche Payload dabei keine Rolle spielt und sich so auch für andere Einsatzzwecke eignet.

Als Java-Anwendung entwickelt ist *Compendium* für das Windows Betriebssystem ebenso verfügbar wie für Mac OS und Linux. Zur Datensicherung wird eine SQL-Datenbank (aktuell MySQL oder Derby[9]) eingesetzt. Als Open Source-Software wurde es unter der LGPL-Lizenz veröffentlicht. [SIERHUIS, 2006]

### 2.2.4 CoPe_it!

Als Teil des *PALETTE-Projekt*[10] wurde mit *CoPe_it!*[11] ein Tool zur Unterstützung von synchroner und asynchroner argumentativer Zusammenarbeit entwickelt. Dabei wurde auf die konsequente Nutzung von heutigen Web-Technologien gesetzt, um die Flexibilität dieser neuen Technik mit den erweiterten Funktionen strikt formalisierter Argumentations- und Entscheidungssysteme früherer Zeiten zu vereinen. Realisiert werden kann dies dank einer offenen semantischen Schicht und des Konzepts der inkrementellen Formalisierung von argumentativer Zusammenarbeit.

*CoPe_it!* erlaubt die schrittweise Entwicklung von Argumentationsräumen, bei dem die Formalisierung nicht durch das Programm selbst, sondern durch die Nutzer, durchgeführt und gleichzeitig kontrolliert wird. Wurde gemeinschaftlich ein gewisser Grad an Formalität erreicht kann CoPe_it! aktiv den Entscheidungsfindungsprozess, durch eine aufbereitete Präsentation der Daten, erleichtern. [TZAGARAKIS et al., 2009]

Nutzer können verschieden geartete Daten- und Wissenselemente erstellen und in den Argumentationsraum hochladen. Dabei können sie bestimmten vorgegebenen oder eigens definierten Typen von Elementen zugeordnet werden. Wie in Abbildung 2.3 zu sehen, können neben klassischen Textpassagen ebenso Fotos als auch Videos mit eingebettet und miteinander verknüpft werden. [KARACAPILIDIS et al., 2009]

Als technische Plattform wurde Microsofts *.NET Framework*, als Beschreibungssprache der Daten XML und zur Sicherung dergleichen der Microsoft SQL Server gewählt.

[9] Bei *Apache Derby* handelt es sich um eine relationale Open Source-Datenbank, veröffentlicht unter der Apache License 2.0, die komplett in Java entwickelt wurde. Sie basiert auf den Standards von Java, JDBC und SQL und lässt sich komplett in ein Java-Softwareprojekt einbetten. http://db.apache.org/derby/ – Stand 31. Juli 2010

[10] Das von der EU gegründete PALETTE-Projekt (Pedagogically Sustained Adaptive Learning through the Exploitation of Tacit and Explicit Knowledge) bezweckt die Vereinfachung und Erhöhung des individuellen und organisatorischen Lernens in praxisbezogenen Gemeinschaften, die seit mehr als zehn Jahren als wirksame Umgebungen zur Unterstützung des Lernens angesehen werden.

[11] http://copeit.cti.gr – Stand 31. Juli 2010

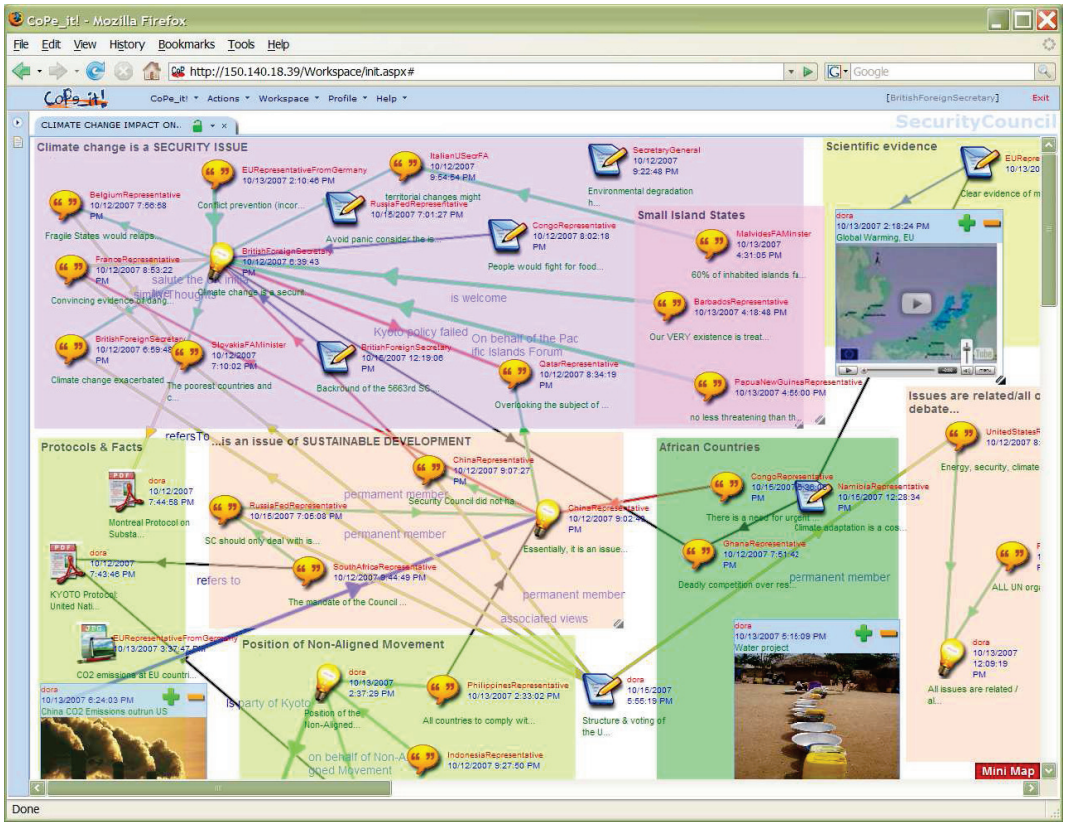


**Abbildung 2.3:** CoPe_it! Oberfläche

### 2.2.5 gIBIS

Als Pionier der computergestützten grafischen Argumentationssysteme erblickte *gIBIS* Ende der Achtziger Jahre das Licht der Welt. Fortschritte in der Computertechnik machten es erstmals möglich, solche Programme mit vergleichsweise wenig Aufwand zu realisieren. [CONKLIN und BEGEMAN, 1988] griffen die von [RITTEL und KUNZ, 1970] geschaffene *IBIS* Notation auf und entwickelte eine Software, mit dessen Hilfe sich sowohl eine grafische Repräsentation der IBIS-Elemente samt Verbindungen in Form eines Diagramms, siehe Abbildung 2.4, als auch das kollaborative Arbeiten über das Netzwerk realisieren lies.

Die bereits durch die IBIS-Methode bekannten Elemente ließen sich in Echtzeit von mehreren Nutzern an verschiedenen System erstellen, verbinden und modifizieren. Erweitert wurden sie durch einige weitere Elemente, die beispielsweise das Einbinden externer Dokumente erlaubten. Die Implementierung einer Nachrichtenschicht sorgte

für die Synchronisation sämtlicher Daten über die Systemgrenzen hinweg.

Da *gIBIS* für aktuelle SUN Workstations mit Farbmonitoren entwickelt wurde, konnte die Oberfläche konsequent auf die Nutzung von Farben als Indikator für verschiedene Arten von Elementen genutzt werden. Zur Unterstützung der damals noch weit verbreiteten monochromen Monitore repräsentierten Icons dennoch weiterhin die verschiedenen Elementtypen.

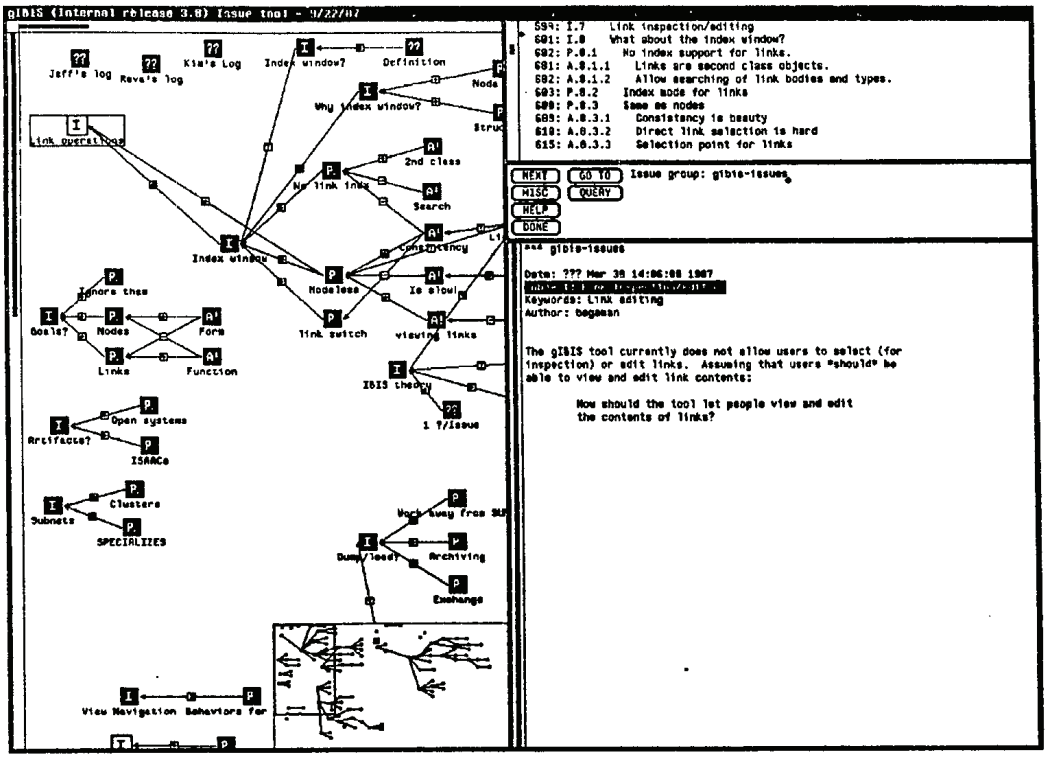


**Abbildung 2.4:** gIBIS Screenshot

### 2.2.6 Rationale

*Rationale*[12], ein proprietäres käuflich zu erwerbendes Tool zur Gestaltung von Argumentationsdiagrammen, bewirbt sich als effektive Hilfe beim Schreiben von argumentativen Essays, Meinungsbildern oder Textanalysen sowie zur Dokumentation von Gruppendiskussionen und zur Vorbereitung von Reden. Es soll weiterhin die Fähigkeiten des Analysierens und des kritischen Denkens seiner Nutzer schulen.

[12]http://rationale.austhink.com – Stand 31. Juli 2010

Die Oberfläche zeigt ein modernes Auftreten und entspricht dem Look&Feel, siehe Abbildung 2.5[13], der seit Microsoft Office 2007 eingeführten neuen Oberflächenstruktur. Das schnelle Erstellen, Verändern, Betrachten und Veröffentlichen von Diagrammen ist einer der Hauptmerkmale von *Rationale*. [VAN GELDER, 2007]

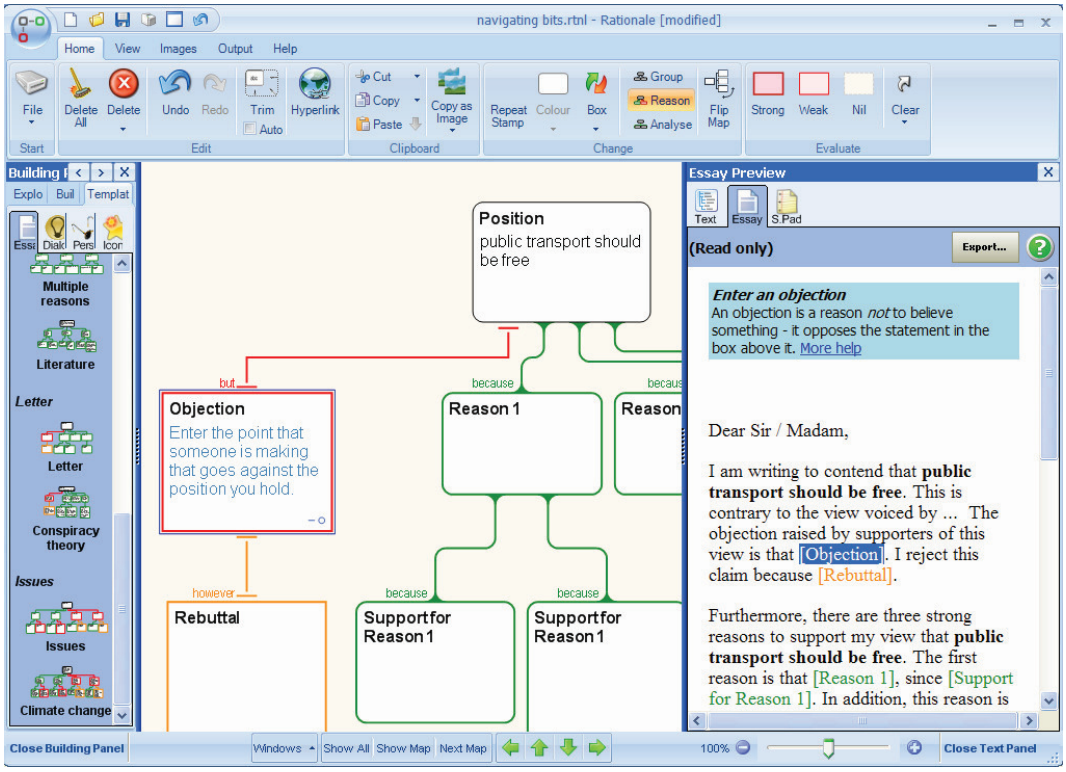


**Abbildung 2.5:** Rationale Oberfläche

Erstellen lassen sie sich mit Hilfe von Elementen wie beispielsweise Boxen, die durch Pfeile verbunden werden können. Da *Rationale* sich als universelles Tool versteht, werden bereits über 30 verschiedene Elementtypen mitgeliefert. Dennoch können weitere Elemente den eigenen Bedürfnis entsprechend konfiguriert werden. Eine Overview-Map, sowie das automatische Positionieren und Anordnen von Elementen vereinfachen das Arbeiten mit großen Diagrammen. [SCHEUER et al., 2010]

[13] Entnommen http://assets.rationale.austhink.com/pdf/gettingstartedguide.pdf – Stand: 23.7.2010

## 2.3 Zusammenfassung

Auch wenn in diesem Kapitel nur ein kleiner Teil der auf dem Markt befindlichen Argumentationssysteme betrachtet wurde, lassen sich dennoch die schier grenzenlosen Einsatzzwecke solcher System erahnen. Die Erkenntnis, dass jeder Anwendungsbereich seine spezifischen Anforderungen an solche System stellt, zeigt zudem die Herausforderungen, an denen sich das in dieser Arbeit betrachtete LASAD-Framework (siehe Kapitel 6), beweisen muss.

# 3 LARGO, ein intelligentes Tutorensystem für Jura-Studenten

*LARGO*, als intelligentes Tutoren- und Argumentationssystem entwickelt, soll Studierende der Rechtswissenschaften beim Erlernen von juristischen Argumentationstrukturen unterstützen.

Hauptbestandteil ist dabei ein Transcript (siehe Abbildung 3.1 links) einer Gerichtsverhandlung, desen Handlungs- und Argumentationsverlauf in Form eines Diagrammes visualisiert werden soll. Relevante Textpassagen werden durch bestimmte Boxtypen (*Fact*, *Hypothetical* und *Test*) repräsentiert und mit verschieden geartete Verbindungen untereinander verknüpfen. Zur Wahl stehen *modified to*, *destinguished from*, *analogized to*, *leads to* und eine *general relation*.

Nutzer werden dabei aktiv von einer Feedback-Engine unterstützt, die den Nutzer während des gesamten Prozesses auf „Schwächen“ im seinem erstellen Diagramm hinweist.

In Abschnitt 3.1 wird die zugrunde liegende Analyse-Technologie nochmals näher beschreiben.

Als technische Plattform wurde das *Cool Modes*-Framework[1] [PINKWART, 2003] verwendet. Eine in Java verfasste plugin-basierte Architektur für graphenbasierte kollaborative Modellierungssysteme. Weitere Details werden im Abschnitt 3.2 behandelt.

Abschnitt 3.3 befasst sich mit der Evaluierung des *LARGO*-Systems im praktischen Einsatz. Nicht zuletzt die Frage „Are Diagrams Better Than Text for Teaching Argumentation Skills?“ [PINKWART et al., 2008] war hierbei von zentraler Bedeutung.

[1] http://www.collide.info/index.php/Cool_Modes/de – Stand: 31. Juli 2010

Der letzte Abschnitt fasst die gesammelten Erkenntnisse und Probleme zusammen und skizziert mögliche Lösungsansätze, die schlussendlich zu dieser Arbeit geführt haben.

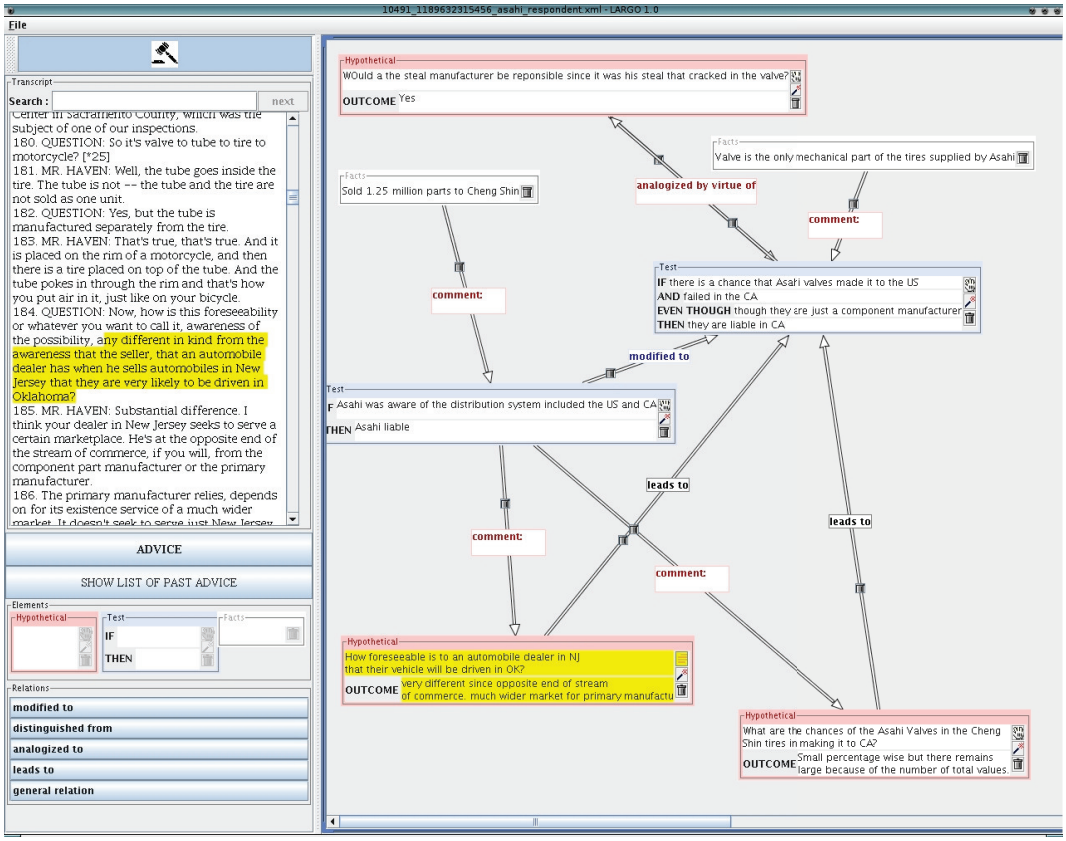


**Abbildung 3.1:** Oberfläche des alten *LARGO*-Systems

## 3.1 Diagrammanalyse und Feedback-Engine

Wie schon zuvor erwähnt ist *LARGO* in der Lage dem Studierenden ein direktes Feedback beim Erstellen der Diagramme zu geben. Da es in dieser Domäne meist keine eindeutigen Lösungen gibt und eine inhaltliche Interpretation der Textinhalte mit heutigen Verfahren ebenso unmöglich ist, können inhaltliche Fehler nicht auf direktem Wege ermittelt werden.

Dennoch versucht *LARGO* „Schwächen“ im Diagramm zu ermitteln. Hierbei werden strukturelle, kontextbezogene und inhaltliche Schwächen unterschieden.

[PINKWART et al., 2006a]

Strukturelle Schwächen lassen sich durch Analyse der Elemente eines Diagramms ermitteln. Hypothesen stehen normalerweise in Verbindung mit Fakten und überprüfen Tests auf Ihre Glaubwürdigkeit. Werden solche Verbindungen in der Struktur nicht erkannt deutet dies auf eine Schwäche im Diagramm hin. Solche Schwächen sind unabhängig vom verwendeten Protokoll und lassen sich als Graph-Grammatiken implementieren.

Kontextbezogene Schwächen sind dahingegen durchaus vom verwendeten Transcript abhängig sind. Werden für wichtige Teile des Protokolls keine Elemente erstellt oder mit falschen Elementen verbunden, stellt dies eine Schwäche im Kontext des Protokolls dar. Zur Ermittlung solcher Schwächen sind zusätzliche protokollabhängige Parameter notwendig.

Konnten strukturelle und kontextbezogene Schwächen noch auf verhältnismäßig einfache Weise durch entsprechend detaillierte Grammatiken ermittelt werden, fällt es selbst Tutoren schwer inhaltliche Schwächen eines Argumentationsdiagramms zu ermitteln. Zu diesem Zweck verwendet *LARGO* eine Art kollaborativen Filteralgorithmus. Hierbei wird ausgenutzt, dass durch die strukturellen und kontextbezogenen Regeln systemseitig bekannt ist welches Diagrammelement welchen Teil des Protokolls beschreibt. Wurde eine entsprechende Passage des Protokolls durch den Nutzer annotiert bietet *LARGO* eine Auswahl von Beschreibungsalternativen der gleichen Textpassage an, die durch andere Nutzer erstellt wurde. Dabei wird der Nutzer aufgefordert die Alternativen zu bestimmen, die er ebenfalls als gut erachtet. Mit einem Abgleich dieser Daten und einer bereits existierenden heuristischen Bewertung kann somit abgeschätzt werden, ob der Nutzer die entsprechende Passage richtig verstanden hat.

## 3.2 Technische Plattform

*LARGO* wurde auf Basis des *Cool Modes*-Framework entwickelt. *Cool Modes* ist ein Framework für kollaborative Systeme. Entwickelt, um Diskussionen und kooperative Modellierungsprozesse zu unterstützen. Kooperation wird dabei allgemein in Form von gemeinsamen Arbeitsbereichen realisiert. Sämtliche Objekte sowie ihre grafischen Repräsentanten können dazu synchronisiert werden und lassen sich samt

der zugrunde liegenden Semantik frei konfigurieren. Dies ermöglicht das Erstellen von domänenspezifischen Plugins mit der Möglichkeit interpretative Funktionen wie Datenanalyse und Feedback-Engine zu integrieren. [PINKWART, 2003]

*LARGO* selbst wurde als Einzelnutzer-Tool ohne eine kollaborative Zusammenarbeit der Teilnehmer konzipiert. Dies bedeutet, dass der eigene Arbeitsbereich nicht mit anderen Nutzern geteilt wird, selbst wenn dies dank des *Cool Modes*-Framework generell möglich gewesen wäre. Ganz verzichtet *LARGO* aber nun doch nicht auf die Möglichkeiten der Kollaboration. Wie schon in Abschnitt 3.1 erwähnt, werden die kollaborativen Filter-Methoden auf diese Art und Weise realisiert.

*Cool Modes* und damit auch *LARGO* wurden in Java entwickelt. Dessen Plattformunabhängigkeit ist es zu verdanken, dass *LARGO* sowohl für Microsoft Windows als auch für Linux zur Verfügung steht. Allerdings setzt dies auf Clientseite eine installierte *JVM* voraus, ebenso muss die Software auf jedem System einzeln und bei einem Update erneut installiert werden und sorgt so für erheblichen administrativem Aufwand. Im späteren Einsatz zeigte sich, dass *LARGO* nicht unter jeder Java-Version gleichermaßen funktioniert. Diese Einschränkung steigerte den erforderlichen Setup-Aufwand in heterogenen Rechnernetzen, wie sie beispielsweise an Universitäten üblich sind enorm.

Eine zentrale Datenspeicherung der einzelnen Instanzen wurde nicht implementiert. Zur späteren Datenanalyse müssen daher die erstellten Log-Dateien einzeln von den Client-Rechnern kopiert und zusammengetragen werden.

## 3.3 Frühere Studien

Das *LARGO*-System wurde 2006 und 2007 im Zuge zweier Studien an der *University of Pittsburgh* evaluiert. Dabei wurde die aufgestellte Hypothese überprüft, ob *LARGO*'s grafische Repräsentationen und Feedback-Mechanismen Studierenden helfen würden, besser zu lernen als mit der Hilfe eines rein textbasierten Tools.

Bestätigte die erste, mit freiwilligen Teilnehmern, durchgeführte Studie noch die entsprechende Hypothese, konnte das *LARGO*-System in einer zweiten, für die Teilnehmer verpflichtenden, Studie nicht länger überzeugen.

Die erste Studie wurde begleitend zum im ersten Studienjahr stattfindenden „Legal Process Course“ geführt. [PINKWART et al., 2007]

28 Studierende meldeten sich freiwillig und nahmen an den vier Sitzungen der Studie teil. Sie wurden in zwei Gruppen eingeteilt. Die eine Hälfte der Teilnehmer nutzte das *LARGO*-System, die andere Hälfte fungierte als Kontrollgruppe unter Verwendung eines rein textbasierten Programms. Zur späteren Erfolgskontrolle wurde in Sitzung 1 ein Pre- und in Sitzung 4 ein Post-Test durchgeführt. Diese Tests sollten mit multiple choice-Fragen die Argumentations- und Beweisführungsfähigkeiten der Teilnehmer vor und nach den Sitzungen 2 und 3 bewerten. Während der Sitzungen 2 und 3 wurden den Teilnehmern jeweils ein Verlaufsprotokoll eines Falls gegeben, den sie ihrer Gruppe entsprechend entweder grafisch oder in Textform repräsentativ aufbereiten sollten.

Auf Grundlage dieser Daten konnte belegt werden, dass die Verwendung des *LARGO*-Systems besonders bei weniger erfahrenen Studierenden zu einer, im Verhältnis zur Kontrollgruppe, stärkeren Verbesserung der entsprechenden Fähigkeiten führte. Bereits erfahrene Teilnehmer konnten ihre Bewertung indes nicht nachweisbar verbessern. Das Ergebnis dieser Studie bestätigt somit die zuvor aufgestellte Hypothese.

Ein weitere Studie fand 2007 unter ähnlichen Bedingungen statt. [PINKWART et al., 2008] Im Unterschied zur ersten Studie war die Teilnahme im Rahmen des „Legal Process Cource“ diesmal jedoch für alle 85 Teilnehmer verpflichtend. Eine finanzielle Entschädigung gab es im Vergleich zum vorherigen Jahr ebenfalls nicht.

Sechs Sitzungen mussten absolviert werden, wobei Sitzung 1 und 5 dem Pre- und Post-Test aus der ersten Studie entsprachen. Während der Sitzungen 2, 3 und 4 arbeiteten die Teilnehmer an verschiedenen Fällen unter Verwendung des *LARGO*-Systems und einem textbasierten-Tool. Sitzung 6 wurde genutzt, um unter den Teilnehmern einen Wissenausgleich herbeizuführen, dazu wurden die Ergebnisse beider Gruppe präsentiert.

Die zweiten Studie wollte die gewonnenen Ergebnisse aus dem Jahr zuvor jedoch nicht bestätigen. Es gab keine signifikanten Unterschiede zwischen beiden Gruppen. Die aufgestellte Hypothese konnte nicht bestätigt werden.

Mögliche Gründe für die Unterschiede zwischen den beiden Studien könnten unter anderem die Motivation der Teilnehmer und deren Umgang mit dem System selbst sein.[PINKWART et al., 2008]

Freiwillige Teilnehmer von Studien sind meist von vornherein hoch motiviert und bereit mitzuarbeiten. Finanzielle Anreize fördern diese Motivation zusätzlich. Die zweite Studie indes hatte keinerlei Anreize für die Teilnehmer. Sie wurden weder finanziell entlohnt noch hatte der Grad der Mitarbeit der Studierenden Auswirkungen auf ihre späteren Noten. Kurzum. Es gab neben der Pflicht, an der Studie teilzunehmen, keinerlei weitere Anreize, die Details von *LARGO* kennenzulernen und anzuwenden. Dementsprechend konnte man feststellen, dass Schlüsselfunktionen von *LARGO* wie beispielsweise die Feedback-Engine kaum von den Teilnehmern genutzt wurden. Da laut der Hypothese aber gerade diese Funktionen von *LARGO* für die Lernerfolge der Studierenden verantwortlich sein sollen, ist es nicht verwunderlich, dass die Fertigkeiten sowohl von wenig- als auch vielerfahrenen Teilnehmern keine feststellbaren Verbesserungen gegenüber der Kontrollgruppe zeigten.

Wie bereits erwähnt wurden die Schlüsselfunktionen des *LARGO*-Systems von den Teilnehmern kaum benutzt. Dies lässt sich mit der fehlenden Motivation, Funktionen des Systems zu erforschen, erklären, verdeutlicht aber gleichzeitig die Notwendigkeit, *LARGO* so zu modifizieren, dass Funktionen weiterhin attraktiv für die Nutzer bleiben. Dies scheint die aktuelle Version nicht im nötigen Maße zu leisten.

## 3.4 Zusammenfassung

Die Studien haben gezeigt, dass das Prinzip hinter dem *LARGO*-System funktionieren kann. Sie zeigten ebenso deutlich, dass *LARGO* bei seinem jetzigen Entwicklungsstand Schwächen aufweist, die einen praktischen Einsatz als Tutorensystem nicht erlauben. Für den erfolgreiche Einsatz ist es unabdingbar die fehlende Motivation der Teilnehmer durch Veränderungen am System selbst zu kompensieren.

Dass das *LARGO*-System nicht nur Probleme mit der Motivation der Nutzer hat, zeigten die in Abschnitt 3.2 beschriebenen Probleme der technischen Plattform selbst. Gerade in Fachbereichen wie den Rechtswissenschaften, die naturgemäß nicht direkt etwas mit Informationstechnologien gemein haben, muss ein solches System einfach zu installieren und zu warten sein. Es muss schlichtweg ohne großen Aufwand in heterogenen Rechnernetzen zu betreiben sein und daher technische Abhängigkeiten auf ein Minimum reduzieren.

Dies führte schlussendlich zu der Entscheidung das *LARGO*-System grundlegend zu überarbeiten. Um die technischen Probleme, die zumeist der aktuellen Plattform selbst zuzuschreiben waren, zu lösen erschien ein Wechsel der Plattform unabdingbar. Im Bereich der Web-Technologien war in den letzten Jahren ein enormer Fortschritt zu verzeichnen. Es lag daher nahe das *LARGO*-System auf diese neue Technologie zu migrieren, bietet sie doch adäquate Lösungen für einen Großteil der skizzierten Probleme.

Damit einhergehend erschien es sinnvoll das Bedienkonzept an die Gepflogenheiten aktueller Desktop-Anwendungen anzugleichen. Der vermehrte Einsatz von Drag&Drop-Techniken und ein modernes Look&Feel sollten ebenfalls mit in die neue Version einfließen und einen intuitiven Umgang mit dem System ermöglichen.

Da das *LARGO*-System als reine Einzelnutzer-Anwendung konzipiert wurde, erschien die Frage interessant, ob durch die Erweiterung des Systems um kooperative Funktionen eine nachhaltige Verbesserung erzielt werden könne.

Ein Großteil dieser Ideen soll im Rahmen dieser Arbeit implementiert und abschließend auf ihren Nutzen hin evaluiert werden. Eine detaillierte Beschreibung der Ziele lässt sich daher im nächsten Kapitel finden.

# 4 Die Ziele dieser Arbeit

**„Migration und Erweiterung von Software unter Berücksichtigung neuer Webtechnologien, mit abschließender Evaluierung, am Beispiel von „LARGO“, eines intelligenten Tutorensystems für Jurastudenten.“**

So hätte ein etwas ausführlicherer Titel dieser Arbeit lauten können. In ihm sind die Hauptziele klar erkennbar.

Bisher gab Kapitel 2 einen allgemeinen Überblick über die Welt der computergestützte Argumentationssysteme. Kapitel 3 befasste sich im speziellen mit dem alten *LARGO*-System, bot einen Einblick in dessen Geschichte und vermittelte zudem die Motive, die zur Formulierung der im Folgenden aufgeführten Ziele führte.

*Migration*, *Erweiterung* und *Evaluation* sind die Schlagwörter des Titels, die sich in den Zielen gleichsam wiederfinden.

Migration Wie der Titel schon propagiert war die „Migration“ sämtlicher Funktionen von LARGO auf eine neue Webtechnologie eines der gesteckten Hauptziele. Im Detail beinhaltete dies:

- Die Suche nach einem aktuellen *Rich Internet Application* Framework, welches den Installations- und Wartungsaufwand auf ein Minimum reduziert.
- Die Übernahme sämtlicher Funktionen des alten LARGO-Systems auf die neue Softwarearchitektur.
- Dessen Einbettung in eine, dem State of the Art entsprechenden, neuen Oberfläche, die durch das vermehrte Nutzen von Techniken, wie Drag&Drop, sowie einer graphischen Neugestaltung dem Empfinden aktueller Anwendung entspricht und daraus resultierend den Umgang mit LARGO erleichtern soll.

**Erweiterung** Frühere Studie sowie der praktische Einsatz an der Universität Pittsburgh zeigten, dass das bestehende LARGO-System durchaus Möglichkeiten zur Optimierung bietet. In Absprache mit den Partnern in Pittsburgh soll, im Rahmen dieser Arbeit, das bestehende Konzept durch folgende Punkte erweitert werden:

- Erweitertes Transcript-Highlighting
- Integriertes Tutorial- und Questionnaire-Framework

**Evaluierung** Die Neuimplementierung von LARGO sollte im Rahmen einer kleinen Usability-Studie in Pittsburgh auf ihre Lauffähigkeit und Funktion getestet werden.

# 5 RIA - Rich Internet Application Frameworks - Die Qual der Wahl!

## 5.1 Was ist eine Rich Internet Application?

*Rich Internet Applications* haben in den vergangen Jahren die Gattung der Web-Anwendungen revolutioniert. Kämpften traditionelle Web-Anwendungen noch mit ihrer großen Beschränktheit in Bereichen der Benutzeroberflächen und Bedienbarkeit [DEITEL und DEITEL, 2008], glänzen die neuen Techniken mit breit gefächerter Offenheit. Effiziente und mächtige graphische Bedienoberflächen lassen die Grenzen zu klassischen Desktop-Anwendungen zunehmend verschwimmen. Sie vereinen moderne Bedienkonzepte wie Drag&Drop-Techniken und Shortcuts sowie ein zeitgemäßes Look&Feel.

*Rich Internet Applications* sind komplexe Web-Anwendungen mit gesteigerten Anforderungen an die Client-Architektur, asynchrone Kommunikation mit Serverdiensten und einer großen Auswahl an UI-Widgets.
Sie sind ein Resultat der verbesserten Technologien im *World Wide Web* der heutigen Zeit. Nicht zuletzt die gesteigerten Übertragungsraten moderner Techniken wie DSL und leistungsfähigeren Clientrechnern sind maßgeblich an der steigenden Verbreitung beteiligt. [BUSCH und KOCH, 2009]

Erstmals ist es möglich komplexe Anwendungen von den Zwängen klassischer Desktop-Anwendungen zu befreien. Anwendungen, die aufgrund ihrer Anforderungen zuvor nur als lokal installierte Software zur Verfügung standen, lassen sich nun auf der Basis aktueller *Rich Internet Application Frameworks* entwickeln und ohne clientseitige Installation in webbasierten Laufzeitumgebungen betreiben. Dies ist zumeist ein aktueller Browser der gegebenenfalls mit Plugins um entsprechende Fähigkeiten erweitert wird.

Anwendungen wie *Google Apps*[1], die sich erfolgreich als Konkurrenz zu ihren klassischen Desktop-Pendants etabliert haben belegen die Leistungs- und Konkurrenzfähigkeit von Rich Internet Applications auf.

In den folgenden Abschnitten werden einige der aktuellen *Rich Internet Application Frameworks* eingehender betrachtet sowie ihre Vorzüge und Nachteile aufgezeigt.

Zur Realisierung der in Kapitel 4 beschrieben Ziele stellen *Rich Internet Applications* die ideale Plattform da. Client-Rechner werden von den Zwängen lokaler Installationen befreit und reduzieren somit den anfallenden Setupaufwand im täglichen Einsatz auf ein Minimum. Zudem erlauben sie es gleichzeitig eine zeitgemäße graphische Benutzeroberfläche zu implementieren, sodass die Usability von *LARGO* weiter verbessert werden kann.

Entscheidende Kriterien bei der Wahl des passenden *Rich Internet Application Framework* für die Migration des *LARGO*-Systems waren unter anderem die Abhängigkeiten von bestimmten Betriebssystemen, Browsern und etwaigen benötigten Plugins. Ebenso wichtig ist die zugrunde liegende Lizenz und die damit möglicherweise einhergehenden Kosten sowie die nötige Programmiersprache und die zur Verfügung stehenden Entwicklungsumgebungen.

## 5.2 Plugin oder AJAX basiert - zwei Wege zum Ziel

Im Laufe der Zeit haben sich zwei verschiedene Ansätze herauskristallisiert, *Rich Internet Applications* zu realisieren. Der eine Ansatz setzt dabei auf spezielle Plugins, die Web-Browser um entsprechende Fähigkeiten erweitern. Der andere Ansatz nutzt Standardtechniken aktueller Web-Browser und verzichtet auf jegliche zusätzlich zu installierende Software.

Der besseren Übersicht wegen werden die in diesem Kapitel vorgestellten Frameworks in folgende Kategorien unterteilt:

- Frameworks, die weitere Plugins nutzen
- Frameworks, die einzig auf Standardtechniken (*AJAX*) setzen

[1] siehe: http://www.google.com/apps/ – Stand: 31. Juli 2010

Da die Anzahl der aktuell zur Verfügung stehenden Implementierungen den Rahmen dieser Arbeit sprengen würde, werden im Folgenden nur einige besonders relevante Frameworks behandelt. Eine umfassendere Liste an konkreten Implementierungen halten Quellen im Internet vor.[2]

## 5.3 Frameworks auf Plugin-Basis

### 5.3.1 Adobe Flash

| | |
|---|---|
| Entwickler | Adobe Systems |
| Lizenz | proprietär |
| Sourcecode | Action Script |
| Runtime | Flash Player |
| www | www.adobe.com |

Zu allererst findet *Adobe Flash*, das momentan wohl bekannteste Browser-Plugin, seine verdiente Beachtung. Ursprünglich 1996 von *Macromedia* als multimediales Browser-Plugin veröffentlicht, sollte es visuelle und akustische Animationen in die damals noch recht triste Welt der meist statischen Web-Auftritte bringen.

Mittlerweile wird *Flash* von *Adobe Systems* weiterentwickelt und befindet sich nach eigenen Aussagen [ADOBESYSTEMS, 2010] auf mehr als 99% aller Desktoprechner. Laut [RIASTATS][3] sind es immerhin vergleichbare 97%, was *Flash* die absolute Spitzenposition bei vergleichbaren Browser-Plugins sichert. Das proprietäre Plugin ist aktuell für alle bekannten Desktop-Betriebssysteme wie Microsoft Windows, Mac OS und Linux unentgeldlich verfügbar. An weiteren Versionen für mobile Betriebssysteme wird ebenso gearbeitet. Als erster Vertreter dieser Zunft wurde 2010 *Android 2.2*[4] mit *Flash*-Support ausgestattet.

[2] z.B. Wikipedia: http://en.wikipedia.org/wiki/List_of_web_application_frameworks – Stand: 31. Juli 2010

[3] *RiaStats* ist ein unabhängiger Webservice, der kontinuierlich die Web-Browser der Besucher von 100 Webseiten auf die zur Verfügung stehenden *Rich Internet Application Frameworks* untersucht. Dabei wird sowohl die Verfügbarkeit von *Adobe Flash* und *Microsoft Silverlight*, als auch von Java untersucht und entsprechend graphisch aufbereitet. Dabei kann tagtäglich auf eine hohe einstellige Millionenzahl an unabhängigen Web-Browsern zurückgegriffen werden.

[4] Android wird von Google als freies Betriebssystem für Mobiltelefone und andere mobile Endgeräte wie Netbooks und Tablet-PCs entwickelt. Weitere Details: www.android.com – Stand: 31. Juli 2010

1999 wurde *Flash* mit der Einführung von *ActionScript* um vollwertige Scripting-Fähigkeiten erweitert und entwickelte sich fortwährend zur verbreitetsten Plattform für Onlinespiele und -werbung. Mittlerweile dient es als vollwertige Basis für *Rich Internet Applications*.

Anwendungen lassen sich in einer objektorientierten Sprache namens *ActionScript* entwickeln. Sie folgt dabei weitestgehend den Spezifikationen des *ECMAScript*-Standard[5] und gleicht damit dem bekannteren *JavaScript* in Syntax und Semantik. Entsprechende Entwicklungsumgebungen werden von *Adobe* als Teil der *Adobe Flash Plattform*[6] kostenpflichtig vertrieben.

Die Entwicklung der Benutzeroberfläche findet getrennt vom Programmcode statt. Der Ursprung von *Flash* als Animations-Plugin ist noch deutlich zu erkennen. Es stehen schier unbegrenzte Möglichkeiten bei der Gestaltung der Oberflächen zur Verfügung. *Flash Catalyst* als entsprechendes Tool erinnert dabei stark an ein ausgewachsenes Grafikprogramm wie *Adobe Photoshop* und verlangt dem Designer weitreichende Fähigkeiten ab. Bei entsprechenden Kenntnissen ermöglicht es aber eine graphisch hochwertige und individuelle Oberfläche.

Einmal kompiliert lassen sich die Anwendungen in jeden *Flash*-Player, unabhängig vom eingesetzten Betriebssystem, ausführen. Dank der breiten Unterstützung von Audio-, Video- und Bildformaten, einem integrierten Streaming-Protokoll (*RTMP*) und entsprechender Hardware-Unterstützung für Grafikbeschleuniger, Mikrofon und Webcam lassen sich mit *Flash* hochwertige und multimedial aufwändige Anwendungen erstellen. Selbst das Streamen und Abspielen von Videos in HD-Auflösung stellt mittlerweile kein Problem mehr dar.

[5] *Ecma International* ist eine weltweit operierende Non-Profit-Organisation, die sich der Standardisierung im Informations- und Kommunikationsbereich widmet. Prominente Standards sind unter anderem *ECMAScript* (JavaScript), C# und die *CLI*. Weitere Details: www.ecma-international.org – Stand: 31. Juli 2010

[6] Unter der *Adobe Flash Platform* werden sämtliche Entwicklungs- und Laufzeitprogramme zusammenfasst. Integrale Bestandteile sind unter anderem der *Flash Player* als Laufzeitumgebung, *Flash Professional*, *Flex* oder *Flash Catalyst* als Entwickungsumgebung, sowie *Flash Media Server*, *BlazeDS* und *LiveCycle DS* als Serverumgebung.

### 5.3.2 Adobe Flex

| | |
|---|---|
| Entwickler | Adobe Systems |
| Lizenz | proprietär / Open Source |
| Sourcecode | MXML + Action Script |
| Runtime | Flash Player |
| www | www.adobe.com |

Wie schon im Absatz 5.3.1 erwähnt, orientiert sich der Entwicklungsprozess von reinen *Flash*-Anwendungen mehr an Bedürfnissen von Grafik- und Webdesignern als an jenen der klassischen Softwareentwickler. Um im aufkeimenden Markt der *Rich Internet Applications* nicht den Anschluss zu verlieren, wurde von *Adobe* eigens zu diesem Zweck das *Flex Framework* entwickelt. Erstmals 2004 veröffentlicht stellt *Flex* eine Entwicklungsumgebung dar, die von Grund auf auf die Bedürfnisse von Softwareentwicklern ausgerichtet ist und als Basistechnologie auf *Adobe Flash* aufbaut.

Indem Adobe *Flash* und damit eine bereits etablierte und hoch verfügbare Technik als Basis für ihr neues *RIA*-Framework wählte, machte es die Notwendigkeit eines neuen Plugins unnötig. *Flex*-Anwendungen werden von einem Compiler in einem Zwischenschritt in reinen *ActionScript*-Code übersetzt und von da an wie jede andere *Flash*-Anwendung auch behandelt. Im Unterschied zu *Flash* werden *Flex*-Anwendungen in *MXML* verfasst, einer Auszeichnungssprache auf *XML*-Basis mit deren Hilfe sich Benutzeroberflächen samt ihrer Elemente deklarieren lassen. Die eigentliche Programmlogik wird weiterhin in *ActionScript* verfasst und als Code-Teil direkt in die *MXML*-Datei eingebettet. Die *Flex*-SDK wird im Gegensatz zu *Adobe Flash* unter einem Open Source-Modell vertrieben. Man erhofft sich dadurch eine breitere Akzeptanz und Unterstützung von der Entwicklergemeinschaft. [TAPPER et al., 2008]

*Adobe Flex* bringt standardmäßig bereits eine breite Masse an Bedienelementen wie Buttons, Bäumen, Tabellen, etc. mit und lässt sich über Addons um weitere Fähigkeiten wie Diagramme und Graphen erweitern. Die Integration von Serverdiensten lässt sich über integrierte Web-Services, vergleichbar der asynchronen Datenübertragung aus der *AJAX*-Welt (siehe Kapitel 5.4.1.1), realisieren.

Seit Version 3 lässt sich eine *Flex*-Anwendung ebenfalls unter der *Adobe AIR*[7]-

[7] Adobe AIR, auch *Adobe Integrated Runtime*, ist eine plattformunabhängige Laufzeitumgebung die das Erstellen von *Rich Internet Applications* für den Desktop ermöglicht. Siehe auch http://www.adobe.com/de/products/air/ – Stand: 31. Juli 2010

Laufzeitumgebung als Desktop-Anwendung betreiben und kann somit den Fesseln des Web-Browsers entrinnen. [PFEIL, 2008]

### 5.3.3 JAVA Applet

Mit der Bekanntgabe im Jahre 1995, dass der *Netscape Navigator*[8] Java offiziell unterstützen werde, begann der Siegeszug der Java-Technologie sowohl als eigenständige Programmiersprache wie auch im Internet. [BANK, 1995]


| | |
|---|---|
| Entwickler | Sun/Oracle |
| Lizenz | proprietär/GPL |
| Sourcecode | Java |
| Runtime | JVM |
| www | www.java.com |

Bereits die erste veröffentlichte Java-Version brachte die Unterstützung für das Java-Applet-Framework mit. Eine Spezifikation mit der sich in Java geschriebene Anwendungen einfach in Webseiten einbinden ließen. Der Quellcode von Java-Applets, wird wie bei anderen Java-Projekten auch, in betriebssystemunababhängigen Java-Bytecode kompiliert und setzt damit eine *JVM* zur Ausführung auf Client-Seite voraus. Sie stehen aber für alle gängigen Betriebssysteme und Web-Browser zur Verfügung.

Da einem Applet der volle Funktionsumfang der *Java SE* zur Verfügung steht, lassen sich durchaus komplexe Anwendungen erstellen und im Web-Browser betreiben. Ebenso lassen sich spezielle Anwendungsfälle realisieren, die vom Web-Browser standardmäßig nicht unterstützt werden. Durch entsprechende Java-Bibliotheken lassen sich zum Beispiel hardwarebeschleunigte 3D-Umgebungen umsetzen oder spezielle Peripherie ansteuern. Eine Offenheit, die *Flash* und *Silverlight* vermissen lassen.

Negativ fällt allerdings die Dauer der Startphase eines Applets auf. Der eigentliche Download, das Starten der JVM und das anschließende Laden der Anwendung dauert bis heute im Vergleich zu anderen Lösung erheblich länger. Nicht zuletzt ein Grund,

[8] Der *Netscape Navigator* war in den Anfängen des WWW Mitte der 90'er der mit Abstand führende Web-Browser auf dem Markt. Er hatte zwischenzeitlich einen Verbreitungsgrad von ca. 90% erreicht, wurde dann aber Ende der 90'er durch den *Internet Explorer* fast gänzlich verdrängt.

warum die Popularität von Java-Applets in den letzten Jahren stark nachgelassen hat. Unabhängig davon ist die spätere Performance der eigentlichen Anwendung mit nativ installierten Desktop-Anwendungen vergleichbar. Größter Nachteil ist die Abhängigkeit von der auf Client-Seite installierten *Java SE*-Version. So kommt es in heterogenen Umgebungen mit verschiedenen Version häufig zu Inkompatibilitäten, die das fehlerfreie Ausführen eines Applets verhindern.

### 5.3.4 JAVA FX

Mit *JavaFX* bereichert seit 2008 ein Framework für *Rich Internet Applications* die Java-Spezifikationen. Es wurde als plattformunabhängige Lösung[9] konzipiert und soll neben Web-Browsern auch auf Settop-Boxen und mobilen Endgeräten zum Einsatz kommen. Es ist Bestandteil der JRE und setzt eine Installation der *Java SE Runtime* voraus, die auf Grund der großen Verbreitung bereits auf vielen Desktop-PCs wie auch mobilen Endgeräten zur Verfügung steht. Laut [RIASTATS] ist Java und damit *JavaFX* auf ca. 75% der Desktop-PCs lauffähig und befindet sich damit im Mittelfeld der klassichen Plugins. Die direkte Konkurrenz – namentlich *Flash* – kommt auf ca. 95%, *Silverlight* auf eine Verbreitung von insgesamt ca. 60%.


| | |
|---|---|
| Entwickler | Sun/Oracle |
| Lizenz | proprietär/GPL |
| Sourcecode | JavaFX Script |
| Runtime | JVM |
| www | www.javafx.com |

Anwendungen werden in *JavaFX Script* verfasst, eine an die Java-Sprache angelehnte und auf schnelle und grafisch ansprechende Entwicklung optimierte Scriptsprache. Da *JavaFX*-Anwendungen in Java-Bytecode übersetzt werden und daher eine klassische JRE voraussetzen, lassen sich auch reguläre Java-Klassen problemlos einbetten. So lässt sich der standardmäßige Funktionsumfang von *JavaFX* dem konkreten Anwendungsfall entsprechend erweitern.

Die Anwendungen selbst werden von einem Webserver an den Client ausgeliefert und dort per *Java Web Start* oder als *Java Applet* ausgeführt. Dies kann klassisch im

[9] Java und damit JavaFX ist für Microsoft Windows, Mac OS X, Linux und Solaris verfügbar.

Web-Browser geschehen. Es ist aber ebenso möglich entsprechende Anwendungen losgelöst von einem Web-Browser auf dem Desktop laufen zu lassen.

Sämtliche Elemente der Benutzeroberfläche lassen sich in ihrem Aussehen per *CSS* den konkreten Anforderungen entsprechend anpassen. Neben den standardmäßig mitgelieferten Elementen lassen sich auch eigene Elemente definieren. Eine 3D-Engine wird ebenfalls mitgeliefert. *JavaFX* bringt somit alles mit, um ansprechende Benutzeroberflächen schnell und unkompliziert zu erstellen.

Als Kommunikationskänale mit Serverdiensten unterstützt *JavaFX* von Haus aus *HTTP-GET*, *REST* und *Web-Services*. Die zuvor erwähnte Freiheit bei der Integration fremder Java-Bibliotheken eröffnet auch hier die Möglichkeit neuer Wege.

### 5.3.5 Microsoft Silverlight

*Silverlight* wird von Microsoft entwickelt und wurde 2006 erstmals als direkte Konkurrenz zu *Adobe Flex/Flash* veröffentlicht. *Silverlight* ist ein proprietäres Browser-Plugin ähnlich wie *Adobe Flash* und für Microsoft Windows wie auch Apple Macintosh verfügbar. Das mittlerweile in Version 4.0 erschienene Plugin ist offiziell mit Microsofts Internet Explorer sowie Mozilla Firefox und Apple Safari kompatibel.


| | |
|---|---|
| Entwickler | Microsoft |
| Lizenz | proprietär |
| Sourcecode | .NET Sprachen |
| Runtime | Silverlight (.NET Runtime) |
| www | www.silverlight.net |

Die Benutzeroberfläche von Anwendungen samt Multimediainhalten, Grafiken, Animationen und die Interaktion mit dem Benutzer werden bei *Silverlight* unter einer UI-Präsentationsschicht zusammengefasst, die sich vom *Windows Presentation Foundation*, kurz *WPF*, dem großen Bruder aus der Windows-Welt, ableitet. *WPF* ist Teil des aktuellen *.NET-Framework* und dient der Gestaltung von Anwendungsoberflächen. Dabei werden bereits eine Fülle von GUI-Elementen mitgeliefert, mit deren Hilfe sich schnell ansprechende Benutzeroberflächen gestalten lassen. Diese beruhen auf frei skalierbaren Vektorgrafiken und werden in einem *XML*-basierten Textformat, kurz *XAML*, definiert. Die Entwicklung eigener Bedienelemente ist ebenfalls möglich. Nicht

zuletzt durch die Unterstützung von Hardwarebeschleunigung, Webcams und Soundausgabe steht *Silverlight Adobe Flash* in nichts nach und ermöglicht eine performante Darstellung auch von komplexen Anwendungen im Web-Browser.

Die entsprechende Programmlogik lässt sich mit Hilfe einer abgespeckten Version des *.NET Frameworks* erstellen. Dabei liegt der Fokus nicht nur auf einer unterstützten Programmiersprache. Anwendungen für *Silverlight* lassen sich mit allen *.NET*-kompatiblen Sprachen erstellen. Dies sind Sprachen, die sich in die *Common Intermediate Language (CIL)* übersetzen lassen und somit zur Laufzeit von einer entsprechenden *Common Language Runtime (CLR)* in ausführbaren, dem Zielsystem angepassten, Maschinencode transformieren lassen. Prominente Beispiele sind C#, Visual Basic .NET oder C++/CLI.

Der größte Nachteil von *Silverlight* ist bis heute der enge Fokus auf Microsoft-eigene Produkte. Dies macht sich unter anderem in der schlechten Unterstützung anderer Betriebssysteme bemerkbar. So ist *Silverlight* für Betriebssysteme wie Linux und FreeBSD nicht verfügbar. Zwar gibt es mit *Moonlight*[10] seit Januar 2009 eine vom Mono-Projekt[11] entwickelte, freie Implementierung für *Silverlight*, die es ermöglicht entsprechende Anwendungen auch unter anderen Systemen lauffähig zu machen. Der Funktionsumfang hinkt der aktuellen *Silverlight*-Version allerdings um zwei volle Versionsnummern hinterher. So werden von *Moonlight* bisher nur die Spezifikationen von *Silverlight* 2.0 komplett erfüllt. Laut [RIASTATS] kommt *Silverlight* somit momentan auf eine Verbreitung von ca. 60%, davon fallen wiederum 60% auf die aktuelle Version 4.

[10] http://www.mono-project.com/Moonlight – Stand: 31. Juli 2010

[11] Mono ist eine Open Source Impementierung des Microsoft .NET Framework basierend auf den ECMA Standards für C# und der *Common Language Runtime*. http://www.mono-project.com/Moonlight –Stand: 31. Juli 2010

### 5.3.6 OpenLaszlo


| | |
|---|---|
| Entwickler | Laszlo Systems |
| Lizenz | CPL |
| Sourcecode | LZX (XML + Javascript) |
| Runtime | Flash, DHTML |
| www | www.openlaszlo.org |

Schon im Jahre 2002 und damit zwei Jahre bevor *Adobe* mit *Flex* das gleiche Konzept aufgriff, brachte die Firma *Laszlo Systems* den *Laszlo Presentation Server* auf den Markt. Als Reaktion auf die Einführung von *Flex* wurde dieser unter die *CPL* Open Source-Lizenz gestellt und unter dem neuen Namen *OpenLaszlo* veröffentlicht.

*OpenLaszlo* ist dabei ebenso wie *Adobes Flex* ein Entwicklungs-Framework für *Rich Internet Applications*, welches eine deklarative auf *XML* basierende Sprache verwendet, um Anwendungen und ihre Funktionen zu beschreiben. In dieser Sprache geschriebene Anwendungen werden durch einen Compiler in eine entsprechende Laufzeitumgebung, in beiden Fällen zum Beispiel *Adobe Flash*, übersetzt. Im Gegensatz zu *Flex* verfolgt *OpenLaszlo* aber den Ansatz nicht nur ein Zielsystem zu unterstützen.

Konkret werden Anwendungen für *OpenLaszlo* in *LZX*, einer eigenen objektorientierte *XML*-Sprache mit eingebettetem *JavaScript*, geschrieben. Neben *MXML* – welches bei *Adobe Flex* zum Einsatz kommt, ähnelt sie konzeptionell ebenso Microsofts *XAML* als auch Mozillas *XUL*. [LASZLOSYSTEMS, 2006]

Die offene Architektur von *OpenLaszlo* erlaubt die Übersetzung von in *LZX* geschriebenen Anwendungen in verschiedene Zielsysteme. Aktuell wird sowohl *Flash* als auch *DHTML* unterstützt. Im Folgenden werden die Möglichkeiten von *Flash* als Zielsystem genauer behandelt. Eine Betrachtung der *DHTML*-Umsetzung wird im Abschnitt 5.4.3 beschrieben.

Der *Adobe Flash*-Player (siehe auch Abschnitt 5.3.1) findet sich nach eigenen Angaben mittlerweile auf über 99% aller aktuellen Desktop-Rechner. [ADOBESYSTEMS, 2010] Er ist klein und leistungsfähig sowie hoch verfügbar und sorgt für Konsistenz der Anwendungen auch über Betriebssystemgrenzen hinweg. *Flash* stellt somit eine ideale Laufzeitumgebung für *Rich Internet Apllications* dar. Diese Vorteile gelten gleichermaßen für *OpenLaszlo*.

## 5.4 Frameworks auf AJAX-Basis

### 5.4.1 AJAX - die Grundlage vieler moderner Webanwendungen

Der Begriff *AJAX* steht für „Asynchronous JavaScript and XML“ und wurde von Jess James Garret als Oberbegriff für ein neu aufkommendes Designkonzept für Web-Anwendungen im Jahre 2005 eingeführt. [GARRETT, 2005] Er wird heute häufig synonym für alle Web-Anwendungen verwendet, die einen gegenüber herkömmlichen Webseiten erhöhten Bedienkomfort bieten. [JÄGER, 2008]

Abgesehen von dieser allgemeinen Nutzung des Begriffes beschreibt der Name *AJAX* deutlich welche klassischen Voraussetzungen Garrett an eine Web-Anwendung stellte. Zum Einen muss sie Daten *asynchron* mit dem Server austauschen. Die Anwendung muss clientseitig in *JavaScript* geschrieben sein und zum Schluss noch auf ein Austauschformat auf *XML* Basis zurückgreifen um Daten mit dem Server auszutauschen. Keine dieser Techniken wurde eigens für *AJAX* entwickelt, sondern waren schon 2005 verfügbar. Mit *AJAX* wurden sie erstmals in einem Designkonzept zusammengefasst, um gemeinsam eine neue Art von Web-Anwendungen realisieren zu können, die in ihrem Funktionsumfang und Bedienkomfort ihres Gleichen suchte.

Im Folgenden wird auf die drei Grundvoraussetzungen und deren Bedeutung nochmals näher eingegangen.

#### 5.4.1.1 A wie - asynchrone Datenübertragung

Die asynchrone Datenübertragung zwischen Client und Server stellt die Schlüsselkomponente von *AJAX* dar. Zwar ist in technischer Hinsicht – es sind beides *HTTP*-Requests – die asynchrone Übertragung von Daten keineswegs schneller als die klassische synchrone Übertragung. Dennoch sorgt die asynchrone Übertragung für eine erhöhte „Reaktionsbereitschaft“ beim Client.

Vor dem Einsatz von *AJAX* war es nötig, die komplette Seite im Web-Browser zu wechseln, um eine Anfrage an den Server zu stellen (siehe Abbildung 5.1). Jede Anfrage benötigt eine gewisse Zeit. Während dieser Zeit konnte der Anwender nicht weiter arbeiten. Er musste warten bis die Seite im Web-Browser erneut aufgebaut wurde. Diese „Zwangspausen“ führten dazu, dass viele Anwendungen nicht sinnvoll als

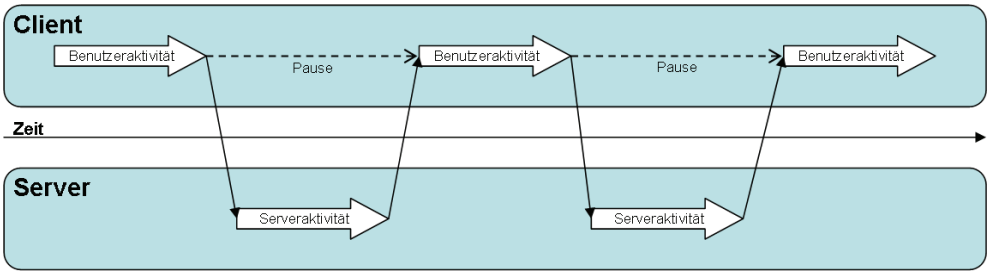


**Abbildung 5.1:** Synchroner HttpRequest

Web-Anwendungen realisiert werden konnten. Als Beispiel kann eine Tabellenkalkulation herangezogen werden. Hier müsste für jede Änderung einer Zelle die gesamte Anwendung samt Tabelle neu geladen werden.

Erst das Konzept der asynchronen Datenübertragung löst dieses Problem durch seine Nebenläufigkeit. Die *AJAX*-Engine erlaubt es Anfragen an den Server zu stellen ohne die Seite neu zu laden oder den Ablauf der Anwendung zu unterbrechen. Genauer heißt dies, der Benutzer kann weiterhin mit der Anwendung interagieren ohne jeweils auf die Antworten gestellter Serveranfragen warten zu müssen (siehe Abbildung 5.2).

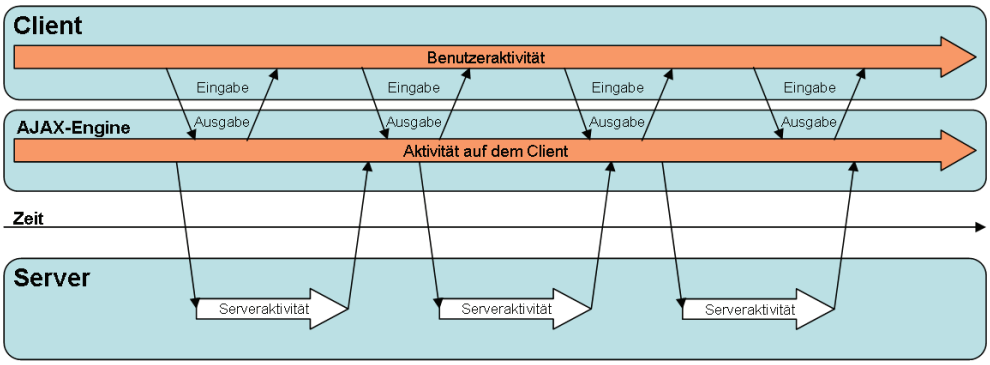


**Abbildung 5.2:** Asynchroner HttpRequest

Grundlage dieser Technik ist die *XMLHttpRequest*-API (kurz *XHR*). Sie dient der asynchronen Übertragung von *XML* oder anders formatierter Daten über das *HTTP*-Protokoll. Eingeführt wurde sie lange bevor *AJAX* ein Begriff war, von Microsoft als ursprünglicher Bestandteil von *Outlook Web Access*, einem Web-Mail-Client für den eigenen *Exchange Server*. Erstmals im *Internet Explorer* integriert, fingen die anderen Web-Browser-Hersteller an das Konzept mit ähnlichen Implementierungen in ihre Web-Browser aufzunehmen. Bisher ist die entsprechende API allerdings noch nicht

standardisiert worden und noch heute sind gewisse Inkompatibilitäten zwischen den verschiedenen Implementierungen vorhanden. Die W3C bemüht sich aktuell um eine entsprechende Standardisierung [W3C, 2009], um die zunehmende Verbreitung dieser vielversprechenden Technik zu fördern.

#### 5.4.1.2 JA wie - JavaScript

Wir wissen bereits, dass *JavaScript* per Definition die Basis sämtlicher *AJAX* Web-Anwendungen ist. Doch auch wenn es der Name vermuten lässt haben Java und *JavaScript* nicht viel gemein. Vielmehr waren es die Interessen des Marketings, die Netscape 1995 dazu verleiten ließ die von Brendan Eich geschriebene Skriptrache *Mocha* zuerst in *Livescript* und später dann in *JavaScript* umzubenennen, um so von dem damaligen Hype um Java zu profitieren. [EICH, 2005]

Zwar hat *Javascript* einen Großteil der Syntax und Benennungskonventionen mit dem Namensvetter Java gemein. Konzeptionell unterscheiden sie sich jedoch in weit größerem Umfang. So findet man bei *JavaScript* unter anderem weitreichende Wurzeln in funktionalen Sprachen wie *Lisp* oder *Scheme*, aber auch Elemente objektorientierter Sprachen wie Java und C# werden auf konzeptionaler Ebene aufgenommen. Dieses breit gefächerte Spektrum führt in vielerlei Hinsicht zu einer größeren Ausdrucksstärke als beispielsweise bei C++ oder Java. [JÄGER, 2008] Eine Tatsache, die in der Öffentlichkeit dennoch häufig verkannt wird und *JavaScript* noch heute als kleinen Bruder von Java abstempelt.

Sämtliche aktuellen Web-Browser beherrschen *Javascript* bereits in ihrer Grundkonfiguration. Eine Nachrüstung per Plugin ist nicht notwendig. Waren die konkreten Umsetzungen noch vor einigen Jahren für ihre schlechte Performance bekannt, hat nicht zuletzt die aufkeimende *AJAX*-Technik und die damit einhergehende Verbreitung von *Javascript* dafür gesorgt, dass aktuelle Implementierungen um ein vielfaches effektiver und schneller arbeiten.

Die allgemeine Verfügbarkeit und gute Performance dieser Technik macht sie zum idealen Kandidaten, um moderne *Rich Internet Applications* in der heutigen Web-Browser-Landschaft zu realisieren.

#### 5.4.1.3 X wie - XML

*XML*, auch *Extendable Markup Language*, bedeutet übersetzt nichts anderes als „Erweiterbare Auszeichnungssprache“. Sie definiert einen Standard[12] zur Speicherung von Daten, um diese zwischen sehr unterschiedlichen Systemen austauschbar zu machen. Dabei ist *XML* eine Meta-Sprache. Das heißt, dass sich mit ihrer Hilfe weitere Untersprachen definieren lassen. Prominente Beispiele sind standardisierte Sprachen wie *XHTML*, *SVG* oder auch *SOAP* und *WSDL*, die bei der Realisierung von *Web-Services* eine wichtige Rolle spielen. All diese Sprachen beruhen auf der durch *XML* vorgegebenen Syntax und werden nur durch ihre spracheigene Grammatik eingeschränkt. Solche Grammatiken lassen sich durch sogenannte Schema-Sprachen wie *DTD* oder *XML-Schema/XSD* beschreiben und definieren somit sämtliche Objekte einer Untersprache.

Eine solche Sprachhierarchie ermöglicht es, ohne großen Aufwand Sprachen für bestimmte Zwecke zu definieren, dabei aber gleichzeitig auf ein breites Spektrum an allgemein verfügbaren Implementierungen und Tools zurückzugreifen. Es lassen sich für jede aktuelle Programmiersprache bereits fertige *XML*-Parser finden, sodass eine große Interoperabilität über Betriebssystems- und Programmiersprachen-Grenzen hinweg ohne großen Aufwand realisiert werden kann.

*XML* mit seiner „Offenheit“ und der stetig steigenden Verbreitung stellt somit den idealen Kandidaten als Austauschformat für das damalige *AJAX*-Konzept dar. Zwar ist es technisch gesehen nicht zwingend notwendig Nutzdaten per *XML* zu formatieren und wird längst nicht in jeder *AJAX*-Anwendung so gehandhabt, dennoch spielt es auch heute noch eine wichtige Rolle in modernen *RIA*-Anwendungen.

[12] http://www.w3.org/TR/xml11/ – Stand: 31. Juli 2010

### 5.4.2 Google Web Toolkit

| | |
|---|---|
| Entwickler | Google |
| Lizenz | Apache-Lizenz |
| Sourcecode | Java |
| Runtime | DHTM/Javascript |
| www | code.google.com/webtoolkit/ |

Wie es der Name schon vermuten lässt, wird dieses Framework zur Entwicklung von Web-Anwendungen federführend von *Google* entwickelt. Erstmals im Jahr 2006 als freie Software[13] veröffentlicht, hat es nach mittlerweile neun Releases die Versionsnummer 2.0 erreicht und wird mehr und mehr von Google selbst bei seinen Softwareprojekten verwendet, was nicht zuletzt von der Stabilität und Qualität dieses Frameworks zeugt. Als prominentes Beispiel ist *Google Wave*[14] zu nennen, ein von Google entwickeltes Echtzeitsystem zur Kommunikation und Zusammenarbeit, dessen Client mit dem *Google Web Toolkit* realisiert wurde.

Anwendungen lassen sich als Java-Projekt entwickeln und werden später vom GWT-eigenen Java-zu-Javascript- Compiler in nativen Javascript-Code übersetzt. Dabei wird die lauffähige Java-Anwendung in eine äquivalente Javascript-Anwendung transformiert. Die Vorteile liegen klar auf der Hand: Die Anwendungen sind schlank und schnell, laufen in jedem aktuellen Web-Browser und das lästige Lernen von Javascript und dessen browserspezifischen Besonderheiten entfällt. Der entsprechende Compiler sorgt dafür, dass sich auf allen aktuellen Web-Browsern das gleiche Look&Feel einstellt.

Zwar fristet die kompilierte Fassung einer *GWT*-Anwendung ein Dasein als Javascript-Anwendung, während der Entwicklung läuft sie jedoch als Java-Code in einer JVM. Dieser als *Development Mode* bezeichnete Modus ermöglicht das einfache Debugging von Anwendungen, da sich die gesamte *Standard Java Debugger*-Infrastruktur nutzen lässt. In Verbindung mit einer Entwicklungsumgebung, wie *Eclipse* und dessen Debugger-Frontend eine effektive Möglichkeit Fehler in der eigenen Anwendung zu finden. Seit Version 2.0 ist es weiterhin möglich das Verhalten der Anwendung im gewünschtem Web-Browser direkt zu testen. Dafür werden für alle gängigen Web-Browser entsprechende Plugins bereitgestellt.

[13] Veröffentlicht unter der Version 2.0 der Apache-Lizenz
[14] http://wave.google.com/ – Stand: 31. Juli 2010

Bei der Gestaltung der Anwendungen lässt schon die umfassende Sammlung an Widgets, die *GWT* standardmäßig mitbringt, kaum Wünsche offen. Sollte dennoch einmal ein gewünschtes Widget fehlen, so ist die Wahrscheinlichkeit hoch in der großen und aktiven Entwicklergemeinde fündig zu werden. Nicht zuletzt durch weitere Frameworks wie *Ext GWT*[15] wird der Funktionsumfang von *GWT* nochmals erhöht. Sie bringen neben erweiterten Funktionen eine grafisch anspruchsvollere Widget-Sammlung mit und machen das Erstellen von *Rich Internet Applications* noch ansprechender und einfacher. Dabei lässt sich das Auftreten der Widgets ohne großen Aufwand mit Kenntnissen von HTML, DOM und CSS den eigenen Wünschen entsprechend anpassen.

Für die Kommunikation mit Serverdiensten bietet *GWT* gleich mehrere Wege. Die einfachste und komfortabelste Art ist das von *GWT* mitgelieferte *GWT RPC Framework*. Bei dieser transparenten Art Anfragen an ein Java-Servlet zu senden, sorgt *GWT* für sämtliche anfallenden Low-level-Aufgaben, wie beispielsweise das Serialisieren der Daten. Ebenso ist es einem freigestellt auch andere RPC-Mechanismen in seine Anwendung zu integrieren. Beispielsweise *JSON* oder weitere externe Bibliotheken. Da serverseitig auf Java-Bytecode gesetzt wird, stellt der Einsatz fremder Java-Bibliotheken auf Serverseite generell keine Hürde dar und erlaubt die einfache Verknüpfung von *GWT* mit den übrigen in der Java-Welt verfügbaren Schnittstellen.

### 5.4.3 OpenLaszlo

Dass *OpenLaszlo* in diesem Kapitel ein weiteres Mal Beachtung findet beruht auf der Tatsache, dass *LZX*-Anwendungen sich in verschiedene Zielsysteme kompilieren lassen. Da neben *Flash LZX*-Codes ebenfalls als reine *DHTML*-Anwendung kompiliert werde kann und somit auf keine weiteren Plug-ins angewiesen ist, sei es der Vollständigkeit halber hier nochmals kurz erwähnt. Die genaueren Details und entsprechende Architektur wurden bereits zuvor im Abschnitt 5.3.6 der pluginbasierten Frameworks erschöpfend behandelt.

Anwendungen, die auf *DHTML* basieren nutzen das *AJAX*-Konzept und beschränken sich daher auf Techniken wie *HTML* und *JavaScript*, um interaktive *Rich Internet Applications* zu realisieren. Techniken, die jeder moderne Web-Browser bereits standardmäßig

[15] http://www.extjs.com/products/gwt/ – Stand: 31. Juli 2010

mit sich bringt und durch kein zusätzliches Plugin realisiert werden muss. *OpenLaszlo*-Anwendungen, die nach *DHTML* kompiliert wurden, sind völlig unabhängig von meist proprietären Plugins und stellen somit nicht zuletzt für Open Source-affine Unternehmen die einzig sinnvolle Lösung für solche Anwendungen dar. Der entsprechende Compiler sorgt dafür, dass sich die *DHTML*-Variante im Look&Feel nicht von der *Flash*-Variante unterscheidet.

## 5.5 Zusammenfassung

Generell ist zu sagen, dass sich alle beschriebenen Frameworks, egal ob plugin- oder *AJAX*-basiert, zur Migration von *LARGO* eignen würden. Sämtliche Frameworks bringen entsprechende Funktionen mit, um sowohl die grafischen als auch die technischen Anforderungen zu erfüllen.

Um erste Erfahrungen insbesondere im Umgang mit den Entwicklungswerkzeugen zu erlangen, wurden zu Beginn der Pilotphase des *LASAD*-Projekts sowohl mit *Adobe Flex*, *GWT*, *JavaFX*, *Silverlight* und *OpenLaszlo* erste Prototypen entwickelt, die sich einzig auf die Darstellung von Graphen und Diagrammen sowie die nötige Client-Server-Kommunikation beschränkten.

Schon hier zeigten sich signifikante Unterschiede im Umgang mit den einzelnen Frameworks. Schnell stellte sich heraus, dass die zu Grunde liegende Programmiersprache ein entscheidender Faktor für effektives und vor allem für Fremde nachvollziehbares Arbeiten darstellte. Für *Adobe Flex*, *Silverlight* und *OpenLaszlo* mussten zunächst die frameworkeigenen Sprachen und Dialekte erlernt werden. *GWT* hingen lässt sich mit reinem Java entwickeln, einer weit verbreiteten Sprache, die sowohl während der Ausbildung als auch im produktiven Einsatz häufig zum Einsatz kommt. Für *JavaFX* gilt dies in ähnlicher Weise. Zwar werden Anwendungen ebenfalls in einer eigenen Scriptsprache namens *JavaFX Script* verfasst. Diese ist auf Grund der engen Verwandtschaft zu Java aber weitaus einfacher zu erlernen.

Plattformunabhängigkeit war ein weiterer entscheidender Faktor bei der Wahl des richtigen Frameworks. *LARGO* als Tutorensystem wird meist in heterogenen Rechnernetzen eingesetzt, wie sie an Universitäten und Schulen häufig anzutreffen sind. Von daher sollte ein entsprechendes Framework sowohl Microsoft Windows, Mac OS als

auch Linux unterstützen. *AJAX*-Frameworks leisten dies für alle benannten Betriebssysteme, sofern ein aktueller Browser zur Verfügung steht. Plugins für *Adobe Flash* und *Flex*, Java Applets, *JavaFX* und *OpenLaszlo* (Flash) sind für alle oben genannten gängigen Betriebssystem verfügbar. Einzig Microsoft *Silverlight* beschränkt sich auf die Unterstützung von Microsoft Windows und Mac OS. Linux und andere System werden offiziell nicht unterstützt.

Letztendlich wurde sich für das *Google Web Toolkit* entschieden. Java als Programmiersprache, *Eclipse* als Standardentwicklungsumgebung, völlige Plattformunabhängigkeit und der Verzicht auf jegliche zu installierenden Plugins machen es zum idealen Kandidaten für *LARGO* und somit für den ersten zu implementierenden *LASAD*-Client.

# 6 LASAD – Ein konfigurierbares Framework für Argumentationssysteme

In Kapitel 2 wurden bereits einige Vertreter computergestützter Argumentationssysteme aufgeführt. Die von [SCHEUER et al., 2010] erstellte Übersicht listet annähernd 50 solcher Systeme auf, erhebt dennoch keinen Anspruch auf Vollständigkeit.

Viele der aufgeführten Systeme sind entweder für einen sehr speziellen Einsatzzweck konzipiert worden, zum Beispiel juristische (*LARGO*, [PINKWART et al., 2006b], siehe auch Kapitel 3) oder auch wissenschaftliche Argumentation (*Convince Me*, [RANNEY und SCHANK, 1998]), oder wurden in ihrer Ausrichtung zu allgemein gehalten (*Athena*, [ROLF und MAGNUSSON, 2002]), um als angemessenes eLearning-Werkzeug zu dienen.

Ein goldener Mittelweg, der etwa eine Vielzahl konfigurierbarer, domänenspezifischer Werkzeuge im Rahmen einer flexiblen, erweiterbaren System-Infrastruktur anbietet, existiert derzeit nicht. [LOLL et al., 2010]

Genau diese Lücke versucht das LASAD-Framework mit einer allgemeine Systemarchitektur zu schließen, die es ermöglicht, mit minimalem Konfigurationsaufwand neue, bei Bedarf auch kollaborative Argumentationssysteme zu erstellen, die dennoch domänenspezifischen Anforderungen genügen.

Ende 2008 gestartet, befindet sich das *LASAD*-Projekt[1] noch in einer frühen Entwicklungsphase. Nicht alle geplanten Funktionen wurden bisher implementiert. Dennoch scheint es ideal dazu geeignet als Basis für das neue *LARGO*-System zu dienen.

Die Entscheidung im Rahmen dieser Arbeit *LARGO* auf der Basis des *LASAD*-Frameworks neuzuimplementieren, schaffte Vorteile für beide Projekte. Das *LASAD*-Projekt bekam bereits in einer frühen Entwicklungsphase einen ernsthaften Praxisbezug. Architekturprobleme ließen sich so schnell erkennen und in frühen Stadien

[1] http://lasad.dfki.de – Stand 31. Juli 2010

des Projektes bereits beheben. Das *LARGO*-System bekam indes eine mächtige und aussichtsreiche Plattform als Basis zur Verfügung gestellt, die große Teile der Anforderungen bereits leisten konnte.

Die erste Client-Plattform des *LASAD*-Frameworks sollte als *Rich Internet Application* realisiert werden und das *LARGO*-System abbilden. Letztendlich hat man sich auf Grundlage der in Kapitel 5 gewonnen Erkenntnisse dazu entschieden das *Google Web Toolkit* (siehe Abschnitt 5.4.2) als Basis zu wählen. Die Offenheit des *LASAD*-Frameworks und dessen Schnittstellen erlauben es ebenso Clients auf Basis anderer Technologien zu entwickeln und an dieses anzubinden.

Im folgenden Abschnitt wird die allgemeine Architektur des *LASAD*-Frameworks näher betrachtet.

## 6.1 Architektur

Das *LASAD*-Framework wurde auf Basis einer klassischen Schichtenarchitektur konzipiert und, wie in Abbildung 6.1 dargestellt, implementiert. Die nächsten Abschnitte thematisieren die entsprechenden Schichten und beschreiben die aktuelle technische Umsetzung. Diese sind nicht fest vorgeschrieben. Vielmehr lassen sie sich mit anderen Technologien realisieren und sind so ein Beweis für die *Offenheit* und *Flexibilität* des *LASAD*-Frameworks.

### 6.1.1 Client-Schicht

Das *LASAD*-Framework sieht zwei unterschiedliche Arten von Clients vor: Zum Einen normale Benutzer-Clients (Abbildung 6.1 oben links), zum Anderen sogenannte Analyse-und-Feedback-Clients (Abbildung 6.1 oben rechts). Beide Clients nutzen dieselben Schnittstellen zum Server und haben daher prinzipiell die gleichen Möglichkeiten. Sie können jedoch durch ein serverseitiges Rollen- und Rechte-Management eingeschränkt werden.

Ein Benutzer-Client dient der Erstellung von Argumentationen menschlicher Nutzer, *Analyse-und-Feedback-Clients* dienen hingegen der Untersuchung von Argumentationsstrukturen, um zum Beispiel auf mögliche Schwächen oder Inkonsistenzen hinweisen zu können. Im Fall von *LARGO* würden die entsprechenden Graph-Grammatiken, die

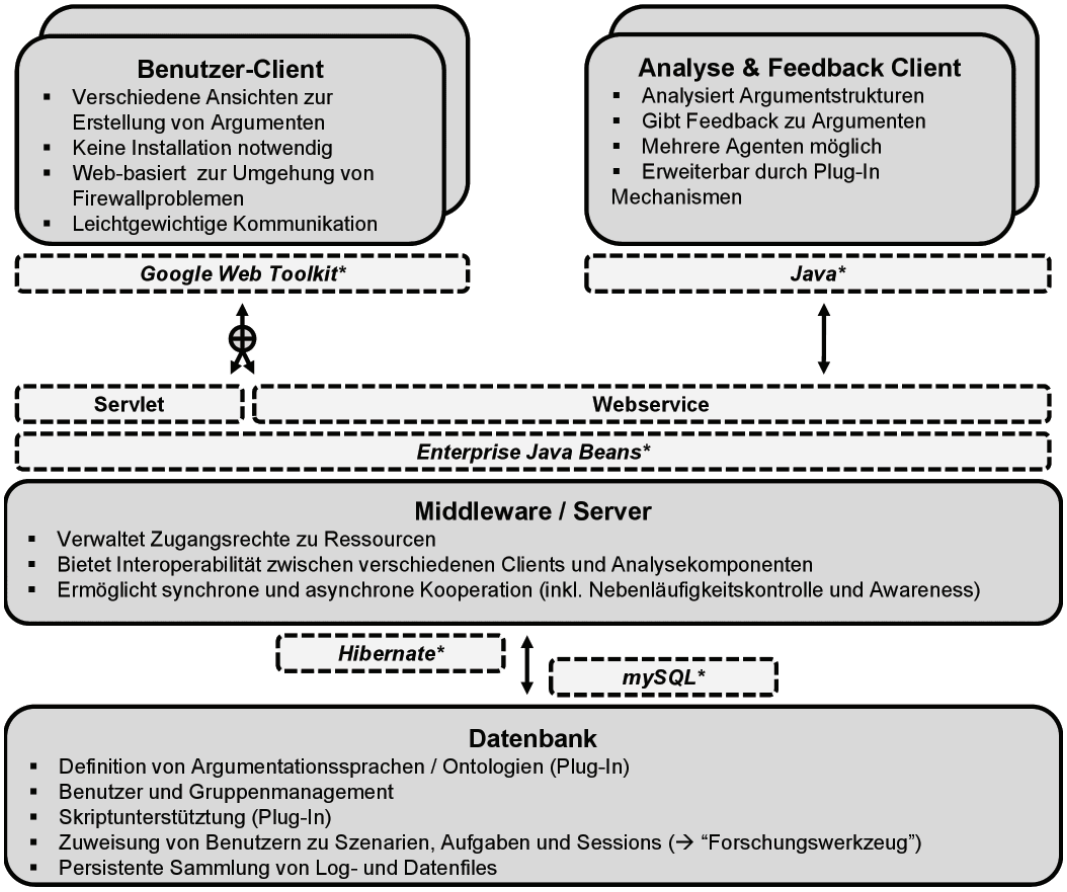


**Abbildung 6.1:** Die Architektur des LASAD-Frameworks [LOLL et al., 2010]

in Kapitel 3.1 bereits erwähnt wurden, in Form eines solchen Analyse-und-Feedback-Client realisiert werden.

Der erste Benutzer-Client wurde als *Rich Internet Application* auf der Basis von *GWT* entwickelt. Der Client ist plattformunabhängig, benötigt keine clientseitige Installation und ist in jedem aktuellen Web-Browser lauffähig. Weitere Vorteile und eine genauere Beschreibung von *GWT* sind Kapitel 5 zu entnehmen.

Der erste Analyse-und-Feedback-Client wurde vom *DFKI*[2] [SCHEUER et al., 2009] auf Java-Basis entworfen und nutzt ebenso wie *GWT* die angebotenen Server-Schnittstellen auf Basis von *Web-Services*.

[2]http://www.dfki.de – Stand 31. Juli 2010

### 6.1.2 Server-Schicht

In der aktuellen Version werden die serverseitigen Aufgaben (Abbildung 6.1 Mitte) durch *Enterprise Java Beans* realisiert. Zu diesen gehört neben der Nebenläufigkeitskontrolle auch das Rollen- und Rechtemanagement. Jede Aktion (beispielsweise das Modifizieren und Erstellen eines Elements oder das Absenden einer Chat-Nachricht), die ein Client ausführt, wird an den Server geschickt und von diesem verarbeitet. Der *GWT*-Client verbindet sich momentan über eine Servlet-Schnittstelle. Der aktuelle Analyse-und-Feedback-Client nutzt *Web-Services*, die als Hauptschnittstelle für Clients geschaffen wurden. Antworten werden vom Server an die beteiligten Clients mittels Server-Pushs verschickt. Ein Polling der Clients ist nicht notwendig, was sowohl Client- als auch Server-Ressourcen spart.

### 6.1.3 Daten-Schicht

Sämtliche Log- und Konfigurationsdaten werden zentral und persistent gespeichert und vom Server schlussendlich an die Clients verteilt. Momentan verwendet der Prototyp ein *Hibernate Mapping* mit einem *MySQL-Datenbank-Backend*. (Abbildung 6.1 unten)

## 6.2 Ontologien und ihre Konfiguration

*Ontologien* beschreiben die in einem Argumentationssystem zur Verfügung stehenden Elemente, deren Beziehungen untereinander sowie die möglichen Modifikatoren. Auf Sprachen übertragen würden sie der zugrundeliegenden Grammatik entsprechen.

Im Falle von *LARGO* wären dies zum Beispiel die Box-Elemente wie *Fact*, *Test* und *Hypothetical* sowie die in Kapitel 3 erwähnten Verbindungstypen samt der ihnen zur Verfügung stehenden Elemente, wie zum Beispiel verschiedene Arten an Textfeldern. Ebenso kann das grafische Erscheinungsbild von Elementen beispielsweise in Form von unterschiedlichen Umrandungen und Farben beeinflusst werden.

Neben der eigentlichen *Ontologie* können auch weitere Konfigurationselemente gespeichert werden. Im Fall von *LARGO* fallspezifische Transcripts oder ein Tutorial wie es im Kapitel 7.3 beschrieben wird.

Im *LASAD*-System werden sie in einem *XML*-Format definiert, serverseitig in der Daten-Schicht gespeichert und bei Bedarf an die Clients verteilt. Sie sind der Hauptgrund für die hohe Flexibilität des *LASAD*-Frameworks.

## 6.3 Zusammenfassung

Statt für *LARGO* eine eigene neue Architektur zu schaffen, konnte man sich dank des *LASAD*-Frameworks darauf beschränken *LARGO*-spezifischen Eigenheiten, wie das Transcript sowie das für die Studie benötigte Tutorial- und Questionaire-Framework, in die *LASAD*-Architektur zu integrieren. Dank der Offenheit des *LASAD*-Frameworks stehen diese Funktionen auch anderen System zur Verfügung, die zukünftig auf Basis des *LASAD*-Framework realisiert werden.

Die entsprechende Implementierung des *LARGO*-Clients wird im Kapitel 8 thematisiert. Das nächste Kapitel beschäftigt sich mit der Planung der abschließenden Usability-Studie an der *University of Pittsburgh*, bei der die allgemeine Funktionsfähigkeit der neuen LARGO-Implementierung unter realistischen Bedingungen getestet wurde.

# 7 Studienplanung

Der Fokus dieser Arbeit lag von Beginn an auf der Migration und Erweiterung des bereits existierenden *LARGO*-Systems. Im Rahmen einer abschließenden Studie sollte untersucht werden, ob das neue System sowie die erarbeiteten Veränderungen positive Auswirkungen auf die in früheren Studien gezeigten Schwächen von *LARGO* (siehe Abschnitt 3.3) haben.

Kurzum. Die Hypothese, dass sich das *LASAD*-Framework prinzipiell eignet um ein domänenspezifisches Tutorentool abzubilden, sollte am Beispiel von *LARGO* verifiziert werden.

Durchgeführt wurden die Studien durch unsere Partner an der *University of Pittsburgh School of Law*. Zur Realisierung der Studie musste das *LASAD*-Framework um ein Tutorial-Framework und ein Framework für interaktive Fragebögen erweitert sowie das studienbegleitende Tutorial und der dazugehörige Fragebogen in Kooperation mit den Partnern vor Ort erarbeitet werden. Diese enge Zusammenarbeit war umso bedeutender, da die Studie nicht persönlich in Pittsburgh begleiten werden konnte.

Zu beachten waren dabei die neuen Möglichkeiten des kooperativen Arbeitens. Trotz der noch rudimentären Funktionen des jungen *LASAD*-Frameworks sollte die allgemeine Funktionsfähigkeit am Beispiel von *LARGO* ein erstes Mal getestet werden. Das begleitende Tutorial musste von einem Teil der Probanden als Einzelnutzer bearbeitet werden. Der übrige Teil anschließend in Zweiergruppen an getrennten PCs. Die Fragebögen wurden nach jedem einzelnen Tutorialschritt von den Probanden selbst beantwortet.

Im Verlauf dieses Kapitels werden die Durchführung und die spätere Auswertung näher erläutert. Die Implementierung der benötigten Erweiterungen des *LASAD*-Frameworks finden in Kapitel 8 Beachtung. Die abschließende Auswertung ist in Kapitel 9 nachzulesen.

## 7.1 Studienbedingungen

Bei der Gestaltung der Studie waren zwei Bereiche von besonderen Interesse. Zum Einen die Arbeit als Einzelperson um Vergleiche mit früheren Erfahrungen und Studien ziehen zu können, zum Anderen die Arbeit in der Gruppe, um die grundlegende Möglichkeit des kooperativen Arbeitens im Kontext von *LARGO* zu bewerten.

Dies führte zu folgender Aufteilung der Probanden:

- **Bedingung 1: Einzelstudie** Eine Gruppe von Einzelpersonen bearbeitet das Tutorial für sich alleine.
- **Bedingung 2: Gruppenstudie** Gruppen von zwei Personen an zwei verschiedenen Rechnern bearbeiten den ersten Teil des Tutorials für sich alleine, einen zweiten Teil gemeinsam.

## 7.2 Teilnehmer

Die Studie selbst wurde an der *University of Pittsburgh* durchgeführt und durch dortige Mitarbeiter betreut. Die Teilnehmersuche wurde ebenfalls vor Ort durchgeführt und per E-Mail-Verteiler, Aushängen und durch den direkten Kontakt in Vorlesungen bekannt gemacht. Gesucht wurden Studierende, die sich für das Thema Recht interessierten. Dahingehende Vorkenntnisse wurden allerdings nicht vorausgesetzt.

Es fanden sich insgesamt elf Teilnehmer, die auf vier Terminen verteilt an der Studie teilnahmen. Die Aufteilung, entsprechend der Studienbedingungen im Absatz, zuvor stellt sich wie in Tabelle 7.1 beschrieben dar.

| Termin | Einzelstudie | Gruppenstudie | Gesamtteilnehmer |
|---|---|---|---|
| 3.3.2010 | 2 | 1 | 4 |
| 4.3.2010 | 1 | 0 | 1 |
| 2.4.2010 | 1 | 1 | 3 |
| 7.4.2010 | 1 | 1 | 3 |

**Tabelle 7.1:** Teilnehmerverteilung der einzelnen Studientermine

Nachdem erfolgreichen Abschluss der Studie wurde jeder Teilnehmer mit 30$ entlohnt.

## 7.3 Aufgabenstellung

Die Studie wurde von vornherein als Usability-Studie geplant und sollte dementsprechend die Machbarkeit und den gesteigerten Nutzen der im Rahmen dieser Arbeit getätigten Veränderungen am LARGO-System zeigen. Beide Teilnehmergruppen mussten während der Studie ein Diagramm eines Transcripts erstellen. Dazu wurde ihnen neben dem Transcript ein Tutorial zur Seite gestellt, welches den Teilnehmer in neun Schritten durch die gesamte Studie führte. Am Ende jedes Tutorialschritts wurde der Teilnehmer aufgefordert einen Fragebogen auszufüllen. Sowohl die Wahl des Transcripts als auch die Konzeption des Tutorials sowie der Fragebogen selbst fand in enger Kooperation mit Pittsburgh statt. Im weiteren Verlauf dieses Abschnitts wird auf diese drei Elemente nochmals genauer eingegangen.

### 7.3.1 Transcript und Tutorial

Wie schon zuvor erwähnt, wurde die Studie in ihrer Gesamtheit von unseren Partnern in Pittsburgh durchgeführt. Ein direktes Eingreifen bei Problemen oder aufkommenden Fragen war nur beschränkt möglich. Von daher kam dem Tutorial als „Roter Faden“ eine umso bedeutendere Rolle zu. Es musste den Teilnehmern den Umgang mit dem System näherbringen, die konkreten Aufgabenstellungen kommunizieren und die zur späteren Auswertung nötigen Fragebögen integrieren.

Angelehnt an frühere Studien wurde das Transcript des Carney Falls als Grundlage gewählt. Die ersten acht Schritte dienten dazu dem Teilnehmer sämtliche Funktionen von *LARGO* näherzubringen. Dabei wurde das Erstellen und Manipulieren von allen möglichen Elementen unter Verwendung der zur Verfügung stehenden Methoden erläutert.
Die acht Schritte glichen sich für Bedingung 1 und Bedingung 2. Mit vollenden des achten Schritts hatte jeder Teilnehmer das Grundgerüst eines Diagramms erstellt. Der neunte Schritt des Tutorials stellte die Unterscheidung beider Bedingungen dar. Setzten die Teilnehmer der Bedingung 1 ihr selbst erstelltes Diagramm bis zum Ende des Transcripts fort, wurden die Teilnehmer einer Gruppe der Bedingung 2 aufgefordert, kooperativ ein Referenzdiagramm fortzusetzen. Dieses Diagramm entsprach dem erwarteten Ergebnis der ersten acht Schritte, Abbildung 7.1 zeigt diese Vorlage.

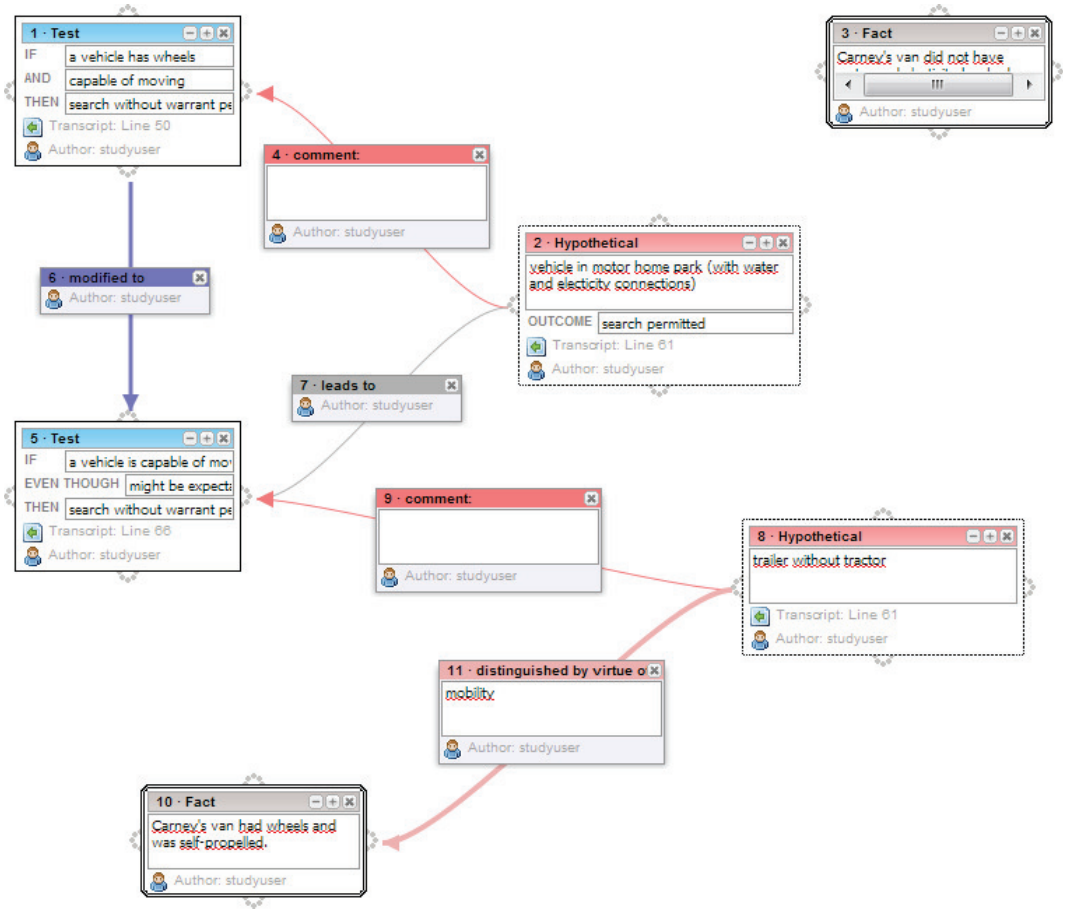


**Abbildung 7.1:** Diagrammvorlage der Multiuser-Studie

## 7.4 Auswertung

Am Ende jedes Tutorialschrittes mussten die Teilnehmer einen Fragebogen beantworten. Neben der Erhebung von persönlichen Informationen zum Studienjahr und der allgemeinen Computererfahrung jedes einzelnen Teilnehmers dienten die Fragebögen größtenteils zur Erfolgskontrolle der einzelnen Tutorialschritte und gab den Teilnehmern gleichzeitig die Möglichkeit über ihre eigenen Erfahrungen und Präferenzen zu berichten. Bis einschließlich Schritt 8 glichen sich die Fragebögen beider Studienbedingungen. Schritt 9 stellte dann die Differenzierung beider Gruppen dar. Den Teilnehmern der Bedingung 2 wurden drei zusätzliche Fragen gestellt, die das kooperative Arbeiten thematisierten.

Der letzte Fragebogen in Schritt 10 implementierte die zehn Fragen des *SUS* von

John Brooke [BROOKE, 1996]. Jede dieser Fragen ließ sich auf einer Skala von 1 bis 5 beantworten und erlaubte so eine rechnerische Auswertung der allgemeinen Usability von *LARGO* und schaffte gleichzeitig Vergleichsmöglichkeiten mit späteren Versionen und anderen Programmen.

Waren im Rahmen dieser Arbeit Erkenntnisse über die Usability von besonderer Bedeutung, wurden die Fragen des Schritt 8 indes von unseren Partnern in Pittsburgh gestellt. Sie zielten auf das Verständnis des zu Grunde gelegten Transcripts ab und waren für die eigenen Forschungen von Bedeutung.

Sämtliche Fragebögen sind im Anhang A zu finden.

### 7.4.1 Datenerfassung

Nicht zuletzt da die Studie in Pittsburgh durchgeführt wurde, war es wichtig, dass sämtliche relevante Daten für eine spätere Auswertung der Ergebnisse dauerhaft gespeichert wurden.

Neben den Antworten der Fragebögen wurden durch das System ebenfalls folgende Informationen gesammelt:

- Den Login- und Logout-Zeitpunkt eines Teilnehmers
- Die Aufenthaltsdauer auf den Diagrammen
- Sämtliche Aktionen auf den Diagrammen im Zeitverlauf, sprich das Erstellen, Verändern und Löschen von Elementen und deren Eigenschaften
- Die in Studienbedingung 2 anfallenden Chat-Nachrichten.

Die Gesamtheit dieser Daten erlaubt es auch im Nachhinein durch die Teilnehmer erstellte Diagramme wiederherzustellen und auf ihre Struktur hin zu analysieren.

# 8 Design und Implementierung

Kapitel 5 und Kapitel 6 haben technische Möglichkeiten aufgezeigt, um einen Großteil der in Kapitel 3 beschrieben Probleme des alten *LARGO*-Systems zu lösen. Das *LASAD*-Framework mit seiner offenen und flexiblen Architektur wurde speziell zur Realisierung von Projekten wie *LARGO* entwickelt und musste nur in einigen *LARGO*-spezifischen Funktionen nennenswert modifiziert und erweitert werden.

Die Ziele dieser Arbeit sahen die vollständige Migration des alten *LARGO*-System auf das neue *LASAD*-Framework vor. Einzig der *LARGO*-spezifische Analyse-und-Feedback-Client zur Realisierung des aus *LARGO* bekannten Feedbacks basiert auf der Arbeit des *DFKI* und war nicht Bestandteil dieser Arbeit.

Als Ausgangspunkt der Implementierung dienten die Prototypen des in Abschnitt 6.1.1 bereits erwähnten Benutzer-Clients auf Basis des *Google Web Toolkits* sowie der in Abschnitt 6.1.2 erwähnte Server-Part realisiert als *EJB*.

Sowohl der Prototyp des Clients als auch der Prototyp des Servers gingen aus einem ersten Forschungsbesuch im März 2009 in Pittsbrugh hervor und wurden seither stetig im Rahmen des *LASAD*-Projekts und dieser Arbeit weiterentwickelt und entsprechend modifiziert bis letztendlich Anfang 2010 eine lauffähige Version beider Programmteile existierte, die einen ersten Einsatz von *LARGO*, basierend auf dem *LASAD*-Framework, während der im Kapitel 7 geplanten Studie, ermöglichte.

Im September 2009 wurden im Zuge eines Forschungsaufenthalts in Pittsburgh die weiterentwickelten Prototypen den Partnern in Pittsburgh präsentiert und im Verlauf des Aufenthalts weiter an die Bedürfnisse des rechtswissenschaftlichen Einsatzgebiets angepasst. Speziell die Konzeption des Tutorial- und Questionnaire-Frameworks sind aus diesem Aufenthalt hervorgegangen.

Schlussendlich mussten sowohl der Client als auch der Server entsprechend angepasst werden, um das *LASAD*-Framework um geforderte Funktionen wie das Transcript und das Tutorial- und Questionnaire-Framework zu erweitern.

Im den folgenden Abschnitten wird auf die nötigen client- wie auch serverseitigen Veränderungen näher eingegangen und die endgültig Implementierung aufgezeigt.

## 8.1 Implementierung des Clients

Bis auf die fehlende Transcript-Unterstützung waren im *LASAD*-Framework bereits alle nötigen Funktionen und Elemente der alten *LARGO*-Oberfläche vorgesehen, sodass für die funktionale Migration von *LARGO* einzig die Unterstützung eines Transcripts nachgerüstet werden musste.

Erweitert wurde der Client um ein Tutorial- und Questionnaire-Framework, welches sowohl die in Kapitel 7 geplante als auch zukünftige Studien mit dem *LASAD*-Framework erleichtern soll.

### 8.1.1 Transcript

Das Transcript (siehe Kapitel 3) ist einer der wichtigsten Bestandteile des *LARGO*-Systems. Es erfüllt gleich mehrere grundlegende Funktionen: Zum Einen stellt es das lineare Verlaufsprotokoll eines konkreten Falls dar und ist damit zentraler Dreh- und Angelpunkt von *LARGO*. Zum Anderen sorgt es für die nötigen Highlight-Funktionen die eine Verknüpfung von Textpassagen mit Diagramm-Elementen erlauben und schlussendlich sorgt letztlich für die grafische Repräsentation bereits erstellter Verknüpfungen im Transcript selbst.

Alle drei aufgeführten Punkte wurden im Vergleich zur alten *LARGO*-Version im Funktionsumfang und ihrer Darstellung erweitert. (Siehe Abbildung 8.1)

So gab es zwar bereits in der älteren Version eine Zeilennummerierung des Transcripts, diese wurde aber weder in der reinen Darstellung noch bei der späteren Referenzierung von Textpassagen genutzt. Die neue Version sorgt für eine konsequente zeilenbasierte Darstellung, die auch bei Breitenänderungen des Transcript-Panels erhalten bleibt und so eine dauerhafte Referenzierung auch über Client-Grenzen hinweg erlaubt. In kollaborativen Umgebungen ist dies besonders wichtig, um Konsistenz zu gewährleisten. Weiterhin erhöht es allgemein die Übersichtlichkeit und schafft so Vorteile im Anwendungsfall eines Einzelnutzers.

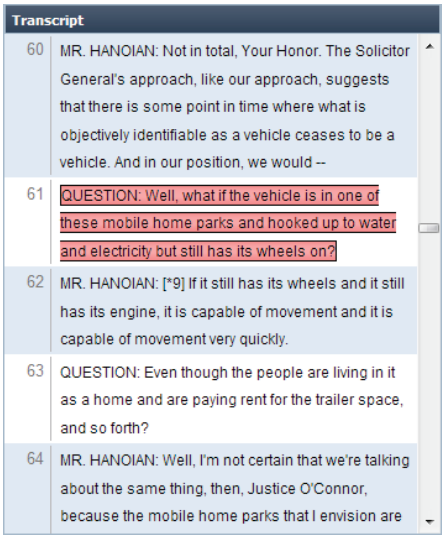


**Abbildung 8.1:** Screenshot des Transcripts im *LASAD*-Framework

Textpassagen werden, wie aus anderen Programmen (beispielsweise *Microsoft Word*) bekannt, mit der Maus markiert. Ebenso können gleich mehrere Zeilen komplett markiert werden, indem nicht die Textpassagen selbst, sondern die Zeilennummern direkt markiert werden. Ein Textbereich wird mit Zeilennummer und Zeichenposition der Anfangs- und der Endstelle referenziert.
Im Gegensatz zur alten Version wurden die Highlight-Funtionen erweitert. So lassen sich ausgewählt Textpassagen nun direkt per Drag&Drop mit dem Diagramm verknüpfen. Zieht man die Textpassage auf einen leeren Bereich der Arbeitsfläche, kann man eine Box erstellen die sofort mit der Textpassage verknüpft ist. Ebenso kann eine Textpassage auf eine bereits existierende Box gezogen werden. Diese wird dann mit der entsprechenden Passage verknüpft.

Eine Verknüpfung wird an zwei Stellen im *LARGO*-Client dauerhaft dargestellt. Zum Einen erhält eine verknüpfte Box ein zusätzliches Element, welches die Referenzierung samt Zielennummern zeigt. Zum Anderen wird die Textpassage im Transcript in der repräsentierenden Farbe der Box hervorgehoben, siehe Abbildung 8.2. Wurde in der alten Version nur die Textpassage der aktuell ausgewählten Box gezeigt, sind in der neuen Version stets alle verknüpften Textpassagen sichtbar. Dies bewahrt dank der optimierten Darstellung dennoch seine Übersichtlichkeit.

Klickt man auf das Transcript-Element der Box, wir die Textpassage fokussiert und bei Bedarf in den sichtbaren Bereich gebracht. Gleiches passiert beim Klick auf ei-

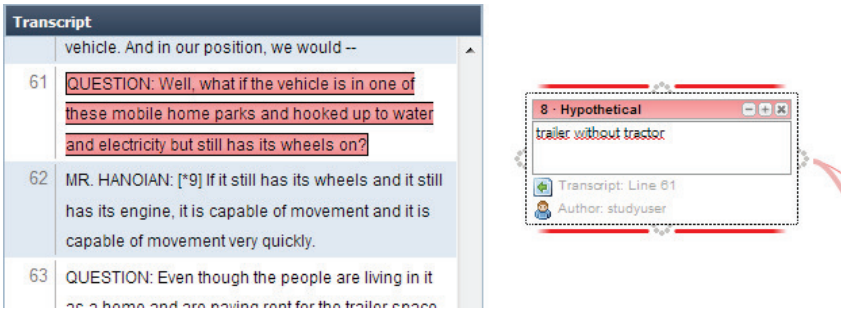


**Abbildung 8.2:** Screenshot des Transcripts mit verlinkter Box im *LASAD*-Framework

ne Textpassage. Die entsprechende Box wird markiert und in den Arbeitsbereich gebracht.

### 8.1.2 Tutorial-Framework

Wurden den Teilnehmern früherer Studien die späteren Arbeitsanweisungen und die Einführung ins System noch in Papierform ausgehändigt, sollten für die geplante Studie (siehe Kapitel 7) die entsprechenden Informationen in die Oberfläche direkt integriert werden.

Dazu wurde der *LASAD*-Client um ein Tutorial-Framework erweitert. Dessen Panel kann an der rechten Seite des Arbeitsbereiches eingeblendet werden und sich in mehrere Schritte unterteilen, siehe Abbildung 8.3. Diese lassen sich komplett in *HTML* formatieren und können standardmäßig bereits auf Text und Grafiken zurückgreifen. Setzt man zum Beispiel das *Flash*-Plugin (siehe Kapitel 5.3.1) auf Client-Seite voraus, ist sogar die Integration von Audio- und Videobeiträgen möglich.

Es werden jeweils nur die Beiträge des aktuell ausgewählten Schrittes angezeigt, wobei am Ende ein Fragebogen integriert werden kann, dessen Funktion im folgenden Abschnitt genauer erläutert wird.

### 8.1.3 Questionnaire-Framework

Fragebögen sind ein adäquates Mittel, um im Verlauf einer Studie detaillierte Informationen der Teilnehmer zu erlangen und stellen somit meist einen wichtigen Teil von Studien dar. Es lag nahe, das *LASAD*-Framework um eine Funktion für elektronische

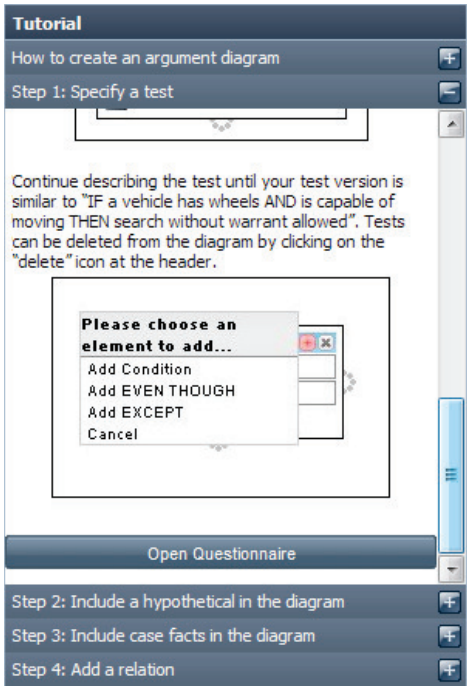


**Abbildung 8.3:** Screenshot des Tutorial-Panel im *LASAD*-Framework

Fragebögen zu erweitern. In Verbindung mit dem Tutorial-Framework stehen nun alle Funktionen bereit, die eine völlig papierlose und ins *LASAD*-System integrierte Studienplanung und -durchführung ermöglichen.

Ein Fragebogen kann aus einer beliebigen Anzahl an Fragen bestehen. Neben reinen Textantworten (siehe Abbildung 8.4) besteht ebenfalls die Möglichkeit, eine Skala vorzugeben, wie sie beispielsweise für Fragen des *SUS*-Test (siehe Abschnitt 7.4) benötig wird (siehe Abbildung 8.5).

Die Antworten der Nutzer werden sofort an den Server übertragen und dort zentral zur späteren Auswertung gesichert.

## 8.2 Implementierung des Servers

Damit auf Basis des *LASAD*-Frameworks auch später das *LARGO*-System als Client abgebildet werden kann, bedarf es noch der servereitigen Konfiguration der *LARGO*-Ontology selbst. Weiterhin mussten für die im vorherigen Abschnitt erwähnten Erweiterungen des LASAD-Clients auch serverseitige Modifikationen vorgenommen

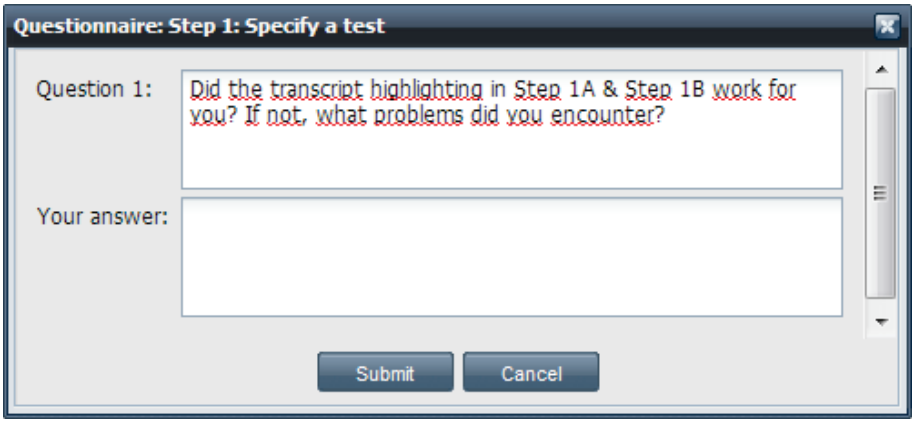


**Abbildung 8.4:** Screenshot eines Fragebogens mit Text-Antwortfeld im *LASAD*-Framework

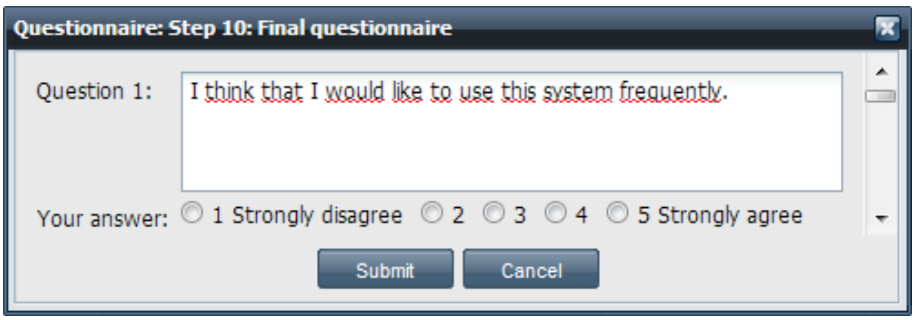


**Abbildung 8.5:** Screenshot eines Fragebogens mit Skalen-Antwortfeld im *LASAD*-Framework

werden. In den nun folgenden Abschnitten werden die nötigen Änderungen an der Server-Architektur im Detail erläutert.

### 8.2.1 Ontology

Die Ontology eines Argumentationssystems, wie es *LARGO* ist, wird im *LASAD*-Framework in einer XML-Datenstruktur konfiguriert (siehe auch Abschnitt 6.2). Dabei werden neben den möglichen Elementen, sprich Box- und Verbindungstypen, auch deren äußeres Erscheinungsbild konfiguriert. Header-Farben, Umrandungen und Linienstärken können dazu beliebig variiert werden.

Im Folgenden wird am Beispiel der Fact-Box die Konfiguration von Elementen erklärt:

```
<element elementid="fact" elementtype="box" quantity="" minquantity=""
    maxquantity="">
  <elementoptions heading="Fact" />
  <uisettings width="190" height="120" resizable="true" border="double"
      background-color="#C0B7B4" font-color="#000000" />
  <childelements>
    <element elementid="text" elementtype="text" quantity="1" minquantity="
        1" maxquantity="1">
      <elementoptions />
      <uisettings background-color="#FFFFFF" font-color="#000000" minheight
          ="40" />
    </element>
    <element elementid="transcriptlink" elementtype="transcript-link"
        quantity="0" minquantity="0" maxquantity="1">
      <elementoptions manualadd="false" longlabel="Transcript-Link" />
      <uisettings minheight="16" maxheight="16" />
    </element>
    <element elementid="awareness" elementtype="awareness" quantity="1"
        minquantity="1" maxquantity="1">
      <elementoptions label="AWARENESS" manualadd="false" />
        <uisettings background-color="#FFFFFF" font-color="#000000"
            minheight="16" maxheight="16" />
    </element>
  </childelements>
</element>
```

Eine Fact-Box mit der Überschrift „Fact“ hat eine anfängliche Größe von 190x120 Pixeln, lässt sich im Nachhinein in der Größe ändern, hat eine doppelte Umrandung und „#C0B7B4“ als Header-Farbe.

Sie besteht aus einem Text-Element, einem optionalen Transcript-Element und einem nicht veränderbaren Awareness-Element. Der „quantity“-Parameter gibt die anfängliche Anzahl des Elements an. „Minquantity“ und „maxquantity“ geben die Grenzen bezüglich der Anzahl an. Der „manualadd“-Parameter hat Einfluss auf die Möglichkeiten des Nutzers ein solches Element manuell hinzuzufügen. So kann das Transcript-Element, was eine Verknüpfung einer Box mit einer Textpasssage repräsentiert, nicht manuell hinzugefügt werden, sondern wird durch die Verknüpfungsmechanismen des Transcripts automatisch erstellt.

Die gesamte Ontology des *LARGO*-Systems findet sich im Anhang B.

### 8.2.2 Transcript, Tutorial- und Questionnaire-Framework Implementierung und Konfiguration

Jedem Arbeitsbereich, auf dem ein Nutzer später arbeitet, liegt ein serverseitiges Template zugrunde, welches wiederum auf einer konfigurierten Ontology beruht. Ein Template beinhaltet weitere Konfigurationen, die nicht die Ontology selbst, aber die Arbeitsumgebung beschreiben. Ein Arbeitsbereich der Studie samt entsprechendem Transcript, dem Tutorial und den dazugehörigen Fragebögen wird in Form eines Templates, ebenfalls auf XML-Basis, konfiguriert und versorgt den Client später mit den nötigen Daten.

Serverseitig musste die Struktur des Templates um die entsprechenden Datenstrukturen erweitert werden.

Nachfolgend sind Beispielkonfigurationen für ein Transcript, das Tutorial und einen Fragebogen zu sehen:

#### 8.2.2.1 Transcript-Konfiguration

```
<transcript>
  <lines>
    <line number="3" text="CALIFORNIA, Petitioner, v. CHARLES B. CARNEY,
        Respondent" />
    <line number="4" text="No. 83-859" />
    <line number="5" text="SUPREME COURT OF THE UNITED STATES" />
    <line number="6" text="1984 U.S. TRANS LEXIS 209" />" +
    <line number="7" text="October 30, 1984, Tuesday, Washington, D.C." />
  </lines>
</transcript>
```

Jede Zeile eines Transcripts hat ihr eigenes <line>-Element mit zwei Parametern. Einen für die Zeilennummer, einen weiteren Parameter für den entsprechenden Text.

#### 8.2.2.2 Tutorial- und Questionnaire-Konfiguration

```
<tutorial>
  <steps>
    <step title="How to create an argument diagram">
      <text><![CDATA[
```

```
        The following instructions explain in several steps how to use <b>
            LARGO</b> to diagram the reasoning with hypotheticals in a
            SCOTUS oral argument using the Carney case as an example....
      ]]></text>
      <questions>
        <question qid="s0_1" type="text"><![CDATA[
          Which law school year you are (1st, 2nd, 3rd)?
        ]]></question>
        <question qid="s10_1" type="score" minscore="1" maxscore="5"><![
            CDATA[
          I think that I would like to use this system frequently.
        ]]></question>
      </questions>
    </step>
  </steps>
</tutorial>
```

Da Fragebögen Teil eines Tutorial-Schritts sind, werden diese auch in einer gemeinsamen Konfiguration erstellt. Ein Tutorial besteht dabei aus bestimmten Anzahlen an <step>-Elementen, die neben einem Titel einen in *HTML* kodierten Text besitzen. Jedem Tutorial-Schritt können beliebig viele <question>-Elemente untergeordnet werden, die jeweils einer Frage entsprechen. Über den „type“-Parameter lässt sich die Art der Antwortmöglichkeit definieren, „text“ entspricht dabei einem einfachen Textfeld, „score“ einer Skala, deren Start- und Endwerte über die Parameter „minscore“ und „maxscore“ angegeben werden können. Der „qid“-Parameter kann genutzt werden um Fragen ein eindeutiges Kürzel zuzuweisen, dies kann bei der späteren Auswertung hilfreich sein, hat aber keine direkte Funktion im *LASAD*-System selbst.

# 9 Durchführung und Auswertung der Studie

An insgesamt vier Terminen im März und April dieses Jahres wurde die in Kapitel 7 geplante Studie an der *University of Pittsburgh* durchgeführt.

Im Anschluss an die Beschreibung des technischen Aufbaus und den Ablauf der Studie selbst wird die Auswertung der aus Pittsburgh gelieferten Daten stehen.

## 9.1 Technischer Aufbau

Das neue *LARGO*-System beruht vereinfacht gesagt auf einer Client-Server-Architektur (siehe Kapitel 6.1). Um mögliche Performance-Probleme aufgrund der hohen Latenzen der Verbindung zwischen Europa und Amerika zu vermeiden, wurden sowohl die nötige Server- als auch Clientinfrastruktur an der *University of Pittsburgh* betrieben.

Für den Fall, dass der dortige Server ausgefallen wäre stand als Fail-Over-Lösung eine vergleichbare Serverinstallation in Clausthal bereit.

### 9.1.1 Client

An die Clients wurden keine besonderen technischen Bedingungen gestellt. Einzig ein aktueller Web-Browser, z.B. Mozilla Firefox sowie eine funktionierende LAN-Anbindung waren nötig.
Die Studie konnte daher ohne großen Aufwand in einem der existierenden Rechnerlabors durchgeführt werden. Da *LARGO* auf eine lokale Installation verzichtet war ein vorheriges Einrichten der Rechner nicht notwendig.

### 9.1.2 Server

Konnte für die Clients auf vorhandene Installationen zurückgegriffen werden, musste für den *LASAD*-Server eigens ein Rechner in Pittsburgh eingerichtet werden. Diese Aufgabe wurde ebenfalls direkt in Pittsburgh realisiert.
Zum Einsatz kam eine aktuelle *Gentoo*-Installation mit einem installierten *JBOSS*- und *Tomcat*-Server sowie einer *MySQL*-Datenbank zur Datenhaltung.

### 9.1.3 Datenerfassung und -haltung

Die in Abschnitt 7.4.1 bereits genannten Daten wurden seitens des *LASAD*-Server in ihrer Gesamtheit in einer *MySQL*-Datenbank gesichert.

Durch die Weitergabe dieser Datenbank-Dateien ließ sich der in Pittsburgh stationierte Server auf einen lokalen Server abbilden. Dies führte zu einer 1-zu-1-Abbildung der Studiendaten auf einem lokalen System und erlaubte das spätere Auswerten der Daten direkt im *LASAD*-System.

## 9.2 Ablauf

Die Anzahl und die Zusammensetzung des Teilnehmerfeldes von insgesamt elf Personen wurde bereits in Abschnitt 7.2 erläutert.

Für einen Durchlauf wurde ein Zeitraum von 1 1/2 Stunden veranschlagt. Dieser beinhaltete neben der Bearbeitung des in Anschnitt 7.3 beschrieben Tutorials auch die kurze Einführung der Teilnehmer durch den Betreuer vor Ort. Jedem Teilnehmer wurden Login-Daten in Form eines Kürzels ausgehändigt, anhand dessen sich die zugehörige Studienbedingung identifizieren ließen. Weitere Rückschlüsse auf die Person waren durch das anonymisierte Kürzel nicht möglich. Mit diesem konnte man sich im Web-Browser am System anmelden und mit dem Tutorial beginnen.

Sobald ein Teilnehmer das Tutorial beendet hatte wurde seitens des Betreuers festgestellt, ob alle Aufgaben bearbeitet wurden. Danach wurde er entlassen und durfte gehen. Der Obolus von 30$ wurde im Nachhinein an die Teilnehmer ausgezahlt.

Um auf etwaige Probleme und Fragen der Teilnehmer schnell reagieren zu können und dem Betreuer vor Ort bei seiner Arbeit zu unterstützen, bestand jeweils eine Chat-Verbindung mit dem Betreuer in Pittsburgh.

Am Ende eines jeden Durchlaufs wurden die gesammelten Daten des Tages als Dump der *MySQL*-Datenbank bereitgestellt.

### 9.2.1 Aufgetretene Probleme

Trotz der Bemühungen im Vorfeld der Studie eine stabile und lauffähige Version des *LASAD*-Frameworks sowie der *LARGO*-Komponenten bereitzustellen, kam es während der Studie mehrfach zu kleineren Problemen mit der Software.

Dabei ist es vereinzelt zu den folgenden Problemen gekommen:

- Probleme beim Login der Teilnehmer
- Probleme beim Beitreten einer Map
- Darstellungsprobleme im Diagramm, z.B. keine Pfeilspitzen sichtbar

Diese Probleme sind allesamt mit dem noch recht jungen Entwicklungsstand des *LASAD*-Frameworks, insbesondere des Server-Parts, zu Zeiten dieser Studie erklärbar und waren durchaus zu erwarten.
Da die Probleme nur geringe Auswirkungen auf die Durchführung der Studie selbst hatten und sämtliche Daten fehlerfrei erfasst wurden, sei dies hier nur am Rande erwähnt.

Bereits eine zweite Studie, im Rahmen des *LASAD*-Projekts durchgeführt von Frank Loll, verlief einige Monate später fast völlig fehlerfrei. Einzig serverseitige Performance-Probleme verlangsamten die Reaktionen des Systems im Verlauf der fünfstündigen Studie, führten letztendlich aber ebensowenig zu nachhaltigen Problemen bei ihrer Durchführung.

Man kann dem System also durchaus attestieren, dass es trotz seines jungen Alters bereits eine Stabilität erreicht hat, die Studien dieser Art erlauben lassen.

## 9.3 Auswertung

### 9.3.1 Erfahrung der Teilnehmer

Zur Einordnung der Jura-Erfahrungen der Teilnehmer wurde im *Schritt 0* nach ihrem aktuellen Jahr an der *Law School* gefragt. Sieben und damit die Mehrzahl aller Teilnehmer studierten bereits im dritten Jahr. Ebenfalls gab es einen *LLM Student*[1] unter den Teilnehmern. Sie können somit allesamt als erfahren eingestuft werden. Tabelle 9.1 gibt einen Überblick über die Verteilung der Teilnehmer.

| Law School Year | Anzahl der Teilnehmer |
|---|---|
| 2nd year | 1 |
| 3rd year | 7 |
| LLM student | 1 |
| Not answered | 2 |

**Tabelle 9.1:** Law School-Jahre der Studienteilnehmer

Um zusätzlich eine Einschätzung der Computererfahrung der Teilnehmer zu ermöglichen, wurden die Teilnehmer gebeten ihre Erfahrung auf einer Skala (very high, high, medium, low, very low) zu bewerten. Die genau Verteilung ist Tabelle 9.2 zu entnehmen. Sieben und damit mehr als die Hälfte aller Teilnehmer stuften ihr Wissen mit *high* oder *very high* ein.

| Erfahrungslevel | Anzahl der Teilnehmer |
|---|---|
| low | 1 |
| medium | 1 |
| high | 6 |
| very high | 1 |
| Not answered | 2 |

**Tabelle 9.2:** Computererfahrung der Studienteilnehmer

Allgemein kann man das Teilnehmerfeld also durchaus als erfahren im Umgang mit Computern im allgemeinen und Jura im speziellen bezeichnen.

[1] Ist Teilnehmer des *Master of Laws (LL.M.) Program for Foreign Law Graduates* an der University of Pittsburgh. Dies setzt eine abgeschlossene Jura-Ausbildung im Heimatland voraus und ist somit ebenfalls als erfahren einzustufen. Weitere Details: http://www.law.pitt.edu/academics/cile/llmprogram – Stand: 31. Juli 2010

### 9.3.2 Gemeinsame Fragebögen der Bedingung 1 und 2

Im Folgenden wird es eine Zusammenfassung der Ergebnisse zu jeder einzelnen der insgesamt 17 Fragen geben, die sowohl bei Bedingung 1 als auch bei Bedingung 2 von den Teilnehmern beantwortet werden mussten.

**Schritt 1 - Frage 1: Did the transcript highlighting in Step 1A & Step 1B work for you? If not, what problems did you encounter?**

Bei sechs der insgesamt elf Teilnehmer funktionierte das Hervorheben bestimmter Transcript-Abschnitte wie gewünscht, alle anderen Teilnehmer hatten mehr oder weniger Probleme. Meist führte das Auswählen bestimmter Teilabschnitte einer Transcript-Zeile nicht zum erhofften Ziel.

Da das Transcript ein zentrales Element des *LARGO*-Systems ist kann man dieses Ergebnis keinesfalls als Erfolg verbuchen. Weitere Detailverbesserungen beim Selektieren einzelner Abschnitte sind daher unumgänglich.

**Schritt 2 - Frage 1: Did you have any problems with using the „Add“ button to extend elements like the hypothetical in Step 2 and the test in Step 1?**

Bei durchweg allen Teilnehmern funktionierte der *Wizzard/Add Button* wie beschrieben. Einzig der Name „Wizzard“, ein Relikt aus früheren LARGO-Versionen, führte teilweise zur Verwirrung der Teilnehmer. Er wird der eigentlichen Bedeutung des Buttons nicht länger gerecht und sollte daher geändert werden.

**Schritt 3 - Frage 1: It is possible to create a „Fact“ by using the Add menu or by right-clicking on the workspace. Which method have you used in Step 3, and did it work without problems?**

Beide genannten Methoden wurden bereits in den Schritten zuvor eingeführt. Hier hatten die Teilnehmer zum ersten mal die Wahl und mussten sich für eine der Methoden entscheiden. Fünf der Teilnehmer nutzten das *Add Menu* zum Erstellen des Fact-Elements, sechs der Teilnehmer hingegen das *Kontextmenu*, welches durch Drücken

der rechten Maustaste geöffnet werden kann. Beide Lösungen bereiteten den Teilnehmern keine Probleme.

Zumindest in dieser Teilnehmergruppe gibt es eine annähernde Gleichverteilung bei der Verwendung beider zur Verfügung stehenden Methoden. Daher gibt es keinen Grund, sich zukünftig für eine dieser zu entscheiden.

**Schritt 4 - Frage 1: Did you have any problems with the drag&drop method for creating a relation?**

Verbindungen zwischen einzelnen Elementen per Drag&Drop zu erstellen gehört zu einer der bedeutenden neuen Funktionen, die im Rahmen dieser Arbeit integriert wurden. Um so interessanter ist es, ob diese Technik den Ansprüchen der Testkandidaten gerecht wurde.

Mit neun Teilnehmern hatten grob 2/3 der Teilnehmer keinerlei Probleme beim Erstellen der Verbindungen. Bei einem Teilnehmer erschienen keine Linien, ein weiterer war sich nicht sicher, ob er alles richtig gemacht hatte.

**Schritt 4 - Frage 2: Do you understand the function of the connector icons?**

Zeigte die technische Umsetzung der Drag&Drop-Technik noch einige Probleme wie die Antworten auf die Frage zuvor nahe legen, war die generelle Funktion dieser neuen Technik mit ihren „connector icons“ jedem der elf Teilnehmer durchaus verständlich.

**Schritt 5 - Frage 1: You have now added your second test element to the diagram. Were you able to do this without reading the introductions in Step 1 again?**

Ein Indikator für den intuitiven Charakter von Bedienkonzepten stellt die Anzahl an Wiederholungen dar, die eine Person benötigt, um ein bestimmtes Vorgehen zu reproduzieren ohne dabei auf eine Anleitung angewiesen zu sein. Die Antworten auf diese Frage waren besonders interessant, da die Gestaltung eines intuitiven Bedienkonzepts eine der Hauptmotivationen dieser Arbeit war.

Um so erfreulicher war es, dass zehn der Teilnehmer keine weitere Anleitung benötigten, um ein weiteres Element zu erstellen.

**Schritt 5 - Frage 2: Do you have any suggestions for improving the adding and editing of boxes?**

Mit Vollendung des fünften Schritts konnten die Teilnehmer bereits erste Erfahrungen beim Hinzufügen und Modifizieren der Boxen sammeln. Acht der Teilnehmer hatten keinerlei Vorschläge zur Verbesserung des Bedienkonzepts.
Ein Teilnehmer verlieh seiner in Schritt 1 getätigten Aussage nochmal Nachdruck, indem er die Probleme mit dem Transcript anführte. Ein weiterer Teilnehmer wünschte sich eine schnellere Oberfläche, da seine Oberfläche die Tendenz hatte häufiger mal einzufrieren.

Die genannten Punkte kritisierten indes nicht das Konzept an sich, sondern die teils fehlerhafte technische Umsetzung. Somit konnten hier keine neuen Erkenntnisse gesammelt werden.

**Schritt 6 - Frage 1: You have linked now some more elements with relations. Were you able to do this without reading the introductions in Step 4 again?**

Wie auch schon bei der artverwandten Frage *Schritt 5 - Frage 1* waren die Teilnehmer einhellig der Meinung, dass auch beim Erstellen von Relations keine erneute Anleitung vonnöten war, um die gestellte Aufgabe zu bewältigen.

**Schritt 6 - Frage 2: Do you have any suggestions for improving the way that linking boxes is done?**

Sechs der Teilnehmer gefiel die aktuelle Umsetzung, Boxen miteinander zu verbinden. Sie hatten keine weiteren Vorschläge zur Verbesserung. Ein Teilnehmer vermisste die Möglichkei, die Richtung einer Verbindung, repräsentiert durch dessen Pfeilspitze, im Nachhinein zu tauschen. Ein weiterer Teilnehmer wiederholte seine in Schritt 4 getätigte Aussage nochmals, indem er darauf hinwies, dass bei ihm keine Linien zu sehen gewesen seien, was mehr ein Bugreport als ein konkreter Verbesserungsvorschlag war.

**Schritt 7**

Schritt 7 hatte keinen abschließenden Fragebogen.

**Schritt 8**

Die zwei am Ende von Schritt 8 gestellten Fragen haben im Kontext dieser Arbeit keinerlei Bedeutung und werden nicht näher behandelt. Sie sind rein juristischer Natur, dienten der Verständniskontrolle und wurden auf Wunsch unserer Pittsburgher Partner aufgenommen.

**Schritt 9 - Frage 1: You have now worked with the LARGO system for some time. Did you encounter any problems linking elements to the transcript of the oral argument?**

Mit Vollendung von Schritt 9 wurden auch alle im Rahmen dieser Studie gestellten Aufgaben mit dem System vollendet. Somit entsprachen die nunmehr gegebenen Antworten einem abschließenden Urteil der Teilnehmer.

Nur fünf der Teilnehmer hatten keinerlei Probleme beim Verlinken des Transcripts mit Elementen des Diagramms. Alle übrigen Teilnehmer hatten hingegen Schwierigkeiten bei der Benutzung. Dabei spielten die schon zuvor genannten Probleme beim Selektieren des Textes sowie gewisse Verzögerungen in der Reaktion eine entscheidende Rolle.

**Schritt 9 - Frage 2: We have introduced three methods to add facts, hypotheticals and tests to the diagram. What do you think about each of them? ( 1. by dragging transcript parts to the workspace, 2. by the Add menu, 3. by the workspace context menu (right click) )**

Alle drei Möglichkeiten wurden gleichermaßen von den Teilnehmern angenommen und würden so auch Verwendung im späteren Einsatz finden. Trotz der schon zuvor beschriebenen technischen Probleme einzelner Methoden konnte keine eindeutige Präferenz für eine der genannten Möglichkeiten festgestellt werden.

**Schritt 9 - Frage 3: Did you have any problems while creating relations between elements?**

Bei neun der elf Teilnehmer sind im Verlauf der Studie keinerlei Probleme beim Erstellen von Verbindungen aufgetreten.

Ein Teilnehmer musste Verbindungen teils häufiger erstellen damit sie erschienen. Ein weiterer Teilnehmer wünschte sich einen Verbindungstyp dem man eigene Bezeichner geben können sollte.

**Schritt 9 - Frage 4: Do you have any comments on the user interface in general? Anything you particularly like or dislike?**

Nachfolgend einige Kommentare der Teilnehmer:

- Das Transcript war nicht lesbar, wenn andere Fenster, wie zum Beispiel der Fragebogen, geöffnet waren. Da sich einige der Fragen auf dieses bezogen, sollte hier in zukünftigen Versionen nach einer anderen Lösung gesucht werden.
- Einer der Teilnehmer mochte die Möglichkeit, Elemente zu verschieben und diese miteinander zu verbinden. Eine Funktion, die nur mit Wörtern so nicht möglich ist.
- Der Chat war furchtbar.
- Es sei schwer Passagen, im Transcript hervorzuheben sowie Elemente miteinander zu verbinden.
- Ein weiterer Teilnehmer wünschte sich ein automatisch ausrichtendes Diagramm, was die Übersicht und die Lesbarkeit der Elemente steigern würde.

**Schritt 9 - Frage 5: Does the software miss any features that you'd want to have? Please write down your suggestions.**

Die Teilnehmer würden folgende Funktionen gerne in *LARGO* integriert wissen:

1. Mehr Freiheiten bei der Formatierung von Texten, zum Beispiel verschiedene Farben sowie die Möglichkeit Textpassagen zu unterstreichen

2. Die Möglichkeit eigene Verbindungen zwischen Elementen zu definieren
3. Die Nummerierung der Elemente solle sich anpassen, sobald einzelne Elemente gelöscht wurden und nicht wie bisher die aktuelle Nummerierung fortsetzen
4. Einen Korrekturmodus wie in Word, der die Verfolgung von Änderungen vereinfacht

Die in Punkt 2 gewünschte Funktion findet immer wieder Erwähnung bei Befragungen von *LARGO*-Anwendern. Es fällt ihnen teils schwer die in ihren Vorstellungen existierenden Verknüpfungen auf die in *LARGO* zur Verfügung stehenden Elemente abzubilden. Doch genau diese vordefinierte Menge an Elementen macht es später möglich die Lösungen der Anwender miteinander zu vergleichen. Ebenso ist sie Voraussetzung für eine effektive Feedback-Engine, wie sie *LARGO* mit sich bringt.

Dass sich die Nummerierung, wie in Punkt 3 gewünscht, nicht automatisch anpasst und quasi „Lücken" bei der Nummerierung entstehen, sobald ein Element gelöscht wurde, ist so durchaus gewollt und sinnvoll. Denn gerade beim kollaborativen Arbeiten werden Elemente anhand dieser ID zum Beispiel in Nachrichtensitzungen identifiziert. Würde sich die Nummerierung der Elemente verändern, wären die Referenzen auf Elemente nicht länger konsistent.

Hingegen sind die in Punkt 1 und Punkt 4 genannten Anregungen durchaus realisierbar und kollidieren keineswegs mit dem *LARGO*-Konzept. Es sind funktionserweiternde Wünsche an die Client-Oberfläche und könnten in die nächsten Versionen einfließen.

**Schritt 9 - Frage 6: Is this a tool that you think might help you understand legal argumentation? Why or why not?**

Sämtliche Teilnehmer äußerten sich positiv über die Software und bescheinigten ihr die Fähigkeit das Verständnis von juristischen Argumentationsverläufen zu verbessern. Dabei hoben sie besonders die graphische Aufarbeitung der Argumentationsstränge in Form eines Diagramms hervor, es zeigt dem Nutzer stets die Zusammenhänge einer Argumentation und liefert somit eine Übersichtlichkeit, die beim reinen Lesen eines Falls häufig verloren geht.

### 9.3.3 Zusätzlicher Fragebogen der Bedingung 2

Stellte Frage 6 die letzte Frage im Schritt 9 für die Teilnehmer der Bedingung 1 dar, mussten die sechs Teilnehmer der Bedingung 2 noch drei weitere Fragen bezüglich des kollaborativen Arbeitens beantworten.

**Schritt 9 - Frage 7: You tried to work in collaboration with your partner on step 9. Did you have any problems while working together?**

Bei einer Gruppe schien die Synchronisation der Arbeitsflächen nicht zu funktionieren. Der eine konnte nicht sehen, was der andere tat. Kollaboratives Arbeiten war daher nicht möglich. Dahingegen funktionierte bei einer anderen Gruppe das gemeinsame Arbeiten ohne Probleme. Um sich auszutauschen machten sie regen Gebrauch von der Chat-Funktion, kritisierten allerdings dessen Umsetzung. Darstellungsfehler in längeren Nachrichtensitzungen hätten das Arbeiten verkompliziert.

Beide angesprochenen Probleme haben das gemeinsame Arbeiten entschieden gestört und sind auf Fehler in der Server- und Clientsoftware zurückzuführen. Sie wurden im Anschluss an die Studie behoben.

**Schritt 9 - Frage 8: Does the software miss any information about your partner, which would helps you working in collaboration?**

Nur einer der Teilnehmer hatte hier etwas beizutragen und wünschte sich eine Live-Chat-Funktion sowie die Möglichkeit eigenen Nachrichten mehr Nachdruck zu verleihen, indem sie „geschriehen“ werden. Dies wird meist durch die konsequente Nutzung von Großbuchstaben oder fetteren Schriftzeichen realisiert.

**Schritt 9 - Frage 9: One last question. Do you think it’s beneficial working with more than one person on a diagram? Could you shortly explain your answers?**

Die Mehrzahl der Teilnehmer empfand das kollaborative Arbeiten als nützlich in diesem Zusammenhang. Es schaffte die Möglichkeit voneinander zu lernen. Allerdings könne ein Chat niemals eine Unterhaltung von Angesicht zu Angesicht ersetzen und hemme somit zum Teil das Arbeiten in der Gruppe.

Nur ein Teilnehmer sah keinen Mehrgewinn im gemeinsamen Arbeiten. Er empfand es als verwirrend.

### 9.3.4 SUS - System Usability Scale

Der zehnte und letzte Fragebogen war wieder für beide Bedingungen gleich geartet und bestand aus zehn standardisierten Fragen die sich wie schon in Kapitel 7.4 beschrieben auf den *SUS* von John Brooke [BROOKE, 1996] beziehen. Jede dieser Fragen musste auf einer Skala von 1 bis 5 beantwortet werden. Konnte oder wurde eine Frage von einem Teilnehmer nicht beantwortet, war diese mit einer 3 zu bewerten. Dieser Fall ist im Verlauf der Studie zweimal eingetreten.

Jedes Element des Tests wird letztendlich mit einem Wert von 0 bis 4 bewertet. Für die Fragen 1, 3, 5, 7 und 9 entspricht dieser Wert der Antwort des Teilnehmers minus 1. Für die Fragen 2, 4, 6, 8 und 10 indes wird der Wert der Antwort vom Wert 5 abgezogen. Anschließend werden die Werte allesamt summiert, mit einem Faktor von 2,5 multipliziert und repräsentieren sodann den individuellen SUS-Wert eines Teilnehmers. Der Wert kann rechnerisch zwischen 0 und 100 liegen, wobei 100 dem best- und 0 dem schlechtmöglichst zu erreichenden Wert entspricht.

| Fragen / Teilnehmer | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | SUS-Wert |
|---|---|---|---|---|---|---|---|---|---|---|---|
| s1 | 3 | 2 | 4 | 2 | 3 | 2 | 5 | 3 | 5 | 1 | 75,0 |
| s2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 47,5 |
| s3 | 4 | 3 | 4 | 3 | 4 | 2 | 4 | 3 | 4 | 2 | 67,5 |
| s4 | 4 | 2 | 5 | 1 | 3 | 2 | 5 | 1 | 5 | 1 | 87,5 |
| m3_1 | 2 | 2 | 4 | 1 | 2 | 2 | 5 | 4 | 2 | 3 | 57,5 |
| m1_1 | 2 | 2 | 5 | 4 | 4 | 2 | 5 | 2 | 4 | 1 | 72,5 |
| m1_2 | 4 | 2 | 4 | 4 | 4 | 2 | 4 | 1 | 4 | 2 | 72,5 |
| m2_1 | 2 | 4 | 3 | 5 | 3 | 3 | 2 | 5 | 3 | 2 | 35,0 |
| m2_2 | 3 | 2 | 2 | 3 | 3 | 2 | 4 | 3 | 2 | 1 | 57,5 |
| m4_1 | 3 | 2 | 5 | 3 | 5 | 3 | 5 | 3 | 5 | 1 | 57,5 |
| m4_2 | 2 | 1 | 4 | 1 | 4 | 1 | 3 | 1 | 5 | 4 | 77,5 |
| | | | | | | | | | | Durchschnitt | 65,9 |
| | | | | | | | | | | Standardabweichung | 15,14 |

**Tabelle 9.3:** Ergebnisse des SUS-Test der Studie

Nach Auswertung der in Tabelle 9.3 zusammengefassten Daten bewerteten die Teilnehmer das *LARGO*-System mit *SUS*-Werten von geringen 35 bis beachtlichen 87,5 Punkten. Bei einer Standardabweichung von 15,14 lag der Durchschnitt bei 65,9 Punkten.

[BANGOR et al., 2009] haben, um die Bedeutung einzelner *SUS*-Werte besser einordnen zu können, in einer eigenen Studie den *SUS*-Test um eine elfte Frage erweitert. Ihr Wortlaut war wie folgt, „Overall, I would rate the user-friendliness of this product as:“ und sollte entgegen den Standardfragen nicht auf einer Skala von 1 bis 5 bewertet werden. Vielmehr wurde eine siebenteilige Skala herangezogen, deren Werte mit Adjektiven von „Worst Imaginable“ bis „Best Imaginable“ belegt waren. Siehe auch Abbildung 9.1.

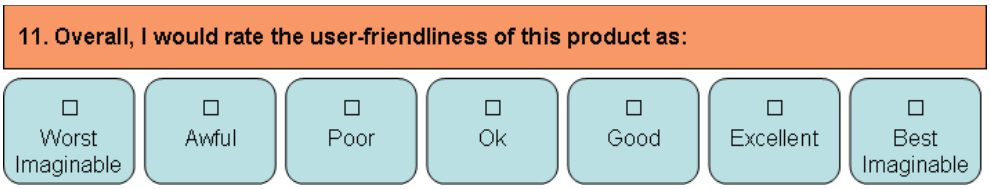


**Abbildung 9.1:** Erweiterung des SUS-Tests nach Bangor et al.

Der erweiterte Fragebogen wurde nun 964 Teilnehmern verschiedener Studien gestellt, die sonst den originalen Fragebogen erhalten hätten. Auf Grundlage dieser Daten wurde die in Abbildung 9.2[2] eingeführte Kategorisierung der Werte vorgenommen.

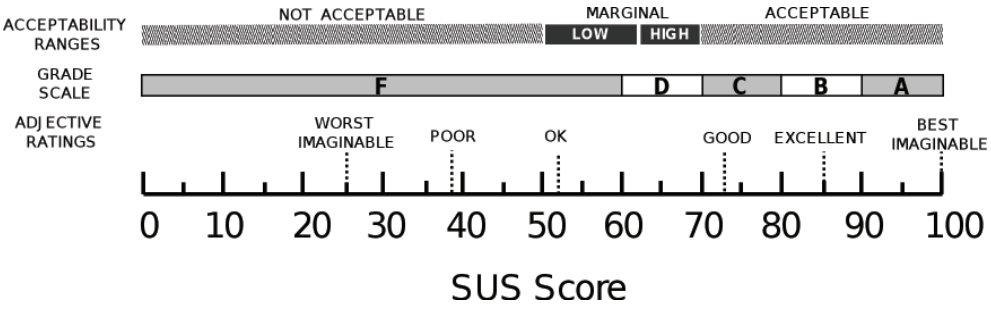


**Abbildung 9.2:** Kategorisierung der SUS-Werte nach Bangor et al.

[2] [BANGOR et al., 2009]

Dass sich die bereits zuvor erwähnten Software-Fehler insgesamt negativ auf die *SUS*-Bewertung auswirken würden war schon absehbar. Der erreichte Mittelwert von 65,9 Punkten entspricht nach [BANGOR et al., 2009] nur der Schulnote „D“[3] und liegt zwischen den Adjektiven „Ok“ und „Good“.

Während der von Frank Loll einige Monate später durchgeführten Studie mit dem *LASAD*-System wurde bereits ein durchschnittlicher *SUS*-Wert von 81,5 Punkten erreicht, was nach [BANGOR et al., 2009] der Schulnote „B“ entsprechen würde. Man kann also davon ausgehen, dass bei einer Wiederholung der Studie mit der aktuellen, von Fehlern bereinigten, Softwareversion der SUS-Wert in ähnlichen Bereichen liegen würde.

### 9.3.5 Diagramme

Die Abbildungen 9.3 und 9.4 zeigen exemplarisch zwei Diagramme, die während der Studie von den Teilnehmern erarbeitet wurden.

Abbildung 9.3 zeigt dabei ein typisches Ergebnis eines Teilnehmers der Studienbedingung 1. Die Tatsache, dass die Diagramme der Teilnehmer meist aus sehr eng aneinander positionierten Elementen bestehen, zum Teil sogar mit großflächigen Überlappungen, legt den Schluss nahe, dass ihnen trotz existierender Scrollbars nicht bewusst war, dass die zur Verfügung stehende Arbeitsfläche ein vielfaches der Monitorauflösung entsprach.

Da sich das Referenzdiagramm des kollaborativen Teils der Studienbedingung 2 bereits über die Grenzen der Monitorauflösung hinweg erstreckte, nutzten auch die später erstellen Diagramme der Teilnehmer durchweg mehr Raum und machten einen übersichtlicheren Eindruck. Siehe zum Beispiel Abbildung 9.4.

Allgemein ist dieses Phänomen dennoch nicht neu und ließ sich so bereits in der alten *LARGO*-Version beobachten. Es zeigt jedoch unmissverständlich die noch bestehende Notwendigkeit die Benutzeroberfläche zu optimieren.

[3] Das amerikanische Notensystem beruht meist auf einem System von 5 Buchstaben. Diese teilen sich wie folgt auf: A (Höchstnote, hervorragend), B (über dem Durchschnitt), C (Durchschnitt), D (genügend) und F (fail, durchgefallen). Weitere Details: http://de.wikipedia.org/wiki/Schulnote#USA – Stand: 31. Juli 2010

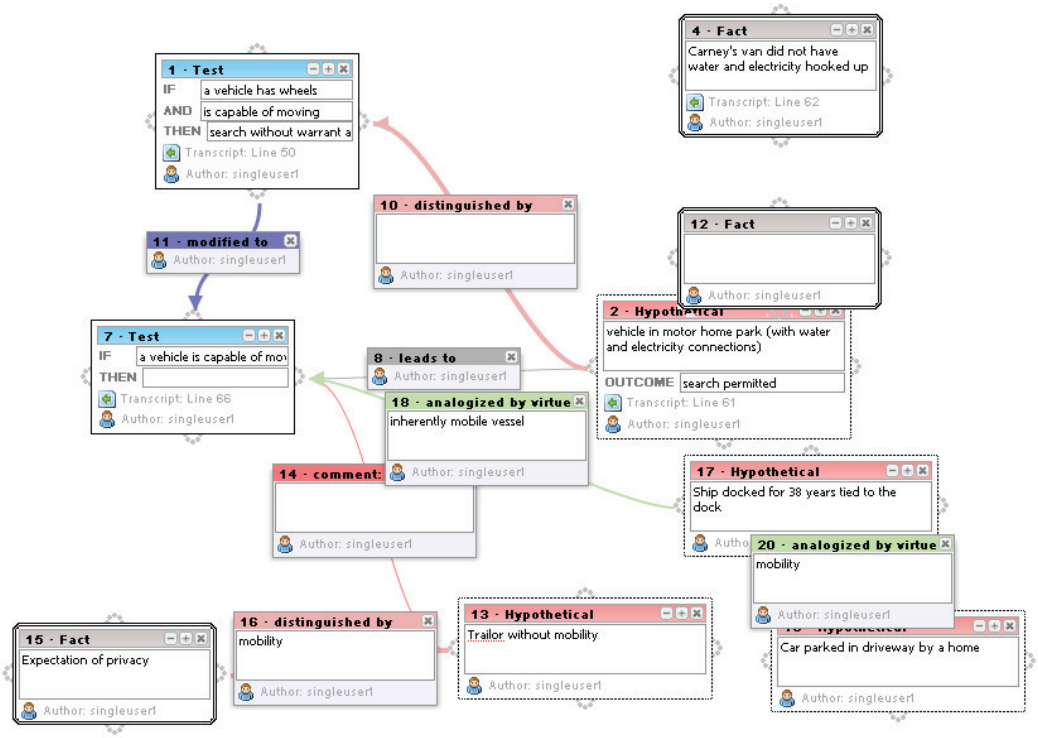


**Abbildung 9.3:** Studienergebnis: Diagramm s_1

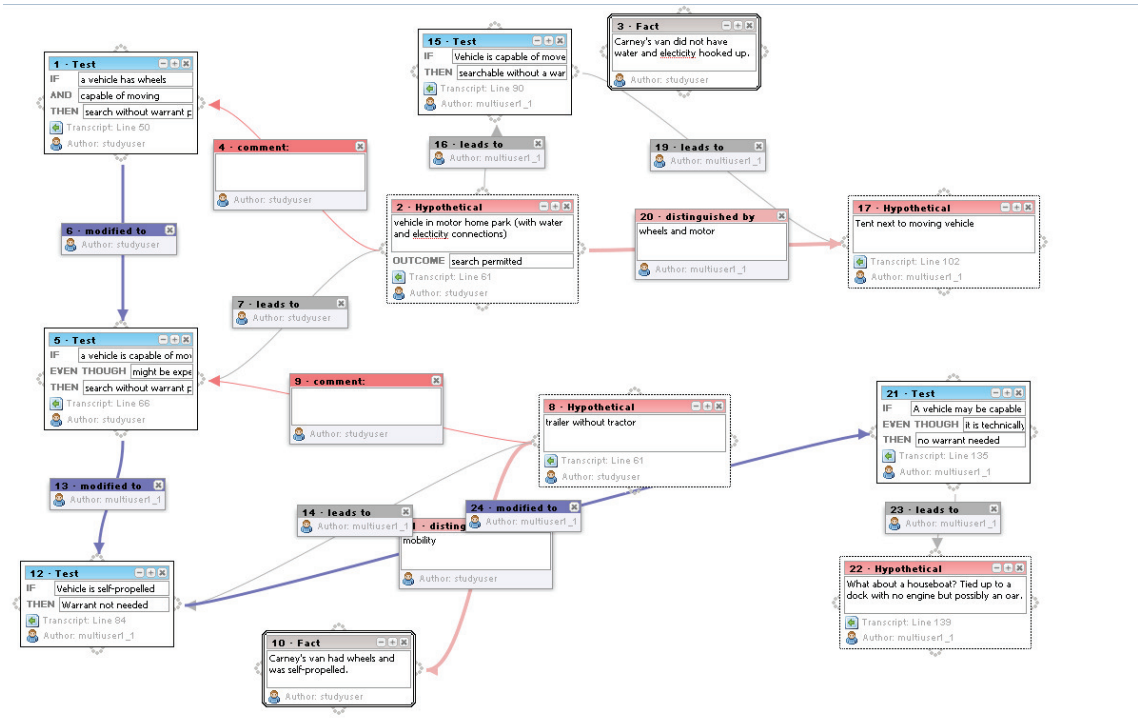


**Abbildung 9.4:** Studienergebnis: Diagramm m1

### 9.3.6 Zusammenfassung

Sowohl die während der Studie aufgetretenen technischen Probleme, als auch die teils schlechte Bewertung des Systems durch die Teilnehmer, zeugt von dem frühen Entwicklungsstand des LASAD-Frameworks zu Zeiten der Studie. Das System konnte dennoch beweisen wozu es zukünftig in der Lage sein wird. Bereits jetzt waren, bis auf die fehlende Feedback-Engine, sämtliche Funktionen des alten *LARGO*-Systems erfolgreich migriert worden und bestätigten die in Kapitel 7 aufgestellte Hypothese, dass das *LASAD*-Framework alle nötigen Funktionen mit sich bringt um eine domänenspezifisches Tool, wie *LARGO* es ist, erfolgreich abzubilden.

Auch wenn die kollaborativen Funktionen des *LASAD*-Frameworks sich bisher auf einen gemeinsamen Arbeitsplatz und einen Chat beschränken, konnte gezeigt werden, dass kollaboratives Arbeiten im Kontext von *LARGO* eine Option ist, die auf ihre Funktionalität hin in zukünftigen Studien überprüft werden sollte.

Ebenso zeigte die Studie, dass die im Rahmen dieser Arbeit durchgeführten Erweiterungen am Bedienkonzept des *LARGO*-Systems durch die Benutzer angenommen wurden. Zwar hatten einige der Teilnehmer bei der Nutzung Probleme, sodass die *SUS*-Bewertung dementsprechend abgestraft wurde, doch lassen die von Frank Loll einige Monate später ermittelten Werte (siehe Abschnitt 9.3.4) vermuten, dass eine Wiederholung dieser Studie ähnlich positive Bewertungen hervorbringen würde.

# 10 Fazit und Ausblick

Am Beispiel von *LARGO* konnte belegt werden, dass sowohl die konkrete *LARGO*-Implementierung erfolgreich war als auch das *LASAD*-Framework in der Lage ist ein domänenspezifisches Argumentationssystem wie *LARGO* abzubilden.

Das LASAD-Framework wird seinen gesteckten Zielen also durchaus gerecht und wurde durch die Migration von *LARGO* um einige nützliche Funktionen reicher. Zu nennen sind hier die Transcript-Unterstützung, das Tutorial-Framework sowie die Möglichkeit Fragebögen zu integrieren.

Die während der Studie aufgetretenen Probleme waren zumeist technischer Natur und mit dem frühen Entwicklungstand des *LASAD*-Frameworks zum Zeitpunkt der Studie erklärbar. Eine weitere Studie, die einige Monate später durchgeführt wurde, unterstützt diese These, kam es bei ihr doch zu weit weniger Problemen. Ebenso fiel ihre Bewertung im Zuge des *SUS*-Tests signifikant besser aus.

Die Neuimplemenierung von *LARGO* beruht auf dem *LASAD*-Framework. Daher wirken sich Verbesserungen im Zuge des *LADAD*-Projekts nunmehr ebenso positiv auf das *LARGO*-System aus.

Mit dem *LARGO*-System wurde der erste Benutzer-Client des *LASAD*-Frameworks auf Basis des *Google Web Toolkit* realisiert. Dies setzte sich gegenüber anderen *Rich Internet Application*-Frameworks als Client-Basis durch.

Das *LASAD*-Projekt ist ein Beweis für die Möglichkeiten die moderne *Rich Internet Application*-Frameworks bieten. Den direkten Vergleich mit klassischen Desktop-Anwendungen müssen sie nicht länger fürchten. Die Entwicklung auf diesem Bereich geht stetig weiter und wird die Grenzen zwischen den beiden Lagern weiter verschwimmen lassen.

Sobald die Implementierung des Analyse-und-Feedback-Clients durch das *DFKI* abgeschlossen wird, kann der letzte fehlende Teil des *LARGO*-Systems, die Feedback-Engine, auf die neue Architektur migriert werden. Dies ist Voraussetzung, um die fehlende Motivation der Nutzer in weiteren Studien zu untersuchen und das *LARGO*-System zu optimieren.

Das *LASAD*-Projekt hat seine Feuerprobe bestanden und muss nun beweisen, dass ein allgemeines Framework geeignet ist hochgradig spezialisierte Arguementationssysteme abzubilden.

# Literaturverzeichnis

[AdobeSystems, 2010] AdobeSystems (2010). *Flash Player penetration*. http://www.adobe.com/products/player_census/flashplayer/ - Abfrage vom 3. Juli 2010.

[Bangor et al., 2009] Bangor, Aaron, P. Kortum und J. Miller (2009). *Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale*. Journal of Usability Studies, 4:114–123.

[Bank, 1995] Bank, David (1995). *The Java Saga*. http://www.wired.com/wired/archive/3.12/java.saga.html - Abfrage vom 29. Juli 2010.

[Belgiorno et al., 2008] Belgiorno, Furio, R. Chiara, I. Manno, M. Overdijk, V. Scarano und W. Diggelen (2008). *Face to Face Cooperation with CoFFEE*. In: *EC-TEL '08: Proceedings of the 3rd European conference on Technology Enhanced Learning*, S. 49–57, Berlin, Heidelberg. Springer-Verlag.

[Brooke, 1996] Brooke, John (1996). *SUS - A quick and dirty usability scale In: Usability Evaluation in Industry*, Kap. 21, S. 189–194. Taylor & Francis Ltd.

[Busch und Koch, 2009] Busch, Marianne und N. Koch (2009). *Rich Internet Applications - State of the Art*. Technischer Bericht 0902, Programming and Software Engineering Unit (PST) Institute for Informatics Ludwig-Maximilians-Universität München.

[Conklin und Begeman, 1988] Conklin, Jeff und M. L. Begeman (1988). *gIBIS: a hypertext tool for exploratory policy discussion*. In: *CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, S. 140–152, New York, NY, USA. ACM.

[Deitel und Deitel, 2008] Deitel, Paul J. und H. M. Deitel (2008). *AJAX, Rich Internet Applications, and Web Development for Programmers*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[EICH, 2005] EICH, BRENDAN (2005). *JavaScript at Ten Years*. http://www.mozilla.org/js/language/ICFP-Keynote.ppt - Abfrage vom 29. März 2010.

[GARRETT, 2005] GARRETT, JESSE JAMES (2005). *Ajax: A New Approach to Web Applications*. http://www.adaptivepath.com/ideas/essays/archives/000385.php - Abfrage vom 29. März 2010.

[VAN GELDER, 2007] GELDER, TIM VAN (2007). *The rationale for Rationale*. In: *Law, Probability and Risk*, Bd. 6, S. 23–42.

[JÄGER, 2008] JÄGER, KAI (2008). *Ajax in der Praxis*. Springer Berlin Heidelberg.

[KARACAPILIDIS et al., 2009] KARACAPILIDIS, N., M. TZAGARAKIS, N. KAROUSOS, G. G. V. KALLISTROS, S. CHRISTODOULOU, C. METTOURIS und D. NOUSIA (2009). *Tackling cognitively-complex collaboration with CoPe_it!*. In: *International Journal of Web-Based Learning and Teaching Technologies*, Bd. 4, S. 22–38.

[LASZLOSYSTEMS, 2006] LASZLOSYSTEMS (2006). *OpenLaszlo - An Open Architecture Framework for Advanced Ajax Applications*. http://www.openlaszlo.org/whitepaper/LaszloWhitePaper.pdf - Abfrage vom 7. März 2010.

[LOLL et al., 2010] LOLL, FRANK, N. PINKWART, O. SCHEUER und B. M. MCLAREN (2010). *Ein generisches Framework zur Erstellung von argumentationsunterstützenden Systemen*. In: *Tagungsband der Multikonferenz Wirtschaftsinformatik (MKWI)*, S. 1427 – 1438.

[MISTRIK et al., 2006] MISTRIK, IVAN, B. P. SPRINGER, S. J. B. SHUM, S. J. B. SHUM, A. M. SELVIN, A. M. SELVIN, M. SIERHUIS, M. SIERHUIS, J. CONKLIN, J. CONKLIN, C. B. HALEY, C. B. HALEY, B. NUSEIBEH und B. NUSEIBEH (2006). *Hypermedia Support for Argumentation-Based Rationale*. In: *15 Years on from gIBIS and QOC. In: Rationale Management in Software Engineering (Eds*, S. 111–132. Springer-Verlag: Berlin.

[PFEIL, 2008] PFEIL, CHRISTIAN (2008). *Adobe AIR: RIAs für den Desktop entwickeln Know-how für HTML/Ajax- und Flash*. Addison-Wesley, München.

[PINKWART et al., 2006a] PINKWART, N., V. ALEVEN, K. D. ASHLEY und C. LYNCH (2006a). *Schwachstellenermittlung und Rückmeldungsprinzipen in einem intelligenten Tutorensystem für juristische Argumentation*. In: MÜHLHÄUSER, M., G. RÖSSLING und R. STEINMETZ, Hrsg.: *Tagungsband der 4. e-Learning Fachtagung Informatik (DeLFI)*,

Nr. P-87 in *GI Lecture Notes in Informatics*, S. 75 – 86, Bonn, Germany. Köllen Verlag. Best Paper Award.

[PINKWART, 2003] PINKWART, NIELS (2003). *A Plug-In Architecture for Graph Based Collaborative Modeling Systems*. In: *Proceedings of the 11 th Conference on Artificial Intelligence in Education*, S. 535–536. IOS Press.

[PINKWART et al., 2006b] PINKWART, NIELS, V. ALEVEN, K. ASHLEY und C. LYNCH (2006b). *Toward legal argument instruction with graph grammars and collaborative filtering techniques*. In: *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, S. 227–236. Springer Verlag.

[PINKWART et al., 2007] PINKWART, NIELS, V. ALEVEN, K. ASHLEY und C. LYNCH (2007). *Evaluating Legal Argument Instruction with Graphical Representations Using LARGO*. In: *Proceeding of the 2007 conference on Artificial Intelligence in Education*, S. 101–108, Amsterdam, The Netherlands, The Netherlands. IOS Press.

[PINKWART et al., 2008] PINKWART, NIELS, C. LYNCH, K. ASHLEY und V. ALEVEN (2008). *Re-evaluating LARGO in the Classroom: Are Diagrams Better than Text for Teaching Argumentation Skills*. In: *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*.

[RANNEY und SCHANK, 1998] RANNEY, M. und P. SCHANK (1998). *Toward an integration of the social and the scientific: Observing, modeling, and promoting the explanatory coherence of reasoning.*. In: READ, S. und L. MILLER, Hrsg.: *Connectionist models of social reasoning and social behavior*, S. 245–274.

[RIASTATS] RIASTATS. *Rich Internet Application Statistics*. http://www.RIAStats.com - Abfrage vom 25. Juni 2010.

[RITTEL und KUNZ, 1970] RITTEL, H. und W. KUNZ (1970). *Ïssues as elements of information systems."*. Technischer Bericht, University of Stuttgard.

[ROLF und MAGNUSSON, 2002] ROLF, B. und C. MAGNUSSON (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 et al., 2009] SCHEUER, O., B. M. MCLAREN, F. LOLL und N. PINKWART (2009). *An Analysis and Feedback Infrastructure for Argumentation Learning Systems*. In: DIMITROVA, V., R. MIZOGUCHI, B. DU BULAY und A. GRAESSER, Hrsg.: *Proceedings*

*of the 14th International Conference on Artificial Intelligence in Education (AIED)*, Nr. 200 in *Frontiers in Artificial Intelligence and Applications*, S. 629 – 631, Amsterdam, The Netherlands. IOS Press.

[SCHEUER et al., 2010] SCHEUER, OLIVER, F. LOLL, N. PINKWART und B. M. MCLAREN (2010). *Computer-supported argumentation: A review of the state of the art*. International Journal of Computer-Supported Collaborative Learning, 5(1):43–102.

[SCHNEIDER et al., 2007] SCHNEIDER, DAVID C., C. VOIGT und G. BETZ (2007). *Argunet - A software tool for collaborative argumentation analysis and research*.

[SIERHUIS, 2006] SIERHUIS, MAARTEN (2006). *Compendium*. Presented at Computational Semantics Laboratory, Stanford University. http://compendium.open.ac.uk/institute/library/Stanford.SemLab.2009.09.28.pdf - Abfrage vom 23. Juli 2010.

[TAPPER et al., 2008] TAPPER, JEFF, M. LABRIOLA, M. BOLES und J. TALBOT (2008). *Adobe Flex 3: Das offizielle Trainingsbuch von Adobe Systems*. Addison-Wesley.

[TZAGARAKIS et al., 2009] TZAGARAKIS, M., N. KAROUSOS, N. KARACAPILIDIS und D. NOUSIA (2009). *Unleashing Argumentation Support Systems on the Web: The case of CoPe_it!*. In: *In Proceedings of the Web Science 2009 Conference*, Athens, Greece.

[W3C, 2009] W3C (2009). *XMLHttpRequest - W3C Working Draft 19 November 2009*. http://www.w3.org/TR/XMLHttpRequest/ - Abfrage vom 6. April 2010.

# Anhang: Studie

## A Fragebögen

- **Step 0**
  - **s0_1** Which law school year you are (1st, 2nd, 3rd)?
  - **s0_2** Which experience with computers you have in general (very high, high, medium, low, very low)?
- **Step 1**
  - **s1_1** Did the transcript highlighting in Step 1A & Step 1B work for you? If not, what problems did you encounter?
- **Step 2**
  - **s2_1** Did you have any problems with using the Ädd"button to extend elements like the hypothetical in Step 2 and the test in Step 1?
- **Step 3**
  - **s3_1** It is possible to create a "Fact"by using the Add menu or by right-clicking on the workspace. Which method have you used in Step 3, and did it work without problems?
- **Step 4**
  - **s4_1** Did you have any problems with the drag&drop method for creating a relation?
  - **s4_2** Do you understand the function of the connector icons?
- **Step 5**

  - **s5_1** You have now added your second test element to the diagram. Were you able to do this without reading the introductions in Step 1 again?
  - **s5_2** Do you have any suggestions for improving the adding and editing of boxes?

- **Step 6**
  - **s6_1** You have linked now some more elements with relations. Were you able to do this without reading the introductions in Step 4 again?
  - **s6_2** Do you have any suggestions for improving the way that linking boxes is done?

- **Step 7**
  - no Questionnaire

- **Step 8**
  - **s8_1** In the second test(5) the advocate modified the test presented in test(1) in response to the first hypothetical(2). Is it necessary for them to modify the second test(5) in response to hypothetical(8) "trailor without tractor". If so why and what should they say? If not then why not?
  - **s8_2** You have related hypothetical(8) "trailor without tractor"to both the last fact(10) and the modified test(5). The diagram faithfully represents the argument contained in the transcript. Now please state your own opinion. Think about the current fact situation and the hypothetical in the light of the test and of the underlying principles and policies. What would be your argument in court if you were confronted with the hypothetical(8) "trailor without tractor"?

- **Step 9**
  - **s9_1** You have now worked with the LARGO system for some time. Did you encounter any problems linking elements to the transcript of the oral argument?

- **s9_2** We have introduced three methods to add facts, hypotheticals and tests to the diagram. What do you think about each of them? ( 1. by dragging transcript parts to the workspace, 2. by the Add menu, 3. by the workspace context menu(right click) )
- **s9_3** Did you have any problems while creating relations between elements?
- **s9_4** Do you have any comments on the user interface in general? Anything you particularly like or dislike?
- **s9_5** Does the software miss any features that you'd want to have? Please write down your suggestions.
- **s9_6** Is this a tool that you think might help you understand legal argumentation? Why or why not?
- **s9_7** You tried to work in collaboration with your partner on step 9. Did you have any problems while working together?
- **s9_8** Does the software miss any information about your partner, which would helps you working in collaboration.
- **s9_9** One last question. Do you think it's beneficial working with more than one person on a diagram? Could you shortly explain your answers?

- **Step 10**
  - **s10_1** I think that I would like to use this system frequently.
  - **s10_2** I found the system unnecessarily complex.
  - **s10_3** I thought the system was easy to use.
  - **s10_4** I think that I would need the support of a technical person to be able to use this system.
  - **s10_5** I found the various functions in this system were well integrated.
  - **s10_6** I thought there was too much inconsistency in this system.

- **s10_7** I would imagine that most people would learn to use this system very quickly.
- **s10_8** I found the system very cumbersome to use.
- **s10_9** I felt very confident using the system.
- **s10_10** I needed to learn a lot of things before I could get going with this system.

# Anhang: Design & Implementierung

## B LARGO-Ontology

```
<ontology type="LARGO">
  <elements>

    <element elementid="test" elementtype="box" quantity="" minquantity=""
        maxquantity="">
      <elementoptions heading="Test" />
      <uisettings width="190" height="160" resizable="true" border="
          standard" background-color="#55C3FF" font-color="#000000" />
      <childelements>
        <element elementid="if" elementtype="text" quantity="1" minquantity
            ="1" maxquantity="1">
          <elementoptions label="IF" texttype="textfield" />
          <uisettings background-color="#ffffff" font-color="#e62bdb"
              minheight="16" maxheight="16"/>
        </element>
        <element elementid="and" elementtype="text" quantity="0"
            minquantity="0" maxquantity="5">
          <elementoptions label="AND" texttype="textfield"  longlabel="
              Condition" />
          <uisettings background-color="#FFFFFF" font-color="#000000"
              minheight="16" maxheight="16"/>
        </element>

        <element elementid="even" elementtype="text" quantity="0"
            minquantity="0" maxquantity="1">
          <elementoptions label="EVEN THOUGH" texttype="textfield" />
          <uisettings background-color="#FFFFFF" font-color="#000000"
              minheight="16" maxheight="16"/>
        </element>
        <element elementid="then" elementtype="text" quantity="1"
            minquantity="1" maxquantity="1">
          <elementoptions label="THEN" texttype="textfield" />
```

```
      <uisettings background-color="#FFFFFF" font-color="#000000"
          minheight="16" maxheight="16"/>
    </element>
    <element elementid="except" elementtype="text" quantity="0"
        minquantity="0" maxquantity="1">
      <elementoptions label="EXCEPT" texttype="textfield" />
      <uisettings background-color="#FFFFFF" font-color="#000000"
          minheight="16" maxheight="16"/>
    </element>
    <element elementid="transcriptlink" elementtype="transcript-link"
        quantity="0" minquantity="0" maxquantity="1">
      <elementoptions manualadd="false" longlabel="Transcript-Link"/>
      <uisettings  minheight="16" maxheight="16"/>
    </element>
    <element elementid="awareness" elementtype="awareness" quantity="1"
         minquantity="1" maxquantity="1">
      <elementoptions label="AWARENESS"  manualadd="false"/>
      <uisettings background-color="#FFFFFF" font-color="#000000"
          minheight="16" maxheight="16"/>
    </element>
  </childelements>
</element>

<element elementid="fact" elementtype="box" quantity="" minquantity=""
    maxquantity="">
  <elementoptions heading="Fact" />
  <uisettings width="190" height="120" resizable="true" border="double"
      background-color="#C0B7B4" font-color="#000000" />
  <childelements>
    <element elementid="text" elementtype="text" quantity="1"
        minquantity="1" maxquantity="1">
      <elementoptions />
      <uisettings background-color="#FFFFFF" font-color="#000000"
          minheight="40"/>
    </element>
    <element elementid="transcriptlink" elementtype="transcript-link"
        quantity="0" minquantity="0" maxquantity="1">
      <elementoptions manualadd="false" longlabel="Transcript-Link"/>
      <uisettings minheight="16" maxheight="16" />
    </element>
    <element elementid="awareness" elementtype="awareness" quantity="1"
         minquantity="1" maxquantity="1">
      <elementoptions label="AWARENESS"  manualadd="false"/>
      <uisettings background-color="#FFFFFF" font-color="#000000"
```

```
            minheight="16" maxheight="16"/>
      </element>
    </childelements>
  </element>

  <element elementid="hypothetical" elementtype="box" quantity=""
      minquantity="" maxquantity="">
    <elementoptions heading="Hypothetical" />
    <uisettings width="190" height="120" resizable="true" border="dashed"
        background-color="#FF8080" font-color="#000000" />
    <childelements>
      <element elementid="text" elementtype="text" quantity="1"
          minquantity="1" maxquantity="1">
        <elementoptions />
        <uisettings background-color="#FFFFFF" font-color="#000000"
            minheight="40"/>
      </element>
      <element elementid="outcome" elementtype="text" quantity="0"
          minquantity="0" maxquantity="1">
        <elementoptions label="OUTCOME" texttype="textfield" />
        <uisettings background-color="#FFFFFF" font-color="#000000"
            minheight="16" maxheight="16"/>
      </element>
      <element elementid="transcriptlink" elementtype="transcript-link"
          quantity="0" minquantity="0" maxquantity="1">
        <elementoptions manualadd="false" longlabel="Transcript-Link"/>
        <uisettings minheight="16" maxheight="16"/>
      </element>
      <element elementid="awareness" elementtype="awareness" quantity="1"
          minquantity="1" maxquantity="1">
        <elementoptions label="AWARENESS"  manualadd="false"/>
        <uisettings background-color="#FFFFFF" font-color="#000000"
            minheight="16" maxheight="16"/>
      </element>
    </childelements>
  </element>


  <element elementid="modified" elementtype="relation" quantity=""
      minquantity="" maxquantity="">
    <elementoptions heading="modified to" endings="true"/>

    <uisettings width="140" height="120" resizable="false" border=""
        background-color="#8080FF" font-color="#000000" linewidth="3px"
```

```
      linecolor="#8080FF"/>
  <childelements>
    <element elementid="awareness" elementtype="awareness" quantity="1"
        minquantity="1" maxquantity="1">
      <elementoptions label="AWARENESS"  manualadd="false"/>
      <uisettings background-color="#FFFFFF" font-color="#000000"
          minheight="16" maxheight="16"/>
    </element>
  </childelements>
</element>

<element elementid="distinguished" elementtype="relation" quantity=""
    minquantity="" maxquantity="">
  <elementoptions heading="distinguished by virtue of" endings="true"/>
  <uisettings width="215" height="60" resizable="false" border=""
      background-color="#edb3b3" font-color="#000000" linewidth="4px"
      linecolor="#edb3b3"/>
  <childelements>
    <element elementid="text" elementtype="text" quantity="1"
        minquantity="1" maxquantity="1">
      <elementoptions />
      <uisettings background-color="#FFFFFF" font-color="#000000"
          minheight="40"/>
    </element>
    <element elementid="awareness" elementtype="awareness" quantity="1"
        minquantity="1" maxquantity="1">
      <elementoptions label="AWARENESS"  manualadd="false"/>
      <uisettings background-color="#FFFFFF" font-color="#000000"
          minheight="16" maxheight="16"/>
    </element>
  </childelements>
</element>

<element elementid="analogized" elementtype="relation" quantity=""
    minquantity="" maxquantity="">
  <elementoptions heading="analogized by virtue of" endings="true"/>
  <uisettings width="205" height="60" resizable="false" border=""
      background-color="#c2edb2" font-color="#000000" linewidth="2px"
      linecolor="#c2edb2"/>
  <childelements>
    <element elementid="text" elementtype="text" quantity="1"
        minquantity="1" maxquantity="1">
      <elementoptions />
      <uisettings background-color="#FFFFFF" font-color="#000000"
```

```
            minheight="40"/>
      </element>
      <element elementid="awareness" elementtype="awareness" quantity="1"
          minquantity="1" maxquantity="1">
        <elementoptions label="AWARENESS"  manualadd="false"/>
        <uisettings background-color="#FFFFFF" font-color="#000000"
            minheight="16" maxheight="16"/>
      </element>
    </childelements>
  </element>

  <element elementid="leads" elementtype="relation" quantity=""
      minquantity="" maxquantity="">
    <elementoptions heading="leads to" endings="true"/>
    <uisettings width="115" height="120" resizable="false" border=""
        background-color="#B4B4B4" font-color="#000000" linewidth="1px"
        linecolor="#B4B4B4"/>
    <childelements>
      <element elementid="awareness" elementtype="awareness" quantity="1"
          minquantity="1" maxquantity="1">
        <elementoptions label="AWARENESS"  manualadd="false"/>
        <uisettings background-color="#FFFFFF" font-color="#000000"
            minheight="16" maxheight="16"/>
      </element>
    </childelements>
  </element>

  <element elementid="general" elementtype="relation" quantity=""
      minquantity="" maxquantity="">
    <elementoptions heading="comment:" endings="true"/>
    <uisettings width="160" height="60" resizable="false" border=""
        background-color="#FF8080" font-color="#000000" linewidth="1px"
        linecolor="#FF8080"/>
    <childelements>
      <element elementid="text" elementtype="text" quantity="1"
          minquantity="1" maxquantity="1">
        <elementoptions />
        <uisettings background-color="#FFFFFF" font-color="#000000"
            minheight="40"/>
      </element>
      <element elementid="awareness" elementtype="awareness" quantity="1"
          minquantity="1" maxquantity="1">
        <elementoptions label="AWARENESS"  manualadd="false"/>
        <uisettings background-color="#FFFFFF" font-color="#000000"
```

```
                minheight="16"  maxheight="16"/>
          </element>
        </childelements>
      </element>

  </elements>
</ontology>
```