Fachhochschule Nordwestschweiz

IP8 - Algoria: Kollaboration im Schulzimmer

Projektbericht

University of Applied Sciences Northwestern Switzerland - Computer Sciences

Studierender:	Reto Frey
Betreuender Dozent:	Prof. Dr. Christoph Stamm
Revision:	1.0
Datum:	28. September 2012

Zusammenfassung / Abstract

Zusammenfassung

Algoria Worksheet ist ein Programm, welches Studierenden und Dozierenden die Möglichkeit bietet, den Computer im Algorithmen und Datenstrukturen Unterricht sinnvoll und effektiv einzusetzen. Dazu behilft es sich einer interaktiven Zeichenfläche, welche aus gezeichneten Figuren Datenstrukturen erkennen und auswerten kann. Zusätzliche Funktionen sind das Stellen von Aufgaben und das automatische Überprüfen der gezeichneten Lösungen. Das Lösen von Aufgaben umfasst nicht nur das Zeichnen einer geforderten Datenstruktur, sondern auch die Beantwortung von Multiple Choice Fragen. Während das Überprüfen von Multiple Choice Antworten einfach ist, ist für die Überprüfung von Datenstruktur-Lösungen mit der geforderten Musterlösung eine Metrik entwickelt worden, welche eine Ähnlichkeit zwischen Musterlösung und der abgegebenen Lösung angibt.

Während sich die vorgehenden Arbeiten auf das Programm als Einzelplatzlösung fokussierten, legt diese Arbeit das Hauptaugenmerk auf den Einsatz von *Algoria Worksheet* im Klassenzimmer. Die Hauptgebiete des Klassenzimmer-Einsatzes liegen auf einer Kollaboration zwischen den Studierenden, was es ihnen ermöglicht, gemeinsam Aufgaben zu lösen, auf der Möglichkeit Arbeitsblätter für *Algoria Worksheet* im Programm zu versenden und einem Weg Aufgabenblätter parallel darstellen zu können.

Für das gleichzeitige Darstellen von mehreren Arbeitsblättern und dem Versenden solcher sind konkrete Erweiterungen implementiert worden. Bezüglich Kollaboration werden bereits bestehende Konzepte vorgestellt und analysiert. Nach der Analyse der verschiedenen Möglichkeiten wird ein Grundgerüst präsentiert, welches es erlaubt, eines oder mehrere dieser Konzepte in das Programm zu integrieren. Dieses Grundgerüst basiert auf dem Versenden von Aktionen, welche in der lokalen Applikation vollzogen werden. Je nach Konzept sollen diese Aktionen in einer späteren Arbeit synchronisiert werden.

Abstract

Algoria Worksheet is an application which provides the ability to use the computer in the algorithm and data structure lectures in a useful and effective way to students as well as lectures. To achieve that it uses an interactive drawing board which is able to recognize and analyse data structures from drawn figures. Additionally the application provides the functionality to create worksheets and the automatic correction of solution to these worksheets. The solving of tasks within a worksheet does not only contain the drawing of an asked data structure but also the answering of multiple choice questions. The correction of a multiple choice task is relatively easy to achieve. For the marking of drawing-based tasks a metric has been developed. This metric describes the similarity of the given sample solution to the solution a student provides when answering a question.

Where the previous reports mainly focussed on the application in a single place environment this report directs its attention to the service of *Algoria Worksheet* within a classroom. The main areas of the classroom service are the collaboration between multiple students, which allows them to work simultaneously on the same task, the ability to send and receive worksheets within the *Algoria Worksheet* application and a way to show different worksheets at the same time.

For the concurrent displaying of different worksheets as well as for the sending and receiving of worksheets concrete implementations have been made during this work. Concerning the collaboration different existing concepts are presented and analysed. After the analysis of these concepts a framework is presented which allows it to integrate one or more of these concepts within the application. This framework is based on the sending of actions which are made in the local application. Based on the choice of the concept used these actions should be synchronized in an upcoming work.

Inhaltsverzeichnis

1. Disposition

Die Disposition gibt in vier Seiten einen Überblick über die Ausgangslage und die Anforderungen des Projektes. In der Ausgangslage werden die bereits bestehenden Softwareteile vorgestellt, was den Einstieg in die Thematik und das Verständnis erleichtern soll. Die Anforderungsbeschreibung, welche danach beschrieben wird, enthält zwei konkrete Fragestellungen, sowie die daraus abgeleiteten Anforderungen an die zu erweiternde Applikation.

1.1. Ausgangslage

In den folgenden Abschnitten wird die Ausgangslage dieses Projektes beschrieben. Da es sich um ein Fortsetzungsprojekt handelt, werden die Arbeiten, welche in den voraus gegangenen Projekten geleistet wurden, beschrieben. Das erste Projekt, welches in dieser Projektreihe steht, ist die Entwicklung einer interaktiven Zeichenfläche. Dies wird unter 1.1.1 behandelt. Das darauf aufbauende Programm *Algoria Worksheet* wird im Kapitel 1.1.2 beschrieben.

1.1.1. Algoria

Algoria ist eine interaktive Zeichenfläche, welche sich mit Vorteil mit einem Tablet-Computer bedienen lässt. Sie erlaubt das einfache Zeichnen von Datenstrukturen, welche durch die *Algoria*-Engine erkannt werden. Für diesen Schritt werden zuerst die gezeichneten Striche getrennt verarbeitet. Somit entsteht ein Bild aus einzelnen Strichen, welche nun weiter analysiert werden. Anhand von so genannten Ladder-Definitionen wird nun versucht, eine beschriebene Form zu finden. Solche Formen können z.B. Pfeile sein. Hier wird beschrieben, wie drei Striche zueinander stehen sollen (Winkel zwischen den Strichen) und wo sie sich schneiden sollen (jeweils an einem Ende der jeweiligen Striche). Eine andere Definition wäre, wie ein Array gezeichnet werden soll. Hier werden die bereits erwähnten Attribute so gesetzt, dass sich 4 Striche zu einem Rechteck vereinen. Wenn die Applikation eine Datenstruktur erkannt hat, wird diese im Speicher nachgebildet.

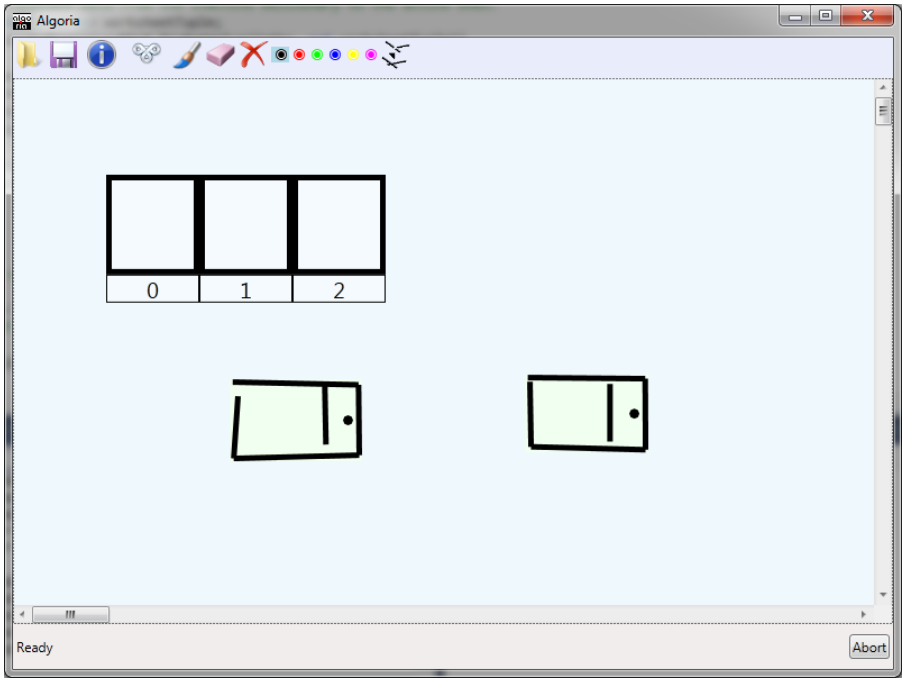

Abbildung 1.1.: Algoria

Die Abbildung 1.1 zeigt einen Screenshot von *Algoria*. In der Zeichenfläche ist bereits ein Array sowie zwei Linked-List Elemente erkannt worden. Weitere Funktionen, welche *Algoria* bietet, sind die folgenden:

- Vordefinierte Algorithmen auf den erkannten Datenstrukturen ausführen
- Verschieben von Datenstrukturen
- Ändern von Datenstrukturen (Hinzufügen und Löschen von Elementen innerhalb einer Datenstruktur)
- Manipulieren von Werten innerhalb einer Datenstruktur
- Speichern und Laden von Zeichenflächen

1.1.2. Algoria Worksheet

Algoria Worksheet wurde während der gleichnamigen Bachelor Thesis[1] geplant und entwickelt. Ziel dieses Programmes ist es, den Algorithmen und Datenstrukturen Unterricht dahingehend zu bereichern, dass der Computer eine sinnvolle Rolle übernehmen kann. So soll es Studierende während und neben dem Unterricht unterstützen. Dozierende haben die Möglichkeit, Aufgabenblätter, bestehend aus mehreren Aufgaben, zu erstellen. Diese Aufgaben können zum einen vom Typ Multiple Choice sein, zum anderen kann der Dozierende eine Musterlösung in Form von gezeichneten Datenstrukturen eingeben. Beim zweiten Typ müssen Studierende so nah wie möglich an die geforderte Musterlösung kommen. Damit Studierende möglichst schnell eine Rückmeldung zu ihrer Lösung erhalten, wurde eine automatische Lösungsprüfung in die Applikation integriert. Diese Rückmeldung besteht darin, dass dem Studierenden mitgeteilt wird, zu wie viel Prozent sich seine Lösung mit der Musterlösung deckt. Für die Berechnung dieses Prozentwertes wurde eine Metrik für die Messung der Ähnlichkeit zwischen zwei Datenstrukturen entwickelt. Die Metrik zählt, die notwendigen Aktionen, welche notwendig sind, um aus einer gegebenen Datenstruktur (Studierendenlösung) eine Gesuchte (Musterlösung) zu machen. Mögliche Aktionen sind das Hinzufügen von zusätzlichen Elementen oder das Ändern eines Wertes in einem Element. Jede dieser Aktionen hat ein Gewicht, das heisst, dass das Fehlen einer Datenstruktur schlimmer ist als wenn nur ein Wert in der Datenstruktur falsch ist. Mithilfe dieser Schritte können den Studierenden Tipps gegeben werden, wie sie ihre Lösung weiter verbessern können.

Für beide Aufgabentypen kann ein Aufgabentext mit Anweisungen für die Aufgabe eingegeben werden. Für das Einlesen von Datenstrukturen verwendet *Algoria Worksheet* die interaktive Zeichenfläche, welche bereits in *Algoria* realisiert wurde. Da die Anforderungen von Dozierenden und Studierenden an das Programm sehr verschieden sind, wurden zwei verschieden Versionen des Programmes entwickelt, die Studierenden- und die Admin-Version.

[1] http://webapache.imvs.technik.fhnw.ch/~christoph.stamm/reports/P6_2011_Algoria_Worksheet.pdf[Frey11]

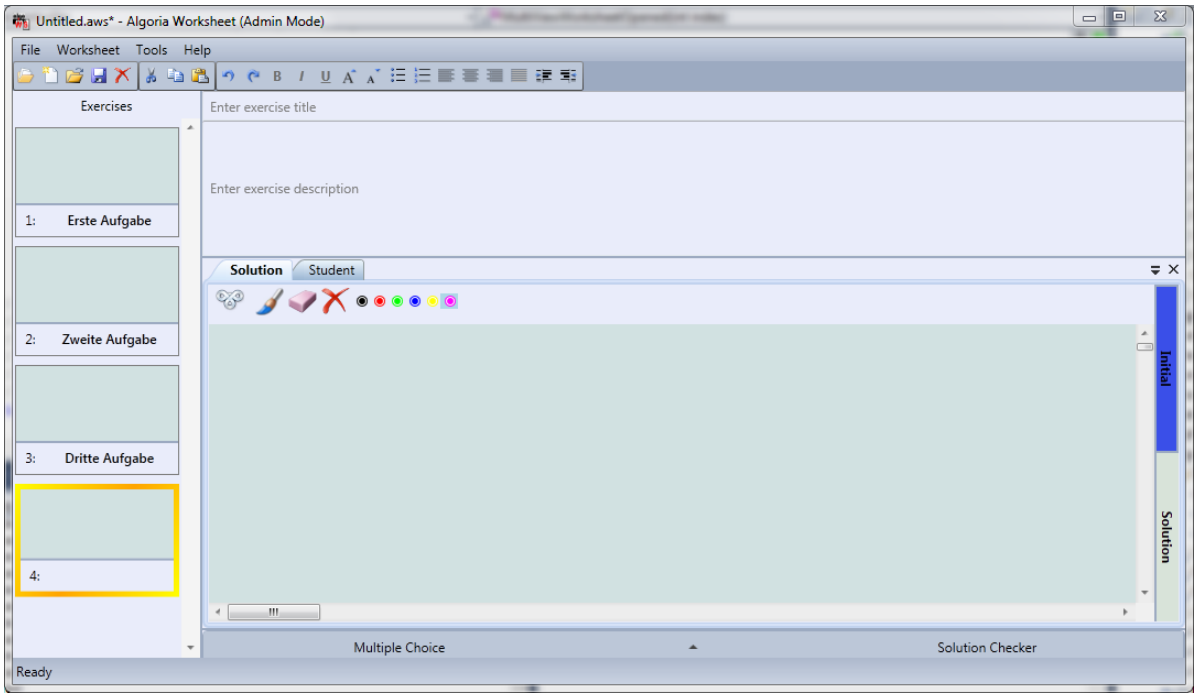

Abbildung 1.2.: Algoria Worksheet

Die Abbildung 1.2 zeigt *Algoria Worksheet* in der Admin-Version mit einem erstellten Arbeitsblatt. Das Arbeitsblatt enthält zum jetzigen Zeitpunkt vier verschiedene Aufgaben. Mithilfe der linken Seitenliste kann, analog zu Power Point, zwischen den verschiedenen Aufgaben gewechselt werden. Die Seitenliste verfügt über Drag & Drop Funktionalitäten, um die Aufgabenreihenfolge verändern zu können. Multiple Choice Antworten können im unten aufklappbaren Bereich erfasst werden.

Nebst den bereits beschriebenen Arbeiten während der Bachelor Thesis wurde die Applikation im Verlauf der Projektarbeit 7 mit dem Titel „P7: Algoria – Persönliche Lernumgebung“[2] weiterentwickelt. Hauptaugenmerk der Projektarbeit war die Verbesserung der automatischen Lösungskorrektur. Vor der Projektarbeit 7 wurde die Metrik mit fest eingestellten Gewichtungen für jeweils notwendige Aktionen erstellt. Nach der Projektarbeit haben Dozierende die Möglichkeit pro Datenstrukturtyp, welcher in einer Musterlösung vorhanden ist, Gewichtungen für einzelne Aktionen zu setzen. Eine weitere Änderung an der Lösungsprüfung ist das Verhalten bei mehreren Datenstrukturen in der Musterlösung und in der Studierendenlösung. Die Paarung zwischen den Datenstrukturen der Muster- und Studierendenlösung wird so erstellt, dass die resultierende Summe aus allen Metriken der gewählten Paare möglichst klein ist. Das Problem des Findens einer optimalen Paarung wird durch das Assignment Problem[3] beschrieben. Lösungsalgorithmen für dieses Problem können somit auch für das Finden von Paarungen zwischen Datenstrukturen angewendet werden.

[2] http://webapache.imvs.technik.fhnw.ch/~christoph.stamm/reports/P7_2012_Algoria_Lernumgebung.pdf [Frey12]
[3] http://www.me.utexas.edu/~jensen/models/network/net9.html [Jens12]

1.2. Anforderungsbeschreibung

Bereits in der vorhergegangenen Projektarbeit P7 sind Überlegungen angestellt worden, wie sich *Algoria Worksheet* ändern müsste, um eine Kollaboration zwischen mehreren Beteiligten zu ermöglichen. Hierbei soll nicht nur ein homogenes Arbeiten (nur unter Studierenden bzw. nur unter Dozierenden) sondern auch eine Kollaboration zwischen Dozierenden und Studierenden ermöglicht werden. Dieser Aspekt ist das Hauptaugenmerk der vorliegenden Arbeit. Die folgenden Fragen stammen aus der Aufgabenstellung[4] und sollen während des Projektes beantwortet werden.

- Welchen Anforderungen muss eine kollaborative Lernumgebung für das Klassenzimmer genügen?
- Wie lässt sich Algoria Worksheet fürs Klassenzimmer anpassen bzw. umbauen und wie überprüfen Sie den Nutzen?

Diese Fragen ziehen konkrete Anforderungen an *Algoria Worksheet* nach sich. Die Tabelle 1.1 listet diese auf. Zusätzlich wird angegeben, welches Kapitel dieses Berichtes sich mit der Anforderung befasst.

Anforderung	Behandelndes Kapitel
Studierende können gleichzeitig an einer Aufgabe arbeiten.	Kapitel 2, Seite 8
Dozierende können Aufgaben oder Arbeitsblätter an Studierende versenden, ohne dazu eine externe Datenablage oder E-Mail zu verwenden.	Kapitel 3.1, Seite 17
Studierende können ihre Lösung einer Aufgabe oder eines Arbeitsblattes an den Dozierenden senden, ohne dazu eine externe Datenablage oder E-Mail zu verwenden.	Kapitel 3.1, Seite 17
Dozierende können den Arbeitsfortschritt der Studierenden verfolgen. Dies ist als 'Live-Übertragung' der Bearbeitung zu verstehen, Dozierende haben also keine Möglichkeit in die Bearbeitung einzugreifen.	Kapitel 3.2, Seite 22
Dozierende können auf einem grossen Display mehrere Lösungen gleichzeitig darstellen.	Kapitel 3.3, Seite 25

Tabelle 1.1.: Anforderungen an *Algoria Worksheet*

[4] Siehe Anhang A, Seite 34

2. Konzept kollaborative Bearbeitung

Die Einführung einer kollaborativen Komponente in *Algoria Worksheet* stellt zusätzliche Anforderungen an die Applikation:

- Die, an der Kollaboration teilnehmenden, Instanzen müssen stets mit dem aktuellen Stand des zu bearbeitenden Dokumentes versorgt sein.
- Der gleichzeitige Zugriff auf dasselbe Dokument muss ermöglicht und synchronisiert werden.
- Der, durch ein grosses Display ermöglichte, zusätzliche Platz muss sinnvoll ausgenutzt werden.

Einige dieser Anforderungen sind Implementationsanforderungen, bei welchen die Schwierigkeit primär in der Planung und Durchführung der Implementation steckt. Diese Aspekte werden in Kapitel 3 ab Seite 17 behandelt. Die Synchronisation des gleichzeitigen Zugriffs auf ein Dokument erfordert jedoch einen Mechanismus zur Sicherstellung der Konsistenz.

Im Folgenden wird das Problem des gleichzeitigen Zugriffs auf eine Ressource analysiert und formal aufgezeigt. Anschliessend werden bereits verwendete Lösungsansätze gezeigt und bezüglich *Algoria Worksheet* analysiert. Den Abschluss dieses Kapitels machen weitere, spezifisch auf *Algoria Worksheet* ausgelegte, Überlegung zur Konsistenzsicherung.

2.1. Sicherstellung der Konsistenz

Damit sichergestellt werden kann, dass ein gemeinsam editiertes Dokument in allen bearbeitenden Instanzen denselben Inhalt hat, bedarf es einer Konsistenzsicherung. Diese dient dazu, Änderungen in einer Instanz konfliktfrei auf allen anderen Instanzen durchzuführen. Für die Darstellung einer solchen kollaborativen Bearbeitung eines Dokumentes in *Algoria Worksheet* wird folgendes Szenario betrachtet: die Studierenden A und B arbeiten kollaborativ an demselben Arbeitsblatt. Dieses besteht aus den Aufgaben E_1 und E_2. Für dieses Szenario gibt es zwei denkbare Ausgangslagen.

- A und B arbeiten an verschiedenen Aufgaben
- A und B arbeiten an derselben Aufgabe (beide an E_1 oder E_2)

Solange A und B an verschiedenen Aufgaben innerhalb eines Arbeitsblattes arbeiten, besteht keine zusätzliche Aufgabe für das Sicherstellen der Konsistenz. Die Änderungen einer Seite können problemlos auf der gegenüberliegenden Seite nachgetragen werden. Wenn die Studierenden aber an derselben Aufgabe arbeiten kann dies zu Problemen bezüglich der Konsistenz führen. Die Bearbeitung einer Aufgabe in *Algoria Worksheet* besteht zumeist aus dem Zeichnen einer geforderten Situation in der interaktiven Zeichenfläche. Aus diesem Grund kann im Folgenden die kollaborative Bearbeitung einer Aufgabe auf das gleichzeitige Bearbeiten einer Zeichenfläche reduziert werden.

Die Tabelle 2.1 zeigt, wie zwei Studierende (A und B) die gleiche Zeichenfläche editieren. Dazu arbeiten sie jeweils in eigenständigen Instanzen von *Algoria Worksheet*, arbeiten aber beide an derselben Aufgabe beispielsweise E_1. Die aufgeführten Aktionen werden immer zeitgleich versendet. Fälle, in welchen jeweils nur einer der beiden Studierenden eine Aktion durchführt, werden nicht aufgeführt. Es besteht in solchen Fällen ein Potential für einen Konflikt. Die Zeichenflächen werden im Folgenden als W_A und W_B bezeichnet.

Für das Beispiel wird angenommen, dass alle aufgeführten Aktionen als atomar gelten. Es wird also in Szenario 1 erst dann eine Aktion ausgelöst, wenn das Array A_1 in W_A fertig gezeichnet und erkannt wurde. Dadurch, dass beide Arrays in verschiedenen Instanzen erzeugt werden, kann gesagt werden das $A_1 \neq A_2$. Wenn die Granularität der Aktionen verfeinert wird, ist jedoch auch hier das Potential eines Konfliktes vorhanden. Dieses Problem wird im Kapitel 2.3.1 betrachtet.

Szenario	Ausgangslage	Aktion in W_A	Aktion W_B	Konsequenz
1	Eine leere Zeichenfläche	Array A_1 wird gezeichnet	Array A_2 wird gezeichnet	kein Konflikt
2	Array A_1 mit 4 Elementen ist auf der Zeichenfläche	Der Wert im ersten Element von A_1 wird geändert	Array A_1 wird auf der Zeichenfläche verschoben	Lösbarer Konflikt: A_1 ist am neuen Ort und hat den neuen Wert
3	Array A_1 mit 4 Elementen ist auf der Zeichenfläche	Array A_1 wird an den linken Rand der Zeichenfläche verschoben	Array A_1 wird an den rechten Rand der Zeichenfläche verschoben	Nicht lösbarer Konflikt: Einer der beiden Änderungen wird keinen Effekt haben

Tabelle 2.1.: Beschreibung der Szenarien bei gemeinsamer Bearbeitung

Das beschriebene Szenario hat zwei unterschiedliche Arten von Konflikten aufgezeigt. Bei einem lösbaren Konflikt muss sichergestellt werden, dass dieser in beiden Zeichenflächen aufgelöst wird (siehe Szenario 2 in Tabelle 2.1). Ein solcher Konflikt kann also so behandelt werden, dass keine zusätzliche Benutzerinteraktion notwendig ist und das System in einem konsistenten Zustand bleibt. Für unlösbare Konflikte muss eine separate Strategie entwickelt werden. Hier ist es denkbar, einen der Konfliktverursacher zu bevorzugen und die Änderung des anderen zu ignorieren. Eine Alternative dazu ist es, einem Benutzer die Möglichkeit zu geben, interaktiv zu entscheiden, welche Operation angewendet und welche verworfen werden soll. Generell ist zu sagen, dass Mechanismen zur automatischen Konfliktbewältigung nur bei lösbaren Konflikten eingesetzt werden können. Unlösbare Konflikte können nicht automatisch behoben, verhindern werden können sie jedoch (zum Beispiel mithilfe von Locking).

Damit bereits bestehende Lösungsansätze für die Sicherstellung der Konsistenz in kollaborativen Systemen analysiert werden können, muss eine Konfliktsituation formal definiert werden. Nach [Elli89] ist ein Konflikt folgendermassen formal definiert:

Algoria Worksheet Instanzen: W_1, W_2, $W_1 \neq W_2$
Operationen: O_1, O_2,
O_1 wird ausgeführt bei W_1, O_2 wird ausgeführt bei W_2
Konfliktfreie Situation: $O_1 \circ O_2 = O_2 \circ O_1$
Konfliktsituation: $O_1 \circ O_2 \neq O_2 \circ O_1$

Auch Verwendung des Symboles $\circ$ als Komposition ist nach [Elli89]. Damit diese Definition stimmt, muss folgende Annahme gelten: Zur Zeit der Ausführung von O_2 bei W_2 darf die Operation O_1 noch nicht übertragen worden sein. Die Operationen müssen sich überlappen. Ein Sequenzdiagramm zweier Operationen ist in der Abbildung 2.1 zu sehen. Der linke Ablauf zeigt eine Situation ohne überlappende Operationen. Auf der rechten Seite überschneiden sich jedoch die Operationen.

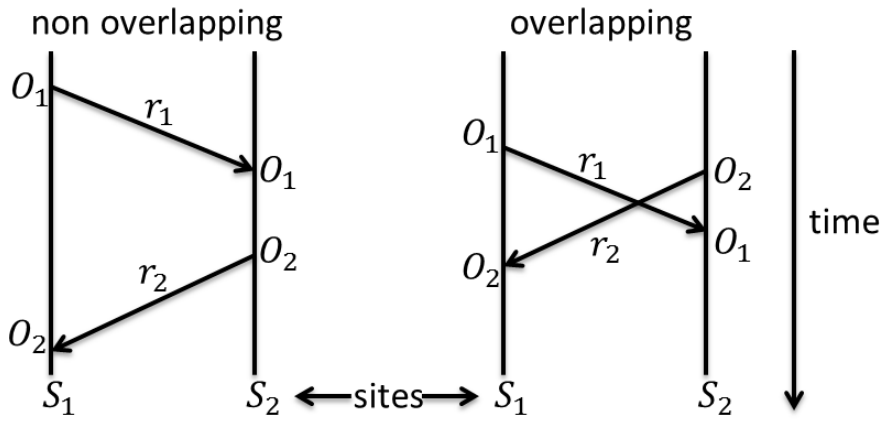

Abbildung 2.1.: Nicht-überlappende und überlappende Operationen aus [Elli89]

2.2. Lösungsansätze zur Konsistenzsicherung in kollaborativen Systemen

Wenn es um kollaborative Systeme geht, ist Google Wave (verfügbar bis April 2012, nun Google Docs[1]) eine der bekanntesten Anwendungen. Mit der webbasierten Applikation ermöglicht es Google mehreren Benutzern gleichzeitig an einem Dokument zu arbeiten. Es werden einige Dokumenttypen angeboten. Die Spanne reicht von Textdokumenten, über Tabellenkalkulationen, Präsentationen und Zeichnungen bis hin zu Umfrageformularen. Welche dieser Anwendungen jedoch betrachtet wird, ist nicht entscheidend. Das Potential, dass bei der gleichzeitigen Bearbeitung Konflikte entstehen, ist in jeder dieser Anwendungen gegeben. Wichtig ist die Art und Weise, wie es ermöglicht wird, dass mehrere Benutzer, trotz dem Auftreten von Konflikten, an einem Dokument arbeiten können. Die Lösung bei Google Wave heisst Operational Transformation (OT). Nach dem Einstellen des Google Wave Projekts wurden mehrere Whitepaper[2] über das Protokoll veröffentlicht, welches es erlaubt, die einzelnen Instanzen synchron zu halten. Auch das heute aktive Google Docs soll auf vergleichbaren Protokollen beruhen. Die Konfliktlösung mit OT und weitere Möglichkeiten mit Konflikten umzugehen, werden im Folgenden besprochen und deren Relevanz für *Algoria Worksheet* aufgezeigt.

2.2.1. Operational Transformation

Eine Anwendung von Operational Transformation wird sowohl in [Elli89] als auch in [Goog12b] beschrieben. Das Herzstück dieser Methode ist es, mithilfe einer Transformation der Operationen den Konflikt zu beseitigen. Das heisst, es wird eine Funktion T benötigt, welche folgende Eigenschaft hat:

$O_1 \circ O_2 \neq O_2 \circ O_1$
$O_1^\tau = T(O_1, O_2)$, $O_2^\tau = T(O_2, O_1)$
$O_1 \circ O_2^\tau = O_2 \circ O_1^\tau$

Die Operationen wurden so verändert, dass sie keinen Konflikt mehr verursachen. Im Folgenden wird ein Beispiel für diese Veränderung einer Operation in *Algoria Worksheet* angegeben. Das Beispiel ist angelehnt an eine Variante, welche in [Elli89] auf Seite 402 zu finden ist.

Ausgangslage: Array $A_1 = [1, 2, 3, 4]$
Operation O_1: Löschen des zweiten Elementes im Array
Operation O_2: Löschen des dritten Elementes im Array
$O_1 \circ O_2(A_1) = O_2([1, 3, 4]) = [1, 3]$
$O_2 \circ O_1(A_1) = O_1([1, 2, 4]) = [1, 4]$

[1] http://docs.google.com[Goog12a]
[2] http://www.waveprotocol.org/whitepapers[Goog12c]

Es ist klar zu erkennen, dass hier ein Konflikt entsteht, wenn die Reihenfolge der Operationen vertauscht wird. Nun wird die Transformationsfunktion auf die Operationen angewendet.

$O_1^\tau = T(O_1, O_2)$: Löschen des zweiten Elementes
$O_2^\tau = T(O_2, O_1)$: Löschen des zweiten Elementes
$O_1 \circ O_2^\tau(A_1) = O_2^\tau([1,3,4]) = [1,4]$
$O_2 \circ O_1^\tau(A_1) = O_1^\tau([1,2,4]) = [1,4]$
Dieses Beispiel ist ausgelegt für einen überschaubaren Konflikt in *Algoria Worksheet*. Es zeigt, dass eine grundsätzliche Machbarkeit besteht. Das Problem des Findens einer solchen Transformationsfunktion besteht jedoch weiterhin.

2.2.2. Operation Serialization

Der Ansatz der Operation Serialization versucht, eine ‚happens-before'-Relation zwischen allen Operationen zu erkennen und diese Relationen stets zu bewahren. Da Aktionen aber auch vollkommen unabhängig voneinander ablaufen können, ist es nicht immer trivial zu entscheiden, welche Operation jetzt vor welcher sein muss. Das Ziel dieser Serialisierung ist es, eine Operations-Liste zu pflegen, welche die ausgeführten Operationen enthält. Wird nun eine Operation von einer anderen *Algoria-Worksheet*-Instanz empfangen, muss diese Operation am richtigen Ort in der Liste eingefügt werden. Gesucht wird also auch hier eine Funktion, hier S.

Momentane Operationsliste: $OL_n = [O_1, O_2, O_3]$
Neu empfangene Operation: O_{new}
Gesuchte Funktion: $OL_{n+1} = S(OL_n, O_{new})$,
sodass $O_{new} \in OL_{n+1}$ mit den zuvor beschriebenen Eigenschaften.

Beim Eintreffen einer neuen Aktion, werden alle bereits bestehenden Operation in der Operations-Liste rückgängig gemacht. Die neu empfangene Operation wird mithilfe der Operation Serialization an der korrekten Stelle in die Liste eingefügt. Werden nun alle Operationen in der Liste in der neuen Reihenfolge ausgeführt, haben alle teilnehmenden *Algoria Worksheet*-Instanzen denselben Stand. Das Problem des Suchens einer Ordnung aller Operationen inkl. der neu empfangenen ist in [Igna06] auf ein Graphen-Problem reduziert worden. Durch die Möglichkeit, bereits Operationen in *Algoria Worksheet* zu versenden, wäre auch das Führen einer Liste von Operationen machbar.

2.2.3. Locking

Im Gegensatz zu den zwei bereits vorgestellten Varianten der Konfliktbewältigung beschäftigt sich Locking nicht mit dem Auflösen von Konflikten, sondern mit dem Verhindern dieser. Beispielsweise wäre es denkbar, die gesamte Zeichenfläche für alle anderen Benutzer zu sperren, sobald ein Benutzer beginnt, etwas zu verändern oder neu zu zeichnen. Dieses Sperrverhalten kann beliebig adaptiert werden. In [Chen01] werden verschiedene Versionen von Locking vorgestellt und besprochen. Objekt- oder Region-Locking wären in *Algoria Worksheet* realisierbar. Das Konzept, es einem Benutzer zu verunmöglichen gleichzeitig die interaktive Zeichenfläche zu verwenden, widerspricht aber dem Grundgedanken der Kollaboration. Aus diesem Grund wird auf diese Konfliktbewältigungslösung nicht weiter eingegangen.

2.2.4. Differential Synchronization

Auch aus dem Hause Google stammt das Prinzip Differential Synchronization. Neil Fraser beschreibt in [Fras09] eine Möglichkeit der Synchronisation, welche weder auf Locking, dem Versenden von Operationen (gepaart mit Operational Transformation oder Synchronization) noch auf einem Three-way

merge beruht. Letzterer wird vor allem im Umfeld von Subversion-Systemen gefunden. Der Grundgedanke des Systems beruht darauf, in zyklischen Abständen immer die gemachten Änderungen seit dem letzten Update zu versenden. Für ein leichteres Verständnis ist in Abbildung 2.2[Fras12] der Aufbau des Systems zu sehen.

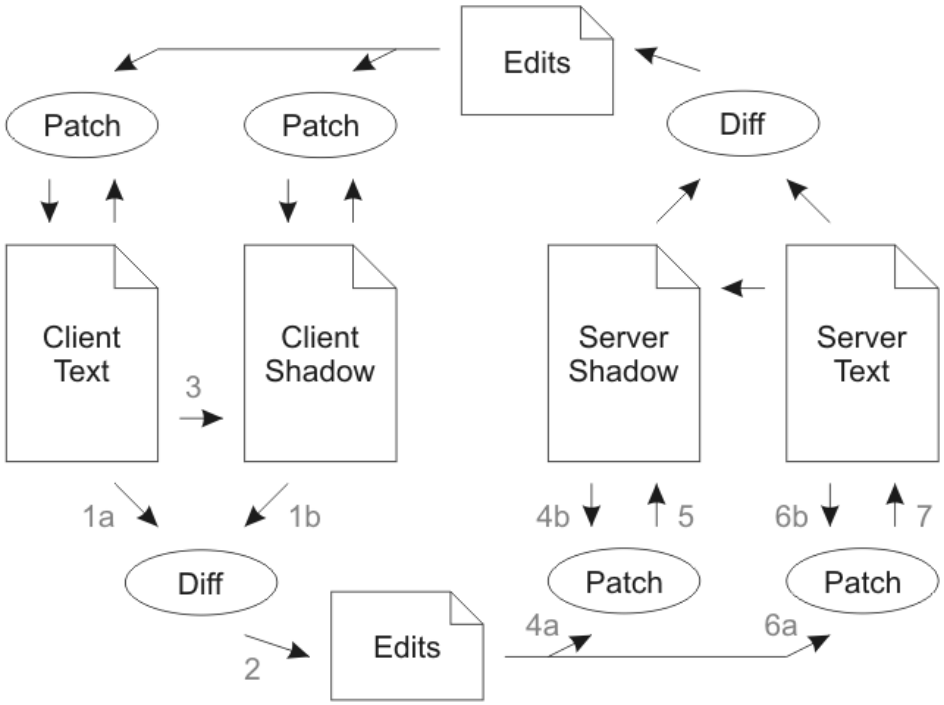

Abbildung 2.2.: Differential Synchronization Systemaufbau mit der Dual Shadow Methode

Sowohl Server als auch Client verfügen über eine Kopie ihres letzten Standes. Wenn dem Synchronisationszyklus zufolge die gemachten Änderungen wieder versendet werden müssen, wird der momentane Stand mit der Kopie verglichen und die Differenzen versendet. Im Anschluss daran wird die Sicherheitskopie auf den neusten Stand gebracht. Auf der Empfängerseite werden sowohl Sicherheitskopie als auch der produktive Stand aktualisiert. Nun wird wieder die Differenz zwischen Sicherheitskopie und produktivem Stand gesucht und versendet. Dies ist ein unendlicher Zyklus. Auch wenn dieses System scheinbar über eine bessere Fehlerresistenz verfügt als die Anderen, hat es gewisse Nachteile bei einem denkbaren Einsatz in *Algoria Worksheet*.

Sowohl die Diff- (Suchen von Unterschieden zwischen aktuellem Stand und Sicherheitskopie) als auch die Patchfunktion (Aktualisieren von Produktivstand und Sicherheitskopie) sind im Umfeld von *Algoria Worksheet* zum jetzigen Zeitpunkt schwer vorzustellen. Ein weiterer Nachteil ist, dass alle Benutzer mit dem Server einen getrennten Update-Zyklus haben. Dies hat zur Folge, dass der Server für jeden Benutzer im Minimum eine Sicherheitskopie hat und bedeutet eine sehr viel höhere Anforderung an die Funktionalität des Servers.

2.2.5. Handlungsvorschlag für Algoria Worksheet

Nachdem einige Varianten der Sicherstellung der Konsistenz vorgestellt worden sind, ist bereits erkennbar, dass Locking keine valide Alternative ist. Durch die Sperrung von ganzen Zeichenflächen, oder Regionen dieser, wird die Ansprechbarkeit verringert. Dies ist nicht im Sinne der Kollaboration. Ein Differential Synchronization-Ansatzes verfügt nicht über die Schwächen eines Locking-Ansatzes. Jeder Benutzer kann zu jeder Zeit in der lokalen Zeichenfläche arbeiten. Die Verwendung von Update-Zyklen hat aber einen Nachteil. Wenn diese Zyklen zu lang sind, werden Änderungen erst sehr spät

übertragen. Dies kann zu schwerwiegenderen Konflikten führen, da eine ganze Reihe von Aktionen in demselben Update-Zyklus versendet werden. Ist der Update-Zyklus hingegen sehr kurz, führt dies zu einer hohen Auslastung von Server und Client Applikation. Da ein Server für jeden angemeldeten Client einen eigenen Zyklus hat, kann dies schnell zu massiven Leistungseinbussen führen.
Die zwei Varianten, welche für *Algoria Worksheet* am erfolgversprechendsten scheinen, sind Operation Transformation und Operation Serialization. Operation Serialization kann mithilfe von Undo-Redo Aktionen realisiert werden. Die Schwierigkeit bei einem Operation Transformation Ansatz ist das Finden einer Transformationsfunktion.

2.3. Synchronisation in Algoria Worksheet

Nach der Betrachtung von möglichen Synchronisierungsalgorithmen, welche teilweise bereits auf zeichnerischen Programmen basieren, werden im Folgenden spezifische Überlegungen bezüglich *Algoria Worksheet* angestellt.

Eine erste Überlegung bezieht sich darauf, wie die Anzahl auftretender Konflikte reduziert werden kann, ohne dass ein Locking die Interaktion zwischen Benutzer und Applikation negativ beeinflusst. Eine einfache Möglichkeit die Wahrscheinlichkeit zu reduzieren, dass zwei Benutzer an demselben Objekt oder in derselben Region arbeiten, ist es, die Benutzer über die Aktionen der anderen Teilnehmer zu informieren. So wäre es denkbar, dass die Mausbewegungen eines Benutzers für andere sichtbar gemacht werden. Realisierbar wäre dies in einer Form wie sie in Abbildung 2.3 zu sehen ist.

Drei aufgeführte Benutzer scheinen sich mit dem Eingabegerät auf der Zeichenfläche zu befinden. Damit die Position des Eingabegerätes nicht ständig aktualisiert werden muss, wäre es eine Möglichkeit, die Fläche in einem gewissen Radius um die tatsächliche Position anzugeben. Wenn ein Benutzer weiss, dass bereits jemand anders an einer Stelle potentiell etwas editiert, ist die Wahrscheinlichkeit kleiner, dass er dazwischen funkt.

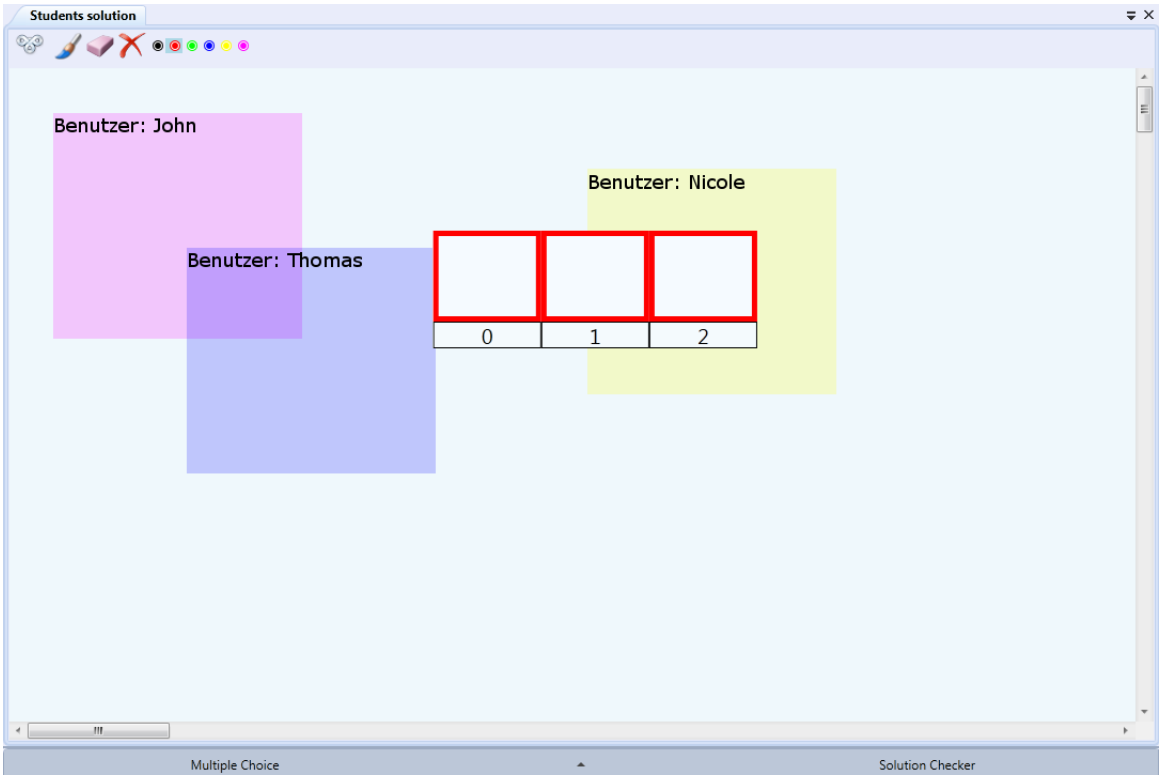

Abbildung 2.3.: Anzeigen der Bewegungen von anderen Benutzern

2.3.1. Granularität der Übertragenen Aktionen

Ein weiterer Faktor, welcher bei einer gemeinsamen Bearbeitung betrachtet werden muss, ist die Granularität der versendeten Aktionen. Die vorstellbaren Granularitätsstufen werden in der folgenden Tabelle 2.2 aufgelistet. Darin sind nur die Operationen aufgeführt, welche direkt mit dem Zeichnen zu tun haben. Datenstrukturen können aber auch über ein Kontextmenü bearbeitet werden. In diesem Falle gibt es aber keine unterschiedliche Granularität. Eine Änderung ist nach einem Klick erledigt.

Granularitätsstufe	Beschreibung
1	Übertragen jedes Berührungspunktes des Eingabegerätes mit der interaktiven Zeichenfläche
2	Übertragen der Eingabe, wenn das Eingabegerät die Eingabe beendet hat (bspw. wenn der Stift abgesetzt oder die Maustaste losgelassen wurde)
3	Übertragen der von der *Algoria*-Engine erkannten Liniensegmente
4	Übertragen der fertig erkannten Datenstrukturen

Tabelle 2.2.: Beschreibung der Granularitätsstufen

Klar zu erkennen ist, je später die Übertragung stattfindet, desto später und weniger häufig muss etwas übertragen werden. Die abnehmende Granularität hat jedoch zur Folge, dass es weniger zu einem Live-Gefühl kommt. Neu gezeichnete Datenstrukturen erscheinen plötzlich anstatt durch das Zeichnen von Linien zu entstehen. Feingranulare Übertragung hat jedoch zusätzliche Anforderung an die Synchronisation. Beispielsweise könnten Benutzer die Linien anderer Benutzer verwenden um eine Datenstruktur zu vervollständigen. Wenn sich nun die Situation ergibt, dass zwei Benutzer dieselbe Linie für zwei unterschiedliche Datenstrukturen verwenden, ist ein Konflikt die Folge. Ferner muss beachtet werden, dass in gewissen Situationen derselbe Input in *Algoria* nicht denselben Output hat. Je nachdem, welche Eingabe zuerst bei einer Zeichenfläche prozessiert wird, können unterschiedliche Resultate folgen. Dieses Phänomen kommt vor allem bei den Granularitätsstufen 1 und 2 zum Tragen.

2.3.2. Update-Aktionen

Durch die, im vorhergehenden Kapitel genannten, Stufen der Granularität ändern sich auch die Aktionen, welche in *Algoria Worksheet* versendet werden. Es folgen Auflistungen der denkbaren Aktionen in den verschiedenen Granularitätsstufen. Die jeweiligen Tabellen haben keinen Anspruch auf Vollständigkeit und müssen gegebenenfalls erweitert werden.

Name der Aktion	Verwendung
MouseLeftClick	Während die linke Maustaste gedrückt wird, werden alle Berührungspunkte der Maus mit der interaktiven Zeichenfläche übertragen.
MouseRightClick	Während die rechte Maustaste gedrückt wird, werden alle Berührungspunkte der Maus mit der interaktiven Zeichenfläche übertragen.

Tabelle 2.3.: Beschreibung von denkbaren Aktionen (Granularitätsstufe 1)

Die Aktionen in Tabelle 2.3 ermöglichen ein sehr flüssiges Darstellen von Zeichenvorgängen. Durch das Übertragen von Koordinaten der Berührungspunkte wird eine Situation erzeugt, wie wenn zwei Benutzer an derselben Instanz mit zwei verschiedenen Zeigegeräten arbeiten. Nachteil dieser Variante ist es, dass solange ein Benutzer an einer Zeichenfläche arbeitet, ein konstanter Fluss von Daten stattfindet. Auch die bereits angesprochene Möglichkeit, dass die Verarbeitung des Inputs in der

Algoria-Engine nicht immer dasselbe Resultat zur Folge haben kann, ist nicht optimal.

Name der Aktion	Verwendung
MouseLeftReleased	Solange die linke Maustaste gedrückt wird, werden alle Berührungspunkte der Maus mit der interaktiven Zeichenfläche gesammelt. Wird die Maustaste losgelassen, werden die Punkte übertragen.
MouseRightReleased	Solange die rechte Maustaste gedrückt wird, werden alle Berührungspunkte der Maus mit der interaktiven Zeichenfläche gesammelt. Wird die Maustaste losgelassen, werden die Punkte übertragen.

Tabelle 2.4.: Beschreibung von denkbaren Aktionen (Granularitätsstufe 2)

Mit den Aktionen aus Tabelle 2.4 wird die Flüssigkeit von Zeichenvorgängen verringert. Die gesammelten Punkte werden in einer Übertragung versendet. Durch dieses Sammeln wird kein konstanter Fluss an Informationen mehr produziert. Es kann aber analog zur Granularitätsstufe 1 nicht vollkommen sicher gesagt werden, welchen Output die *Algoria*-Engine liefert.

Name der Aktion	Verwendung
SegmentAdded	Das hinzugefügte Liniensegment wird übertragen.
SegmentRemoved	Die Koordinaten des zu entfernenden Liniensegments wird übertragen.

Tabelle 2.5.: Beschreibung von denkbaren Aktionen (Granularitätsstufe 3)

Werden die gezeichneten Linien in der Zeichenfläche bereits durch die Engine in *Algoria* analysiert, können Aktionen gemäss Tabelle 2.5 definiert werden. Die Häufigkeit von Übertragungen ist zu vergleichen mit derjenigen in der Granularitätsstufe 2. Durch ein Vorverarbeiten des Inputs ist jedoch die Wahrscheinlichkeit, dass unterschiedliche Resultate auf den Zeichenflächen erscheinen, reduziert.

Wird jeweils ein ganzer Analysedurchlauf auf den Zeichenflächen gemacht, werden Aktionen erst dann versendet, wenn die *Algoria*-Engine eine Veränderung an einer Datenstruktur gefunden hat. Die Übertragungshäufigkeit nimmt, im Vergleich zu den drei bisherigen Granularitätsstufen, weiter ab. Der Zeitpunkt, wann eine Änderung versendet werden kann, ist jedoch später. Dies, weil die Verarbeitung von Input auf der Zeichenfläche mehr Zeit benötigt. Viele Aktionen verfügen über einen Identifikator für die Datenstruktur. Dieser wird verwendet, um eine Datenstruktur auf einer Zeichenfläche zweifelsfrei identifizieren zu können (siehe 3.2.2 für nähere Informationen).

Name der Aktion	Verwendung
DatastructureAdded	Die hinzuzufügende Datenstruktur wird serialisiert und an die restlichen Teilnehmer versendet.
DatastructureRemoved	Es wird ein Löschbefehl mit dem Identifikator der zu entfernenden Datenstruktur versendet.
DatastructureMoved	Die neuen Koordinaten einer Datenstruktur werden versendet (zusammen mit dem dazugehörenden Identifikator).
DatastructuresMerged	Zwei zuvor unabhängige Datenstrukturen werden zusammengefügt. Dazu wird der Identifikator der betroffenen Datenstrukturen versendet.
DatastructuresSeparated	Eine Datenstruktur wird entzweit. Versendet wird sowohl der Identifikator, als auch die Stelle, wo getrennt werden soll.
ArrayFieldAdded	Der Identifikator des betroffenen Arrays sowohl Position und Anzahl neuer Felder werden versendet.
ArrayFieldRemoved	Der Identifikator des betroffenen Arrays sowohl Position und Anzahl zu löschender Felder werden versendet.

Tabelle 2.6.: Beschreibung von denkbaren Aktionen (Granularitätsstufe 4)

Bereits nach relativ wenigen Aktionen ist erkennbar, dass es in der Granularitätsstufe 4 Aktionen gibt, welche spezifisch auf eine bestimmte Datenstruktur passen. Hier muss entschieden werden, ob versucht wird möglichst generelle Aktionen zu definieren oder ob auch Datenstrukturtyp-spezifische Aktionen implementiert werden sollen.

2.3.3. Empfehlung bezüglich der Granularität

In Bezug auf die Sicherstellung der Konsistenz ist eine Lösung mit Granularitätsstufe 4 zu bevorzugen. Die Analyse der verschiedenen Stufen hat gezeigt, dass erst nach einer vollständigen Analyse einer Eingabe zwischen verschiedenen Operationen unterschieden werden kann. Von der ersten bis zur dritten Stufe bezüglich Granularität kann nicht klar gesagt werden, welchen Effekt dieser Input auf die Zeichenfläche hat. Erst wenn die Verarbeitung durch die *Algoria*-Engine abgeschlossen ist, können die resultierenden Aktionen klar getrennt werden. Zudem müssen für beide, in Kapitel 2.2 vorgeschlagenen Systeme klare Definitionen der Operationen vorhanden sein.

3. Implementationen

Mit einer klaren Vision bezüglich des zu realisierenden kollaborativen Systems, werden in den folgenden Unterkapiteln konkrete Implementationen besprochen. Die Realisierungen in Kapitel 3.1 und 3.2 behandeln ein Netzwerk-Framework. Dieses Framework soll als Grundlage für die Implementation eines Synchronisierungsalgorithmus dienen. Abschliessend wird besprochen, wie mehrere Aufgabenblätter gleichzeitig dargestellt werden können.

3.1. Senden und Empfangen von Arbeitsblättern

Die Thematik des Versenden und Empfangen von Arbeitsblättern wird im Folgenden analysiert. Für die Realisierung eines, auf der Analyse aufbauenden, Netzwerk Frameworks für *Algoria Worksheet*, werden zwei Möglichkeiten vorgestellt und verglichen. Die folgenden Seiten dokumentieren diese Vorgänge.

3.1.1. Analyse / Konzeption der Implementation

Vor dieser Projektarbeit ist von *Algoria Worksheet* nur über eine Möglichkeit ein Arbeitsblatt zu verteilen zur Verfügung gestellt worden. Die Dozierenden erstellten ein neues Arbeitsblatt mit beliebig vielen Aufgaben. Diese sind in eine Datei auf die Festplatte geschrieben und dann via Active Directory oder E-Mail an die Studierenden verteilt worden. In der vorliegenden Arbeit wird eine weitere Art der Verteilung eines Arbeitsblattes geplant und implementiert.

Es handelt sich hierbei um eine Verteilung von Arbeitsblättern über das Netzwerk. Vorerst wird nur der Fall betrachtet, indem alle Kommunikationspartner in demselben Netzwerk sind. Eine Erweiterung, welche das Internet miteinschliesst, muss zusätzlich berücksichtigt werden. Es wird hier ausdrücklich von Partnern gesprochen. Ein Arbeitsblatt verteilen kann sowohl ein Dozierender als auch ein Studierender. Letzterer wäre beispielsweise denkbar, wenn die Studierenden ihre ausgefüllten Arbeitsblätter zur Besprechung wieder an den Dozierenden senden sollen. Dieser könnte sie dann öffnen und mithilfe des Multi-Viewers (siehe 3.3 miteinander vergleichen.

Der allgemeine Aufbau sieht vor, dass eine Instanz von *Algoria Worksheet* das Arbeitsblatt öffnet und einen Server startet. Andere *Algoria Worksheet*-Instanzen können nun in einer Client-Rolle Verbindung zum Initiator aufnehmen. Bei diesem Kontakt wird das vom Initiator geöffnete Arbeitsblatt zum Client übertragen. Da es bereits einen Serialisierungsmechanismus für Arbeitsblätter gibt (wenn dieses in eine Datei geschrieben werden soll), kann der in diesem Mechanismus erzeugte Stream für das Versenden verwendet werden.

Für das Herstellen einer Verbindung mit C# und dem .Net-Framework werden im Folgenden zwei Möglichkeiten analysiert und soweit implementiert, dass eine begründete Entscheidung gefällt werden kann. Es handelt sich um das Implementieren der Kommunikation mithilfe von asynchronen Sockets, sowie um das Verwenden der Windows Communication Foundation.

Asynchrone Sockets

Bevor die Implementierung mithilfe von Sockets besprochen wird, soll zunächst geklärt werden, wo der Unterschied zwischen konventionellen, synchronen Sockets und der asynchronen Variante liegt. Als Beispiel wird hier eine Serverapplikation betrachtet, welche es einem Client erlaubt eine Verbindung zu ihm aufzubauen. In beiden Systemen instanziiert der Server ein Socket Objekt mit den gewünschten Parametern, wie der gewünschten IP-Adresse, dem Port, dem Protokoll-Type etc. Bei einem traditionellen, synchronen System beginnt das Horchen auf einkommende Verbindungen mit der Methode ‚accept'. Diese blockiert den Server so lange, bis eine Verbindung von einem Client her aufgebaut

wurde und verarbeitet dann diese. Damit in dieser Variante die Serverapplikation trotz des Wartens auf eine eingehende Verbindung weiter verwendet werden kann, müsste dieses in einem eigenen Thread erledigt werden. Dies führt zu Synchronisationsaufwand, um welchen sich der Entwickler kümmern muss.

Damit diese Auslagerung in einen eigenen Thread nicht notwendig ist, liefern asynchrone Sockets eine Alternative. Der Unterschied zwischen synchroner und asynchroner Variante liegt in der Weise, wie das Lauschen auf eingehende Verbindungen gestartet wird. Die Methode ‚beginAccept' ist asynchron und blockiert im Vergleich zur synchronen Version nicht. Hier wird der Methode ein Delegate auf eine Methode mitgegeben, welche beim Eingehen einer Verbindung ausgeführt werden soll. Im Falle von *Algoria Worksheet* verwaltet diese Delegate-Methode dann die Verbindung und sendet das verlangte Arbeitsblatt zum Client. Der Codeausschnitt 3.1 zeigt, wie eine Verbindung akzeptiert und ein Arbeitsblatt versendet wird.

```
1  private void ListenForClients()
2  {    ...
3      var localEndPoint = new IPEndPoint(ipAddress, ServerPort);
4      var listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
           ProtocolType.Tcp);
5      ...
6      while (serverRunning)
7      {
8          allDone.Reset();
9          listener.BeginAccept(this.AcceptCallback, listener);
10         allDone.WaitOne();
11     }
12 }
13 public void AcceptCallback(IAsyncResult ar)
14 {
15     allDone.Set();
16     var listener = (Socket)ar.AsyncState;
17     var handler = listener.EndAccept(ar);
18     ...
19     this.Send(handler, new RequestWorksheetAnswer(this.serializedWorksheet));
20     ...
21 }
22 private void Send(Socket handler, ICollaborationAction action)
23 {
24     byte[] byteData = CollaborationHelper.SerializeAction(action);
25     handler.BeginSend(byteData, 0, byteData.Length, 0, this.SendCallback,
           handler);
26 }
```

Codeausschnitt 3.1: Asynchroner Socket Server

Zu erwähnen ist jedoch, dass auch hier eine gewisse Synchronisierung nötig ist. Die Aufrufe ‚allDone.Reset();' und ‚allDone.WaitOne();' übernehmen diese Aufgabe. Wenn der Server auf eine einkommende Verbindung wartet, wird mit einem ‚WaitOne' gestoppt um nicht unendlich viele BeginAccept-Aufrufe zu erzeugen. Mit dem Aufruf von ‚allDone.Set();' wird dieses Warten dann abgebrochen und auf die nächste Verbindung gewartet.

Windows Communication Foundation

Nebst der eigenständigen Implementierung der Kommunikation unter der Verwendung von Sockets, wird die Windows Communication Foundation (WCF)[1] betrachtet. WCF bietet dem Entwickler die Möglichkeit, eine serviceorientierte Applikation zu entwickeln. Häufig handelt es sich hierbei um Webservices, welche dann auf den IIS (Microsoft Internet Information Services) anderen Benutzern via Webbrowsern zugänglich gemacht werden. Es kann aber auch eine Client-Server Applikation mithilfe von WCF entwickelt werden, welche danach unabhängig von einem IIS gestartet werden kann. Für das Anwendungsgebiet in *Algoria Worksheet* bietet es sich an, sowohl die Server- als auch die Clientapplikation direkt zu integrieren. Die Art und Weise, wie Client und Server entwickelt werden, unterscheidet sich bei WCF stark von derjenigen, die bei Sockets zum Zuge kamen. Für den Server wird zu Beginn ein Contract-Interface erstellt (siehe ICollaborationContract.cs). Basierend auf dem Contract-Interface können die Klassen für die Client-Seite generiert werden. Der Entwickler hat somit mit der direkten Kommunikation zwischen Client und Server nichts mehr zu tun.

Mit den aufgeführten Methoden kann der Client mit dem Server kommunizieren. So hat ein Client die Möglichkeit, eine Aktion vom Typ ICollaborationAction an den Server zu senden, sich am Server an- und wieder abzumelden. Die Serverapplikation wird mit der Methode ‚Start' in der Klasse CollaborationServer (CollaborationServer.cs, ab Zeile 103) gestartet. Die auf dem Server bekannte Implementation des Contract-Interfaces wird verwendet, um einen Service Host zu instanziieren. Dieser kann nun vom Client mit den generierten Klassen aufgerufen werden. Um eine Verbindung zum Service auf dem Server aufbauen zu können, wird in der CollaborationClient-Klasse (CollaborationClient.cs, ab Zeile 44) ein Proxy-Objekt des Servers instanziiert. Mit diesem Objekt kann der Client nun auf Methoden des Contracts auf dem Server zugreifen. Gleichzeitig hat der Server die Möglichkeit mit einer Callback-Methode auf den Client zuzugreifen. Wie dies genau in *Algoria Worksheet* verwendet wurde, zeigt das Kapitel 3.1.2. Für das Erhalten des Arbeitsblattes sendet nun der Client eine ‚RequestWorksheet' Aktion. Diese wird auf dem Server verarbeitet und mithilfe des Callbacks wird eine ‚RequestWorksheetAnswer'-Aktion zum Client gesendet. Diese Antwort enthält dann das serialisierte Arbeitsblatt.

Entscheidung

Die Implementation desselben Services mit WCF und einer asynchronen Socket-Lösung erlaubt es eine fundierte Entscheidung bezüglich der zu verwendenden Technologie zu fällen. Die weitere Implementation wird auf WCF basieren. Im Folgenden werden die ausschlaggebenden Gründe für diesen Entscheid zusammengefasst.

- Die Verwaltung der Verbindungen auf Socket-Ebene ist bereits sehr tief. So muss der Entwickler sich um sehr viele Aspekte, wie beispielsweise der Problembehandlung, selbst kümmern.
- WCF bietet durch die Verwendung von Callbacks auch die Möglichkeit einer bidirektionalen Verbindung.
- In Sachen Performanz ist eine reine Socket-Lösung besser. Jedoch gilt dies nur solange, wie auch das Error-Handling genügend schnell ist. In WCF ist dies bereits integriert und muss vom Entwickler nicht speziell implementiert werden.
- Die Entwicklung des Clients ist bei einem WCF-Ansatz sehr einfach. Die Möglichkeit, einfach eine Methode auf dem Server aufzurufen, gibt dem Entwickler eine grosse Flexibilität.
- WCF ermöglicht es mithilfe eines Discovery-Services im Netzwerk nach laufenden Instanzen eines bestimmten Services zu suchen.

[1] http://msdn.microsoft.com/en-us/library/ms731082.aspx[Micr12]

- Durch die Verwendung von Data-Contracts ist das Versenden von Daten zwischen Client und Server einfach zu implementieren. Die Data-Contracts werden im Kapitel 3.1.2 kurz behandelt.

3.1.2. Ausführungen zur WCF-Implementation

Basierend auf der Entscheidung zu WCF wird hier spezifischer auf die Implementierung der Kommunikation zwischen verschiedenen Instanzen von *Algoria Worksheet* eingegangen. In Abbildung 3.1 ist eine Skizze zu sehen, wie die einzelnen Komponenten des Kommunikations-Frameworks zusammenspielen. Die gelbe Box symbolisiert hierbei die Serverinstanz, während die grünen Boxen Clients darstellen. Auffallend hierbei ist, dass in der Serverinstanz auch ein Algoria Collaboration Client zu sehen ist. Wie schon erwähnt, ist auch die Verwendung von *Algoria Worksheet* in getrennten Netzwerken, beispielsweise via Internet, angedacht. Um dies gewährleisten zu können, ist es von Vorteil, dass der Server möglichst wenig Funktionalität bietet und nur zwischen den einzelnen Clients die Nachrichten verteilt. Hiermit lässt sich auch das sehr schmale Contract-Interface erklären. Die einzige Veränderung an der Skizze in Abbildung 3.1 für einen eigenständigen Server wäre das Auslagern des "Algoria Collaboration Service Host" auf eine allgemein zugängliche Maschine im Internet.

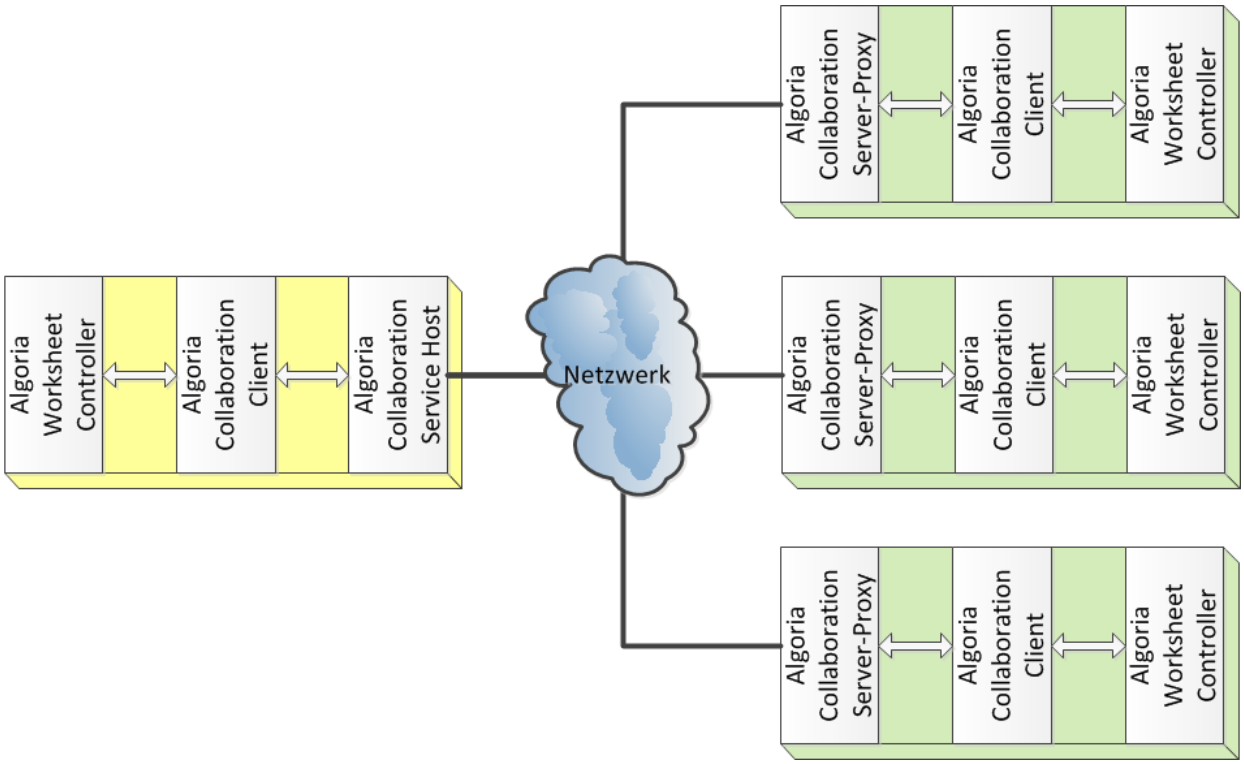

Abbildung 3.1.: Aufbau der Netzwerkkommunikationsumgebung in *Algoria Worksheet*

Callback Interface

Das Contract-Interface hat in der Service-Annotation das Feld für ein Callback-Interface gesetzt. Das Interface für diesen Callback ist in der Datei ICollaborationCallback.cs zu finden. Clientseitig wird dieses Interface implementiert und ermöglicht es so vom Server Nachrichten an den Client zu senden. Wie dieses Callback-Interface auf der Serverseite verwendet wird, ist in der CollaborationContract Klasse zu sehen (CollaborationContract.cs ab Zeile 31). Auf das verwendete Interface ‚ICollaborationAction' – welches auch für die SendAction-Methode im Contract-Interface verwendet wird - wird im Kapitel 3.2 näher eingegangen.

Client Verwaltung

Eine weitere Aufgabe, welche die Implementation des CollaborationContract-Interfaces übernehmen muss, ist das Verwalten aller Clients, genauer der Callback-Referenzen. Wie bereits erwähnt, meldet sich jeder Client über die Methode ‚Subscribe(bool isServer, Guid guid)' beim Server an und mit der Methode ‚Unsubscribe(Guid guid)' wieder ab. Die Anmelde-Methode gibt bereits Aufschluss darüber, ob es sich beim Client um den Initiator handelt oder nicht. Der noch wichtigere Parameter ist aber der ‚globally unique identifier' (Guid). Beim Starten eines Clients wird dieser erzeugt und der Anmelde-Methode mitgegeben. Während dieser Anmeldung hat der Service Host die Möglichkeit, eine Referenz zum Callback des Clients abzuspeichern. Diese Referenz kann mit der Guid des Clients verknüpft abgelegt werden. Verlangt nun ein Client die neuste Version des Arbeitsblattes beim Server, sendet er eine Anfrage zum Server. Dieser leitet die Anfrage zum Client des Initiators weiter, welcher bei der *Algoria Worksheet* Instanz eine serialisierte Version des momentanen Standes abholt. Dadurch, dass die Anfrage zusätzlich den Guid des Anfragestellers beinhaltet, weiss der Initiator-Client, wer die Anfrage gestellt hat und kopiert den Guid zusammen mit dem Arbeitsblatt in die Antwort. Diese geht wieder mit der SendAction-Methode zum Service Host und von dort aus via Callback zum Anfragesteller. Die allgemeine Verarbeitung von Antworten und Anfragen auf dem Service Provider wird im Codeausschnitt 3.2 auf Seite 23 gezeigt.

EndserviceDiscovery

Wie bereits in einem der Entscheidungspunkte beschrieben, verfügt die Windows Communication Foundation über die Möglichkeit, bestimmte Services innerhalb eines Netzwerkes zu suchen. Durch das Hinzufügen der Zeilen 121 - 129 in der Klasse CollaborationServer reagiert der Service Host auf die Anfragen von DiscoveryClients. Mithilfe eines DiscoveryClients kann nun ein Benutzer im gesamten Netzwerk nach einem Service Host suchen, welcher ein bestimmtes Kriterium erfüllt. In diesem Fall ist es, dass der Service Host einen Service mit dem ICollaborationContract-Interface anbietet. Zusätzlich wird dem Suchenden noch mitgeteilt, wie das Arbeitsblatt heisst, welches bei einem Initiator geöffnet ist. Dies liefert dem Benutzer des Clients zusätzliche Informationen über den Service Host, auf welchen er sich verbinden will.

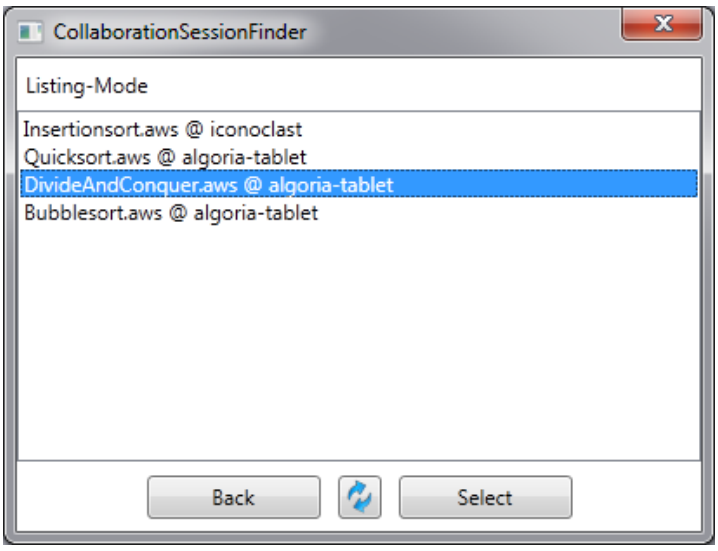

Abbildung 3.2.: Resultat des DiscoveryClients

Die Abbildung 3.2 zeigt das Resultat, welches eine Suche mithilfe eines DiscoveryClients liefert.

3.2. Live Übertragung

Damit ein Dozierender den Arbeitsfortschritt seiner Studierenden mitverfolgen kann, wird im Folgenden eine Live Übertragung im Umfeld von *Algoria Worksheet* besprochen.

3.2.1. Analyse / Konzeption der Implementation

Mit der Grundlage, welche durch das Versenden und Empfangen von ganzen Arbeitsblättern vorhanden ist, wird im Folgenden die Frage geklärt, wie eine Art Live-Übertragung für *Algoria Worksheet* realisierbar ist. Diese Live-Übertragung kann beispielsweise so eingesetzt werden, dass ein Dozierender sich bei seinen Studierenden einklinken und die Bearbeitung einer Aufgabe mitverfolgen kann. Der Fortschritt, welcher in einer Aufgabe durch deren Bearbeitung erreicht wird, bezieht sich hier auf die interaktiven Zeichenflächen. Aufgaben, welche eine gezeichnete Lösung fordern, bilden das Herzstück von *Algoria Worksheet*. Aus diesem Grund werden Multiple Choice Aufgaben hier nicht berücksichtigt. Damit ein Dozierender also sehen kann, wie sich die Zeichenfläche bei seinen Studierenden verändert, muss er immer wieder mit der aktuellsten Version des Arbeitsblattes versorgt werden. Es werden zwei verschiedene Wege präsentiert, wie diese Versorgung möglich ist.

Als erstes wäre es denkbar, in kurzen Abständen die Aufgabe, welche gerade bearbeitet wird, zu übertragen. Falls möglichst wenig an der bereits bestehenden Implementation des Services geändert werden soll, wird das gesamte Arbeitsblatt vom Server zum Client übertragen. Sollte diese Methodik für die Live-Übertragung verwendet werden, müsste nur eine wiederholende Arbeitsblatt-Anfrage vom Client zum Server gesendet werden und der Benutzer des Clients ist immer auf dem neusten Stand.

Eine andere Variante dieser Übertragung ist das Nachführen von Änderungen, welche auf der Serverseite gemacht werden. Für die Realisierung muss die Zeichenfläche des Servers stetig überwacht werden. Es müssen die Änderungen an der Zeichenfläche erkannt und an alle übertragen werden, welche sich beim Server angemeldet haben.

Die folgende Abbildung 3.3 stellt die Vor- und Nachteile der beschriebenen Lösungen einander gegenüber.

Variante	Vorteile	Nachteile
Ganzes Arbeitsblatt	+ Kaum Änderungen an der bestehenden Implementierung	– Kein „Live-Feeling“ – Stetige Netzwerkauslastung (wenn das Arbeitsblatt periodisch aktualisiert wird) – Kann für das kollaborative Bearbeiten einer Aufgabe nicht verwendet werden
Nur Änderungen	+ Kann als Grundlage für das kollaborative Bearbeiten einer Aufgabe verwendet werden + Unmittelbares Anzeigen von Änderungen + Netzwerkauslastung nur wenn etwas geändert wird	– Das Hinzufügen eines Mechanismus für das Versenden von Aktionen ist notwendig

Abbildung 3.3.: Vor- und Nachteile der Live-Übertragungslösungen

Nach Betrachtung der jeweiligen Stärken und Schwächen kann eine Entscheidung getroffen werden. Am sinnvollsten ist es, nur die jeweiligen Änderungen in einer Zeichenfläche zu versenden. Ausschlaggebend für diese Entscheidung ist die Möglichkeit, die Live-Übertragung als Grundlage für eine kollaborative

Bearbeitung einer Aufgabe zu verwenden. Diese Live-Übertragung kann auch als einfachster Fall der Synchronisation angesehen werden. Wie bereits in Kapitel 2.1 ist der Fall, in welchem nur ein Benutzer etwas editiert, ein konfliktfreier.

3.2.2. Ausführungen zur Implementation

Änderungen zu übermitteln, bedeutet in diesem Falle das Versenden der gemachten Aktionen auf einer interaktiven Zeichenfläche, sodass diese Änderung auf einer anderen Zeichenfläche ebenfalls gemacht werden kann. Um dieses Verhalten zu ermöglichen, werden konkrete Aktionen definiert. Zwei dieser Aktionen sind bereits im Kapitel 3.1 aufgelistet. Es handelt sich um das Anfordern eines Arbeitsblattes und um die Antwort auf eine solche Anfrage. Damit das Contract-Interface weiterhin möglichst einfach gehalten werden kann, bietet das ICollaborationAction-Interface einen gemeinsamen Nenner für alle Aktionen. Dieses Interface definiert ein Property-Feld, welches den Guid des Absenders der Aktion enthält. So benötigt es im Contract-Interface nur eine einzige Methode, um die Übertragung der Aktionen zum gewünschten Empfänger zu gewährleisten. Die Implementation ist im folgenden Codeausschnitt beschrieben.

```
1  FUNCTION SendAction(ICollaborationAction action)
2  IF action ist eine Arbeitsblatt Anfrage
3      Sende action via Server-Callback zum Initiator
4  ELSE IF action ist eine Arbeitsblatt Antwort
5      Sende action via Client-Callback zum Anfragesteller
6  ELSE
7      FOR EACH Callback ohne Callback des Absenders der Aktion
8          Sende action via Callback Interface
9      END
10 END IF
11 END FUNCTION
```

Codeausschnitt 3.2: Pseudo Code der SendAction Methode

Es ist klar ersichtlich, dass das Anfordern von Arbeitsblättern und die Antwort darauf ein Spezialfall darstellt. Alle anderen Fälle können allgemein behandelt werden, ohne die genaue Implementation der Aktion auf der Service Host-Seite zu kennen. Mit dieser Grundlage werden hier zwei weitere, häufig verwendete Aktionen präsentiert. Es handelt sich um das Hinzufügen und das Entfernen einer Datenstruktur. Die Klasse DatastructureAdded (siehe DatastructureAdded.cs) zeigt die Implementation der Aktion, welche eine oder mehrere Datenstrukturen hinzufügt. Der Empfänger einer solchen Aktion kann dem Objekt sowohl die neu hinzugefügte Datenstruktur (Feld NewItems in der DatastructureAdded Klasse) als auch den Index der dazugehörigen Aufgabe entnehmen.

Für das Löschen einer Datenstruktur kann hingegen dieser Weg nicht gegangen werden. Es könnte zwar die gelöschte Datenstruktur, analog zum Hinzufügen, übertragen werden. Der Versuch, diese aber dann zweifelsfrei auf der Empfängerseite zu identifizieren, ist hingegen, aufgrund der Implementierung in *Algoria*, schwierig. Eine Abhilfe für dieses Problem ist, dass bei der Erstellung einer Datenstruktur auf der interaktiven Zeichenfläche jeder Datenstruktur ein Guid zugewiesen wird. Für die DatastructureRemoved-Aktion bedeutet dies auch, dass nur ein Guid-Objekt übertragen werden muss. Die Datenstruktur mit der gleichen Guid zu identifizieren, ist zweifelsfrei möglich.

Auffallend bei allen Klassen, welche das ICollaborationAction-Interface implementierten, sind die Annotationen der Klassen und der Properties. WCF erlaubt es, primitive Datentypen ohne zusätzliche Vorkehrungen zu übertragen. Im Falle der Aktions-Klassen handelt es sich aber nicht um einen primitiven Datentyp. Abhilfe schaffen die Annotationen ‚DataContract' für Klassen und ‚DataMember' für Felder. Bei Feldern, welche übertragen werden sollen, muss sichergestellt werden, dass diese serialisier-

bar sind. Eine weitere Annotation, welche für das Funktionieren des Systems notwendig ist, ist die ‚ServiceKnownType' für das Contract- und Callback-Interface. Bei Methoden, welche über einen Service zur Verfügung gestellt werden, müssen die Parameter serialisierbar sein. Im Falle von *Algoria Worksheet* enthält die Signatur der ‚SendAction'- (im Contract-Interface) und der ‚OnCallback'-Methode (im Callback-Interface) das ICollaborationAction Interface als Typ für den Parameter. Damit WCF trotzdem mit dieser Situation zurecht kommen kann, muss deklariert werden, welche Implementationen des Interfaces bekannt sind. Die Annotationen sind ersichtlich in den Klassen ICollaborationCallback und ICollaborationContract.

3.3. Paralleles Darstellen verschiedener Arbeitsblätter

In den folgenden Unterkapiteln wird die Möglichkeit des parallelen Darstellens von mehreren Aufgabenblättern beschrieben. Dazu wird die Anforderung zuerst analysiert, ein Konzept für die Ausführung präsentiert und die konkrete Implementation in *Algoria Worksheet* besprochen.

3.3.1. Analyse / Konzeption der Implementation

Algoria Worksheet ist in den Anfängen als reine Einzelplatzapplikation entwickelt worden. Dieses Anwendungsgebiet wird in der vorliegenden Arbeit erweitert. So sollen Dozierende die Möglichkeit haben, das Programm auf dem Beamer oder auf sehr grossen Touch-Screens (Giga-Display) zu verwenden. Um den Zuwachs an Platz, welcher auf einem Giga-Display möglich ist, auszunutzen, wird *Algoria Worksheet* mit neuen Funktionen angereichert. Der Fokus der hier beschriebenen Anforderung liegt dabei darin, verschiedene Arbeitsblätter gleichzeitig darstellen zu können.

Für die Integration dieser Funktionalität sind zwei Möglichkeiten vorhanden. Zum einen wäre es möglich eine weitere Version von *Algoria Worksheet* zu entwickeln (eine ‚Multi-View-Version'). Ein anderer Weg ist es, die bestehende Admin-Version dahingehend zu erweitern, dass mehrere Arbeitsblätter gleichzeitig geöffnet werden können. Die Möglichkeit, mehrere Arbeitsblätter gleichzeitig darzustellen, ist ein Anwendungsgebiet, in welchem sich nur die Dozierenden bewegen. Deshalb wurde entschieden, die Admin-Version zu erweitern. Wie dies, ohne grosse Anpassungen vornehmen zu müssen, in der Benutzeroberfläche integriert werden kann, ist in der Abbildung 3.4 zu sehen.

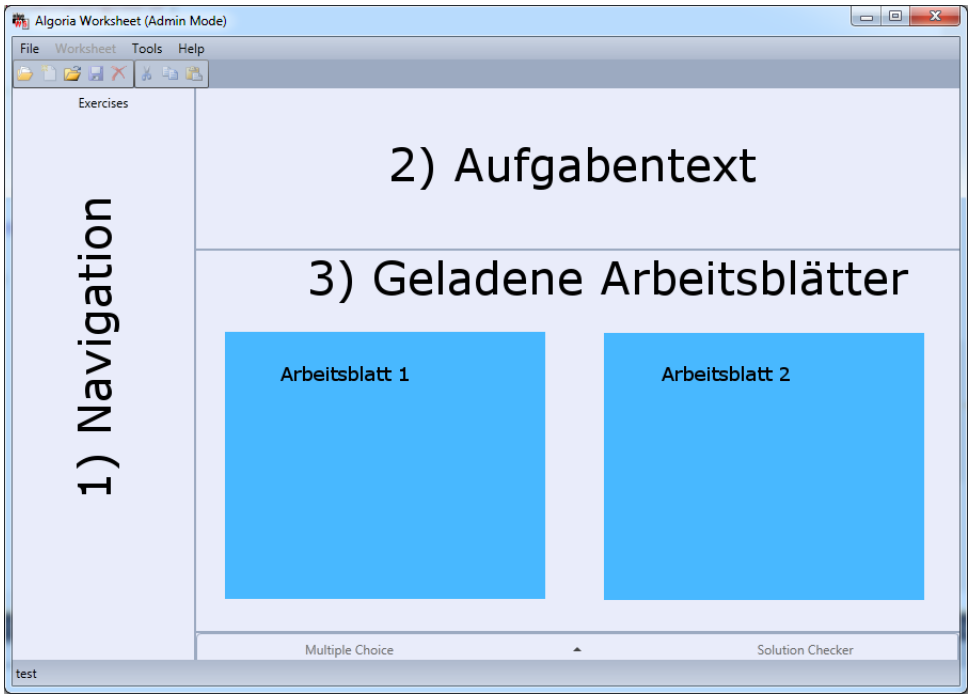

Abbildung 3.4.: Design der Multi-View Darstellung

Im Bereich, wo sich normalerweise nur eine interaktive Zeichenfläche befindet, sind mehrere Arbeitsblätter zu sehen. Ein solches Arbeitsblatt beinhaltet die dazugehörigen Zeichenflächen. Um in einem Arbeitsblatt navigieren zu können, und für diese Navigation nicht zusätzlicher Platz benötigt wird, passt sich die bereits bestehende Navigation dem jeweils aktiven Arbeitsblatt an. So soll diese wie gewohnt die Aufgaben des gerade aktiven Arbeitsblattes darstellen. Dasselbe gilt für den Aufgabentext. Dieser stellt den Titel und die Aufgabenstellung der aktiven Aufgabe des aktiven Arbeitsblattes dar.

Dozierende erhalten durch diese Darstellung die Möglichkeit zwei Lösungen miteinander zu vergleichen. Da es aber auch möglich sein soll Arbeitsblätter, welche von einer ganzen Klasse gelöst wurden, zu besprechen, führte dies dazu, bis zu 30 Arbeitsblätter parallel darstellen zu müssen. Damit der Ladeaufwand für eine so grosse Menge von Daten verteilt werden kann, bietet es sich an, für die jeweiligen Arbeitsblätter eine Vorschau zu generieren und die benötigten Zeichenflächen nur dann zu instanziieren, wenn der Dozierende diese auch benötigt. Auch das Darstellen der geöffneten Zeichenflächen in eigenen Fenstern bietet sich an, um auf einem grossen Bildschirm zwei Lösungen direkt nebeneinander zu platzieren.

Für die Darstellung der Vorschaubilder wird ein Konzept ähnlich dem des Windows Media Centers in Betracht gezogen. Für das Auflisten von Bildern in einem Ordner werden kleinere Vorschau-Bilder erzeugt, welche auf den Mauszeiger reagieren, sobald dieser sich über ein Bild bewegt. Konkret werden die Bilder mit einem goldenen Rahmen versehen. Ein ähnliches Verhalten könnte für *Algoria Worksheet* verwendet werden.

3.3.2. Ausführungen zur Implementation

MultiView Architektur

Für die Integration der neuen MultiView-Ansicht und der damit verbundenen Vorschau-Bilder bzw. Zeichnungsflächen wurde die folgende Klassenhierarchie aufgebaut.

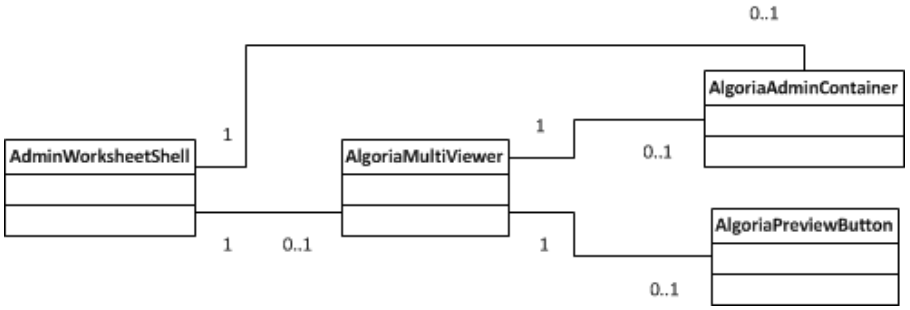

Abbildung 3.5.: Multi-View Klassendiagramm

Wie in Abbildung 3.5 zu sehen, verfügt die Klasse AdminWorksheetShell (diese Klasse stellt das Grundfenster der Applikation dar) über maximal eine Referenz zu einem AlgoriaMultiViewer (hier werden die verschiedenen Vorschaubilder oder Zeichenflächen verwaltet und dargestellt) und zu einem AlgoriaAdminContainer. Letzterer beinhaltet die notwendigen Zeichenflächen für die Musterlösung, die Studierendenlösung und die Ausgangslage einer Aufgabe. Der AlgoriaMultiviewer kann sowohl die bereits genannten AlgoriaAdminContainer beinhalten, als auch AlgoriaPreviewButtons. Diese Schaltflächen ermöglichen es dem Dozierenden ein gewünschtes Arbeitsblatt zu öffnen oder es ganz zu schliessen. Für eine einfache Verwaltung des zur Verfügung stehenden Platzes innerhalb der Darstellungsfläche wird ein UniformGrid verwendet. Dieses erlaubt es, beliebig viele Elemente anzuzeigen, wobei die Elemente immer gleich gross dargestellt werden. Das Resultat von vier parallel angezeigten Arbeitsblatt-Vorschaubildern ist in Abbildung 3.6 zu sehen. Die Instanz der Klasse AlgoriaMultiViewer beinhaltet nun vier AlgoriaPreviewButton-Objekte.

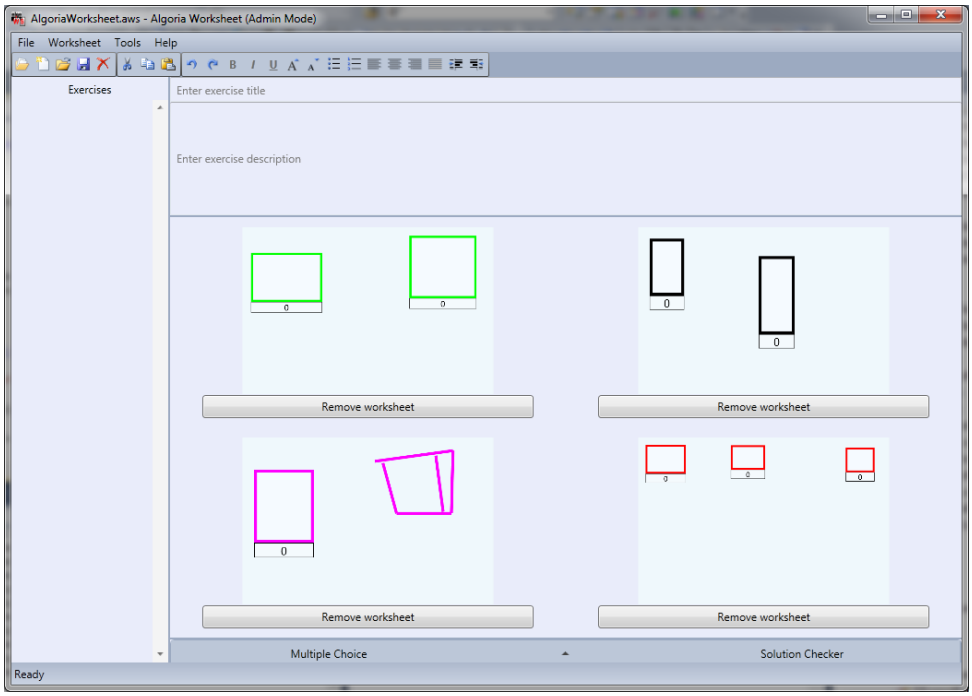

Abbildung 3.6.: Multi-View Ansicht mit geschlossenen Arbeitsblättern

Klickt ein Dozierender auf einen dieser AlgoriaPreviewButtons, wird für die entsprechende Aufgabe ein AlgoriaAdminContainer instanziiert und die Aufgaben des Arbeitsblattes geladen. Die Abbildung 3.7 zeigt die Benutzeroberfläche, wenn zwei der Arbeitsblätter aus Abbildung 3.6 durch Klicken geöffnet wurden.

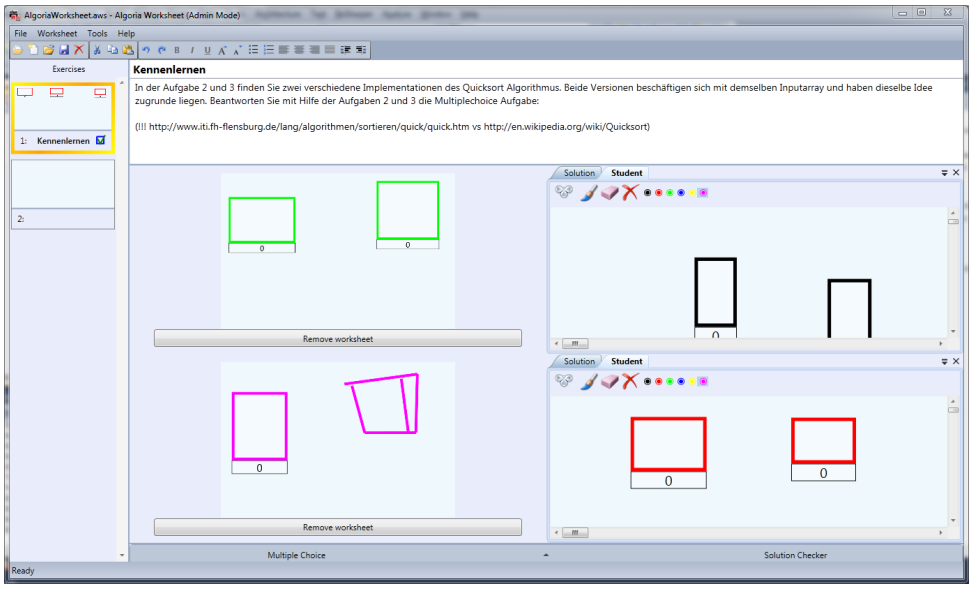

Abbildung 3.7.: Multi-View Ansicht mit offenen Arbeitsblättern

Die Gewährleistung der Kommunikation zwischen dem Controller der Applikation, dem MultiViewer, dem AlgoriaAdminContainer und den PreviewButtons ist solange einfach, wie der Fluss von Informationen in der Hierarchie tiefer geht. Beispielsweise ist das Aufrufen einer Funktion auf einem PreviewButton vom MultiViewer her kein Problem, da der MultiViewer eine Referenz auf alle PreviewButtons hat. Der umgekehrte Fall ist aber genauer zu betrachten.
Klickt ein Benutzer auf einen PreviewButton muss der MultiViewer darüber in Kenntnis gesetzt wer-

den, um den Button durch einen AlgoriaAdminContainer zu ersetzen und den Controller anzuweisen das dazugehörige Arbeitsblatt zu laden. In diesem Beispiel werden zwei Nachrichten von einer tieferen auf eine höhere Hierarchiestufe gesendet (vom Button zum MultiViewer und vom MultiViewer zum Controller). Um eine solche Kommunikation zu realisieren, wäre es möglich, dass jedes Objekt jeweils eine Referenz zur nächsthöheren Hierarchiestufe erhält. Dieses Vorgehen führt aber zu einer sehr engen Kopplung zwischen den einzelnen Klassen. Eine Alternative dazu sind Delegates. Mithilfe eines Delegates kann eine Klasse eine Referenz auf eine Funktion erzeugen, welche dann an ein untergeordnetes Objekt mitgegeben werden kann. Durch das Aufrufen dieser Referenz erhalten nun tiefere Hierarchiestufen die Möglichkeit, mit höheren zu kommunizieren, ohne eine enge Kopplung in Kauf nehmen zu müssen. Die Definition eines Delegates ist in der Klasse AlgoriaPreviewButton (AlgoriaPreviewButton.xaml.cs, ab Zeile 58) zu sehen. Weitere Delegates sind in der MultiViewer-Klasse (AlgoriaMultiViewer.xaml.cs, ab Zeile 75) zu finden.

Windows Media Center Style

Damit die Vorschau-Schaltflächen einen ähnlichen Effekt zeigen wie die Vorschaubilder im Windows Media Center wird im Folgenden ein WPF-Style für den AlgoriaPreviewButton vorgestellt. Dieser Style definiert die Reaktion auf ‚EventTrigger'. Es handelt sich um die Events ‚MouseEnter' und ‚MouseLeave'. Fährt also der Dozierende mit der Maus über einen PreviewButton, wird ein solcher Event ausgelöst. Damit wird eine Animation gestartet, welche durch ein Storyboard beschrieben wird. Die zwei relevanten Storyboards sind im Codeausschnitt 3.3 aufgelistet.

```
1  <Storyboard x:Key="sizeUp">
2    <Int32Animation Duration="00:00:0.0" To="1" Storyboard.
       TargetProperty="(Panel.ZIndex)" />
3    <DoubleAnimation Storyboard.TargetProperty="RenderTransform.ScaleX"
       To="1.2" Duration="0:0:0.5" />
4    <DoubleAnimation Storyboard.TargetProperty="RenderTransform.ScaleY"
       To="1.2" Duration="0:0:0.5" />
5  </Storyboard>
6  <Storyboard x:Key="sizeDown">
7    <DoubleAnimation Storyboard.TargetProperty="RenderTransform.ScaleX"
       To="1.0" Duration="0:0:0.5" />
8    <DoubleAnimation Storyboard.TargetProperty="RenderTransform.ScaleY"
       To="1.0" Duration="0:0:0.5" />
9    <Int32Animation BeginTime="0:0:0.2" Duration="00:00:0.0" To="0"
       Storyboard.TargetProperty="(Panel.ZIndex)" />
10 </Storyboard>
```

Codeausschnitt 3.3: Storyboards für size up und size down

Die verwendeten Animationen ändern den Wert eines Feldes kontinuierlich. Beispielsweise wird mithilfe des RenderTransform.ScaleX Feldes des AlgoriaPreviewButtons die Grösse des Vorschaubildes während 0.5 Sekunden auf 120% der ursprünglichen Grösse in X-Richtung skaliert. Ebenfalls der Z-Index wird mit derselben Methode gesetzt. Je höher dieser ist, desto weiter rückt das Element in den Vordergrund. Analog zum Vergrössern der Elemente werden sie beim Verlassen der Maus wieder auf die ursprüngliche Grösse herunterskaliert.

4. Fazit

Nachdem die Arbeiten in diesem Projekt beschrieben wurden, werden hier die Fragen aus Kapitel 1.2 noch einmal aufgegriffen und Antworten dazu formuliert.

> Welchen Anforderungen muss eine kollaborative Lernumgebung für das Klassenzimmer genügen?

Kollaboration als solches beschäftigt sich mit der Zusammenarbeit von mehreren Personen. Für eine Applikation im Umfeld der Bildung kann dies zwei verschiedene Szenarien nach sich ziehen. Zum einen ist eine Kollaboration zwischen Dozierenden und Studierenden möglich, zum anderen können auch Studierende unter sich kollaborativ in einer Lernumgebung arbeiten.

Der erste Fall zielt auf das Arbeiten während des Unterrichts im Plenum ab. Die Lernumgebungsapplikation ist auf Seiten des Dozierenden verfügbar. Studierenden muss es also ermöglicht werden, diese Applikation sehen zu können. Dazu können technische Hilfsmittel wie ein grosses Display oder ein Beamer verwendet werden.

Der Unterricht im Plenum kann auch für den zweiten Fall die Grundlage bilden, wahrscheinlicher sind aber Gruppenarbeiten. Hier beschäftigen sich mehrere Personen mit derselben Aufgabe, arbeiten an demselben Dokument. Eine kollaborative Lernumgebung muss also in der Lage sein, den gleichzeitigen Zugriff auf ein Dokument und dessen gleichzeitige Bearbeitung zu gewährleisten. Die Lernumgebungen sollten jeweils getrennt voneinander laufen können, nur das Dokument soll dasselbe sein. Eine solche Funktionalität bringt aber einige Anforderungen mit sich. Nebst der Frage, wie die Änderungen von einer Instanz der Lernumgebung zur nächsten kommen, muss geklärt werden, wie die Konsistenz des Dokumentes sichergestellt werden kann.

> Wie lässt sich *Algoria Worksheet* fürs Klassenzimmer anpassen bzw. umbauen und wie überprüfen Sie den Nutzen?

Die Kollaboration im Plenum kann in *Algoria Worksheet* durch den Einsatz eines grossen Displays ermöglicht werden. Dozierende können die Applikation auf diesem Display anzeigen und erhalten sofort ein grosses Angebot an Platz. Um dieses ausnutzen zu können, ist die Möglichkeit implementiert worden, mehrere Aufgabenblätter parallel öffnen zu können. Dadurch ist es möglich, zusammen mit den Studierenden, Lösungen einer Aufgabe zu vergleichen und zu besprechen. Noch mehr der Kollaboration dienlich ist eine Kombination dieser neuen Funktionalität mit dem, was im folgenden Abschnitt beschrieben wird.

Auf *Algoria Worksheet* bezogen bedeutet die Bearbeitung einer Aufgabe als Gruppe, dass mehrere Personen ein Arbeitsblatt zeitgleich bearbeiten können. Damit diese nicht am selben Computer sitzen müssen, wurde ein Netzwerk-Framework entwickelt, welches es erlaubt, einzelne *Algoria Worksheet* Instanzen miteinander kommunizieren zu lassen. Die Sicherstellung der Konsistenz des gemeinsam editierten Dokumentes soll mithilfe von bekannten Synchronisierungsalgorithmen geschehen.

Der Nutzen einer solchen Änderung kann direkt im Klassenverband getestet werden. So wäre es denkbar, dass ein Teil der Studierenden eine Aufgabe kollaborativ löst und ein anderer Teil jeweils die Aufgabe als Einzelaufgabe angeht. Leider konnte dies während der Projektarbeit nicht durchgeführt werden.

5. Reflexion

5.1. Projektmanagement

Die Kommunikation zwischen Studierendem und Advisor hat, wie schon in der vorausgegangenen Projektarbeit, gut funktioniert. Wöchentliche Sitzungen haben zu einem konstruktiven Dialog geführt. Auch durch diese Sitzungen konnte das Ziel nie aus den Augen verloren werden, da automatisch jede Woche ein Blick auf den Terminplan geworfen wurde, um zu überprüfen, ob das Projekt sich noch auf einem guten Weg befindet.
Im Gegensatz zum vorhergehenden Projekt wurde die Dokumentationsphase in einem grösseren Teil vor dem Abschluss des Projektes durchgeführt. Durch ein geführtes Sitzungsprotokoll der jeweiligen Woche waren genügend Notizen vorhanden, um die Dokumentation zu schreiben.
Im Allgemeinen wurde aber der Zeitaufwand für die theoretische Arbeit falsch eingeplant. Es sollte sich um einen konzentrierten Block zu Beginn der Arbeit handeln. Dieser Block sollte aber nicht die gesamte Zeit für theoretische Arbeiten einnehmen. Die restliche Zeit sollte sich, in jeweils kleineren Blöcken, über die gesamte Projektlaufzeit ziehen. Der Zeitaufwand für die theoretische Arbeit bleibt insgesamt derselbe, es handelt sich aber um einen kontinuierlichen Prozess. Dies erlaubt es auch, Ideen über die Zeit reifen zu lassen. Derselbe Ansatz sollte während der Master Thesis auch für die Dokumentation verwendet werden.

5.2. Anforderungen

Die folgenden Absätze sollen die Ergebnisse der Arbeiten zu den einzelnen Anforderungen analysieren und reflektieren. Arbeiten, welche nicht direkt im Zusammenhang mit spezifischen Anforderungen erledigt wurden, finden auch hier eine Erwähnung. Auf das genaue Dokumentieren dieser Arbeiten wurde verzichtet, da es sich meist um Arbeiten handelt, welche für die Beantwortung der Fragen in diesem Projekt an sich nicht relevant sind.

Gleichzeitiges Darstellen verschiedener Arbeitsblätter

Mit der Implementierung, welche zum jetzigen Zeitpunkt vorliegt, ist es möglich, mehrere Arbeitsblätter darzustellen. Auch das Auslagern der geöffneten Arbeitsblätter in einzelnen Fenstern sollte keine grosse Herausforderung mehr darstellen, da dies in einem Testprojekt bereits mit leeren Fenstern realisiert wurde. Für die Erstellung der Vorschaubilder könnte es eventuell noch effizientere Lösungen geben. Im Moment wird die erste Aufgabe des Arbeitsblattes in einem versteckten Fenster geöffnet und mithilfe eines Rendering-Vorgangs ein Vorschaubild erzeugt. Auch das Öffnen und Schliessen hat noch Potential hinsichtlich der Performanz, so ist vor allem das Öffnen eines geladenen Arbeitsblattes eine zeitintensive Angelegenheit.

Senden und Empfangen von Arbeitsblättern

Durch den direkten Vergleich von zwei Ansätzen für das Implementieren einer verteilen Applikation konnte viel Neues gelernt werden. Bisher war mir die Windows Communication Foundaiton nur für Webservices ein wirklicher Begriff. Deshalb lag der Fokus zu Beginn auf der Implementation mit Sockets. Diese Implementierung überlässt jedoch sehr viel dem Entwickler selbst, so wurde auch WCF wieder zum Thema. Im Nachhinein ist es für mich überraschend, wie gut sich WCF für diese Anforderung einsetzen liess. Hinsichtlich des gesamten Netzwerk-Frameworks bleiben aber noch einige Fragen offen. Beispielsweise ist nicht klar, was passiert, wenn der Server abstürzt. In einem solchen Fall könnte ein Notfallplan implementiert werden, in welchem einer der Clients die Serverrolle übernimmt.

Live Übertragung

Die Live Übertragung der Bearbeitung einer Aufgabe ist in dem Sinne keine wirkliche passive Übertragung. Es wird dasselbe System verwendet, welches später bei der vollständigen kollaborativen Bearbeitung einer Aufgabe zum Einsatz kommt. Möglicherweise würde es Sinn machen, eine Read-Only Version eines Kollaborations-Clients zu entwickeln. Dieser Client erlaubt nur das Mithorchen, welche Änderungen andere Benutzer auf ihren Zeichenflächen vollführen. Auch eine Verwendung der Live Übertragung in Verbindung mit dem Darstellen von verschiedenen Arbeitsblättern wäre eine denkbare Erweiterung.

Gleichzeitige Bearbeitung

Es wurden viele gute Ideen gefunden, wie ein kollaboratives System funktionieren könnte. Viele dieser Ideen fokussieren sich auf eine Anwendung im Textverarbeitungsbereich. Es ist versucht worden, dieses Vorgehen auf *Algoria Worksheet* zu übertragen. Dies gestaltete sich schwieriger als erwartet. Schon nur die Unklarheit darüber, was genau von Instanz zu Instanz gesendet wird, konnte während dieser Arbeit nicht restlos geklärt werden. Es wurde jedoch ein Framework errichtet, auf welchem diese Frage nun geklärt werden kann. Auch die Implementierung einer Synchronisierung wurde in dieser Projektarbeit nicht angegangen.

Sonstige Ergebnisse

Nebst den im Voraus bekannten Anforderungen wurden während der Projektarbeit immer wieder kleinere und grössere Arbeiten notwendig. Herausstechend ist das Verändern der interaktiven Zeichenfläche, welche durch *Algoria* zur Verfügung gestellt wird. Diese Fläche wurde während der Bachelor Thesis kopiert und parallel zur im Programm *Algoria* verwendeten Version entwickelt. Dies wurde behoben. Es gibt nun nur noch eine interaktive Zeichenfläche in der gesamten Solution. Ein weiterer erfolgreicher Punkt war das Ersetzen des Worksheet-Shell-Interfaces durch eine abstrakte Basisklasse. Dieses Interface definierte die Grundlage der Benutzeroberfläche in der Studierenden- und Dozierendenversion. Da in der Implementierung dieses Interfaces viel duplizierter Code gefunden wurde, hat das Ersetzen durch eine abstrakte Basisklasse viel zur Wartbarkeit beigetragen. Eine noch nicht vollendete Änderung ist das Entfernen von Prism aus der interaktiven Zeichenfläche. Nach mehrmaligem Probieren wurde Prism vorerst in *Algoria* gelassen.

5.3. Ausblick

Die Vorschläge bezüglich Synchronisationsvarianten für die kollaborative Bearbeitung von Aufgaben in *Algoria Worksheet* wurden in dieser Arbeit gemacht. Die nächste Arbeit verfügt somit über eine Grundlage um eine solche Synchronisierung zu testen. Weiter ist eine ganze Reihe von Tests vorstellbar. Dies umfasst das Testen der Synchronisierung, möglicherweise auch das Gegenüberstellen von mehreren Implementationen. Auch der MutliViewer kann in Bezug auf die Leistungsfähigkeit getestet und optimiert werden. Zum einen bezieht sich dies auf die maximale Anzahl Arbeitsblätter, welche parallel dargestellt werden und zum anderen auf die Ladezeit für diese Arbeitsblätter. Die Projektarbeit 7 [Frey12] lässt auch noch Fragen unbeantwortet. Die wichtigste hierbei ist, wie kann die automatische Lösungsprüfung getestet werden.

6. Ehrlichkeitserklärung

Hiermit bestätigt der Autor, diese Arbeit ohne fremde Hilfe und unter Einhaltung der gebotenen Regeln erstellt zu haben.

Reto Frey

Ort, Datum Unterschrift

Literaturverzeichnis

[Chen01] Chen, D. and Sun, C. *Optional Instant Locking in Distributed Collaborative Graphics Editing Systems*. In Proceedings of *International Conference on Parallel and Distributed Systems* (ICPADS 2001), S. 109-116, 2001.

[Elli89] Ellis, C.A. und Gibbs, S.J. *Concurrency Control in Groupware Systems*. In Proceedings of *ACM SIGMOD Conference on Management of Data* (SIGMOD '89), S. 399-407, 1989.

[Fras09] Fraser, N. *Differential Synchronization*. In Proceedings of *ACM Symposium on Document Engineering* (DocEng'09), S. 13-20, 2009.

[Fras12] Fraser, N. Differential Synchronization [online] *http://neil.fraser.name/writing/sync/* Zugegriffen am 10.09.2012.

[Frey11] Frey, R. und Zogg K. *IP6 Algoria Worksheet*, Projektbericht zur Bachelor Thesis, 2011.

[Frey12] Frey R. *P7: Algoria - Personliche Lernumgebung*, Projektbericht zum Informatik Projekt 7, 2012.

[Goog12a] Google. Google Docs [online] *http://docs.google.com* Zugegriffen am 10.09.2012.

[Goog12b] Google. Google Wave Operational Transformation Whitepaper [online] *http://wave-protocol.googlecode.com/hg/whitepapers/operational-transform/operational-transform.html* Zugegriffen am 10.09.2012.

[Goog12c] Google. Google Wave Federation Protocol White Papers [online] *http://www.waveprotocol.org/whitepapers* Zugegriffen am 20.09.2012.

[Igna06] Ignat, C.-L. and Norrie, M.C. *Drwa-Together: Graphical Editor for Collaborative Drawing*. In Proceedings of *International Conference on Computer Supported Cooperative Work* (CSCW'06), S. 269-278, 2006.

[Jens12] Jensen, P. [online] *http://www.me.utexas.edu/~jensen/models/network/net9.html* Zugegriffen am 10.09.2012.

[Micr12] Microsoft. MSDN Artikel zur Windows Communication Foundation [online] *http://msdn.microsoft.com/en-us/library/ms731082.aspx* Zugegriffen am 10.09.2012.

[Stam12] Stamm, C. *P8 im Frühlingssemester 2012*, Aufgabenstellung zu dieser Projektarbeit, 2012.

A. Aufgabenstellung P8

Fachhochschule Nordwestschweiz
Hochschule für Technik

MASTER OF SCIENCE IN ENGINEERING

Vertiefungsausbildung

Modulbeschreibung

P8 im Frühlingssemester 2012

Institut:	**IMVS**	
Projekttitel:	**Algoria: Kollaboration im Schulzimmer**	
Thema: (Kurzbeschreibung)	Interaktive Lernumgebungen erlauben die gezielte und selbständige Auseinandersetzung mit einem vorgegebenen Lernstoff. Sie leiten den Lernenden durch den Lernprozess und helfen ihm, sein Wissen zu überprüfen und zu festigen. Auf Basis der Magic-Ink-Software Algoria soll eine interaktive Lernumgebung für den Algorithmen und Datenstrukturen Unterricht in der Informatik entwickelt und in der Praxis erprobt werden. Die Interaktion soll sich an den gleichen Paradigmen wie bei Algoria orientieren (möglichst natürliche Interaktion ausgerichtet auf [Multi-]Touchscreens).	
Projektteam:	Auftraggeber: (Adresse)	IMVS, Christoph Stamm
	Student/-in	Reto Frey
	Betreuer (extern):	
	Advisor (intern):	Christoph Stamm
Fragestellungen: (erwartete Arbeiten)	Welchen Anforderungen muss eine kollaborative Lernumgebung für das Klassenzimmer genügen? Welche Funktionalitäten sind notwendig, sinnvoll, wünschenswert, umsetzbar? Wie lässt sich Algoria Worksheet fürs Klassenzimmer anpassen bzw. umbauen und wie überprüfen Sie den Nutzen? Realisieren Sie eine klassenzimmertaugliche, kollaborative Lernumgebung auf der Basis von Algoria Worksheet in Absprache mit Ihrem Betreuer. Planen Sie die Erstellung eines wissenschaftlichen Papers.	
Ergänzenden Veranstaltungen:	Informatik-Seminar	
Starttermin:	29.02.2012	
Abgabetermin:	31.08.2012	

Ort, Datum: _____ _____

Unterschriften: _____ _____

Student/Studentin Advisor