In: Bekavac, Bernard; Herget, Josef; Rittberger, Marc (Hg.): Informationen zwischen Kultur und Marktwirtschaft. Proceedings des 9. InternationalenSymposiums fur Informationswissenschaft (ISI 2004), Chur, 6.-8.Oktober2004. Konstanz: UVK Verlagsgesellschaft mbH, 2004. S. 1 – 22

Analyse und Unterstützung virtueller Lernteams bei der objektorientierten Softwareentwicklung

Ralph Kölle[1], *Glenn Langemeier*[2]

[1] Angewandte Informationswissenschaft
Universität Hildesheim
ralph@vitaminl.de

[2] Technik, Physik und Wirtschaft
Universität Hildesheim
glenn@vitaminl.de

Zusammenfassung

Die konstruktivistische Lerntheorie geht davon aus, dass Wissen vom Lernenden auf Basis seiner Erfahrungen, seines Vorwissens und des sozialen Kontextes aktiv und individuell konstruiert wird. Folglich gibt es sehr viele Möglichkeiten der Wissenskonstruktion, ein Lernprozess ist kaum vorhersagbar. Ziel ist daher „die Entwicklung von Lernumwelten, in denen kognitive Lernprozesse in handelnder Auseinandersetzung mit der Umwelt stattfinden können“ (Schulmeister 1997, 78).

Das Projekt VitaminL (*Vi*rtuelle *T*eams: *A*nalyse und *M*odellierung *i*n *n*etzbasierten *L*ernumgebungen, Kölle & Langemeier 2004) greift diese Idee auf und überträgt sie auf virtuelle Lernteams. In solchen Lernteams findet eine spezielle Art von Wissensmanagement als kooperativer Prozess statt. Individuelles Wissen muss kommunikativ in das Team übertragen werden, indem es reflektiert und verarbeitet wird.

In der universitären Praxis finden solche Prozesse innerhalb von Gruppenübungen, insbesondere in Tutorien zur objektorientierten Programmierung, statt. Dort werden Aufgaben gestellt, die im Team zu lösen sind. Treten Probleme auf, so werden anwesende Tutoren um Hilfe gebeten. Für die Übertragung auf virtuelle Teams wird an der Universität Hildesheim im Rahmen des Projekts VitaminL eine CSCL-Umgebung entwickelt, die solche Teams bei der kurzzeitigen, synchronen Bearbeitung von Programmieraufgaben aus dem Bereich der objektorientierten Software-Entwicklung unterstützt. Die Lehre der objektorientierten Software-Entwicklung an der Universität Hildesheim ist durch hohe Interdisziplinarität geprägt, Lerngruppen bestehen aus Studierenden verschiedenster Studiengänge, die bzgl. Programmierkenntnissen als Anfänger bezeichnet werden müssen. So findet auch die Weiterentwicklung des VitaminL-Projekts

in Kooperation des Instituts für Technik (Informatik-Ausbildung) und des Instituts für Angewandte Informationswissenschaft statt.

Die Übertragung der Gruppenübungen (Tutorien) ins E-Learning in Form von virtuellen Tutorien mittels der CSCL-Umgebung hilft zum einen, die personalintensiven Präsenztutorien zu entlasten, zum anderen ermöglicht es überhaupt erst die Durchführung dieser kooperativen Lernform unabhängig von Raum und Zeit. Ähnliche Ansätze verfolgen die Projekte ENFORUM (Kuhlen 2004a) des Hochschulverbandes für Informationswissenschaft (HI), das ebenfalls interdisziplinär ausgerichtet ist und ein virtuelles kollaboratives Wörterbuch mit enzyklopädischen Eigenschaften zur Verfügung stellt und K^3-forum (Kuhlen 2004b), das in Ergänzung zum ENFORUM ein Systems zur kooperativen Produktion und Aneignung von konzeptorientiertem Wissen entwickelt. Auch das Projekt SELIM (Software Ergonomie und Lernen im Multimedialen Kontext, Schudnagis 2002) virtualisiert Tutorien für die informationswissenschaftliche Ausbildung (an der Universität Hildesheim), unterstützt dabei allerdings die individuelle Lernform.

Bei VitaminL wird auf der Basis eines speziell auf den o.g. Anwendungsbereich adaptierten Rollenmodells eine Software-Komponente entwickelt, die ein virtuelles Team während der Zusammenarbeit beobachtet und Informationen über dessen Zusammensetzung liefert. Die Analyse erfolgt mittels eines stochastischen Verfahrens und liefert Rollenprofile der Teammitglieder als Ergebnis. Diese Profile werden von einer zweiten Komponente verarbeitet und auf Defizite untersucht. Mittels *Case Based Reasoning* (*CBR*) werden mögliche Problemsituationen während der Aufgabenbearbeitung erkannt und Lösungsansätze in das Team eingebracht, die zur Zielerreichung beitragen sollen, ohne das Team von den für den Lernerfolg wichtigen Teilaufgaben zu entlasten.

1 Einleitung

Komplexe Aufgaben und Probleme werden heute üblicherweise im Team bearbeitet und gelöst, aber auch kleinere Aufgaben lassen sich im Team oft leichter lösen. Ebenso gilt die Übertragung auf Lernprozesse: geht die Lernsituation über den reinen Wissenserwerb hinaus, so sind „*kooperative Lernformen dem individuellen Lernen oft überlegen*“ (Schulmeister 2001, S.196).

Findet eine räumliche und ggf. auch zeitliche Entkopplung der Teammitglieder statt, so spricht man von virtuellen Teams (vgl. Konradt & Hertel, 2002, S.18). Moderne Informations- und Kommunikations-

technologien stellen solchen Teams verschiedene Werkzeuge für synchrone und asynchrone Kommunikation und Kollaboration zur Verfügung. Man unterscheidet dabei CSCW-Systeme (*Computer Supported Collaborative Work)* für kooperatives Arbeiten und CSCL-Systeme (*Computer Supported Collaborative Learning)* für kooperatives Lernen. Forschungsarbeiten beschäftigen sich in diesem Zusammenhang meistens mit Teams, deren Zusammenarbeit eher langfristig orientiert ist.

Im Projekt VitaminL hingegen wird eine CSCL-Umgebung entwickelt, die Lernteams bei der synchronen und kurzzeitigen Zusammenarbeit unterstützt. Unter Verwendung synchroner Kommunikationswerkzeuge werden dabei Programmieraufgaben der objektorientierten Software-Entwicklung bearbeitet. Neben der Kurzzeitigkeit liegt ein weiterer Schwerpunkt auf der Zusammensetzung der jeweiligen Gruppen. Ausgehend von einem rollenbasierten Arbeitsmodell wird ein virtuelles Team während seiner Zusammenarbeit beobachtet und analysiert. Die Ergebnisse werden im Anschluss an Software-Komponenten weitergereicht, die versuchen, Problemsituationen zu erkennen und das Team zielgerichtet zu unterstützen.

2 Ausgangssituation

2.1 Lernen im Team

Lehrveranstaltungen zur Programmierung setzen sich in der Regel aus Theorie- und Praxisteilen zusammen. Der Theorieteil findet in der traditionellen Lehre häufig als Vorlesung statt, im E-Learning gibt es ganz verschiedene Ansätze, von einfachen Hypertext-Tutorials bis zu hin zu komplexen adaptiven Hyperbook-Systemen (Henze et al. 2001).

Die Vertiefung der Theorie im Praxisteil findet meist in von Tutoren unterstützten Gruppenübungen, sogenannten Tutorien, statt. Die Lerngruppe bekommt eine Aufgabe aus dem aktuellen Lernkontext und hat diese innerhalb einer bestimmten Zeit zu lösen. Die Vorgehensweise der Gruppe entspricht dabei prinzipiell einem der aus der Softwareentwicklung bekannten Vorgehensmodelle, z.B. dem Wasserfallmodell (Rechenberg & Pomberger 1999). Die Aufgabe ist zu analysieren, Lösungsansätze sind zu erkennen, es folgt die Umsetzung und die termingerechte Abgabe der Lösung. Die Bearbeitungszeit der Aufgaben beträgt in der Regel 30-90 Minuten. Bei

Problemen werden Tutoren oder anwesende Dozenten befragt, die dann adäquate Hinweise geben, ohne aber eine fertige Lösung zu präsentieren.

Das Prinzip dieser gemeinsamen Lösungsfindung im Team ist sehr praxisorientiert und entspricht dem konstruktivistischen Lernansatz mit all seinen Vorteilen. Gerade beim Lernen einer ersten Programmiersprache ist die Kombination aus Experimentieren, Lernen aus Fehlern und Selbstreflexion sehr wichtig. Neben den fachlichen Lerninhalten wie Syntax und Semantik müssen sich die Teilnehmer in ihren Gruppen organisieren, Teilaufgaben verteilen und über mögliche Lösungsansätze diskutieren. Verschiedene Niveaus von Vorwissen, bedingt durch Teilnehmer verschiedenster Studienbereiche und verschiedenster Zielsetzungen, aber auch verschiedenartige Kompetenzen sozialer Art erzwingen Diskussionen und Wissenstransfer innerhalb der Gruppe, was wiederum die Notwendigkeit der Reflexion des jeweiligem individuellen Wissens impliziert.

2.2 Problemsituationen

Bei den Teilnehmern der Lehrveranstaltungen zur Programmierung handelt es sich häufig um Anfänger, oft lernen sie ihre erste Programmiersprache. Während der Bearbeitung der Übungsaufgaben stoßen die Gruppen naturgemäß immer wieder auf Probleme verschiedener Art. Das können Verständnisprobleme bei der Aufgabenstellung sein oder aber Syntax- und Semantikprobleme. Ferner können Probleme mit dem objektorientierten Ansatz oder Unstimmigkeiten zwischen den Teammitgliedern auftreten.

Neben dem grundsätzlich didaktische Kernproblem, wie die theoretischen Konstrukte aus der Vorlesung in die Programmierübung umzusetzen sind, liegt einer der Gründe für das Auftreten dieser Problemsituationen oftmals in der personellen Zusammensetzung des Teams: für die Zielerreichung wichtige Funktionen sind unterrepräsentiert oder gar nicht vorhanden. Daher wurden im Rahmen von VitaminL verschiedene Rollenmodelle auf die Eignung der Anwendung auf Lernteams der objektorientierten Software-Entwicklung untersucht (vgl. Kap. 4.3).

2.3 Rollen und Funktionen

Zur effizienten Ausnutzung der Potenziale eines Teams ist eine sinnvolle Aufgabenverteilung unter Berücksichtigung der jeweiligen Fähigkeiten und Neigungen der einzelnen Teammitglieder wichtig. Den Mitgliedern werden im Zuge einer solchen Funktionsdifferenzierung Rollen zugeteilt: *„Eine Rolle*

ist ein sozial definiertes Verhaltensmuster, das von einer Person, die eine bestimmte Funktion in einer Gruppe hat, erwartet wird“ (Zimbardo & Gerrig, 2000, S. 723).

Die Rollenstruktur einer Gruppe wird mittels sog. Rollenmodelle beschrieben. Jede Rolle eines solchen Modells beinhaltet die mit ihr verbundenen Verantwortlichkeiten und deren charakteristische Eigenschaften. Das Lehrer-Lerner-Modell als einfachstes Modell besteht aus dem Lehrer als Experten eines bestimmten Fachgebiets und dem Lerner als demjenigen, der Wissen aus eben diesem Gebiet erwerben möchte (Pilkington et al., 1999; Daradoumis, 1999 ; Soller & Busetta, 2003). Das Projekt *AlgebraJam* von Singley et al. (2000) verfeinert das Lehrer-Lerner-Modell nach dem Ansatz von Vygotsky (*the zone of proximal development*; Vygotsky, 1978), wonach sich der Lernende im Lernprozess über mehrere Stufen vom zunächst passiven Beobachter bis hin zum Trainer entwickelt.

Die Frage, welche Faktoren den Erfolg bzw. den Misserfolg von Teams bestimmen, stand bei den Untersuchungen von Belbin (2003) im Vordergrund. Dazu wurden Teams und ihre Mitglieder bei Planspielen beobachtet, sämtliche Aktionen codiert und aufgezeichnet und letztlich mittels eines *IPA*-basierten Systems (*Interaction Process Analysis*; Bales, 1950) ausgewertet. Als Resultat konnten neun unterschiedliche Teamrollen vom Macher über den Vorsitzenden bis hin zum Spezialisten identifiziert werden.

Mehr an den Persönlichkeiten der Teammitglieder als an deren Aufgaben orientiert sich das Rollenmodell von Eunson (1990), der in seinem Rollenmodell über 20 verschiedene Rollen beschreibt und diese in Aufgabenrollen, sozio-emotionale Rollen und zerstörerische Rollen unterteilt. Neben aufgabenorientierten Rollen wie dem Initiator und dem Informationsgeber sind in diesem Modell auch der Mutmacher und der Friedensstifter, aber auch Schwätzer, Störer und Manipulierer zu finden.

Ein Rollenmodell im Sinne einer Aufgabenverteilung wurde mit dem *Team Management System* (*TMS*) von Margerison & McCann (Margerison, 1990) entwickelt. Dort wird der Begriff *Rolle* mit *Arbeitsfunktion* gleichgesetzt. Basierend auf der These, dass erfolgreiche Teams neun zentrale Funktionen wahrnehmen, entstand das Rad der Arbeitsfunktionen (*Types of Work Wheel*). Die Schwerpunkte können je nach Anforderungsprofil variieren, aber kein Bereich darf durch das Team vernachlässigt werden, sonst *„läuft das Rad nicht rund“*.

2.4 Persönlichkeitstypen - der Ansatz von Spencer und Pruss

Der Rollenbegriff umfasst also unterschiedlichste Interpretationen wie Arbeitsfunktion, Charaktereigenschaft und Verhaltensmuster. Zu dieser Erkenntnis gelangen auch Spencer & Pruss (1995) bei ihren Untersuchungen und Beobachtungen. Sie trennen daher Funktionen und Persönlichkeitstypen voneinander und definieren ein Rollenmodell, das zehn unterschiedliche Persönlichkeitstypen wie bspw. den *Visionär*, den *Trainer*, den *Friedensstifter* und das *Arbeitstier* benennt und charakterisiert (s. Abbildung 1).

Auch Spencer & Pruss weisen darauf hin, dass im Hinblick auf eine ideale Teameffizienz, das sog. Teamgleichgewicht, alle Rollen besetzt sein sollten. Soll ein Team mit neuen Mitgliedern ergänzt werden, so ist folglich darauf zu achten, dass diese geeignete Neigungen besitzen, um das Teamgleichgewicht zu wahren bzw. wieder herzustellen. Mittels eines umfangreichen Fragebogens können tendenzielle Neigungen zu den verschiedenen Rollen des Modells von Spencer & Pruss ermittelt werden.

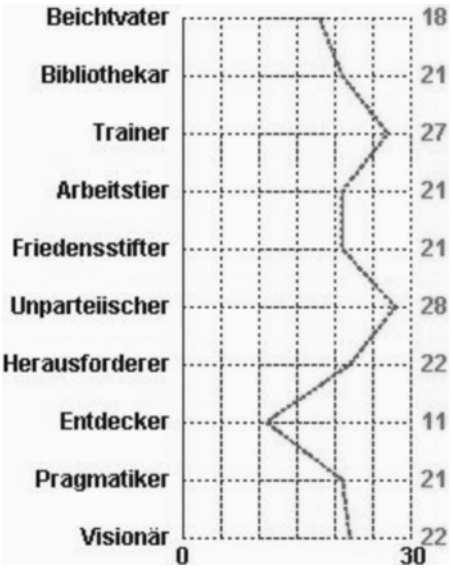

Abbildung 1: Ein typisches Rollenprofil nach Spencer & Pruss

3 Ziele

3.1 Virtuelle Tutorien

Um raum- und zeit-unabhängige Gruppenübungen realisieren zu können und so den diskursiven Vorgang virtueller Lerngruppen bei der objektorientierten Software-Entwicklung zu ermöglichen, ist es ein Ziel, virtuelle Tutorien

durchzuführen, d.h. die reale Tutoriumssituation in das E-Learning zu adaptieren. Ganz nebenbei werden dadurch personalintensive Präsenztutorien entlastet. Eine zentrale Frage bei der Umsetzung ist die, wie mit Problemsituationen umgegangen wird, da im virtuellen Tutorium prinzipbedingt kein Tutor vor Ort ist. Die Funktion des Tutors ist also zu ersetzen. Ansonsten entspricht die Situation der eines realen Tutoriums: einem Team wird eine Aufgabe gestellt, die es gemeinsam analysiert und bearbeitet. Treten Probleme auf, so greift normalerweise der Tutor helfend ein. Dabei ist aus didaktischer Sicht zu beachten, dass nicht zu viel und nicht zu wenig geholfen wird. Zuviel Hilfe verhindert den Lerneffekt, zu wenig Hilfe untergräbt die Motivation. Tritt ein Problem aufgrund mangelnder Rollenbesetzung im Team auf, so nimmt der Tutor automatisch die Rolle ein, die der Problemsituation entspricht bzw. in der Problemsituation hilft.

Genau diese Tatsache wird im Projekt VitaminL technisch umgesetzt. Im virtuellen Tutorium treffen sich Mitglieder eines Teams zur gemeinsamen Aufgabenbearbeitung. Als Kommunikations- und Arbeitsplattform steht eine CSCL-Umgebung zur Verfügung, die Werkzeuge zur Kommunikation und zur gemeinsamen Bearbeitung von Dokumenten enthält. Wie auch in realen Tutorien wird es zu Problemsituationen kommen. Diese Problemsituationen werden vom System erkannt und einer Rolle zugeordnet. Diese entspricht genau der Rolle, die in der realen Situation der Tutor einnehmen würde. Eine Analysekomponente erkennt auf der Basis eines zugrunde liegenden Rollenmodells, welche Rollenstruktur das Team hat. Entspricht die Problemsituation einer Rolle, die dem Team fehlt, so greift das System helfend ein. Entspricht sie einer Rolle, die im Team vorhanden ist, so überlässt es das Team sich selbst, dann muss es die Lösung selbst finden. Hier ist also das bekannte Problem intelligenter tutorieller Systeme zu lösen zu entscheiden, ob helfend eingegriffen werden soll oder nicht.

3.2 Analyse von Teamarbeit

Auf Basis eines entwickelten Rollenmodells für die oben beschriebene Situation eines virtuellen Tutoriums zur Programmierung beobachtet eine Softwarekomponente virtuelle Teams bei der Bearbeitung der Aufgaben und analysiert die Rollenstrukturen. Zur Ermittlung der Rollenstruktur werden z.Z. statistische bzw. probabilistische Verfahren geprüft (vgl. Kap. 5.2). Unter der Voraussetzung einer idealen Rollenstruktur werden Defizite im Team ermittelt und an eine weitere Softwarekomponente weitergeleitet. Diese simuliert somit eine oder mehrere Rollen, indem sie im Rahmen der Rolle bei auftretenden Problemsituationen helfend eingreift. Diese Hilfe wird über die

Kommunikationsschnittstelle umgesetzt und unterscheidet sich in ihrer Form nicht von der „normalen“ Kommunikation der Teammitglieder.

3.3 Kompensation von Defiziten durch Simulation

Wie schon erwähnt findet die Simulation nur im Rahmen der Rolle(n) statt, die der Problemsituation entspricht, um das Problem zu umgehen, dass dem Team zu viel geholfen werden könnte und somit ein Teil des Lernerfolgs verloren ginge. Andererseits ist die Hilfe wichtig, um die Motivation zu erhalten, die benötigt wird, um die Aufgabe erfolgreich zu Ende zu bringen. Das Verfahren der Simulation wird z.Z. erarbeitet, der aktuelle Stand ist in Kap. 5.3 und 5.4 dokumentiert.

4 Vorgehensweise und Konsequenzen

4.1 Projektstudie

Zur Vorbereitung des Forschungsprojekts *VitaminL* wurden in einer dem Projekt vorgelagerten Pilot-Phase erste Daten gesammelt und ausgewertet. Dazu wurde zunächst der Teamfragebogen von Spencer & Pruss (1995, S. 56ff) elektronisch umgesetzt und von mehreren Teams ausgefüllt. Ergänzend dazu wurde ein erster Prototyp der *VitaminL*-Lernumgebung entwickelt und in Benutzertests eingesetzt.

4.1.1 Fragebogen

Um im Vorfeld mögliche Informationen darüber zu bekommen, wie die Rollenstrukturen der Teams aussehen, wurde auf der Basis von Spencer & Pruss (1995) ein elektronischer Fragebogen entwickelt, der als Ergebnis ein Rollenprofil für jedes Teammitglied liefert. So lässt sich später feststellen, ob das von der Analysekomponente erkannte Rollenprofil dem entspricht, das der Fragebogen liefert. Außerdem lassen sich die Ergebnisse des Fragebogens dazu verwenden zu entscheiden, in welchem Bereich sich eine Unterstützung von Teams „lohnt“. Ansonsten müsste mit der Entwicklung der Unterstützungskomponente so lange gewartet werden, bis die Analyse fertig ist. Die Datenbasis der Fragebögen umfasst zur Zeit etwa 70 Datensätze und wächst pro Semester um 30-40. Da der Fragebogen als Web-Formular umgesetzt ist und per *JSP* (*Java Server Pages*) die Daten direkt an das Projekt weiterleitet, ist es sehr leicht, weitere Studierende folgender Programmierkurse einzubeziehen.

Inhaltlich ergibt sich aus der Beantwortung der Fragebögen ein Profil, das für jede vorhandene Rolle und Befragten einen Wert zwischen Null und 30 liefert (vgl. Abbildung 1). Diese Bandbreite von 30 wird dabei über die bisher Befragten relativ stark ausgeschöpft, die Differenz zwischen dem minimalen Wert und dem maximalen Wert bei einer Rolle schwankt zwischen 12 (Bibliothekar, min=10, max=22) und 22 (Friedensstifter, min=5, max=27).

Bricht man die Ergebnisse auf die einzelnen Gruppen herunter, indem man für die jeweiligen Rollen Durchschnittswerte über die Mitglieder berechnet, so ergeben sich starke Unterschiede. Erfüllt eine Gruppe die Entdeckerrolle zu 12,5 Punkten, so gibt es andere mit 24 Punkten in dieser Rolle. In diesem Fall wäre es sinnvoll, die erste Gruppe im Rahmen der Funktionen der Entdeckerrolle zu unterstützen. Bei jeder Unterstützung ist es immer wichtig, dass ein vernünftiges Maß an Unterstützung gefunden wird, um den Lernerfolg der Gruppe zu optimieren.

4.1.2 Benutzertests

Pro Semester finden etwa 15-20 Benutzertests statt, in denen an Teams mit 2-4 Mitgliedern Aufgaben vergeben werden, die dann im Rahmen der CSCL-Umgebung zu lösen sind. Die Mitglieder befinden sich dazu in verschiedenen Räumen und haben als Informationsquellen beliebige Bücher und Scripte zur Verfügung, auch ein Internetzugang ist vorhanden. Als Kommunikationsplattform wird ausschließlich die CSCL-Umgebung des Projekts benutzt.

Die Benutzertests dienen als empirisches Verfahren der Evaluation der erarbeiteten Modelle und der entwickelten Software-Komponenten zur Analyse der Teams und der Simulation. Dazu wird die gesamte Kommunikation aufgezeichnet und kann später nachvollzogen werden (Logfile-Analyse). Das gilt sowohl für die sprachliche Kommunikation (Chat) als auch für die Entwicklung der Quelltexte und Compilermeldungen. Auf Basis der Ergebnisse der Beobachtungen während der Benutzertests und der Logfile-Analyse wird die CSCL-Umgebung weiterentwickelt.

4.2 Folgerungen

Bisherige Benutzertests haben gezeigt, dass der Einsatz einer unterstützenden Umgebung von den Studierenden als durchaus sinnvoll empfunden wird, wenngleich im ersten Prototyp noch einige technische Probleme zu bewältigen waren. So wird als einer der wichtigsten Kritikpunkte die Tatsache angeführt, dass die eigentliche Programmierung jeweils nur von einem

Mitglied durchgeführt werden kann, da der Prototyp lediglich die Bearbeitung eines einzigen Dokuments erlaubt. Diese Beschränkung wird in nachfolgenden Versionen durch Verwendung von *Document Sharing* beseitigt werden.

Die Auswertung der Fragebögen hat gezeigt, dass in vielen Teams für die erfolgreiche Zielerreichung wichtige Persönlichkeitstypen unterrepräsentiert sind oder ganz fehlen. Dies deckt sich auch mit den heuristischen Beobachtungen der Autoren und führt zu dem Schluss, dass eine CSCL-Umgebung, die virtuelle Teams bei ihrer Aufgabenbearbeitung unter Berücksichtigung der jeweiligen Teamzusammensetzung unterstützt, einen gleichermaßen interessanten wie nutzbringenden Ansatz und eine sinnvolle Ergänzung zu schon vorhandenen CSCL-Systemen darstellt.

4.3 Das Arbeitsmodell von VitaminL

Aus den in der Studie gewonnenen Daten und Beobachtungen wird zunächst ein Arbeitsmodell konzipiert, welches den theoretischen Rahmen für die weiteren Forschungen und Entwicklungen bildet.

Auf der einen Seite ist das Rollenmodell zu finden, das die zehn Persönlichkeitstypen gemäß Spencer & Pruss enthält. Dabei handelt es sich zunächst um ein allgemeingültiges Modell, das die üblicherweise in einem Team benötigten Mitglieder sowie deren Stärken und Schwächen charakterisiert und so nicht ohne Einschränkung zur objektorientierten Software-Entwicklung passt.

Auf der anderen Seite stehen die Arbeitsfunktionen, die für die erfolgreiche Zielerreichung durch das Team vollständig wahrgenommen werden müssen. Diese werden in allgemeingültigen Arbeitsmodellen wie dem Rad der Arbeitsfunktionen (s. Abbildung 2) aufgestellt und beschrieben (Margerison, 1990; TMS-Zentrum, 2003). Je nach Aufgabenstellung und Rahmenbedingungen können diese anteilig variieren, aber keine Funktion darf völlig vernachlässigt werden.

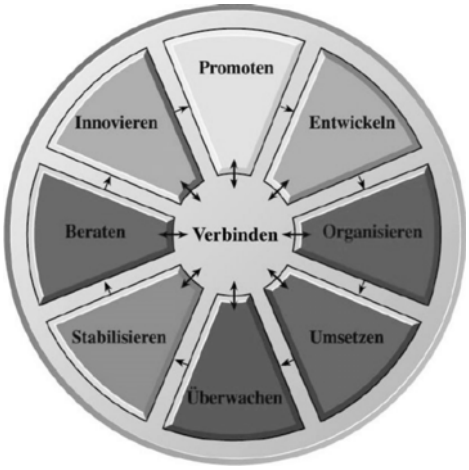

Abbildung 2: Das Rad der Arbeitsfunktionen (Quelle: TMS-Zentrum, 2003)

Diese allgemein gehaltenen Arbeitsfunktionen werden auf ein Kommunikationsmodell abgebildet und dort konkretisiert. Dabei wird eine Unterteilung in rein kommunikative Akte einerseits und Konstruktionsaufgaben andererseits getroffen. Die kommunikativen Akte werden entsprechend den *Collaborative Learning Skills* (McManus & Aiken, 1995) kategorisiert und decken sämtliche Kommunikationsmöglichkeiten der Gruppenarbeit und des Gruppenlernens ab. Die Konstruktionsaufgaben ergänzen die reine Kommunikation um sonstige Interaktionen, die im Rahmen von Gruppenarbeit notwendig sind.

Aus diesen beiden Ausgangsmodellen, dem Rollenmodell und dem Kommunikationsmodell, wird nun ein neues Rollenmodell entwickelt. Dieses ist speziell auf die oben beschriebene Ausgangssituation adaptiert und charakterisiert die Persönlichkeitstypen der objektorientierten Software-Entwicklung (s. *Abbildung 3*).

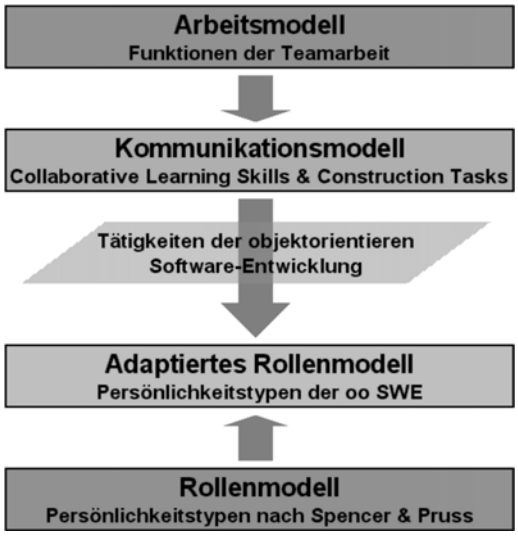

Abbildung 3: Das Arbeitsmodell von Vita*minL*

Dies bedeutet: jedem der Persönlichkeitstypen wird unter Zuhilfenahme eines probabilistischen Modells für ihn typische Kommunikationsakte zugeordnet, anhand der er später identifiziert werden kann. Dabei wird vorausgesetzt, dass jeder Persönlichkeitstyp bestimmte Aufgaben und Funktionen bevorzugt, wohingegen andere, ungeliebte Tätigkeiten eher vernachlässigt und anderen Teammitgliedern überlassen werden. Die für die objektorientierte Software-Entwicklung typischen Tätigkeiten, die hier besondere Berücksichtigung erfahren, können den gängigen Vorgehensmodellen wie dem Wasserfallmodell (Rechenberg & Pomberger, 1999, S. 776ff) oder dem V-Modell (IABG, 2004) entnommen werden.

Nach ersten Analysen der Aufzeichnungen von Benutzertests, der Logfiles und zusätzlichen Interviews, die mit Lernteams und Tutoren durchgeführt wurden, zeichnet sich ab, dass das Rollenmodell von Spencer & Pruss eine sehr gute Grundlage zu sein scheint. Typische Funktionen der Software-Entwicklung, wie bspw. die des Teamleiters, lassen sich auf die vorhandenen Rollen abbilden bzw. werden von anderen Rollen übernommen. Auch deutet sich ein Trend zu wichtigeren und weniger wichtigen Rollen an. So ist der Visionär mit seiner langfristigen Ausrichtung für Lernteams bei der Bearbeitung kurzzeitiger Aufgaben weniger interessant, während der Entdecker für die Informationsbeschaffung und Recherche, der Trainer für die Aufrechterhaltung von Motivation und Kommunikation sehr wichtig sind. Nach Abschluss der z.Z. laufenden Benutzertest-Phase soll das VitaminL-Arbeitsmodell manifestiert werden.

5 Implementierung

5.1 Gesamtmodell

Zunächst zeigt folgende Abbildung die schematische Übersicht über das System, beteiligte Komponenten und prinzipielle Abläufe. Das Kernstück des Systems bildet der Server, auf dem neben einer Benutzerverwaltung (u.a. zur Authentisierung der Teammitglieder) auch eine Komponente für eine verteilte Dokumentenbearbeitung (*Shared Documents*) sowie eine strukturierte Kommunika-tionsschnittstelle realisiert sind.

Mit weiteren Server-Komponenten wird einerseits die gesamte Kommunikation erfasst und analysiert (Analyseagent), um andererseits

Problemsituationen zu identifizieren (Code- und Compileragenten) und zielgerichtet eingreifen zu können (Kommunikationsagent).

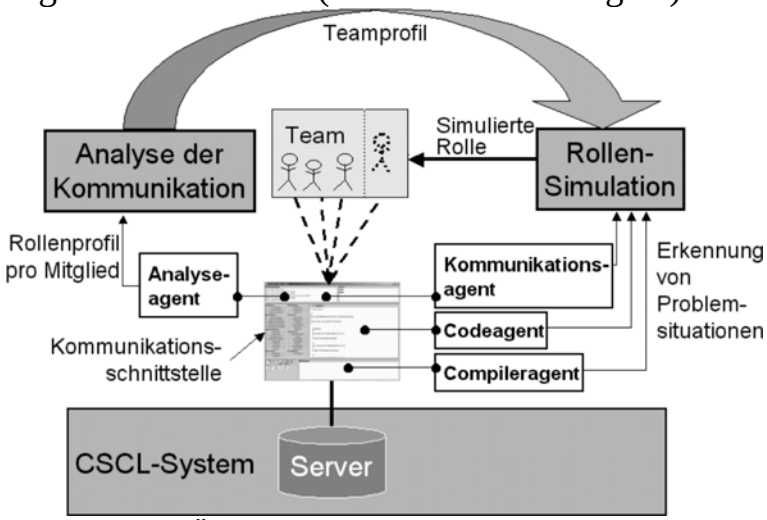

Abbildung 4: Übersicht des CSCL-Systems von VitaminL

5.2 Analyse der Teamarbeit

Die Zusammensetzung eines (realen wie virtuellen) Teams stellt also einen wesentlichen Faktor für dessen Erfolg bzw. Misserfolg dar. Diese Erkenntnis wirft nun die Frage auf, wie sich die Zusammensetzung eines Teams bestimmen lässt.

Die Grundlage des zum Einsatz kommenden Analyseverfahrens bildet das bereits im vorigen Abschnitt skizzierte Arbeitsmodell der Persönlichkeitstypen im Kontext der objektorientierten Software-Entwicklung und die ihnen zugeordneten Kommunikationsmuster. Die *Collaborative Learning Skills* (McManus & Aiken, 1995) ermöglichen eine umfassende Codierung der gesamten Kommunikation und werden mittels einer strukturierten Dialogschnittstelle in die GUI der Client-Anwendung eingebettet: jedem dieser Kommunikations-Codes wird eine Art charakteristischer Satzanfang zugeordnet, der nach Auswahl durch den Anwender von diesem sinngemäß in einem Dialogfenster vervollständigt werden kann. Diese Form der Kommunikation ist nur scheinbar eingeschränkt, tatsächlich haben aktuelle Forschungen (s. Matessa, 2001; Soller & Busetta, 2003) gezeigt, dass dies nicht der Fall ist, vielmehr findet die Kommunikation wesentlich bewusster statt. Ergänzt werden die Möglichkeiten der virtuellen Teamarbeit um eine Komponente für *Document Sharing*, die umfangreiche Editiermöglichkeiten zum Erstellen von Quelltexten bietet.

Abbildung 5: Kommunikation mit VitaminL

Findet sich nun ein virtuelles Team zu einer synchronen Arbeitssitzung zusammen, so wird dessen gesamte Kommunikation auf dem Server von der Analysekomponente erfasst und nach bestimmten Kommunikationsmustern durchsucht, um Hinweise auf die Zusammensetzung des Teams zu erhalten.
Die Zuordnung eines Teammitglieds zu einem oder mehreren Persönlichkeitstypen als Ergebnis einer Analyse lässt sich zwar durch einen menschlichen Experten einigermaßen zuverlässig durchführen, ist aber formal zunächst nur schwer lösbar, da zum einen die Rollen nicht strikt voneinander abgrenzbar sind und zum anderen dieses Problem schlecht strukturiert und in seiner Interpretierbarkeit nicht eindeutig ist. Daher wird für jedes Teammitglied ein sog. *Rollenprofil* angenommen, das Auskunft darüber gibt, zu welchen Persönlichkeitstypen das Teammitglied tendiert. Zur Lösung dieses Problems werden momentan mehrere Verfahren betrachtet und auf ihre Tauglichkeit hin untersucht.

5.2.1 Statistische Auswertungen

Die bislang in Benutzertests gesammelten Daten werden zunächst *offline* mit einfachen statistischen Verfahren untersucht und mit Ergebnissen aus den zugehörigen Fragebögen verglichen. Dieser Schritt dient dazu, grundlegende Erkenntnisse bzgl. der Persönlichkeitstypen des Modells und den von ihnen bevorzugten kommunikativen Handlungen aufzudecken, die ihrerseits zur weiteren Modellbildung herangezogen werden.

Das Spektrum der anzuwendenden Methoden reicht von der Berechnung verschiedener Kennzahlen (wie bspw. arithmetischer und geometrischer Mittelwerte) über Regressionsanalysen (zur Ermittlung linearer, exponentieller oder potentieller Zusammenhänge) bis hin zur Aufstellung von

statistischen Schätzfunktionen, mit denen die beobachteten Daten in ein Modell überführt werden, welches als Grundlage weiterer Analysen dient.

5.2.2 Analyse unter Unsicherheit

Da sowohl das Rollen-Kommunikations-Modell als auch die Analyse mit Unsicherheit behaftet sind, sollen zur Handhabung dieser Unsicherheit in der *online*-Analysekomponente probabilistische lernende Verfahren wie Neuronale Netze, Naive Bayes-Filter oder Markov-Ketten zum Einsatz kommen. Zunächst wird eine Trainingsphase durchgeführt, um eine Wissensbasis aufzubauen, indem Online-Sitzungen bspw. mit Referenzdaten in Übereinstimmung gebracht werden. Anschließend kann auf diese Wissensbasis in einer ersten Anwendungsphase zugegriffen werden, um die Funktionsweise der Analysekomponente zu überprüfen und ggf. auch anzupassen und zu verfeinern.

Letztlich werden die Analyseergebnisse an nachgelagerte Komponenten weitergereicht und dort für eine zielgerichtete Unterstützung des virtuellen Teams in Problemsituationen verarbeitet.

5.3 Erkennen von Problemsituationen

Wie auch in realen Tutorien ist es im virtuellen Tutorium ein zentraler Punkt, Problemsituationen zu erkennen. Das scheint in der realen Situation zunächst einfacher zu sein, denn dort wird einfach nach dem Tutor gerufen, wenn ein Team meint, Hilfe zu benötigen. Allerdings ist es letztlich Aufgabe des Tutors zu entscheiden, ob in diesem Fall wirklich Hilfe nötig ist. Als Technik für die Erkennung solcher Problemsituationen eignet sich Case Based Reasoning (CBR) sehr gut. Als wichtige Vorarbeit für die Benutzung sind zunächst mögliche Problemsituationen zu kategorisieren und für die CBR-Datenbasis zu modellieren.

Typische Problemsituationen bei Programmieranfängern sind anfangs einfache Syntaxfehler (Semikolon vergessen oder an der falschen Stelle, Klammerung, import vergessen), später semantische Fehler z.B. bei der Schleifenprogrammierung (Abbruchbedingung nicht beachtet oder falsch umgesetzt, Arrayindex-Grenzen nicht beachtet) und Fehler bei der objektorientierten Modellierung (public-, private-Attribute, Getter, Setter, Instanziierung von Objekten und wechselseitige Zugriffe). Konkrete Problemsituationen wurden bisher durch Beobachtung in realen Tutorien, bei Benutzertests und durch Interviews mit Studierenden und Tutoren ermittelt. Das gesammelte Material wird z.Z. gesichtet und analysiert, um danach

prototypisch bestimmte, möglichst repräsentative Situationen für die Simulation zu modellieren. Zentrale Frage ist dabei die Bestimmung der Attribute von Problemsituationen und deren Wertebereich. Tritt bspw. aufgrund eines fehlenden Semikolons der gleiche Compilerfehler mehrmals hintereinander auf, ist dies eine einfache, aber sehr typische Problemsituation, die technisch auch relativ leicht zu erkennen wäre, indem Compiler-meldungen zeitabhängig analysiert werden. Häufig gibt es Gruppen, die nur sehr selten compilieren und Syntaxfehler erst sehr spät erhalten. Ähnlich wie bei komplexen Programmierumgebungen kann man in solchen Fällen die Quelltexte im Hintergrund (unbemerkt) parsen und compilieren, mögliche Probleme so schon im Vorfeld erkennen und ggf. helfend eingreifen.

5.3.1 Case Based Reasoning

CBR ist eine Technik, die vorhandenes Erfahrungswissens, das in Form von sogenannten Fällen in einer Fallbasis gespeichert ist, zur Lösung neuer Probleme wiederverwendet. Aufgrund vielfältiger Einsatzmöglichkeiten (zum Beispiel in den Bereichen Help-Desk- und Diagnosesystemen, Entscheidungs-unterstützung, Design- und Planungsaufgaben, Suche in Produktkatalogen) ist CBR gegenwärtig eine vielversprechende und intensiv diskutierte Methode in Wissenschaft und Wirtschaft.

CBR ist ein Feld der künstlichen Intelligenz, bei dem es darum geht, auf der Basis von Kriterien Lösungen in einer Datenbasis zu finden. Darin unterscheidet es sich zunächst nicht von Fakteninformationssystemen, die bspw. mit SQL abgefragt werden. Der entscheidende Unterschied besteht darin, dass CBR ähnliche Lösungen sucht, während SQL entweder exakte oder gar keine Lösungen liefert. Die künstliche Intelligenz liegt somit im Algorithmus, der die Ähnlichkeit berechnet und natürlich in der Modellierung der Fälle, also in der sinnvollen Vergabe der Kriterien und deren Gewichtung.

Beispiel: Ein Kunde möchte ein Auto kaufen und hat bereits die Kriterien festgesetzt: Farbe rot, vier Türen, 150 kW, Seiten-Airbags, ABS, Allrad und Preis kleiner 20.000 Euro. Während eine einfache SQL-Abfrage vielleicht kein Ergebnis liefert, würde CBR die Autos anbieten, die am besten passen, aber wahrscheinlich nicht exakt.

Viele, wahrscheinlich sogar die meisten Ähnlichkeitsalgorithmen basieren auf dem Nearest-Neighbour-Algorithmus. Dieser arbeitet in einem n-dimensionalen Raum und sucht für einen Punkt in diesem Raum den nächsten Nachbarn, also den Punkt mit dem kleinsten Abstand.

Beispiel: 2-dimensionaler Raum, gesucht wird ein Algorithmus A für ein bestimmtes Problem, eine Dimension ist die Performance (y-Achse), eine der Preis (x-Achse), der Wertebereich ist jeweils zwischen eins und zehn. Es stehen drei Algorithmen zur Auswahl: A1 (7,7), A2 (4,4), und A3 (1,6). Gesucht wird ein möglichst günstiger (Preis 1) und schneller (Performance 10) Algorithmus (vgl. Abbildung 6).

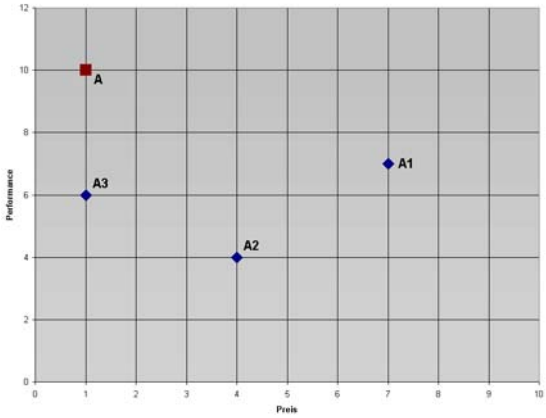

Abbildung 6: Nearest Neighbour

Der Betrachter erkennt sofort, dass A3 der nächste Nachbar von A ist und somit als erstes angeboten würde. Wie aber findet die Umsetzung in einen Algorithmus statt, wie berechnet der Computer die Abstände? Im zweidimensionalen Raum erkennt man die Lösung sehr schnell: Abstände lassen sich mathematisch sehr leicht mit dem Satz des Pythagoras berechnen. Das in Java implementierte Framework Selection-Engine von Baylor Weltzel adaptiert dieses Prinzip auf den n-dimensionalen Raum und erweitert es um Gewichtungen für die einzelnen Dimensionen. Es wird im folgenden Kapitel kurz beschrieben.

5.3.2 Das Selection-Engine-Framework

Das Selection-Engine-Framework ist komplett in Java entwickelt und lässt sich so sehr leicht in die VitaminL-Implementierung integrieren. Die zentrale Klasse für die Berechnungen von Ähnlichkeiten heißt SimilarityEngine. Diese benutzt verschiedene andere Klassen, die aber an dieser Stelle nicht komplett erklärt werden sollen. Wichtig ist die Methode computeSimilarity, die auf der Basis von Kriterien und deren Gewichten eine Liste von Objekten (der Klasse SimilarItems) liefert, die ähnlich der Ergebnisliste eines Information Retrieval Systems nach Ähnlichkeit zum Anfrageobjekt sortiert sind.

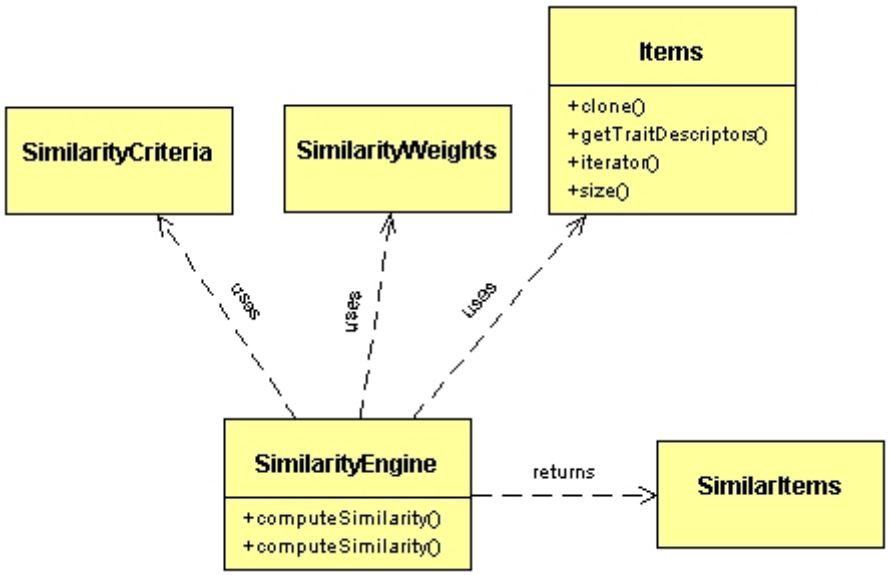

Abbildung 7: Zentrale Klassen des Selection-Engine-Frameworks

5.3 Simulation von Rollen

Die Simulation einer Rolle bedeutet genau genommen die Simulation der Funktionen der Rolle. Diese Funktionen werden letztlich kommunikativ über die CSCL-Umgebung umgesetzt. Dabei ist es der normale Fall, dass die simulierte Rolle in die Kommunikation eingreift und dem Team einen – vielleicht entscheidenden – Tipp gibt. In die Codierung der Quelltexte wird nicht eingegriffen, hier sind und bleiben die realen Teammitglieder eigenverantwortlich.

Die Vorgehensweise entspricht dem Fall einer realen Tutoriumssituation. Einem Team wird eine Aufgabe gestellt, die es gemeinsam analysiert und bearbeitet. An manchen Stellen wird es auf verschiedenartige Probleme stoßen. Tritt eine Problemsituation ein, die das Team offenbar nicht alleine lösen kann, hilft im realen Fall normalerweise der Dozent oder Tutor. Er nimmt somit automatisch eine Rolle ein, die der Problemsituation entspricht.
Diese Situation wird im Projekt auf virtuelle Teams abgebildet. Im virtuellen Team gibt es keinen Tutor und keinen Dozenten, aber es existiert ein virtuelles Teammitglied, das eine zu simulierende Rolle übernimmt. Tritt eine Problemsituation ein, so entscheidet dieses virtuelle Mitglied, ob es seiner Rolle entsprechend helfend eingreift. Im anderen Fall sollte das Team sich selbst helfen können, da die Problemsituation einer Rolle zugeordnet werden kann, die das Team an sich schon ausfüllt.

Die technische Umsetzung dieses Konzepts basiert auf dem o.g. *Case Based Reasoning* (*CBR)* bzw. dem Selection-Engine-Framework und wird z.Z. modelliert und prototypisch umgesetzt. Dazu werden die Problemsituationen nach Rollen klassifiziert und für eine CBR-Datenbasis modelliert. Tritt eine Problemsituation ein, sucht die CBR-Komponente nach ähnlichen Problemen der Vergangenheit, ermittelt die Lösung und setzt diese kommunikativ um. Anhand der Reaktion des Teams wird anschließend ermittelt, ob die Lösung hilfreich war, und aus dem Problem-Lösungspaar wird ein neuer Fall für die Datenbasis generiert. Auf dieses Weise lernt das System kontinuierlich dazu.

5.5 Architektur

Das CSCL-System von VitaminL ist eine vollständig in Java geschriebene Eigenentwicklung der Autoren. Den Kern des Systems bildet eine Client-Server-Architektur, die auf der Basis von *Remote Method Invocation* (*RMI*) arbeitet (s. Abbildung 8). Dabei werden sämtliche Aktionen während der Zusammenarbeit (in Form von Kommunikation und Dokumentenbearbeitung) stets an den Server geschickt, der diese entsprechend weiterleitet. Diese Form der Zentralisierung ermöglicht es, die gesamte Kommunikation eines Teams auf dem Server zu erfassen, zu protokollieren und zu Zwecken der Teamunterstützung zu verwenden.

Jegliche Kommunikation, die zwischen dem Server und den Clients stattfindet, wird unter Zuhilfenahme eines speziell für diese Zwecke entwickelten Nachrichtenprotokolls umgesetzt. Die abstrakte Klasse `VMessage` ist die Basisklasse aller Nachrichten, von ihr werden spezialisierte Nachrichten bspw. für die Kommunikation, für die Bearbeitung von Dokumenten oder auch für Benutzerinformationen abgeleitet.

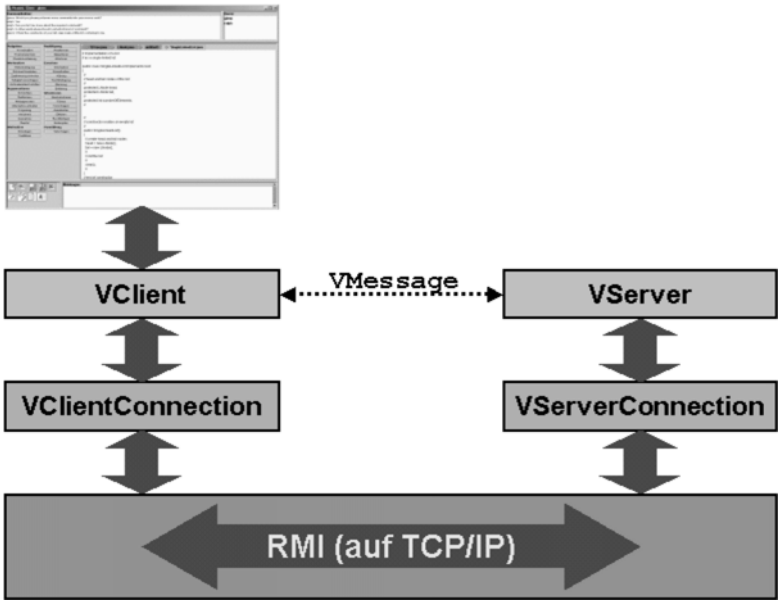

Abbildung 8: Die Architektur des CSCL-Systems von VitaminL

Die Oberfläche der Client-Applikation wird mit den GUI-Klassen der Swing-Bibliothek von Java realisiert und enthält neben der Möglichkeit, mittels einer strukturierten Schnittstelle zu kommunizieren (s. Matessa, 2001; Soller & Busetta, 2003) auch eine Komponente, die die gleichzeitige Bearbeitung mehrerer Dokumente durch die Gruppe mittels *Document Sharing* ermöglicht. Die objektorientierte Architektur mit einfachen Schnittstellen und einem transparenten Nachrichtenprotokoll erleichtert die Erweiterung das Systems durch externen Entwickler. So ist es möglich, Weiterentwicklungen auch im Rahmen von Praktika oder Master-/Magisterarbeiten durchzuführen.

6 Ausblick

Nach ersten Evaluierungen des Projekt *VitaminL* mittels Benutzertests zeichnen sich jetzt schon interessante Möglichkeiten für zukünftige Versionen ab. Neben der Möglichkeit, das CSCL-System um *Shared Whiteboards* für die gemeinsame Erstellung von Skizzen, Grafiken und UML-Diagrammen zu erweitern, bieten sicherlich der Ausbau und die Verbesserung der Analyse- und Simulations-komponenten noch viel Raum für weitere Forschungs- und Entwicklungstätigkeiten.

Am Ende dieser Entwicklung steht ein System, das jedem virtuellen Team optimale Unterstützung im Sinne seiner rollenbasierten Zusammensetzung liefert. Auch ein CSCL-System, das aufgrund seines gespeicherten Wissens für bestimmte Aufgaben optimale Teams aus einer Vielzahl potentieller Teammitglieder zusammenstellt, ist in diesem Zusammenhang denkbar.

7 Referenzen

[Bales 1950] Bales, Robert F.: Interaction Process Analysis - A Model for the Study of Small Groups. Chicago - London : The University of Chicago Press, 1950

[Belbin 2003] Belbin, M.: BELBIN: Home to Belbin Team Roles & Work Roles. 2003. - URL http://www.belbin.com. - Zugriffsdatum: 07.03.2003

[Daradoumis 1999] Daradoumis, T.: Towards an Integrated Model of Dialogue for CSCL. 1999. - URL http://cbl.leeds.ac.uk/~tamsin/dialogueworkshop/daradoumis-dialogue-model.pdf. - Zugriffsdatum: 25.02.2003

[Eunson 1990] Eunson, Baden: Betriebspsychologie. Hamburg - New York: MacGraw-Hill, 1990.

[Hare 1962] Hare, P. A.: Handbook of Small Group Research. New York: The Free Press of Glencoe, 1962

[Henze 2001] Henze, N.; Schmidt, C.; Wolpers, M. (2001): Mediengestützte Didaktik für qualitative Methoden in der Sozialforschung auf der Basis semantischer Modellierung. In: Wagner, E.; Kindt, M. (Hrsg.): Virtueller Campus. Szenarien, Strategien, Studium. Waxmann, Münster, New York, München, Berlin, S.164 – 171

[IABG 2004] IABG: Das V-Modell - Planung und Durchführung von IT-Vorhaben, 2004. - URL http://www.v-modell.iabg.de/. - Zugriffsdatum: 05.03.2004

[Kölle & Langemeier 2004] Kölle, R. ; Langemeier, G.: VitaminL - Eine CSCL-Umgebung für kooperatives Lernen einer Programmiersprache. – URL http://www.vitaminl.de. – Zugriffsdatum: 22.04.2004

[Konradt & Hertel 2002] Konradt, U.; Hertel, G.: Management virtueller Teams - Von der Tele-arbeit zum virtuellen Unternehmen. Weinheim und Basel: Beltz, 2002

[Kuhlen 2004a] Kuhlen, R.: ENFORUM – Enzyklopädisches Forum. – URL http://www.enforum.net – Zugriffsdatum: 10.05.2004

[Kuhlen 2004b] Kuhlen, R.: K3-forum – Kollaboration, Kommunikation, Kompetenz (Informations-kompetenz). – URL http://www.k3forum.net – Zugriffsdatum: 10.05.2004

[Lewe 1995] Lewe, H.: Copmuter Aided Team und Produktivität. Einsatzmöglichkeiten und Erfolgspotentiale. Wiesbaden: Gabler Edition Wissensschaft, 1995

[Margerison 1990] Margerison, C.: Team-Mangagement. London: Management Books 2000 Ltd, 1990

[Matessa 2001] Matessa, M.: The Benefit of Structured Interfaces in Collaborative Communication. 2001. - URL http://www.matessa.org/~mike/pubs/aaai01.pdf. - Zugriffsdatum: 20.02.2004-03-07

[McManus & Aiken 1995] McManus, M. M.; Aiken, R. M.: Monitoring computer-based problem solving. In: Journal of Artificial Intelligence in Education (1995), Nr. 6, S. 307-336

[Pfister 2000] Pfister, H.-R.: Kooperatives computerunterstütztes Lernen (CSCL) - Was ist das und wozu nützt es? Eröffnung des CSCL-Kompetenzzentrums am GMD-IPSI in Darmstadt. In: nfd Information - Wissenschaft und Praxis (2000), Nr. 4 (51. Jg.), S. 227–231

[Pilkington et al. 1999] Pilkington, R. ; Treasure-Jones, T. ; Kneser, C.: The Tutor's Role: An Investigation of Participant Roles in CMC Seminars Using Exchange Structure Analysis. 1999. - URL http://cbl.leeds.ac.uk/~tamsin/dialogueworkshop/ pilkington-tutorsrole.pdf. - Zugriffsdatum: 25.02.2003

[Rechenberg & Pomberger 1999] Rechenberg, Peter (Hrsg.); Pomberger, Gustav (Hrsg.): Informatik-Handbuch. 2., aktualisierte und erweiterte Auflage. München: Hanser, 1999

[Schudnagis & Womser-Hacker 2002] Schudnagis, M. ; Womser-Hacker, C.: Multimediale Lernsysteme softwareergonomisch gestalten. In: Mensch und Computer 2002, Juli 2002

[Schulmeister 1997] Schulmeister, R.: Grundlagen hypermedialer Lernsysteme. 2.Auflage. München: Oldenbourg, 1997.

[Schulmeister 2001] Schulmeister, R.: Virtuelle Universität - Virtuelles Lernen. München: Oldenbourg, 2001

[Singley et al. 2000] Singley, M. K.; Singh, M.; Fairweather, P.; Farrell, R.; Swerling, S.: Algebra Jam: supporting teamwork and managing roles in a collaborative learning environment. In: Proceedings of the 2000 ACM conference on Computer supported cooperative work, ACM Press, 2000, S. 145-154 - URL http://doi.acm.org/10.1145/ 358916.358985. - Zugriffsdatum: 10.02.2003

[Soller & Busetta 2003] Soller, A.; Busetta, P.: An Intelligent Agent Architecture for Facilitating Knowledge Sharing Communication, 2003. - URL http://www.traclabs.com/ ~cmartin/hmas/wkshp_2003/papers/Soller.pdf. - Zugriffsdatum: 19.08.2003

[Spencer & Pruss 1990] Spencer, J.; Pruss, A.: Top Teams - Der Königsweg zu mehr Flexibilität, Effizienz und Erfolg im Betrieb. München: Knaur 1995

[Teufel et al. 1995] Teufel, S. ; Sauter, C. ; Mühlherr, T. ; Bauknecht, K.: Computerunterstützung für Gruppenarbeit. Addison-Wesley, 1995

[TMS-Zentrum 2003] TMS-Zentrum: TMS - das Team Management System, 2003. - URL http://www.tms-online.de. - Zugriffsdatum: 20.03.2003

[Vygotsky 1978] Vygotsky, L. S.: Mind in society. Cambridge, MA: Harvard University Press, 1978

[Zimbardo & Gerrig 2000] Zimbardo, P. G.; Gerrig, R. J.: Psychologie. 7. Auflage. Berlin: Springer, 2000