

Mehrwert der rollenbasierten Umsetzung von kollaborativen Lernumgebungen

Hung Tran Duc

Technische Universität Dresden

Dresden, Deutschland

hung.tduc@yahoo.com

Zusammenfassung—Rollenbasierte Sprachen werden seit Jahrzehnten als Alternative zu objektorientierten Ansätzen untersucht. Die Natur der Rolle erlaubt es Objekten, sich dynamisch an verschiedene Anforderungen anzupassen. Mit zunehmend komplexer und offener Software, besteht eine wachsende Nachfrage nach adaptiven Systemen. Kollaborative Lernumgebungen fördern je nach Anwendung die Sozialkompetenz, Eigeninitiative oder Konzentrationsfähigkeit der Lernenden. In dieser Arbeit soll beleuchtet werden, welche Vorzüge eine Lernumgebung auf der Grundlage von Rollen aufweist.

Stichwoerter—kollaborative Lernumgebung, e-Learning, rollenbasiert

I. EINLEITUNG

Objektorientierte Programmierung ist das verbreitetste und am häufigsten angewendete Programmierparadigma. Dennoch kann es sich lohnen, Alternativen zu untersuchen. Sie könnten zu innovativen Konzepten führen, welche Mittel und Wege aufzeigen, die mit Objektorientierung nicht erkenntlich waren. Beispielalternativen sind funktionsorientierte Programmierung, welche sich auf die Modellierung von Prozessen fokussierte, oder aspektorientierte Programmierung. Letztere bietet die Möglichkeit zentral Verhalten, das mehrere unabhängige Klassen annehmen müssen, zu definieren.

Rollenbasierte Programmierung stellt eine weitere Alternative zur Objektorientierung dar. Zum ersten Mal wurden Rollen 1977 von Bachman et al. charakterisiert. Sie beschreiben Rollen als festgelegtes Verhalten, welches von Objekten verschiedener Klassen angenommen werden kann [18]. Seitdem wurde der Begriff der Rolle stets wieder aufgegriffen und in verschiedenen Bereichen thematisiert. Diese Bereiche umfassen Ontologien [10] [4], Datenmodellierung [5], Konzeptmodellierung [6] und Programmiersprachen [17]. In [16] hat Steimann aktuelle Untersuchungen zu Rollen zusammengefasst und bewertete den Einfluss der Rolle auf die moderne Datenmodellierung als gering. Diese Beobachtung und die Menge an Untersuchungen, sowohl vor als auch nach Steimann, weisen auf ein großes aber ungenutztes Potenzial der Rolle als Programmierparadigma hin. Eine Rolle beschreibt Attribute und Verhalten von Objekten, die in einem bestimmten Kontext miteinander kollaborieren. Ein Objekt kann also je nach Bedarf eine neue Rolle und somit auch neues Verhalten annehmen. Diese Anpassungsfähigkeit kann hilfreich sein, da moderne Softwaresysteme zunehmend komplexer und offener werden [11].

In den meisten Hochschulen werden mittlerweile Lernplattformen verwendet. Neben der Bereitstellung von Lehrmaterialien, dienen sie auch zur Organisation von Lernvorgängen. Als Lernumgebung erleichtern sie den Lernenden die Kommunikation untereinander oder mit den Vortragenden. Der Lernprozess kann somit zu beliebiger Zeit an einem beliebigen Ort mit einem mobilen Endgerät durchgeführt werden. Audience Response Systeme (ARS) werden während Lehrveranstaltungen eingesetzt. Sie ermöglichen es, von allen Zuhörern gleichzeitig Feedback bzw. Input einzuholen. Browserbasiert oder als mobile App verfügbar, benötigen die Systeme keine spezielle Hardware. Primärer Anwendungsfall ist das Stellen von Fachfragen um den Vorbereitungsgrad oder den Lernstand des Publikums zu ermitteln. Es konnte beobachtet, dass das ARS zu einem deutlich höherem Engagement in Vorlesungen führt. Zusätzlich konnten bei Studenten eine kontinuierliche Aufmerksamkeit und höhere Lernleistung festgestellt werden [14]. Eine andere Anwendung stellen kollaborative Lernumgebungen dar. Sie haben ein einfaches Grundprinzip: Lernende werden in Gruppen geteilt um gemeinsam Aufgaben zu lösen. Sie teilen die Aufgaben untereinander auf und tauschen ihre Kenntnisse aus. Auf diese Weise werden Sozialkompetenz und Organisationsfähigkeit gefördert. Auch die Motivation eines Studierenden, kann durch das Gefühl der Gruppenzugehörigkeit gesteigert werden. Der Fokus auf Kollaboration, sowie die verschiedenen Anwendungsfälle legen nahe, dass Lernumgebungen aus einer rollenbasierten Umsetzung einen Vorteil ziehen könnten.

In dieser Arbeit wird diskutiert, welche Vorzüge eine rollenbasierte Lernumgebung gegenüber einer klassisch objektorientierten Umsetzung haben könnte. Dazu werden in Kapitel 2 zuerst Problem und Anforderungen, sowie verschiedene Szenarien in Kollaborativen Lernumgebungen beschrieben. Nachfolgend wird auf die Herausforderungen und Grenzen der objektorientierten Programmierung eingegangen. Kapitel 3 beschäftigt sich mit dem Konzept der Rolle und verschiedenen rollenbasierten Systemen. Kapitel 4 thematisiert die möglichen Vorteile von rollenbasierten Modellierungs- und Programmiersprachen, welche einer Kollaborativen Lernumgebung von Nutzen sein könnten. Die eventuell entstehenden Herausforderungen und Probleme rollenbasierter Ansätze werden in Kapitel 5 erläutert.

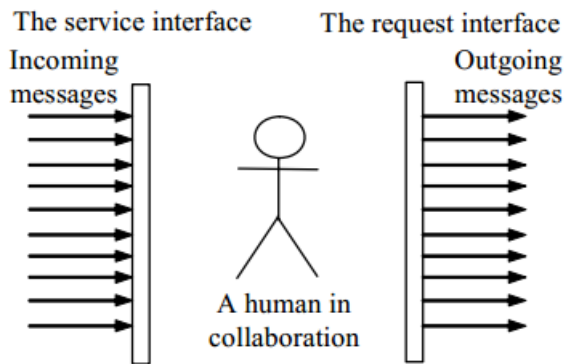


Abb. 1. Eine Rolle unterstützt als Wrapper in einer kollaborativen Umgebung die Kommunikation zwischen Mensch und Computer. [19]

II. VERWANDTE ARBEITEN

Zhu evaluierte die Umsetzung der Rolle in computergestützten Kollaborationssystemen [19]. Und befand, dass in den meisten damaligen Systemen die Rolle als Label für Objekte benutzt wurde. Oft wurde in einer „Switch/Case“-Struktur festgelegt, welche Rolle welches Verhalten auslöst. Waren die Umgebung und die enthaltenen Rollen einmal implementiert, war es sehr aufwendig, an den Rollen Anpassungen vorzunehmen.

In kollaborativen Umgebungen, steht die Zusammenarbeit bzw. der Austausch zwischen Nutzern im Vordergrund. Auf Grundlage dieser Ansicht, stellt Zhu die Behauptung auf, es gäbe lediglich zwei Existenzzustände eines Objektes. Der *Client* formuliert und sendet Anweisungen, während der *Server* eingehende Anfragen abarbeitet. Eine Rolle würde vorgeben, was Nutzer und Umgebung bei einem bestimmten Kontext voneinander erwarten bzw. fordern. Durch das Annehmen einer Rolle, erhält der Nutzer spezielle Rechte, oder auch Verpflichtungen. Eine Rolle stellt im kollaborativen System einen Wrapper dar, der ein Objekt „ummantelt“ und ihm neue Eigenschaften verleiht (Siehe Abb.1). Der Wrapper besteht aus zwei Teilen: Die Schnittstelle für ausgehende Nachrichten, welche Anfragen an das System sendet, und eine Schnittstelle für eingehende Nachrichten, um die Anfragen anderer Objekte zu verarbeiten.

Desweiteren formulierte Zhu mehrere Grundprinzipien für den Aufbau eines Kollaborativen Systems. Das System ist dazu gedacht die Zusammenarbeit von Menschen zu erleichtern, egal ob sie sich am selben Ort befinden oder nicht. Die Grundprinzipien beschreiben die Eigenschaften der Objekte, Rollen und Gruppen. Zusammengefasst loggen sich Nutzer beispielsweise über einen Internetbrowser in das System ein, werden durch ein *Agentenobjekt* repräsentiert und bekommen jeweils eine voreingestellte Gruppe und eine Rolle zugewiesen. Rollen, Klassen und Gruppen werden vom Nutzer erstellt und verwaltet. Eine Gruppe wird spezifiziert durch die Rollen, die angenommen werden müssen und weiteren Eigenschaften wie Sichtbarkeit und Zugänglichkeit. Die Rollen sind Objekte und somit Instanzen einer Klasse von Rollen. Jede Rolle

gibt dem Nutzer vordefinierte Nachrichtenvorlagen, welche dazu genutzt werden, Nachrichten zwischen dem Nutzer und dem System auszutauschen. Sie kann von mehreren Nutzern gleichzeitig gespielt werden, aber ein Nutzer kann zu jedem Zeitpunkt nur eine Rolle tragen. Ist ein Nutzer nicht angemeldet, wird er durch einen Agenten vertreten, der eingehende Nachrichten sammelt und für den Nutzer bereithält. Abschließend betont Zhu wie fundamental das Konzept der Rolle für kollaborative Systeme sei. Weitere Forschung würde in diversen Bereichen Fortschritte, vorteilhaft für Kollaboration, hervorbringen.

III. HERAUSFORDERUNGEN STATE OF THE ART

In diesem Kapitel wird eine beispielhafte kollaborative Lernumgebung für eine Hochschule beschrieben. Es werden Use-Cases genannt und deren relevante Vorgänge beschrieben. Ziel ist es, eine Menge von Anforderungen zu bilden, auf die das Konzept eines rollenbasierten Systems angewandt werden kann.

A. Herausforderungen Kollab e-Learning

Alle Studierende einer Hochschule besitzen einen Login für die Lernumgebung und sind als „Studierende“ gekennzeichnet. Professoren, wissenschaftliche Mitarbeiter und andere Angestellte der Hochschule sind unter „Mitarbeiter“ zusammengefasst und haben ebenfalls eigene Logindaten.

a) *Studiengang*: Studierende immatrikulieren sich in einen Studiengang. Sie haben die Möglichkeit ihn zu wechseln oder einen zusätzlichen Studiengang als Parallelstudium hinzuzufügen. Anhand des Studiengangs wird bestimmt, zu welchen Kursen der Studierende Zugriff hat.

b) *Audience Response System(ARS)*: Das bereits genannte ARS ist ein System um in Vorlesungen simultan von allen Hörern Input einzuholen, beispielsweise um Fachfragen zu beantworten. Diese Fachfragen und die dazugehörigen Antwortmöglichkeiten müssen vorher vom Vortragenden formuliert und im System gespeichert worden sein. Um diese Fragen jederzeit schnell aufrufen zu können, ist eine Zuordnung zur jeweiligen Lehrveranstaltung nötig, möglicherweise in Form eines Katalogs. Es besteht die Möglichkeit, dass ein Mitarbeiter mehrere Kurse leitet, in anderen Kursen als Vortragender erscheint oder die Rolle des Hörers annimmt um selbst teilzunehmen. Ein Kurs kann mehrere wechselnde Vortragende haben. Das Recht, diese Rolle zu übertragen liegt beim Kursleiter, dem Mitarbeiter, der für diesen Kurs verantwortlich ist.

In Seminaren oder ähnlichen Lehrveranstaltung, muss es möglich sein, die Rolle des Vortragenden für einen begrenzten Zeitraum zu übertragen. In dieser Zeit kann der neue Vortragende ebenfalls fachbezogene Fragen stellen und sich Feedback vom Publikum einholen. Die Annahme der Rolle geschieht nur während der Lehrveranstaltung. Die Formulierung der Fragen samt Antworten findet während der Vorbereitung des Vortrags statt.

c) *Arbeitsgruppen*: Im Rahmen einer Lehrveranstaltung wurden Studierende in Gruppen eingeteilt. In diesen Gruppen sollen sie nun gemeinsam verschiedene Aufgaben lösen. Neben einer zufälligen Zuteilung, kann die Gruppenbildung auf der Lernplattform stattfinden, wo Studierende Gruppen eröffnen oder sich einer vorhandenen anschließen können. In [2] befragte Biasutti Studierende zu ihren Eindrücken, nachdem diese in einer kollaborativen Lernumgebung asynchron eine Aufgabe bearbeiteten. Als Kritikpunkte wurde, neben ungleichmäßiger Beteiligung, die Organisation innerhalb der Gruppe genannt. Eine klare Aufgabenverteilung wurde als wünschenswert betrachtet. In [12] und [1] wird betont wie wichtig es sei, dass einerseits ein Ansprechpartner zur Verfügung steht und dass andererseits die Aktivitäten der Gruppe zum Teil überwacht werden. Casamayor et al. stellen den Ansatz vor, relevante Statistiken einer Gruppe aufzuzeichnen und bei bestimmten Ereignissen den Kursleiter zu benachrichtigen [1].

d) *Offenes Forum*: In OPAL¹ können zu Lehrveranstaltungen offene Foren eröffnet werden. Der Kursleiter kann durch einen Eintrag alle Studierenden über neue Informationen in Kenntnis setzen. Studierende haben die Gelegenheit Fragen zur Veranstaltung oder zu Aufgaben zu stellen, welche von anderen Studierenden oder vom Lehrer beantwortet werden kann.

e) *Reviews*: Enthält eine Lehrveranstaltung eine schriftliche Ausarbeitung, kann jedem Studierenden die Arbeit eines anderen zur Bewertung zugewiesen werden. Selbstverständlich können auch Kursleiter die Ausarbeitungen bewerten.

B. Probleme der Umsetzung in objektorientierter Programmierung

So etabliert und verbreitet objektorientierte Programmierung auch sein mag, so hat sie dennoch ihre Grenzen. Laut Reenskaug et al. entstand objektorientiertes Design mit dem Ziel, dass der Programmcode möglichst genau dem mentalem Modell des Endnutzers entspricht [13]. Ein Nutzer soll beim Bedienen der Nutzerschnittstelle eine konkrete Vorstellung damit haben, wie er mit den Objekten des Programms interagiert. Je genauer ein Programm die Veränderungen an inneren Zuständen durch den Nutzer wiedergibt, desto intuitiver die Handhabung. Er kann nicht die tatsächlichen Vorgänge beobachten. Er braucht lediglich eine Repräsentation des Programmzustands um es zu bedienen. Diese Abbildung der Vorstellung des Users auf den Code gelingt nicht überall. Klassische objektorientierte Programmierung bietet keine Möglichkeit, die Kollaboration zwischen Objekten zu beschreiben. Algorithmen, die durch Kollaborationen erfüllt werden, und Relationen haben wie Objekte eine Struktur, die sich im Quellcode nur umständlich repräsentieren lässt. Steimanns Veranschaulichung der Problematik soll im Folgenden auf eine kollaborative Lernumgebung mit den oben genannten Merkmalen übertragen werden [16].

a) *Multiple, dynamische Klassifizierung*: Sowohl Mitarbeiter, als auch Studierende können während der Laufzeit verschiedene Rollen annehmen und wieder ablegen, wie in Abbildung 2 veranschaulicht. Da es unmöglich ist vorherzusehen, wer welche Rolle spielen wird, muss ein Objekt in der Lage sein dynamisch seine Klasse zu wechseln.

b) *Rollen als Unterklassen*: Ein *Studierender* kann gleichzeitig Hörer in einem als auch Vortragender im anderen Kurs sein. Dies würde bedeuten, dass *Hörer* und *Vortragender* Unterklassen von *Studierender* sind. Dasselbe Verhältnis besteht bei *Mitarbeiter*, die ebenfalls Hörer und Vortragende sein können. Bei einer Unterklasse von sowohl *Studierende* als auch *Mitarbeiter* beschränkt sich das Verhalten auf die Schnittmenge der beiden Oberklassen und fällt damit klein oder leer aus.

c) *Rollen als Oberklassen*: Hörer und Vortragender würden Oberklassen für sowohl Mitarbeiter, als auch Studierende darstellen. Das würde bedeuten, dass jeder Mitarbeiter oder Studierender ein Hörer, Vortragender oder beides ist. Es wäre nicht möglich weder Hörer noch Vortragender zu sein.

d) *Statusabhängigkeit*: Eine Rolle als Ober- oder Unterklasse zu implementieren, führt dazu, dass ein Objekt und dessen Rollen, als eine einzelne Instanz dargestellt werden. Diese Instanz kann nur einen Status haben. Sollte ein Mitarbeiter in mehreren Kursen Vortragender sein, würden die Attribute, die mit der Rolle des Vortragenden einhergehen, für jeden Kurs die gleichen Werte haben.

IV. ROLE CONCEPT AND FEATURES

Laut Steimann ist die Rolle neben Objekten und Relationen, ein fundamentales Konzept der Modellierung. Trotz dessen und der langen und breiten Untersuchung des Begriffs, gibt es keine einheitliche Definition. 1647 unternahm Lodwick den Versuch, eine Universalsprache zu entwickeln [7]. Er entwarf ein System mit Vorgängen und Aktivitäten, deren Struktur Rollen enthielten, die von Objekten gespielt wurden. Beim Prozess „Übermittlung einer Nachricht“ gibt es die Rollen *Empfänger* und einen *Sender*. Die Objekte oder Personen, die diese Rollen erfüllen, haben außerhalb der Aktivität eine eigene natürliche Bezeichnung. In [16] analysiert Steimann verschiedene Konzepte wie Rollen in der objektorientierten Modellierung umgesetzt werden. Aus diesen verschiedenen Sichtweisen entnahm Steimann wiederkehrende Merkmale des Begriffs und kategorisierte die Ansätze.

In der ersten Kategorie, werden Rollen als *Teilnehmer einer Relation* gesehen. Eine Rolle wird dadurch definiert, welche Funktion sie innerhalb einer Relation erfüllt. Steimann argumentiert, dass dadurch die individuellen Eigenschaften einer Rolle nicht berücksichtigt werden. Heutzutage ist es vollkommen ausreichend die Relation „empfängt Nachricht von“ auszudrücken, indem die Rollen „Empfänger“ und „Sender“ zugeteilt werden.

In der zweiten Kategorie stellen Rollen Spezialisierungen oder Generalisierungen dar, d.h. Ober- oder Unterklassen. Spezialisierungen scheinen sinnvoll, da Rollen ein Objekt

¹<https://bildungsportal.sachsen.de/opal/>

genauer beschreiben. Andererseits hat zum Beispiel die Klasse *Person* viele Eigenschaften die von der Rolle *Vater* nicht benötigt werden. Erst durch das Annehmen der Rolle, erhält sie die relevanten Attribute und Methoden, was *Vater* zur Oberklasse macht. Im vorherigen Unterkapitel wurden bereits die Probleme dieser Kategorie erklärt.

Eine potentielle Lösung dieser Probleme findet sich in der dritten Kategorie. Viele Autoren vertreten die Auffassung, Rollen seien zum Objekt separate Instanzen. Ein Objekt wird mit seinen Rollen mit der Relation „wird gespielt von“ verbunden und erscheint nach außen als eine Komposition mehrerer Objekte. Das Annehmen und Ablegen einer Rolle wird realisiert durch das Hinzufügen oder Entfernen einer Instanz der Rolle innerhalb der Komposition. Das größte Problem der Umsetzung von Rollen als separate Instanzen ist, dass diese Instanzen sich untereinander eine Identität teilen. In der objektorientierten Modellierung ist grundsätzlich jede Instanz ein Objekt und hat somit eine eigene Identität.

Steimann entwickelte seine eigene Definition der Rolle und setzt sie in der Modellierungssprache *Lodwick* um, mit dem Ziel die verschiedenen Auffassungen einer Rolle zusammenzuführen.

Kühn evaluierte 2017 aktuelle rollenbasierte Modellierungs- und Programmiersprachen und charakterisierte den Begriff der Rolle anhand von drei Aspekten [9].

a) *Aspekt der Anpassung*: Das Verhalten eines Objekts oder eines Spielers ist abhängig von der Rolle, die es oder er spielt. Daraus folgt, dass eine Rolle das Verhalten des spielenden Objekts anpassen kann. Im Beispiel der Lernumgebung bedeutet das, dass ein Studierender zunächst die Rolle des Vortragenden annehmen muss, um entsprechende Rechte zu erhalten. Desweiteren wird jede Rolle nur von einem Objekt gespielt. Im Gegensatz dazu kann ein Objekt mehrere Rollen gleichzeitig annehmen und auch dieselbe Rolle mehrmals. Dies wird veranschaulicht durch den Studenten, der sowohl Vortragender als auch Hörer sein kann und letzteres möglicherweise simultan in mehreren Kursen.

Eine Rolle kann von Objekten unterschiedlichen Typs gespielt werden, wodurch nicht nur Studierende, sondern auch Mitarbeiter Hörer oder Vortragender sein können. Zur Laufzeit kann ein Objekt Rollen dynamisch annehmen und wieder ablegen. Äquivalent dazu kann ein Studierender jederzeit die Rolle des Vortragenden vom Kursleiter aufgetragen bekommen.

b) *Aspekt der Relation*: Hier liegt der Fokus, wie bei Steimanns erster Klassifizierung der Auffassungen, auf die Eigenschaften einer Rolle innerhalb einer Relation oder Beziehung. Der Aspekt beinhaltet, dass Rollen ein eigenes Verhalten und eigene Werte haben und dass Rollen dennoch durch Relationen bestimmt werden können.

c) *Aspekt der Kontextabhängigkeit*: Mobile Applikationen müssen in der Lage sein, sich an einen ständig ändernden Kontext anzupassen. Dies führte dazu, dass rollenbasierte Sprachen der letzten zehn Jahre vermehrt kontextabhängige Rollen anwendeten, um das Verhalten adaptiver Anwendungen zu beschreiben. Der Begriff des Kontexts ist nicht eindeutig definiert. Kühn nennt zwei

häufig vorkommende Ansichten. Nach Dey ist der Kontext die Menge aller Informationen, mit der man die Situation eines Objekts beschreiben kann [3]. Dementsprechend wäre ein Kontext immerwährend und stetig wechselnd. Kamina und Tamai schreiben einem Kontext eine Identität, eigene Bedingungen und eine begrenzte Lebensspanne zu [?]. Kühn unterscheidet die beiden Definitionen voneinander, indem er den Kontextbegriff von Kamina und Tamai als *Behälter* (Original: *Compartments*) bezeichnet. Ein Behälter stellt eine Kollaboration als Objekt dar und gibt vor, wieviele Rollen höchstens zu spielen sind. Als Beispiel für einen Behälter kann die Arbeitsgruppe in einer kollaborativen Lernumgebung dienen. Im Rahmen eines Kurses werden mehrere Arbeitsgruppen gebildet. Sie alle enthalten lediglich die Rolle *Mitglied*, welche mehrmals gespielt wird. Jede Gruppe steht für sich und hat ihren individuellen Status. Damit kann ein Studierender in mehreren Arbeitsgruppen aus verschiedenen Kursen sein. Durch das Behälterobjekt wird die Darstellung einzelner Kollaborationen zwischen Objekten vereinfacht.

Kühn wirft ein, eine Modellierungssprache sollte nicht nur Konzepte und Beziehungen beschreiben, sondern auch die domänenabhängigen Beschränkungen [9]. In vielen Sprachen sind *Relationsbeschränkungen* zu finden. Diese Einschränkungen können *kardinaler* Natur sein und die Anzahl der Beteiligten einer Beziehung begrenzen. Beziehungen zwischen 2 Objekten werden häufig mit Eigenschaften aus der Mathematik eingeschränkt. Im Beispiel der Lernumgebung ist die Beziehung „wird bewertet von“ *irreflektiv*, weswegen keine Person sich selbst bewerten kann. Beschränkungen zwischen Beziehungen besagen unter anderem, dass 2 Beziehungen einander implizieren oder ausschließen. Äquivalent bewirkt die *Rollenprohibition*, dass zwei bestimmte Rollen von einem Objekt gleichzeitig gespielt werden können. In *Rollengruppen* werden alle Rollen, die ein Objekt spielen kann zusammengefasst. Als letztes werden *Vorkommensbeschränkungen* genannt. Sie bestimmt wieviele Objekte höchstens oder mindestens eine Rolle in einem bestimmten Kontext spielen dürfen.

V. HERAUSFORDERUNGEN UND PROBLEME ROLLENBASIERTER SPRACHEN

Wie Steimann und Kühn feststellten, gibt es keine einheitliche Definition der Rolle [9] [16]. Statt einer Definition, welche immer weiter ausgearbeitet und verbessert wird, entstanden viele unterschiedliche Definitionen und Ansichten. Steimann nennt als Ursache, dass Rollen in verschiedenen Kontexten verwendet wurde und Autoren sich auf unterschiedliche Aspekte der Rollen konzentrierten [16]. Diese Ansätze überlappen sich zwar in vielen Bereichen, aber jede neue Definition führte zu einer breiteren Zerstreuung des Begriffs oder gar zu Widersprüchen. Dies erschwert es eine, Definition zu finden, die alle Merkmale umfasst. Die Widersprüche der Ansätze müssen nicht unbedingt

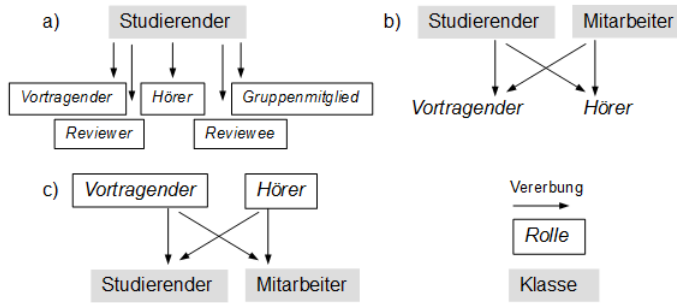


Abb. 2. Modellierung von Rollen als Oberklasse oder Unterklasse.

konzeptionell sein, dass die Merkmale zweier Definitionen einander ausschließen. Kamina und Tamais Definition [8] eines *Kontexts* unterscheidet sich stark von Deys Definition [3], aber wie Kühn feststellte, ließen sich beide Ansichten vereinen, indem lediglich eine neue Bezeichnung hinzugezogen wird.

Möglicherweise durch diese Uneinigkeit bedingt, mangelt es rollenbasierten Modellierungs- und Programmiersprachen an Unterstützung durch Software. Dadurch sind viele Ansätze nicht ohne weiteres in der Praxis anwendbar. Außerdem wurde in nur wenigen Ansätzen, das Konzept der Rolle formal festgelegt bevor es verwendet wurde.

Zhu und Alkins nennen weitere Hindernisse für rollenbasierte Programmierung [20]. Als größte Herausforderung nannten sie die Wahl des Abstraktionsgrades. Bevor eine neue Sprache entwickelt wird, muss klar sein, ob eine Rolle als Objekt, Klasse oder weder noch dargestellt wird. Außerdem sei eine Struktur zur Beschreibung einer Rolle notwendig. Sie soll ausdrücken können ob ein Objekt eine Rolle spielt und um welche es sich handelt.

Schütze und Castrillon analysierten aktuelle rollenbasierte Programmiersprachen mit Schwerpunkt auf deren Verarbeitungszeit [15].

Die Programmiersprachen ROP, Object Teams, LyRT und SCROLL haben einen Befehl ausgeführt im Kontext einer Bank, wodurch alle *CheckingAccounts* mit allen *SavingsAccounts* eine Transaktion durchführten. Im Test konnte ROP am schnellsten und in annehmbarer Zeit die 2,25 Millionen Transaktionen beenden. Object Teams war im Durchschnitt bis zu 60 mal langsamer. Dies scheint aber noch akzeptabel, im Vergleich zu LyRT und SCROLL, welche nur einen Bruchteil der Transaktionen durchführen konnten. Dabei war LyRT etwa 84.000 und SCROLL 600.000 mal langsamer als ROP. Ein weiterer Aspekt ist die Generierung von Daten während der Ausführung. Auf einer Grundlage von 250 MB produzierte ROP 365 MB an Daten und schnitt damit erneut am besten ab, während die anderen drei Sprachen Mengen von 2,8 GB bis 21,7 GB Daten generierten. Schütze und Castrillon begründen die hohe Verarbeitungszeit damit, dass die Sprachen

versuchten alle Aspekte einer Rolle zu berücksichtigen. ROP ist zwar deutlich schneller und effizienter als die anderen Ansätze, leidet aber unter dem Problem der Objektschizophrenie. Jedes Mal wenn ein Objekt eine neue Rolle annimmt, wird eine Instanz des *Rollenobjektes* erzeugt und zum *Kernobjekt* hinzugefügt. Die Identität des Objekts wird somit aufgesplittet.

VI. ROLLENBASIERTE KOLLABORATIVE LERNUMGEBUNG

In diesem Kapitel wird eine kollaborative Lernumgebung modelliert. Das System soll alle Szenarien, beschrieben im Kapitel 3A, darstellen können. Als Vorbild dient das informelle Modell einer Bank von Kühn. Es wird die Reihenfolge der umgesetzten Aspekte der Rolle übernommen.

- trennung von aufgaben/problemen - Trennung von dynamischem und statischem - dynamische veränderungen des systems - ANPASSBARKEIT ZUR LAUFZEIT - gruppen in lernumgebungen nicht vorhersehbar - Langlebigkeit - berücksichtigen Kontextabhängiges und kollaboratives Verhalten von Objekten [9] - Verhalten kann unabhängig vom Objekt sein - ermöglicht adaptives verhalten
- stärker je mehr contextwechsel -je mehr kontexte, desto mehr kontextwechsel -je mehr tasks desto mehr kontextwechsel - Kontexte: - Vorlesung - Gruppenarbeit - Offenes Forum - Reviews - Grafische Darstellung von Use Cases

VII. DISKUSSION UND FAZIT

LITERATURVERWEISE

- [1] Agustin Casamayor, Analia Amandi, and Marcelo Campo. Intelligent assistance for teachers in collaborative e-learning environments. *Computers & Education*, 53(4):1147–1154, 2009.
- [2] Silvia Dewiyanti, Saskia Brand-Gruwel, Wim Jochems, and Nick J Broers. Students' experiences with collaborative learning in asynchronous computer-supported collaborative learning environments. *Computers in Human Behavior*, 23(1):496–514, 2007.
- [3] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.
- [4] Nicola Guarino and Christopher A Welty. An overview of ontoclean. In *Handbook on ontologies*, pages 201–220. Springer, 2009.
- [5] Terry Halpin. Orm 2. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 676–687. Springer, 2005.
- [6] Rolf Hennicker, Annabelle Klarl, and Martin Wirsing. Model-checking helena ensembles with spin. In *Logic, Rewriting, and Concurrency*, pages 331–360. Springer, 2015.
- [7] Michael Hunter. The works of francis lodwick, 2012.
- [8] Tetsuo Kamina and Tetsuo Tamai. Selective method combination in mixin-based composition. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1269–1273. ACM, 2005.
- [9] Thomas Kühn, Max Leuthäuser, Sebastian Götz, Christoph Seidl, and Uwe Aßmann. A metamodel family for role-based modeling and programming languages. pages 141–160, 2014.
- [10] Frank Loebe. Abstract versus social roles—a refined top-level ontological analysis. 2005.
- [11] Stephan Murer, Carl Worms, and Frank J Furrer. Managed evolution. *Informatik-Spektrum*, 31(6):537–547, 2008.
- [12] Manuela Paechter, Brigitte Maier, and Daniel Macher. Students' expectations of, and experiences in e-learning: Their relation to learning achievements and course satisfaction. *Computers & education*, 54(1):222–229, 2010.
- [13] Trygve Reenskaug and James O Coplien. The dcj architecture: A new vision of object-oriented programming. *An article starting a new blog:(14pp)* http://www.artima.com/articles/dcj_vision.html, 2009.
- [14] HFD Winter School. Hochschule im digitalen zeitalter gestalten, 2 2018.

- [15] Lars Schütze and Jeronimo Castrillon. Analyzing state-of-the-art role-based programming languages. In *Companion to the first International Conference on the Art, Science and Engineering of Programming*, page 9. ACM, 2017.
- [16] Friedrich Steimann. On the representation of roles in object-oriented and conceptual modelling. *Data & Knowledge Engineering*, 35(1):83–106, 2000.
- [17] Naoyasu Ubayashi and Tetsuo Tamai. Roleep: role based evolutionary programming for cooperative mobile agent applications. In *Principles of Software Evolution, 2000. Proceedings. International Symposium on*, pages 232–240. IEEE, 2000.
- [18] Charles W. Bachman and Manilal Daya. The role concept in data models., 01 1977.
- [19] Haibin Zhu. Role mechanisms in collaborative systems. *International Journal of Production Research*, 44(1):181–193, 2006.
- [20] Haibin Zhu and Rob Alkins. Towards role-based programming. In *Workshop on Role-Based Collaboration, CSCW*, 2006.