Virtuelle Labore in der Mathematikausbildung: Konzepte und Implementierung

Thomas Richter
TU Berlin
thor@math.tu-berlin.de

Sabina Jeschke
TU Berlin
sabina@math.tu-berlin.de

Ruedi Seiler
TU Berlin
seiler@math.tu-berlin.de

Abstract:

In den letzten Jahren ist die Bedeutung computerunterstützter Darstellungen von Mathematik im Lern- als auch im Forschungsbereich stark gestiegen. Obwohl bereits massive Anstrengungen unternommen werden, die Mathematikausbildung im Ingenieursbereich durch Neue Medien zu unterstützen und auszubauen[Mum], gibt es dennoch nur eine geringe Zahl von Projekten, die auf die Studenten der Mathematik und Physik abzielen. Ferner beschränkten sich viele Projekte nur auf die Verwaltung von Dokumenten, die den Lernenden — obgleich teilweise aufgelockert durch eingestreute aktive Inhalte — zu einem passiven Konsumenten des Lerninhaltes machen und die darum selten geeignet sind, die eigentständige, selbstgesteuerte Auseinandersetzung mit der Materie zu fördern.

Wir präsentieren in diesem Artikel das Konzept des „Virtuellen Labors“, welches die Metapher eines Laborpraktikums innerhalb eines Computernetzwerkes nachbildet und damit das zweite Standbein der universitären Ausbildung in die Neuen Medien abbildet. Das Labor „Cinderella“ [KRG04] zur Untersuchung euklidischer und nicht-euklidischer Geometrie mag hier als ein Beispiel dienen.

Wir werden im ersten Teil ein didaktisches Anforderungsprofil an derartige Labore definieren, dann in einem zweiten Teil die sich daraus untersuchenden Konsequenzen für die Software-Architektur darstellen und im dritten Teil ein am DFG-Forschungzentrum MATHEON der Berliner Universitäten entwickeltes Labor für Elemente der statistischen Mechanik präsentieren.

1 Hintergrund

Das Arbeitsfeld des Ingenieurs, aber auch das des Wissenschaftlers oder Mathematikers ist im Wandel begriffen: Numerische Software und Computer-Algebra-Systeme (CAS) entlasten von aufwendigen Routine-Rechnungen, dafür werden Fähigkeiten wie das schnelle Erfassen und Lernen neuer Methodiken zunehmend bedeutsam. Auch durch den rasanten Wissenszuwachs ist heute ein allumfassendes „Lernen auf Vorrat“ wenig zeitgemäß, stattdessen müssen Lehr- und Lernmethoden vermittelt werden, die auf das effiziente eigenständige Lernen abzielen.

Dieser Anforderung werden die traditionellen Lernmethoden in der universitären Ausbildung nur bedingt gerecht: Frontalunterricht stellt notwendiges Basiswissen bereit, ermöglicht aber kaum eine aktive eigenständige Auseinandersetzung mit der Materie. Klassische Praktika dagegen zielen zwar auf die experimentierenden, selbstgesteuerten Erkenntnisgewinn ab, stossen aber in der Universitätsrealität schnell an Grenzen, die aus den hohen Teilnehmerzahl sowie den Zugangsmöglichkeit und den Ausstattungen der Laboreinrichtung resultieren. Dazu kommt, dass jede experimentelle Auseinandersetzung in “realen” Laboren mit Wissensgebieten auf Experimentaldisziplinen beschränkt ist, die theoretischen Gebiete – Mathematik, theoretische Physik, theoretische Bereiche der Ingenieurwissenschaften – dabei aber ausgeschlossen bleiben oder nur peripher in die Betrachtungen mineinbezogen werden.

Der Einsatz von Neuen Medien und Neuen Technologien im Unterricht stellt deshalb einen Wendepunkt dar. Unter den breiten Möglichkeiten, die eLearning-, eTeaching und eResearch (zusammengefaßt: eLTR-Technologien) uns eröffnen, nehmen *Virtuelle Labore* eine besondere Rolle ein:
Virtuelle Labore sind Umgebungen, die realen Laboren nachempfunden sind, und in denen – computergestützt – Experimente entworfen, "aufgebaut" (erstellt), durchgeführt und ausgewertet werden können. Dabei werden Experimente nicht an realen Aufbauten, sondern an rechnerimplementierten Algorithmen realisiert. Diese Algorithmen bilden dabei entweder reale Geräte und Gegenstände oder aber theoretische Konzepte und Objekte ab (vergl. Kap. 4).

Mit Hilfe multimedialer Techniken können solche experimentellen Lernumgebungen orts- und zeitunabhängig jedem Lehrenden und jedem Studierenden uneingeschränkt zur Verfügung gestellt werden. Experimente, die aus Sicherheits-, Kosten- oder Platzgründen bisher unmöglich bzw. unzugänglich waren, werden realisierbar. Beim Nachbau von Experimenten aus den experimentellen Disziplinen stellen Virtuelle Labore eine wichtige Ergänzung zu "realen" Laboren dar, weil sie eine prägnantere Ausarbeitung des eigentlichen "Phänomens" erlauben und messtechnische Probleme in den Hintergrund rücken. Da der Umgang mit den Gerätschaften und den hierbei entstehenden Problemen aber ein Teil der zu erwerbenden Kompetenz darstellt, darf in experimentellen Gebieten auf reale Laborexperimente nicht verzichtet werden. In den theoretischen Disziplinen schließlich ermöglichen sie überhaupt erst den experimenteller Zugang zu diesen Wissensgebieten, und machen abstrakte Phänomene erst dadurch plastisch verständlich.

Wir entwickeln in diesem Paper zunächst ein detailliertes Anforderungsprofil an ein virtuelles Labor (Kap. 2). Die daraus resultierenden Konsequenzen für eine Implementierung werden beschrieben (Kap. 3). Darauf besierend skizzieren wir eine prototypische Implementierung eines virtuellen Labors für Statistische Mechanik (Kap. 5). Anhand des klassischen Beispiels des Ising-Modelles stellen wir anschließend dar, wie sich ein statistisches Modell als eine konkrete Anwendung des Labors formulieren lässt (Kap. 6). Im letzten Kapitel geben wir einige Ausblicke in die weitere Entwicklungen Virtueller Labore (Kap. 7).

2 Didaktisches Anforderungsprofil

Wir werden im folgenden Kapitel die wesentlichen didaktischen Anforderungen an ein virtuelles Labor zusammenstellen; eine unserer Meinung wesentliche Eigenschaft von derartigen Laboren ist, dass sie aus sich heraus keine „Lernziele" im Sinne von traditionellen Lerneinheiten definieren. Dies bleibt weiterhin dem Leiter einer Vorlesung, eines Seminars oder eines Praktikums überlassen. Sie sind vielmehr Werkzeuge zur Umsetzung einer qualititativ hohen Lehre im Rahmen einer derartigen Veranstaltung.

Die hier formulierten Anforderungen bauen auf den Anforderungen an eines „Next Generation" eLearning-Environments auf, wie sie in [Jes04] entwickelt wurden:

- **Vermittlung von Handlungskompetenz:**

  Das Ziel eines Laboreinsatzes in der Ausbildung, sei es nun ein reales oder ein virtuelles Labor, besteht nicht darin, fertig maßgeschneiderte Problemlösungen anzubieten. Vielmehr soll ein Labor die notwendigen Geräte, bzw. im Falle virtueller Labore Algorithmen bereitstellen, die eine selbständige Entwicklung und Erprobung von Problemlösungsstrategien ermöglichen und dabei die typischen Problemstellungen in Mathematik, Physik und Ingenieurwissenschaften berücksichtigten, um auf das spätere Berufsleben in diesen Bereichen vorzubereiten. Handlungskompetenz lässt sich durch praktische Erfahrung entschieden besser fördern als durch Frontalunterricht.
- **Unterstützung selbstgesteuerten Lernens:**
  Labore bieten Studenten die einzigartige Möglichkeit, den Lernvorgang selbstständig ohne äußere Instanzen zu steuern und somit Entscheidungen über den Lernprozess selbsttätig treffen zu können. Somit

bereitet der Laboreinsatz auf das später im Berufsleben geforderte lebenslange, kontinuierliche Lernen vor.

Wir gliedern die Unterstützung selbstgesteuerten Lernens in die folgenden Unterpunkte:

- **Unterstützung explorativen Lernens:**
  (Virtuelle) Labore ermöglichen ihren Benutzern ungebundenes und effizientes Arbeiten mit den Gerätschaften, um so Sachzusammenhänge selbständig zu erkunden und ein intuitives Verständnis des Fachgebietes aufzubauen. Ein Labor muss also Freiräume schaffen, die zum individuellen Experimentieren auch außerhalb der Schranken eines Curriculums einlädt; ungewöhnliche Anordnungen, Auswahlen, zeitliche Abläufe etc. sind erlaubt und erwünscht. Auf diese Weise werden Fähigkeiten zu selbstbestimmten, lebenslangen Lernen trainiert. Ein Labor vermittelt also Kompetenzen außerhalb seines Faches, indem es „Anleitung zum Lernen“ gibt.
- **Adaption an individuelle Lernstile:**
  Der Support unterschiedlicher Lernstile ist eines der herausragendsten Eigenschaften des Einsatzes multimedialer Technologien in der Lehre. Allerdings nutzen längst nicht alle eLTR-Technologien dieses Potential tatsächlich: Die starren, linearen, kaum interaktiven Kurse der ersten Generation der eLearning-Technologien [JKS04b] erlauben keine individuelle Erarbeitung des Stoffes. Ebenso passen sich vorgefertigte Experimente nicht an das Vorwissen seiner Benutzer, fachlich eng eingegrenzte Umgebungen nicht an unterschiedliche Interessen und Lernziele an und fördern somit nicht die Motivation der Benutzer. Gerade diese Anforderungen jedoch müssen bei der Entwicklung virtueller Labore – mehr noch als bei an allen anderen eLearning-Umgebungen – im Vordergrund stehen.

• **Multiple Einsatzszenarien:**

Labore sollten idealerweise an verschiedene Einsatzszenarien adaptierbar sein. Dies betrifft einerseits den Einsatz in verschiedenen Studienrichtungen, die Schwerpunkte jeweils fachspezifisch anders setzen, und andererseits die Möglichkeit, Labore sowohl zu Demonstrations- Übungs- und Prüfungszwecken einsetzen zu können.

Ein Labor stellt mit seiner Einrichtung zunächst nur verschiedene Einsatzmöglichkeiten bereit, bestimmt diese aber nicht. Durch die hiermit gewonnene Selbstständigkeit und gestalterische Freiheit des Lernenden unterstützt man die Entwicklung der Problemlösekompetenz des Experimentators. Aus diesem Grund sollte ein virtuelles Labor nicht auf einen nicht-erweiterbaren Satz von Experimenten, eine einzige Vorlesung, eine bestimmte Zielgruppe hin ausgerichtet sein; je nach Zielgruppe ergeben sich verschiedene Anforderungsprofile, die wir nach dem Grade aufsteigender Komplexität und Selbstständigkeit der Nutzen beschreiben:

- **Demonstrationsszenario:**
  Im “Demonstrationsmode” werden fertig vorkonzeptionierte Experimente, die nur mit minimalen Benutzerschnittstellen zur elementaren Steuerung ausgestattet sind, oft sogar automatisiert oder semiautomatisiert frontal präsentiert. Ein derart eingesetztes Labor bietet kaum Möglichkeiten zur explorativem Umgang und ist zur Vertiefung der Materie schlecht geeignet, da selbstständige Denkprozesse nur in ungenügendem Maße gefordert werden. Dennoch lassen sich hiermit attraktive Einführungen in ein Fachgebiet geben, deren vordringliches Ziel es ist, beim Betrachter Neugierde auf das entsprechende Fachgebiet zu wecken und Motivation und Anregungen für eigenständige Arbeiten zu geben. Demonstrationsszenarien spielen eine zentrale Rolle gerade außerhalb der reinen Lehrveranstaltungen, etwa bei Präsentationen für Schülergruppen zur Vorstellung von Fachgebieten, etwa in Web-Auftritten o.ä.

  Die Entwicklung des Lernenden wird durch derartige Demonstrationen nur indirekt gefördert, indem durch die Darstellung von ansprechenden Experimenten die Neugierde auf eigenständiges Lernen durch Experimentieren erschlossen wird.
- **Experimente in der Präsenzlehre:**

  In der Präsenzlehre werden Experimente in Virtuellen Laboren analog zu etwa Experimenten in der experimentellen Physik vorgeführt — wobei die Experimente nun sowohl experimentelle als auch theoretische Bereiche umfassen. Experimente im Theoretischen Bereich würden etwa abstrakte Objekte aus dem jeweiligen Fachgebiet durch Visualisierung erfassbar machen und ihre Eigenschaften durch entsprechende Simulation ihres Verhaltens darstellen, beispielsweise kann das Lösungsverhalten von Differentialgleichungen unabhängig von ihrer Realisierung als ein physikaler Prozess untersucht werden.

  Im Unterschied zum reinen Demonstrationsszenario wird hier eine größere Flexibilität, etwa hinsichtlich der Vielfalt an regelbaren Parametern, schnellen Umbau oder einfacher Erweiterung der Versuchsanlage, notwendig. Damit gewinnt aber auch die Qualität der Benutzerführung an Bedeutung.

  Im Vorlesungsbetrieb unterstützen derartige Experimente die Kompetenzentwicklung des Lernenden etwa durch die Verknüpfung von abstrakten Gleichungen und des konkreten Verhaltens seiner Lösungen. Oftmals bauen abstrakt-mathematische Formulierungen Barrieren bei Studierenden auf, die durch Betrachtung von geeignet gewählten Visualisierungen überwunden werden können.

  - **Praktikum und Tutorien:**
    Ein Haupteinsatzgebiet für virtuelle Labore als eLearning-Software der zweiten Generation [JKS04a] besteht in der Durchführung selbständiger Übungen, Projekte und Praktika, die durch Mitglieder des Lehrkörpers unterstützt werden. Eine gute Benutzerführung ist nun entscheidend, um kurze Einarbeitungszeiten in die Laboroberfläche zu gewährleisten. Teil-fertige Aufbauten können, je nach der Komplexität der zu vermittelnden Materie die Zugangsbarrieren abbauen und Anregungen für weitere Experimente schaffen; das Labor muss jedoch einen vollständigen Zugang zu allen Laborfunktionalitäten ermöglichen.

    Für den Studierenden beteutet dies, durch Anleitung und Hilfestellungen durch einen Übungsleiter den Umgang mit den Laboraufbauten zu erlernen um somit die Voraussetzungen für selbstgesteuertes Lernen zu schaffen.

  - **Selbststudium, Hausaufgaben, Klausurvorbereitung:**

    In diesem Einsatzgebiet wird vom Studierenden ein vollständig eigenverantwortliches Arbeiten eingefordert und somit selbstgesteuertes Lernen trainiert. Im Gegensatz zum vorgenannten Einsatz in Rahmen von Praktika und Tutorien ist kein unterstützendes Lehrpersonal verfügbar, Arbeitszeit und -raum sind komplett universitätsunabhängig. Es ist aber auch in diesem Einsatzgebiet wünschenswert, dass das Labor mehrbenutzerfähig ist, um so etwa Kooperationen innerhalb einer Hausaufgaben-Arbeitsgruppe auch außerhalb der Universität zu unterstützen.

  - **Einsatz in der Forschung:**
    Hier werden in weitaus größerem Maße Flexibilität und vor allem die Integration und Vernetzung des Labors mit anderen Softwarekomponenten und sonstiger bestehender Infrastruktur gefordert. Eine gute Benutzerführung ist zwar stets hilfreich und wünschenswert, von der hier angenommenen Benutzergruppe dürfen jedoch gute Vorkenntnisse und ggf. auch eine gewisse Einarbeitungsphase eingefordert werden. Von entscheidender Bedeutung sind hier Komplexität und fachliche Qualität der Algorithmen, die die Frage der Eignung des Labors zur Modellierung relevanter Problemklassen bestimmen.

    In diesem Einsatzgebiet ist ein virtuelles Labor kein Instrument zur Unterstützung der Lehre, sondern vielmehr ein Werkzeug zur Forschung.

  Idealerweise sollte ein virtuelles Labor alle oben genannten Einsatzzwecke addressieren; wir werden im zweiten Kapitel erarbeiten, welche Konsequenzen diese Forderung für die Architektur eines Labors hat.

- **Unterstützung kooperativer Lehr- und Lernszenarien:**
  Wissenschaftliche Leistungen wie Ingenieurleistungen sind im wachsenden Maße Resultate von verteilter Teamarbeit; dies ist bedingt durch die steigende Komplexität der zu lösenden Probleme, die erst durch die Verknüpfung von Experten verschiedener Ausrichtungen effizient angegangen werden können. Teamorintierte Projektarbeit muss deshalb auch Bestandteil der Ausbildung sein. Die Realisierung kooperativer Lernszenarien – insbesondere auch über geographische Distanzen hinweg – ist deshalb eine wichtige Anforderung an das Design Virtueller Labore.

- **Vernetzung mit bestehender Software und Infrastruktur:**
  Ein virtuelles Labor sollte als Teil der von ihm bereitgestellten Infrastruktur auch Standardkomponenten aus dem Alltagseinsatz der jeweiligen Zielgruppe enthalten; dies beinhaltet etwa Produkte wie Maple,

Mathematica oder Matlab, die aus dem täglichen Einsatz des arbeitenden Mathematikers, Physikers oder Ingenieurs nicht mehr fortzudenken sind. Labore müssen also geeignete Schnittstellen anbieten, die eine Einbindung dieser Standardkomponenten zulässt; entsprechende Aufbauten sollten dementsprechend diese Komponenten einsetzen, da der Umgang und der Gebrauch dieser Komponenten Teil des Lernumfanges eines Laboreinsatzes sein soll. Von besonderer Wichtigkeit ist diese Forderung einerseits für das Einsatzziel eines Labors in der aktiven Forschung, in der mit hochspezialisierter Software gewonnene Daten weiterverarbeitet oder das Labor gesteuert werden muss, sowie – in anderer Richtung – für den Einsatz als Demonstrationsobjekte im öffentlichen Raum, für den man Installations- und Nutzungsaufwand minimal halten will.

- **Wiederverwertbarkeit:**
  Laborkomponenten sollten durch Einsatz offener Schnittstellen aus ihrem Laborkontext herauslösbar und damit wiederverwertbar sein. Offene Schnittstellen werden durch Einsatz von offenen, standardisierten Werkzeugen (CORBA,Java,HTML) statt propriertären Lösungen erreicht (Word,PowerPoint,...); ferner ist die Spezifikation der verwendeten Schnittstellen auch vom Labor selbst einzufordern.

  Diese Forderung erlaubt nicht nur das effiziente Zusammenstellen neuer Labore und vermeidet so Parallelentwicklungen und unnötigen Resourcenverbrauch bei der Erstellung eines Labors, sie fördert gleichzeitig durch die Notwendigkeit der Bereitstellung offener Schnittstellen konzeptionell saubere, und damit einfach einzugliedernende Laborbestandteile.

  Derartig aufgebaute Schnittstellen erleichtern auch die Integration des Labors in komplexere Experimente, die der Hilfe von weiteren Softwarekomponenten bedürfen, und erlauben ebenso die Zusammenstellung neuartiger Experimente aus vorgefertigten Komponenten abseits eines geplanten Einsatzgebietes.

  Durch die Zusammenstellung bereits verfügbarer Laborkomponenten mit neuen Bausteinen wird beim Lernenden darüberhinaus bekanntes mit unbekanntem verknüpft und Barrieren beim Übergang in zu erlernende Fachgebiete vermieden.

3 Konsequenzen für die Implementierung

Aus den den Kap. 2 genannten didaktischen Ansprüchen an ein Virtuelles Labor resultieren verschiedene Anforderungen an das Softwaredesign:

- **Granularität des Softwaredesigns:**
  Um unterschiedliche und vielfältige Einsatzszenarien möglichst uneingeschränkt unterstützen zu können, müssen die einzelnen Komponenten flexibel und kreativ kombiniert werden können. Das führt auf ein "strikt anti-monolithisches", fein-granulares Softwaredesign, das in seiner Grundstruktur durch die folgende Dreiteilung charakterisiert ist:
  - **Simulations- und Rechenbausteine:**
    Diese Programmklassen erbringen die eigentliche Simulations- und Rechenarbeit, die zur Durchführung eines Experimentes erbracht werden muss. Sie kommunizieren durch ihre – für den Nutzer nicht sichtbaren – Schnittstellen mit der Außenwelt.
  - **Konnektoren:**
    Konnektoren sind Softwarekomponenten, die der Verknüpfung der Laborbestandteile, etwa beim Aufbau eines Experimentes, dienen. Sie können durch einfache Skripte realisiert werden, die die Meßwerte eines Simulationsbausteines automatisch in ein Algebra-Programm übertragen, aber auch durch komplexe graphische Oberflächen zur Verknüpfung von Einzelbestandteilen wie etwa Oorange [Oor]. Obwohl im Idealfall virtuelle Labore einheitliche Schnittstellen aufweisen sollten, ist in der Praxis eine derartige Uniformität unrealistisch; Konnektoren sollten daher insbesondere Bausteine einer heterogenen Softwarearchitektur miteinander verknüpfen und es so erlauben, Laborkomponenten unterschiedlicher Herkunft miteinander zu einem Experiment zu verknüpfen.
  - **Oberflächen:**
    Benutzeroberflächen stellen die Verbindung zur jeweiligen Zielgruppe des Labores her, etwa Hörer

einer Vorlesung, Teilnehmer eines Praktikums oder forschende Wissenschaftler. Entsprechend den Anforderungen des Einsatzgebietes können diese von komplexen Befehlszeilen bis zu einfachen Applets in einem Web-Browser reichen. Um an verschiedene Einsatzgebiete anpassbar zu sein, werden i.a. mehrere Benutzeroberflächen nötig.

Aus der geforderten Granularität folgt also eine Trennung von Laborbausteinen, Kompositionsmechanismen und Oberflächenfunktionalitäten, so dass sich die Notwendigkeit von geeigneten Schnittstellen zwischen diesen einzelnen Teilen ergibt, die auch wieder durch eine entsprechende Software vermittelt oder aufgebaut werden muss.

- **Verwendung offener Standards, Offenlegung von Schnittstellen:**
  Um den individuellen Anforderungen einer breitgefächerten Nutzergemeinschaft gerecht werden zu können, darf in einem virtuellen Labor nicht der Versuch unternommen werden, alle Anwendungsfelder durch ein einziges Softwarepacket abzudecken. Ein monolithischer Entwurf würde entweder einige Anwendungsfelder nur unzureichend abdecken, oder wäre für einen Einsatz in der Ausbildung zu komplex: Um explorativen Zugang zu Wissengebieten zu ermöglichen, ist die Offenheit der Software zu weiteren, bestehenden Tools wie numerischer Software, ComputerAlgebraSystemen oder alternativen virtuellen Laboren von entscheidenden Bedeutung. Das führt auf hohe Anforderungen hinsichtlich der Schnittstellendefinition und auf die unbedingte Notwendigkeit der Nutzung bzw. Weiterentwicklung bestehender Standards sowie der Offenlegung der verwendeten Schnittstellen[1] mit Konsequenzen an die Offenheit der gesamten Software und - wichtig! – ihrer Dokumentation.

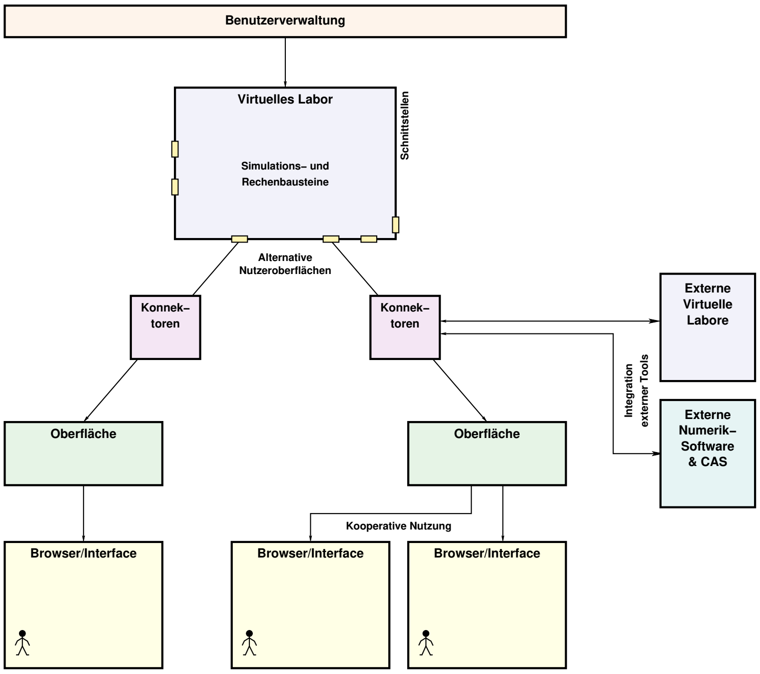

Abbildung 1: Abstraktes Softwaremodell eines Virtuellen Labors

- **Unterstützung von Netzwerkapplikationen:**
  Kooperative Lernformen in Virtuellen Laboren beinhalten insbesondere, dass mehrere Benutzer von verschiedenen Arbeitsplätzen aus gleichzeitig und mit Kenntnis der Aktionen der jeweils anderen an einem Versuchsaufbau arbeiten. Damit entsteht die Notwendigkeit, das Labor als eine mehrteilige Netzwerkapplikation zu entwerfen: Experimente können etwa auf einem Server koordiniert werden, auf den die Studenten als Benutzer gemeinsam zugreifen (Client-Server-Ansatz); symmetrische Lösungen, d.h.

[1] Idealerweise sollten die entstandenen Erfahrungen in eine Standardisierungsgruppe eingebracht werden, um so die weiträumige Kopplung von Laborkomponenten verschiedener Herkunft gewährleisten zu können. Die Notwendigkeit, Simulationsbausteine verschiedenster Herkunft miteinander verknüpfen zu müssen, um etwa Standardbausteine wie Algebrasoftware in einem Labor einsetzbar zu machen, macht eine heterogene Verknüpfung der Bestandteile notwendig – inwieweit eine solche Standardisierung auch über Fachgrenzen hinweg möglich ist jedoch derzeit noch Gegenstand der Forschung.

ohne Herausstellung eines bestimmten Rechners als Server sind ein alternativer Ansatz zur Lösung des Koordinationsproblemes (Peer-to-Peer-Methode).

Zwei kurze Bemerkungen wollen wir zum Abschuss dieses Abschnitten noch machen:

Folgt man der in Kap. 2 skizzierten Argumentation zur Zielsetzung des Einsatzes Virtueller Labore, so wirkt der oben skiziierte Ansatz scheinbar "kanonisch". Deshalb soll hier explizit festgestellt werden, dass heute nur eine Minorität der Software, die im weitesten Sinn den Bereichen eLearning & eTeaching und/oder eResearch & eScience zuordnen sind, einem solchen granularen, offenen und verteilten Ansatz folgt. Bestehende Virtuelle Labore verfügen meist über engen fachlichen Fokus auf ein bestimmes (Teil-)Gebiet und realisieren dieses mittels eines monolithischen Designansatzes – inhaltliche Öffnungen sind nur sehr schwer möglich. Betrachten wir z.B. Computer-Algebra-Systeme wie "Maple" oder "Mathematica", so stellen wir hier eine Verschränkung zwischen in dem Kern (den Simulations- und Rechenbausteinen) und der Oberfläche dar (was wiederum die Integration dieser wichtigen Tools in andere fachspezifische Software erheblich erschwert), die "Konnektoren" werden gegenüber den Nutzer nur im prozeduralen Fluss, in der Ergbnisübergaben von einer Operation zu einer weiteren, sichtbar.

Die zweite Bemerkung betrifft den Aspekt der Nutzerführung und der "Usability" solcher Labore: Die oben beschriebene, flexible granulare Struktur der Software führt unvermeidlich auf eine komplexere Benutzersteuerung und damit auf höhere Einarbeitungszeiten – auf Seiten von Lernenden wie Lehrenden. Die Realisierung einer Umgebung mit vielfältigen Möglichkeiten und Potenzialen für die Verbesserung von Motivation, Verständnis und Lernprozess hat ihren Preis, selbst bei idealem Usersupport. Umgekehrt aber stellen gerade Startschwierigkeiten und technische Anfangsprobleme einen prominenten "Motivationskiller" im eLearning dar – hier einen geeigneten Kompromiss zu finden ist im Einzelfall nicht einfach; ein möglicher Ausweg liegt in der Entwicklung verschiedener, auf unterschiedliche Benutzergruppen zugeschnittenen Oberflächen wie wir ihn i.f. für das Labor VideoEasel darstellen werden.

4 Das Virtuelle Labor VideoEasel

Das am DFG Forschungszentrum MATHEON erstellte virtuelle Labor VideoEasel stellt eine Implementierung eines eLearning-Tools der zweiten Generation dar. Der fachliche Schwerpunkt dieses Labors liegt in einer ersten, inhaltlich prototypischen Phase im Bereich der statistischen Mechanik und verwandter Gebiete. Die Modellbildung erfolgt über zellulären Automaten, die einerseits hervorragend für die Modellierung statistischer Modelle geeignet sind (ausgehend von Gittergasmodellen bis hin zu Fragen der Turing-Vollständigkeit sind viele interessante Themen abgedeckt), andererseits aber eine grosses Potential für die Modellierung anderer Fragestellungen eröffnen. Unsere Wahl für das Fachgebiet einer prototypischen Implementierung fiel deshalb auf die Welt der zellulären Automaten, weil diese einerseits einfach genug ist, als dass mit elementaren Mitteln der Schulmathematik die zugrundeliegende Konzepte erfasst werden können, andererseits die entstehenden Effekte reichhaltig in Anwendung und Vielfältigkeit sind. Das Fachgebiet lädt darum wie kein weiteres zum Experimentieren ein, ohne hierbei durch große fachliche Hürden Barrieren aufzubauen. Anwendungen für derartige Automaten finden sich daher sowohl in den mathematischen Grundvorlesungen — wie etwa partielle Differentialgleichungen — in der aktiven Forschung — statistische Methoden der Bildkompression — oder in mathematischen Spielen. Somit wird einer großen Nutzerbasis der Zugang VideoEaseleröffnet.

Im Folgenden gehen wir auf einige konkreten Auswirkungen des oben erarbeiteten Anforderungsprofils auf die Software ein:

- **Granularität:**
  Entsprechend unserer Forderung nach Wiederverwertbarkeit und Granularität liegen im Labor VideoEasel die Experimente als kleine, bausteinartige Einheiten vor, die nicht fest in den Laborkern hineincompiliert wurden, sondern je nach Bedarf erstellt und hinzugeladen werden können. Diese Einheiten

gliedern sich in zwei Gruppen von Core-Klassen, siehe Bild 4, nämlich einerseits „Automaten“ zur algorithmischen Definition eines physikalischen Phänomens, und „Messtools“ zu deren messtechnischer Erfassung. VideoEasel bietet elementare Methoden der Messwert-Auswertung, stellt aber weder nume-

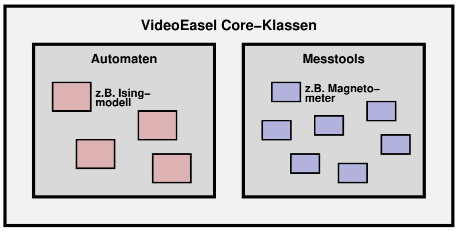

Abbildung 2: Coreklassen VideoEasel

rischen Tools für komplexere Auswertungen noch eine eingebaute Prozesssteuerung für aufwendigere Experimente zur Verfügung. Derartige Aufgaben werden von darauf spezialisierten Werkzeugen übernommen, numerische Auswertung etwa von Algebra-Software wie Maple, Prozesssteuerung etwa von Skript-Sprachen wie Python.

- **Netzwerkapplikation:**
  VideoEasel ist eine klassische mehrteilige Netzwerkapplikation, untergliedert in einen Server und mehrere verschiedenartige Klienten. Der Server ist als eine C++ Applikation ausgeführt, die die Simulationen/Berechnungen der Modellsysteme durchführt und die Animationen und Messresultate an die auf den Arbeitsplatzrechnern laufenden Klienten versendet. Im einfachsten Fall – etwa beim Einsatz auf einem Hörsalrechner zur Unterstützung der Vorlesung – sind Server und Client-Rechner identisch, bei kooperativen Lernszenarien synchronisert der Server mehrere Clients.
- **Schnittstellen zur Eingliederung in bestehende Infrastruktur:**
  Im Labor VideoEasel werden die Schnittstellen zu Standardwerkzeugen, die zur weiteren numerischen Auswertung der Resultate und zur Steuerung des Labors dienen, über die Middleware CORBA [Sca] in der IDL-Sprache realisiert. IDL-Mappings sind für diverse andere Sprachen, etwa Java, C und Python vorhanden, so dass eine Anbindung an viele externe Tools einfach möglich ist. Derzeit sind neben den nativen Java-Oberflächen eine Python-Anbindung zur Skriptsteuerung und eine C-Implementierung einer Maple-Anbindung verfügbar, so dass sich problemlos Daten aus dem Labor heraus und in das Labor hineinbewegen lassen. Ferner wird diese CORBA-Schnittstelle zur Ansteuerung des Labors über diversen graphische Oberflächen verwendet.
- **Wiederverwertbarkeit:**
  Der Wunsch nach Wiederverwertbarkeit von Laborelementen, die eine Zusammenstellung von bereits verfügbaren Laborelementen in diverse Labore erst ermöglicht, schließt eine monolithische Software-Architektur aus. Ein Messtool ist hier beispielsweise für eine Vielzahl von Experimenten wiederverwertbar, ebenso wie Klassen zu in der Oberfläche liegende Visualisierung der Messresultate. Aufgrund der Neuartigkeit des Ansatzes liegen noch keine Erfahrungen bezüglich der Wiederverwertbarkeit von Komponenten außerhalb von VideoEasel vor.
- **Einsatzszenarien:**
  Um den in Kapitel 2 entwickelten Einsatzzielen gerecht zu werden, müssen auf den jeweiligen Zweck zugeschnittene Oberflächen bereitgestellt werden. Eine vollwertige Laboroberfläche ist ungeeignet für eine kurzweilige Demonstration und ein Browser-Applet ebenso für den Forschungseinsatz. Die Wahl des Einsatzszenarios bedingt somit bei VideoEasel die Wahl der Oberfläche. Da alle Oberflächen nur über die dokumentierte CORBA-Schnittstelle auf den Laborkern — den Server – zugreifen, lassen sich auch diverse Clients auf verschiedenen Arbeitsplatzrechnern kombinieren. Es werden aktuell die folgenden User-Clients zur Verfügung gestellt:
  - **Demo-Applet (DemoMode):**
    Zur Durchführung von speziellen Demonstrationsexperimenten und zur Außendarstellung wurde ein konfigurierbares Java-Applet entwickelt, welches nur wenig Interaktion mit dem Benutzer erfordert, allerdings auch nur begrenzte Möglichkeiten zur Interaktion mit dem Labor darstellt. Es lässt sich

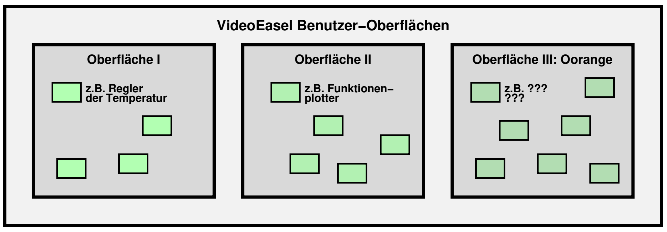

Abbildung 3: Verschiedene Benutzeroberflächen VideoEasel

auf einfache Weise in eine HTML-Seite einbinden und erfordert zur Durchführung der Experimente nur wenig Anleitung, die etwa durch Fließtext auf einer Web-Page erbracht werden kann. Entsprechende Labor-Setups werden Server-seitig gehalten und über ihren Dateinamen auf der Web-Seite referenziert. Abb. 4 zeigt einen entsprechenden Web-Auftritt mit einem einfachen Experiment zur Motivation des Zweiten Hauptsatzes der Thermodynamik; ein einzelner Parameter zur Steuerung der Zeitrichtung ist hier herausgeführt und im Applet verfügbar gemacht. Die Einflussnahmen auf das Labor sind auf die typischen Funktionalitäten eines Zeichenprogrammes (Variation der Geometrie des Experiments etc.) beschränkt.

Das didaktische Ziel beim Einsatz dieser Oberfläche liegt in dem in Kapitel 2 dargestellten „Demonstrationsszenario“, da Experimente unkompiliziert fertig aufgebaut on-line bezogen und durchgeführt werden können.

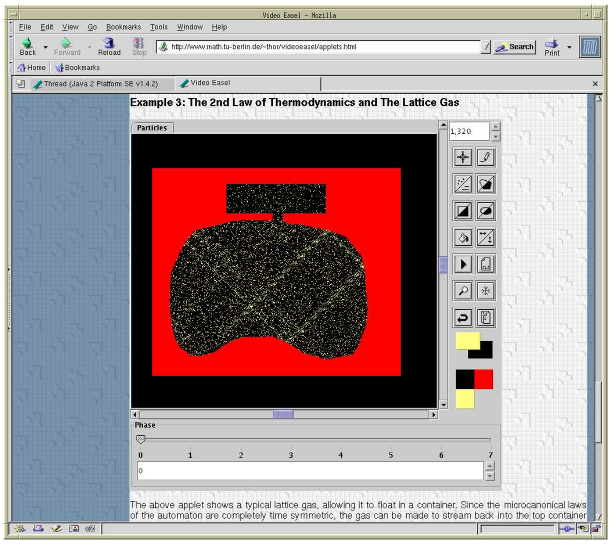

Abbildung 4: VideoEasel Java-Applet in Web-Auftritt, Experiment zum 2. Hauptsatz

– **Java-Application (SimpleExperiments):**
Für kleinere, vorgefertigte Experimente wurde eine Java-Applikation erstellt, die die Applet-Oberfläche zu großen Teilen nachbildet, aber weitaus mehr Möglichkeiten zum Eingriff in die Experimente bietet. Der Experimentalaufbau ist hier, ähnlich wie in einem Praktikum, bereits serverseitig vorgegeben, Parameter zur Einflussnahme auf den Ablauf des Experiments sowie einfache Schnittstellen zur Datenaufnahme stehen jedoch bereit. Aufgrund der geringen Komplexität dieser Oberfläche lassen sich unkompliziert und rasch Praktika erstellen. Abb. 5 stellt einen Screen-Shot dieses Front-Ends dar, hier exemplarisch mit einem weiteren Fenster zur Einstellung eines Parameters. Entsprechende Menüeinträge erlauben die Adaption weiterer Parameter des Experimentes. Die Zeichenoberfläche hingegen ist zum Großteil identisch zum Applet.

Diese Oberfläche ist auf den Einsatz in der Präsenzlehre und für einfache Experimente in Praktika und Tutorien hin optimiert. Die Benutzerführung ist auf einfachen Zugang hin ausgelegt, Eingriffe in die Zusammenstellung von Algorithmen zu einem Labor sind aus Gründen der Übersichtlichkeit nicht oder nur in begrenztem Umfang möglich. Elementare Methoden zum Export aufgenommener Messwerte sind vorhanden, die zum Anfertigen von Hausaufgaben oder Praktikumsaufgaben ausreichen, ohne die Zielgruppe durch technische Details zu sehr von der eigentlichen Messaufgabe abzulenken.

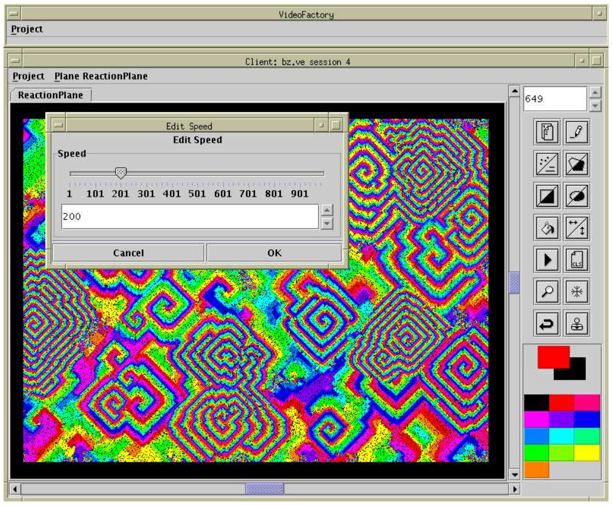

Abbildung 5: VideoEasel Java Front-End mit einem Experiment zur Belousov Zhabotinsky Reaktion

– **Oorange (ResearchScenarios):**
Eine verfeinerte, und vollständige Schnittstelle wird durch das auch an der TU-Berlin entwickelte Oorange Development Tool [Oor] realisiert. Es erlaubt die rein graphische Erstellung eines Experimentes sowie die Ein- und Anbindung an weitere Komponenten über „Java Beans“ [Bea]. Hierbei stellt der Server ähnlich wie auch für die Applet-Schnitte Templates für bereits bestehende Experimente zur Verfügung, die dann Client-seitig in eine Oorange-taugliche XML-Repräsentation umgesetzt werden. Der Nutzer hat hier, anders als bei den einfacheren Schnittstellen, jedoch die Möglichkeit, die so erzeugten Basisexperimente nach Belieben umzuändern und zu ergänzen. Abb. 6 zeigt einen entsprechenden Aufbau in Oorange, hier ein Experiment zum Ising-Modell. Links ist das Oorange-Netzwerk zu sehen, welches die einzelnen Bausteine des Experimentes vernetzt, rechts die Visualisierung eines der Bausteine, wieder mit der gewohnten Oberfläche. Abb. 7 zeigt exemplarisch den Anschluss eines Messinstrumentes zur Aufnahme der Magnetisierung. Die Messresultate werden mit Hilfe eines Plotters dargestellt. In der rechten Bildhälfte ist hier der Quellcode zur Realisierung des Ising-Modells gezeigt; der Code ist edierbar und lässt somit dem Benutzer Freiraum zum Experimentieren mit dem Modell.

Dieser Zugang zu VideoEaselhat nicht den Anspruch, besonders leicht gangbar zu sein, da er in erster Linie für den Forschungseinsatz und nicht für den Einsatz in der Lehre oder in Praktika konzipiert ist. Somit darf vom Benutzer eine zusätzliche Einarbeitungsphase eingefordert werden.

– **Schnittstellen:**
Für wissenschaftliche Anwendungen ist Oorange alleine nicht ausreichend, da es die hierfür teilweise hochspeziellen numerischen Algorithmen nicht bereitstellt. Allerdings existiert für diesen Zweck eine exemplarische Schnittstelle zu Maple, die das Maple C-Interface auf CORBA umsetzt und so einen Zugriff auf die Resultate des Labors liefert. Da sich Oorange und Maple gleichzeitig auf demselben Experiment einsetzen lassen, potenzieren sich hierdurch die Einsatzmöglichkeiten.

Anbindungen zu weiterer Standardsoftware lassen sich erstellen, vorausgesetzt die entsprechende Software selbst bietet Schnittstellen zu externen Programmen. Die Wahl von CORBA als Middleware bietet dabei eine sehr breite Basis zur Implementierung solcher Schnittstellen, da CORBA-Mappings für eine Vielzahl von Sprachen bereitstehen.

Abb. 8 zeigt die Anbindung des schon oben dargestellten Ising-Experimentes an die Software „Maple“.

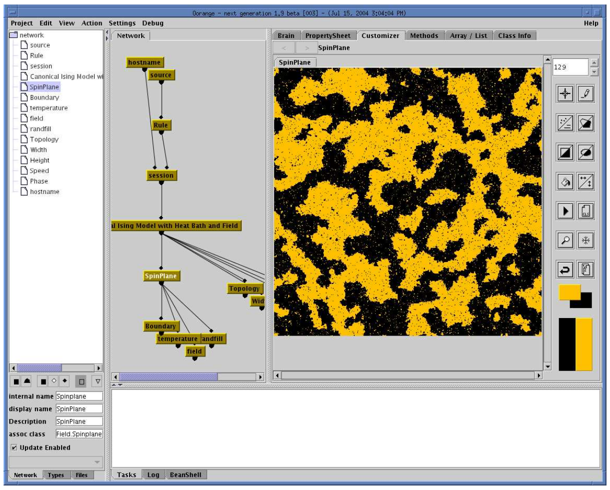

Abbildung 6: Ein typischer Oorange-Aufbau zum Nachweis spontaner Magnetisierung im Ising-Modell

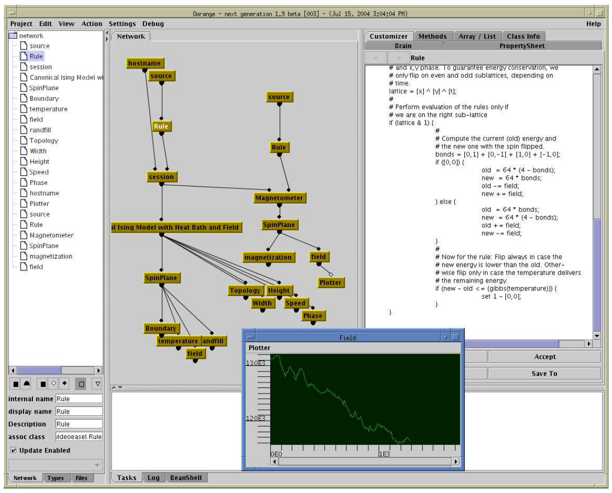

Abbildung 7: Messaufnahme der Magnetisierung im Ising-Modell

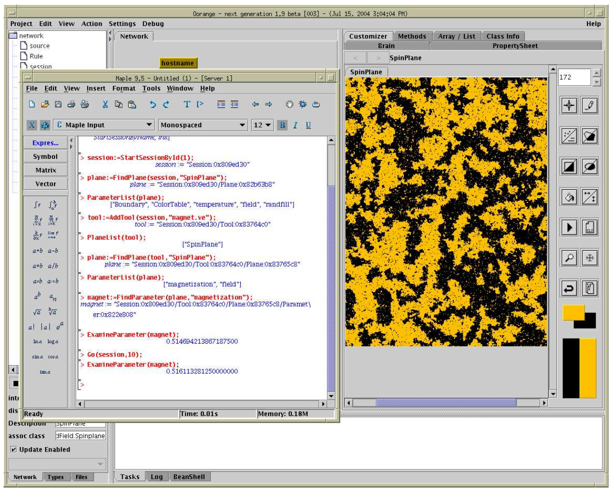

Abbildung 8: Auswertung von Messwerten mit Hilfe von Maple

Betrachten wir nun rückblickend, wie die eingeforderten didaktischen Konzepte innerhalb von VideoEaselumgesetzt werden: Die hier betrachtete Welt der zellulären Automaten ist reichhaltig genug, um einerseits interessante physikalische Effekte simulieren zu können, und andererseits noch so überschaubar, dass Zugangsbarrieren vermieden werden. Das Grundprinzip eines solchen Automaten kann innerhalb kürzester Zeit erlernt werden und mit einfachsten Mitteln lassen sich auch vom ästhetischen Standpunkt aus interessante Experimente durchführen. Durch Anknüpfung an bekannte Konzepte - Zeichenprogramme und Messaparaturen - und Wahl einer geeigneten Oberfläche wird der Benutzer zum Ausprobieren animiert. Das Verständnis über das Verhalten des zu begreifenden Effektes entsteht so beim Umgang mit dem Labor. Exploratives Lernen wird durch die Verknüpfung von ästhetischen und wissenschaftlichen Inhalten gefördert.

Die Bereithaltung diverser Oberflächen erlaubt uns die Addressierung diverser Benutzergruppen mit ganz verschiedenen Ansprüchen an die Labore, und diverse Einsatzzwecke von der reinen Demonstration bis zum Forschungseinsatz werden durchführbar.

Durch den zweiteiligen Aufbau als ein Client/Server Netzwerkwerkarchitektur werden kooperative Einsatzszenarien durchführbar und das Erlernen und Erforschen innerhalb eines Teams ermöglicht. Kooperationen über räumliche Grenzen hinweg sind somit durchführbar.

Letztendlich erlaubt die CORBA-Schnittstelle zum Laborkern das Andocken und Verknüpfen des Labores mit anderen Laboren, Algebra-Systemen und Konnektoren, um so auch komplexe Sachverhalte demonstrieren zu können und so den Benutzer nicht in einer Labortechnologie einzuschließen.

5 Eine VideoEasel Sitzung

Um einen tieferen Einblick in die Arbeit mit einem virtuellen Labor zu ermöglichen, soll nun im Folgenden eine typische VideoEaselSitzung am Beispiel des kanonischen Ising-Modelles beschrieben werden; weitere Modelle stehen ebenso vorgefertigt bereit:

Öffnet der Benutzer die Oorange-Oberfläche, so muss dort zunächst entweder ein Experimental-Aufbau erstellt, oder ein bereits vorgefertigter Aufbau geladen werden. Ersteres erfordert tiefere Einsichten in die Architektur des Labors und soll hier darum nicht beschrieben werden. Lädt man stattdessen den bereitgestellten Ising-Aufbau, so wird auf der linken Seite der Benutzeroberfläche ein Netzwerk der beteiligten Bausteine des Labors aufgebaut, siehe Abbildung 6. Hierbei beinhaltet etwa der Knoten „hostname“ den

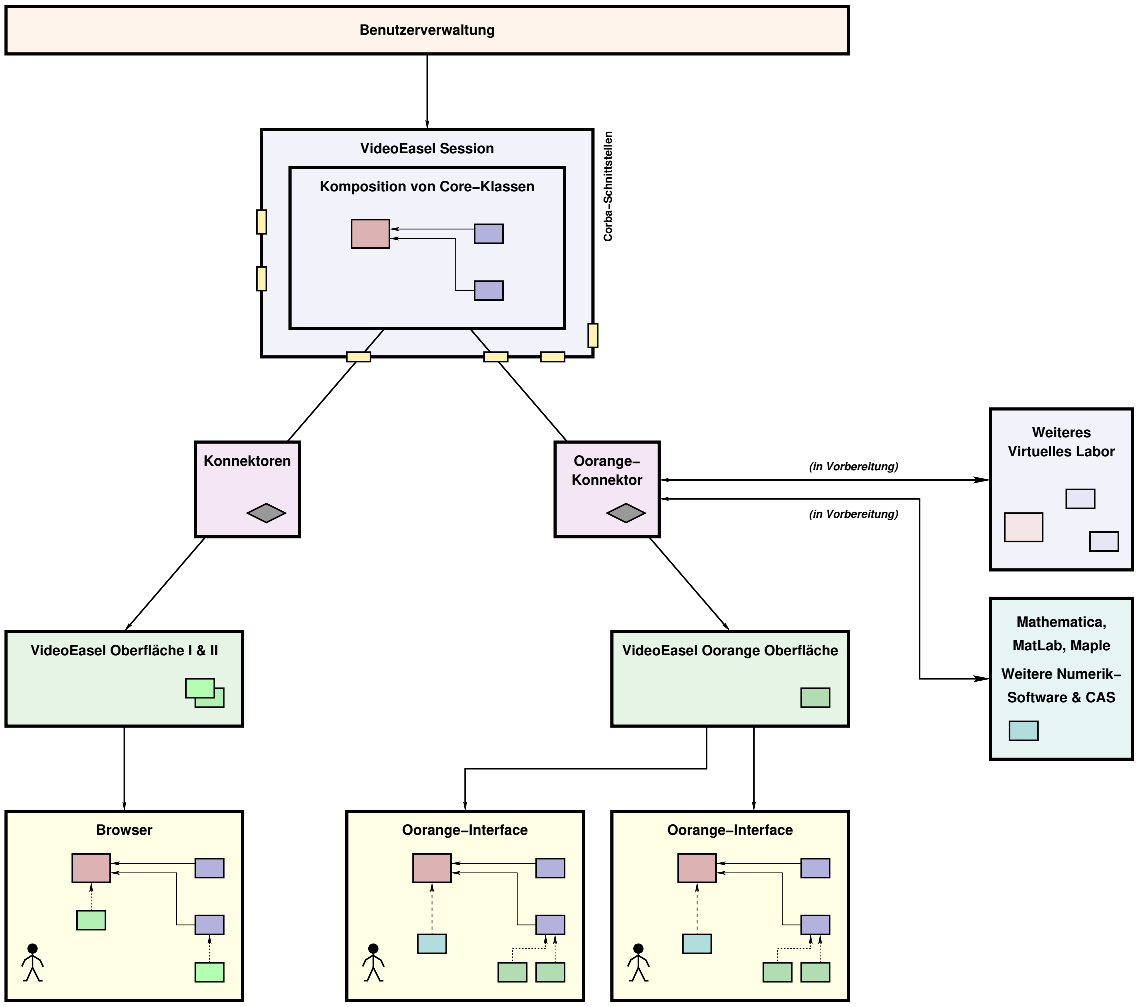

Abbildung 9: Softwaremodell VideoEasel

Namen des VideoEasel-Servers, „Rule“ den eigentlichen Algorithmus und „SpinPlane“ die Elementarmagnete des Ising-Modelles. Durch Anklicken der entsprechenden Knoten des Netzwerkes lässt sich auf der rechten Seite der Oberfläche eine Visualisierung des Knoteninhaltes anzeigen, im Abb. 6 erfolgte dies für den „SpinPlane“-Knoten.

Systeme der statistischen Mechanik, wie das hier gezeigte Ising-Modell, werden in VideoEasel durch zelluläre Automaten realisiert, ein geeignet aufgesetzter Automat bildet die Basis für einen Experimentalaufbau; er erscheint in Programmform im Knoten „Rule“. Der Zustand des physikalische System wird dabei durch eine Einfärbung des Zellengittes, hier immer eine Teilmenge des $\mathbb{Z}^2$, modelliert — in diesem Falle ist der Zustand durch den Knoten „SpinPlane“ repräsentiert. Die zugehörige Visualisierung stellt ebenso eine Oberfläche in der Form eines einfachen Zeichenprogrammes zur Manipulation dieses Zustandes bereit, mit der sich Flächen einfärben oder Kanten ziehen lassen.

Die Zeitevolution eines Systems der statistischen Mechanik wird in der Mathematik durch eine Abbildung Φ definiert, die den Zustand des Systems zur Zeit t auf den Zustand zur Zeit $t+1$ abbildet. Ein modelliertes System ist damit immer dreifach diskret: Diskrete Position, diskrete Zeitschritte und diskrete Zustandsvektoren.

Die Formulierung der Automatenregel erfolgt innerhalb des Labors mit Hilfe einer laboreigenen Sprache, die sich stark an C orientiert, diese jedoch durch zusätzliche Sprachelemente zur effizienten Formulierung von Regeln in der Zellenwelt ergänzt. In Abb. 7 ist auf der rechten Seite ein Ausschnitt aus dem Algorithmus zur Implementierung des Ising-Modelles zu erkennen. Dieser Algorithmus befindet sich im „Rule“-Knoten und kann ebenso wie der Zellenzustand ediert werden; die „Visualisierung“ besteht allerdings in diesem Falle in einem einfachen Text-Editor. Zur Vermeidung von nicht-terminierenden Regeln, die den VideoEasel-Server blockieren würden, werden nur einfachste Sprachkonstrukte wie Verzweigungen, jedoch keinerlei Schleifen unterstützt.

Der Automat lässt sich in der Visualisierung des Spinzustandes, siehe Abb. 6 nun starten: Bei geeignter Wahl der Temperatur und des Magnetfeldes entstehen wie im Bild zu sehen „Weisssche Bezirke“ in der Spinkonfiguration, die bei genügend langer Wartezeit verschmelzen und zu einer spontanen Magnetisierung des Modelles führen. Diese Parameter sind im Bild als tiefere Knoten der „SpinPlane“ angehängt; ihre Visualisierung besteht — hier in den Bildern nicht dargestellt — aus einfachen Schiebereglern, über die sich diese Parameter ändern lassen. Innerhalb des eigentlichen Automaten, der „Rule“ erscheinen diese Regler als numerische Variablen, auf deren Wert innerhalb des Algorithmus zurückgegriffen werden kann. Ebenso können diese Parameter über die CORBA-Schnittstelle erreicht, ausgelesen und verändert werden, so dass sich ein entsprechender Laboraufbau auch von dritter Seite, etwa einem Python-Skript, steuern und automatisieren lässt. Für die einfacheren Benutzeroberflächen wie etwa die Applets sind Parameter die einzige Einflussmöglichkeit, die dem Benutzer eines Experimentes gewährt wird. In der vollständigen Laboroberfläche kann auch die Automatenregel selbst verändert werden.

Zusätzlich zu den Automaten selbst stellt das Labor auch Messinstrumente zur Verfügung, die an ein Laborsystem angeschlossen werden können. Auch diese Messinstrumente sind durch Netzwerke entweder vollständig zu beziehen, oder können von Hand aufgebaut werden. In Abb. 7 ist dies etwa für ein einfaches Magnetometer zur Vermessung der Magnetisierung geschehen.

Die für diese Messinstrumente benötigten Messvorschriften sind auch als zelluläre Automaten realisiert, die auf die Zellkonfiguration des eigentlichen Systems lesend zugreifen können und so Informationen für den Benutzer sammeln. Somit besitzt auch das Magnetometer-Netzwerk einen Knoten des Namens „Rule“, in dem sich der Algorithmus zur Bestimmung der Magnetisierung befindet — hier einfach eine Aufsummierung aller Spinwerte.

Die Messresultate der angeschlossenen Instrumente sind, wie auch die Zellkonfiguration selbst, über die CORBA-Schnittstelle des Labors abrufbar und können auf diese Weise numerisch ausgewertet werden.

6 Die Modelierung eines Physikalischen Systems

Eine klassische Realisierung des oben beschriebenen Ising-Modelles ist der „Metropolis-Algorithmus“ [NMT53], der im Folgenden durch eine Automatenregel im „Rule“-Knoten der VideoEasel-Welt implementiert soll, siehe auch [TM87].

Physikalische Konfigurationen werden im Ising-Modell durch Abbildungen S von einem Teilgitter T des $\mathbb{Z}^2$ in eine Teilmenge der reellen Zahlen modelliert, wobei der Wert der Abbildung an der Stelle x, y als die Ausrichtung des Spins an dieser Position des Gitters interpretiert wird. In der einfachsten Realisierung des Ising-Modelles kann ein Spin nur zwei Zustände annehmen, typischerweise wählt man hier $\{-1, 1\}$ mit den Interpretationen „spin up“ und „spin down“. Ausgehend von einer derartigen Konfiguration S des Gitters ist für das Ising-Modell die Energie dieser Konfiguration definiert durch

$$H(S) := -\sum_{\vec{x},\vec{y}\in T, |\vec{x}-\vec{y}|=1} J \cdot S(\vec{x})S(\vec{y}) - B_0 \sum_{\vec{x}\in T} S(\vec{x}) \; .$$

Hierbei beschreibt $J > 0$ die Größe der Wechselwirkung zwischen benachbarten Spins und B_0 die Energie eines Spins in einem äußeren Magnetfeld. Die vordere Summe läuft nur über benachbarte Spins. Die wichtigsten Eigenschaften dieses Funktionals sind, dass parallel ausgerichtete Spins energetisch bevorzugt werden, ebenso wie Spins, die parallel zum äußeren Magnetfeld B_0 ausgerichtet sind.

Im Metropolis-Modell wird nun mit Hilfe dieses Energiefunktionals eine Dynamik auf den Spinkonfigurationen $T \to \{-1, 1\}$ eingeführt; man beachte, dass somit die hier betrachteten Systeme dreifach diskret sind: Diskret im Ort, diskret in den Zuständen und diskret in der Zeit.

Der Metropolis-Algorithmus

1. Bestimme die Energie des Systems $H(S)$ in der augenblicklichen Konfiguration S.
2. Erwürfele eine Zellposition $\vec{x}$ in T und definiere eine Zellkonfiguration S', die aus S durch Umklappen des Spins an der erwürfelten Position $\vec{x}$ hervorgeht:

$$S'(\vec{y}) := \begin{cases} S(\vec{y}) & \text{für } \vec{x} \neq \vec{y} \\ -S(\vec{y}) & \text{für } \vec{x} = \vec{y} \end{cases}$$

3. Bestimme die Energie des Systems nach Umklappen des Spins, $H(S')$ und die Energiedifferenz durch Umklappen, $\Delta E := H(S') - H(S)$.
4. Falls die Energie der Zelle durch Umklappen abnimmt, d.h. $\Delta E < 0$, so flippe den Spin.
5. Ansonsten flippe den Spin mit einer Wahrscheinlichkeit von $P(\Delta E) \sim \exp(-\beta \Delta E)$ wobei β die inverse Temperatur ist.

Die im letzten Punkt betrachtete Umklappwahrscheinlichkeit kann so verstanden werden, dass $P(\Delta E)$ diejenige Wahrscheinlichkeit Q ist, mit der die termische Energie X eines Wärmebades die zum Umklappen benötigte Energie des Spins aufbringt, wobei X eine Zufallsvariable ist, d.h.

$$P(\Delta E) = Q(X \geq \Delta E) = \int_{\Delta E}^{\infty} p(\xi) d\xi$$

wobei p die Wahrscheinlichkeitsdichte der Energieverteilung des Wärmebades ist. Offensichtlich ist dann $p(\xi) = \beta \exp(-\beta\xi)$ eine Laplaceverteilung.

Wir übersetzen nun Schritt für Schritt den oben definierten Algorithmus in die innerhalb von VideoEasel benutzte Sprache zur Formulierung einer Automatenregel: Zunächst fällt auf, dass die Energiedifferenz ΔE gar nicht von den gesamten Spinkonfigurationen S und S' abhängt, sondern es genügt, S und S' an der Stelle $\vec{x}$ und den vier nächsten Nachbarn zu betrachten. Weiterhin ist die Abbildung H homogen; es genügt damit also zur Berechnung der Energiedifferenz, den Wert eines Spins und der seiner Nachbarzellen zur kennen. Dies ist eine Voraussetzung zur Formulierung eines Modells innerhalb der VideoEasel-Welt. Die Aufgabe einer Automatenregel ist nun die Ermittlung des Spin- bzw. Zellenzustandes im nächsten Zeitschritt, ausgehend von der Konfiguration der Nachbarspins zur jetzigen Zeit. Aufgrund der Homongenität des Modells genügt zur Addressierung der Zell- bzw. Spinzustände also die Angabe einer relativen Position. Diese werden in der Laborsprache als $[\mathtt{d}_1, \mathtt{d}_1]$ notiert, wobei $\vec{d} = (d_1, d_2)$ den Differenzvektor bezeichnen. Somit ist $[\mathtt{0}, \mathtt{0}]$ der Zustand der augenblicklich betrachteten Zelle, und $[\mathtt{1}, \mathtt{0}]$ der Zustand des Spins rechts davon. Ferner werden wir im Folgenden 0 und 1 statt -1 und 1 zur Codierung der Spinausrichtung verwenden[2]

Der Codeabschnitt zur Ermittlung der Energiedifferenz im VideoEasel-Code liest sich dann wie folgt:

```
bonds = [0,1] + [0,-1] + [1,0] + [-1,0];
if ([0,0] == 1) {
    old  = 64 * (4 - bonds);
    new  = 64 * bonds;
    old -= field;
    new += field;
} else {
    old  = 64 * bonds;
```

[2] Diese abweichende Wahl ist nur historisch bedingt und ist ohne weiteren Belang; genauso hätten -1 und 1 gewählt werden können.

```
    new  = 64 * (4 - bonds);
    old += field;
    new -= field;
}
if (new < old) {
    set 1 - [0,0];
} else if (new - old <= {gibbs(temperature)}) {
    set 1 - [0,0];
}
```

Die Multiplikation mit 64 ist hierbei eine beliebige Wahl für den Wechselwirkungsterm J. Die Variablen `old` und `new` enthalten den vom Spin `[0,0]` erzeugten Energiebeitrag, `{gibbs(temperature)}` ist ein Zufallszahlengenerator mit Laplace-Verteilung der oben betrachteten Art und `temperature` und `field` sind hierbei von außen vom Benutzer einstellbare Parameter für die Temperatur bzw. das äußere Magnetfeld.

Entgegen dem Metropolis-Algorithmus berechnet VideoEasel hingegen alle seine Zellen parallel, der zu ändernde Spin wird also nicht erwürfelt. Man überzeugt sich nun leicht, dass bei einer naiven Parallel-Anwendung der obigen Regel die Energieerhaltung nicht mehr gilt, weil etwa eine Viererclique von Spins unter Verletzung der Energieerhaltung gleichzeitig geflippt werden kann. Aus diesem Grunde dürfen nur solche Spins zugleich behandelt werden, die nicht miteinander wechselwirken. Dies sind Spins auf geradem oder ungeraden Teilgittern des $\mathbb{Z}^2$, d.h. solche Spins, für die die Koordinatensumme gerade bzw. ungerade ist. Die Zellenregel muss also so codiert werden, dass in einem Zeitschritt nur das gerade, und in dem folgenden Zeitschritt nur das ungerade Teilgitter betrachtet wird. Für diesen Zweck sind die Zellenkoordinaten $\vec{x} = (x, y)$ unter der Notation `[x]` bzw. `[y]` sowie der Zeitschritt als `[t]` verfügbar. Will man abwechselnd diese beiden Teilgitter betrachten, so geschieht dies über die Abfrage der Bedingung `if (([x]^[y]^[t])&1)`, wobei ^ die Exklusiv-oder Verknüpfung und & die Undierung der Binärdarstellung der Koordinaten ist. Diese Bedingung ist äquivalent zur Abfrage einer von der Zeit abhängigen geraden bzw. ungeraden Koordinatensumme.

Zusätzlich mit den Informationen über die wählbaren Parameter und die Konfiguration des Gesamtsystems entsteht dann der folgende Code (die Zeilennummern sind aus Referenzierungsgründen eingefügt und nicht Teil des Codes):

```
name        =   "Canonical Ising Model";
topology    =   torus in {bounded,torus};
width       =   512 in {64..1024};
height      =   512 in {64..1024};
plane       =   {
    name                    =   "SpinPlane";
    datatype                =   byte;
    boundary                =   0 in {0,1};
    parameter temperature   =   32 in {0..128};
    parameter field         =   0 in {-32..32};
    randomset randfill      =   {0,1};
    colorcode 0             = "spindown"    {0,0,0};
    colorcode 1             = "spinup"  {255,192,0};
    rule    = {
        variable lattice;
        variable bonds,old,new;
        lattice = [x] ^ [y] ^ [t];
        if (lattice & 1) {
            bonds = [0,1] + [0,-1] + [1,0] + [-1,0];
            if ([0,0] == 1) {
                old  = 64 * (4 - bonds);
                new  = 64 * bonds;
```

```
23                  old -= field;
24                  new += field;
25              } else {
26                  old  = 64 * bonds;
27                  new  = 64 * (4 - bonds);
28                  old += field;
29                  new -= field;
30              }
31              if (new < old) {
32                  set 1 - [0,0];
33              } else if (new - old <= {gibbs(temperature)}) {
34                  set 1 - [0,0];
35              }
36          }
37      };
38  };
```

Zeilen 1-4 definieren die globalen Parameter des Modells wie die Breite und Höhe, die Topologie und den Rahmen innerhalb derer sich diese Parameter verändern lassen.

Zeilen 5-13 definieren Parameter für eine „Schicht“ des Automaten; der hier demonstrierte Ising-Automat benötigt nur eine einzige, jedoch ist es für manche Modellsysteme hilfreich, wenn ein Experiment aus mehreren verketteten Automaten aufgebaut wird, die auf jeweils ihrer eigenen Kopie eines Gitters arbeiten. Zugriffe zwischen den entsprechenden Schichten werden durch eine zusätzliche dritte Koordinate bei der Addressierung eines Zellzustandes wie beispielsweise `[0,1,-1]` realisiert.

Die in der einzigen Schicht benötigten Parameter sind die Temperatur und das äußere Feld, zusätzlich kommen noch der Datentyp für den Zellzustand und die Auswahl der Zellfarben hinzu. Der Parameter „randfill“ wird in der Automaten-Regel nicht explizit verwendet, kann aber innerhalb einer Benutzeroberfläche zum Festlegen einer zufälligen Startkonfiguration benutzt werden.

Zeilen 14-37 definieren die eigentliche Automatenabbildung wie oben dargestellt, Zeilen 15-16 stellen dabei typenlose temporäre Variablen bereit.

7 Ausblick

VideoEasel ist noch im Prototypen-Stadium, insofern liegen noch keine konkreten Erfahrungen beim Einsatz des Labors im Praktikumsalltag vor. VideoEasel selbst stellt keine virtuellen Wissensräume im Sinne von [Ham02] zur Verfügung — dies ist jedoch Teil des granularen Konzeptes, da eine entsprechende Funktionalität von hierzu entwickelten Plattformen besser geliefert werden können. Es liefert hingegen die Schnittstellen, um eine solche Einbindung zu ermöglichen, die im weiteren Vorgehen noch präzesiert und standardisiert werden müssen. Weitere Erfahrung mit noch zu erstellenenden Laborkomponenten anderer Fachgebiete sind erforderlich, um eine geeignete Standardisierung zum Datenaustausch von Laboren untereinander voranzutreiben. Da die Schnittstellendefinition im IDL-Format vorliegen und über CORBA implementiert sind, ist eine Anbindung an diverse Applikationen möglich, jedoch ist hier noch viel Handarbeit gefragt.

Die Einbindung in die Oorange-Plattform ist noch von recht vorläufiger Natur und noch mit bestimmten Einschränkungen verbunden, so dass sich etwa Änderungen im Aufbau des Experimentes nur unvollständig in Änderungen der Automatenregel abbilden, insbesondere ist Oorange noch nicht selbst über eine Schnittstelle programmierbar. Entsprechende Ergänzungen seites des Oorange-Teams werden zur Zeit vorgenommen.

Zum Einsatz von Virtuellen Laboren in elektronischen Prüfungen und Praktika muss ein Sicherheitskonzept erstellt werden, welches die Authentizität der ermittelten Daten und den Schutz der Privatsphäre der Benutzer gewährleisten muss. Obwohl die Wahl der CORBA-Middleware eine kurzfristige Bereitstellung von netzwerkverknüpften Applikationen ermöglichte, ist das darunterliegende Netzwerkprotokoll nicht unproblematisch, etwa ist die Durchtunnelung von Firewalls, wie sie zum Schutz von IT-Infrastrukturen innerhalb einer Universität unbedingt eingesetzt werden müssen, mit Problemen verbunden; Obwohl geeignete CORBA-Ergänzungen durchaus verfügbar und in der Standardisierung verfügbar sind, sind diese meist proprietären Lösungen nicht innerhalb von Open-Source Projekten einsetzbar, so dass weitere Alternativen in Erwägung gezogen werden sollten.

Literatur

[Bea] JavaBeans. `http://java.sun.com/products/ejb/`.

[Ham02] T. Hampel, R. Keil-Slawik. sTeam: Structuring Information in a Team - Distributed Knowledge Management in Cooperative Learning Environments. *ACM Journal of Educational Resources in Computing*, 1(2), 2002.

[Jes04] S. Jeschke. *Mathematik in Virtuellen Wissensräumen – IuK-Strukturen und IT-Technologien in Lehre und Forschung*. PhD thesis, Technische Universität Berlin, Berlin, April 2004.

[JKS04a] S. Jeschke and R. Keil-Slawik. Next Generation in eLearning Technology – Die Elektrifizierung des Nürnberger Trichters und die Alternativen. Informationsgesellschaft. Alcatel SEL Stiftung, 2004.

[JKS04b] S. Jeschke, M. Kohlhase, and R. Seiler. eLearning-, eTeaching- & eResearch-Technologien – Chancen und Potentiale für die Mathematik. *DMV-Nachrichten*, Juli 2004.

[KRG04] U. Kortenkamp and J. Richter-Gebert. *Cinderella Interactive Geometry*. Springer, 1999, 2004.

[Mum] Multimediale Mathematikausbildung für Ingenieure. `http://www.mumie.net/`.

[NMT53] M.N. Teller N. Metropolis, A.W. Rosenbluth and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys*, (21):1087–1091, 1953.

[Oor] Oorange: The Oorange development environment. `http://www.oorange.de`.

[Sca] Todd Scallan. a corba primer. `http://www.omg.org`.

[TM87] T. Toffoli and N. Margolus. *Cellular Automata Machines*. MIT Press Cambridge, 1987.