Das Virtuelle Unix Labor

– Statusbericht SS06 –

Ein interaktives Kurssystem mit Tutorieller und Adaptiver Komponente und Ergebnisüberprüfung mittels domänenspezifischer Sprachen

Hubert Feyrer <hubert@feyrer.de>

Inhalt

- Was bisher geschah
- Wissenschaftliche Aspekte
- Aktuelle Themen:
  - Theorie:
    Tutorielle Komponente
    Benutzeradaption
  - Design, Architektur und Umsetzung:
    Tutorielle Komponente
    Benutzeradaption
- Weiteres Vorgehen

Was bisher geschah

Bisher: Überblick

- Basissystem aus Ausgangssituation
- Didaktische Analyse des Stoffgebiets
- Domänenspezifische Sprache zur Ergebnisüberprüfung
- Evaluation und Ergebnisse

Bisher: Basissystem

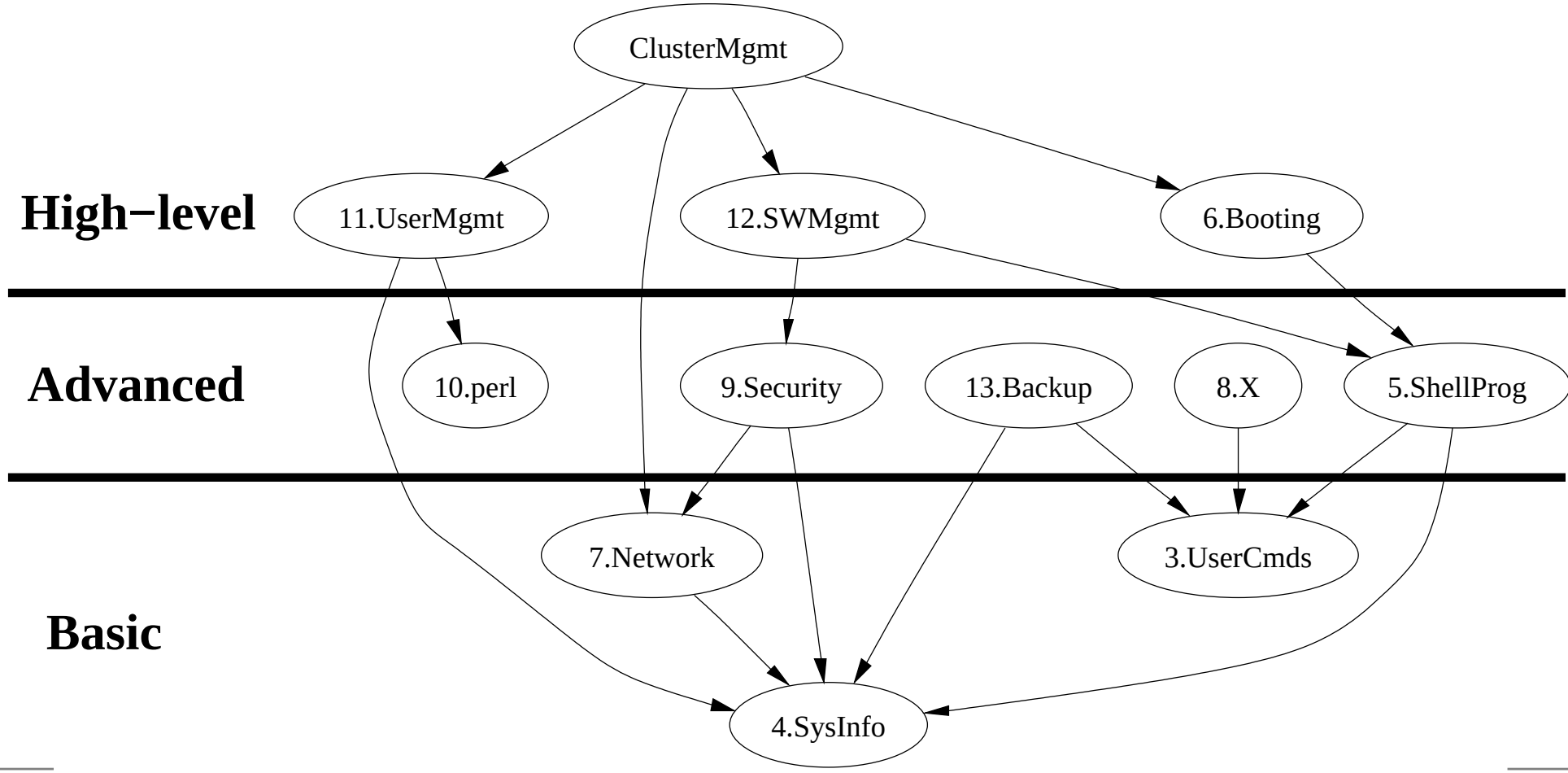
- Vorlesung “Systemadministration” am Fachbereich Informatik/Mathematik der FH Regensburg
- Wahl-Vorlesung besteht seit 1992, seit 2002 Pflichtvorlesung im Studiengang “Allgemeine Informatik”
- Mangel an Übungsmöglichkeiten für fortgeschrittene Themen
- Idee eines dedizierten Übungssystems ca. 2000
- Von 2001 bis 2004 im Rahmen des Hochschul-Wissenschaftsprogramms I (HWP) des BMBF entwickelt

Bisher: Basissystem

- Ergebnis: Webbasierte Lernumgebung zum Buchen von Kursen, vorbereiten dedizierter Übungsrechner, zum Absolvieren von Übungen und Erstellen von Auswertungen
- Vorgestellt auf European BSD Conference 2004 in Karlsruhe
- Erweiterung der Grundfunktionalität durch Themenbereiche der Informationswissenschaft

Bisher: Didaktik

- Didaktische Analyse des Stoffgebietes “Systemadministration”:



Bisher: Didaktik

- Einfache Themen lassen sich mit behavioristischen Lernmethoden lernen: Befehle, Argumente und Anwendungen durch Abfrage
- Fortgeschrittene Themen verlangen nach kognitiven / konstruktivistischen Methoden: praktische Übungen!
- $\Rightarrow$ VUlab als Unterstützung zur existierenden Vorlesung
- Vortrag “Didaktik der Systemadministration” auf dem Frühjahrsfachgespräch 2005 der German Unix User Group in München

Bisher: DSL

- Dient zur Analyse der Übungssysteme am Ende der Übung
- Wird für Diagnose benötigt, um detailiertes Feedback an den Benutzer bzgl. Übungserfolg geben zu können
- Iteratives Vorgehen: Definition von Primitiven, Verallgemeinerung durch Parameter, Erstellen eines Prozessors
- Test-Relevante Daten können beim Aufgabentext gehalten werden (“Data Structure Representation” Pattern)
- Dadurch leichtere Wartung der Übungen (“all in one place”)

Bisher: DSL

- Übungstext erhält Aufruf von Primitiven (Aktivatoren) in Sequenz $\Rightarrow$ Domänenspezifische Sprache (Domain Specific Language, DSL) zur Ergebnisüberprüfung
- Abfragesprache basiert auf PHP $\Rightarrow$ Definition komplexer Abfragen möglich und Erweiterbarkeit gewährleistet (“Language Specialisation” Pattern)
- Iteratives Vorgehen zum Festlegen der DSL zeigt Spezifika der Domäne “Systemadministration” auf: Primitive & Parameter

Bisher: DSL

Beispiel:

feyrer@wl445:/home/feyrer/work/vulab/code/public_html/texte

```
<p>

Aufgaben:
<p>

<h2> Paketverwaltung </h2>
<ol>
<li> Installieren Sie die bash und tcsh Bin\xe4rpaket (Quelle:
     ftp://ftp.netbsd.org/pub/NetBSD/packages/1.6/sparc/All)

     <?php auswertung_teiluebungen(
                  XXX, // vulab1: netbsd-check-installed-pkg PKG=tcsh
                       //         tcsh installiert? (pkg_info -e tcsh)

                  XXX  // vulab1: netbsd-check-installed-pkg PKG=bash
                       //         bash installiert? (pkg_info -e bash)
          ); ?>
</ol>


<h2> Benutzerverwaltung </h2>
<ol>
<li> Richten Sie einen neuen Benutzer "test" ein, Home-Verzeichnis
```

Bisher: Evaluation

- Analyse alter Prüfungen (Klausuren), Fragebogen, Übungsergebnissen
- Lernerfolg durch das Virtuelle Unix Labor bei Wiederholung von Übungen erkennbar
- Bedarf an mehr Informationen während der Übung
- $\Rightarrow$ Tutorielle Komponente, Benutzeradaption

Wissenschaftliche Aspekte

Wissenschaftliche Aspekte

- Grundlagenarbeit: Virtuelles Labor für Übungen zur Systemadministration
- Analyse der Didaktik Methoden im Bereich Systemadministration
- Erstellung einer domänenspezifischen Sprachen für die Verifikation der Übungsergebnisse im Bereich Systemadministration
- Unterstützung bei praktischen Übungen durch tutorielle Komponente
- Anpassung des Systems an unterschiedliche Lernertypen bzgl. Hilfestellung und Feedback

Theorie zu Tutoriellen Komponenten

Theorie zu Tutoriellen Komponenten

- Abgrenzung:
  - Tutor vs. Assistent
  - Einzel- vs. Gruppen-Tutor
  - Natural Language Processing
  - Intelligent Tutoring System
- Generell: Lehre = Wissenskommunikation
- Abbildung der Wissenskommunikation auf das Medium Computer

Theorie zu Tutoriellen Komponenten

- Intelligent Tutoring System als Umsetzung
- Komponenten eines ITS:
  - Pädagogisches (Lehr-) Modell
  - Domänen-Modell
  - Studenten (Benutzer) Model
  - Benutzer-Interface

TK: Pädagogisches Modell

- Allgemeines Vorgehen:
  1. Plan des Studenten erkennen: Bestimmt durch Bandbreite des Kommunikationskanals zwischen System und Student, bestimmt durch Unsicherheit
  2. Kontext für Lehr-Aktion erkennen: Bestimmen des Lehrziels, abgleichen zwischen Plan des Lerners und anpassen an Lehrplan, wobei hier mehrere verschiedene Pläne kombiniert werden können
  3. Entscheidungsbasis festlegen: Auswählen der anzuwendenden Lehrmethode auf Basis der (meist zu wenigen) Fakten, Bereiche: Didaktik, Domäne, Diagnostische Informationen
  4. Ziel-Level beim Studenten festlegen

TK: Pädagogisches Modell

- Allgemeines Vorgehen (Forts.):

  4. Ziel-Level beim Studenten festlegen (Forts.):
     - **Behaviouristisch:** Vorgehen ohne den internen “Status” des Studenten zu berücksichtigen
     - **Epistemisch:** Explizites Beeinflussen des Wissensstand des Studenten, vgl. konstruktives Lernmodell und situiertes Lernen
     - **Individuell:** “Ausserplanmäßige” Aktionen wie ermutigen verzweifelter Studenten, Anpassen der Lehr-Geschwindigkeit, Gratulieren, etc.

TK: Pädagogisches Modell

- Auswahl der Lehrmaßnahme aus Curriculum, getesteten Lehrplänen, Bibliotheken von Aktivitäten und Präsentationstechniken
- Verschiedene Arten der Lehre:
  - Basierend auf kognitivem Wissensnetz: Einbindung aktiver/passiver Assistenten, Erkennung falscher Schritte durch (domänenspezifische!) Theory of Bugs
  - Fakten- vs. Prozedurales Wissen und Zuordnung zum gewählten didaktischen Verfahren (behavioristisch, ...)

TK: Pädagogisches Modell

- Klassische Ansätze:
  - **Overlay Architektur:** Vergleicht Ansatz des Studenten mit dem eines (vorher bestimmten) Experten; Bestimmt, inwieweit einzelne Konzepte erlernt wurden, und wo noch pädagogische Methoden angewandt werden müssen
  - **Differentielles Studenten-Modell:** Prozedurales Netzwerk, das “grosse” Aufgaben in kleine segmentiert, und Fehler ohne Wahrscheinlichkeiten auf Denkfehler zurückführt; Liefert keine Begründung für Entscheidungen; Komplex: 111 observed bugs for substraction

TK: Pädagogisches Modell

- Klassische Ansätze (Forts.):
  - **Leeds Modeling System:** Kennt neben Regeln auch “Mal-Rules” um Fehler und Abweichungen zu erkennen
  - **User Modeling Front End:** Erweitertes Overlay Modell, das einzelne Konzepte zueinander in Beziehung setzt und diese vom diagnostischen Standpunkt nicht isoliert betrachtet
  - **Automated Cognitive Model:** Erzeugt automatisch Bug-Library aus Abweichungen von “guten” Primitiven; Qualität abhängig von deren Granularität

TK: Pädagogisches Modell

- Kognitiver Ansatz: Advanced Computer Tutoring Project
  - Berücksichtigt Vorwissen des Studenten
  - Lernen wird als zweistufiger Prozess angesehen:
    1. Prozeduralisierung vermitteler Informationen und
    2. Regel-Erstellung angesehen
  - Angewandt im Problem-Kontext, um auf konkrete Probleme hinzuweisen, ohne deren Lösung zu verraten. Dies ist dem Studenten als eigene kognitive Leistung aufgrund seines Vorwissens überlassen
  - Zeitnahes Feedback unterstützt den Studenten

TK: Pädagogisches Modell

- Linguistischer Ansatz:
  - Erkennen semantischer und pragmatischer Aktionen aus syntaktischen und semantischen Aktionen
  - Z.B. unter Verwendung von Planning Augmented Transition Networks
  - Vergleichbar zum Parsen von Netzwerk-Protokollen wie TCP/IP in verschiedenen Schichten, mit Ähnlichem Grundsatz: “Be liberal in what you accept, and conservative in what you emit”
  - Bottom-up recognition, top-down expectation
  - Kann auch auf natürlichsprachliche Problembeschreibungen angewandt werden

TK: Pädagogisches Modell

- KI-Ansatz:
  - Modellierung des Systems, und Anwendung von Methoden der Logischen Programmierung zur Erkennung von Vorhaben, Abweichungen und Lösungen
  - Verwenden von Algorithmen der Graphentheorie, Programmiersprachen: u.a. Lisp, Prolog
  - Theoretische vollständig und dadurch auch zur Verifizierung brauchbar
  - Praktisch zu aufwändig, z.B. Backtracking zum Löschen von Dateien oder Abbrechen von Programmen
- Auswahl des Pädagogischen Modells: $\Rightarrow$Umsetzung

TK: Domänen-Modell

- Domänen-Modell gibt Informationen über den zu vermittelnden Stoff
- Unterschied zwischen “artikuliertem” und “compilierten” Domänenwissen
- Compiliertes Wissen ist für den zielgerichteten Einsatz in bestimmten Situationen nötig
- Anwendung von compiliertem Wissen, um Schritte zu überprüfen ($\Rightarrow$ Prozedurales Wissen!), damit ein “Bild” vom Studenten aufzubauen, Lösungsvorschläge zu geben,
- Die Form der Kommunikation (“Artikulation”) kann abgestimmt sein auf Zweck, Struktur, Funktion, Randbedingungen, etc.

TK: Domänen-Modell

- Dazu unterteilen des Stoffes in kleine, abgestimmte Einheiten,
- Auswahl je nach Lehrverfahren und -situation
- Vergleiche Ergebnisse der Didaktischen Analyse oben
- Genauere Analyse des Stoffes: $\Rightarrow$Umsetzung
- Automatische Problemerstellung: nur für einfache Gebiete

TK: Benutzer-Modell

- Feststellen, was im Studenten vorgeht: Erwartungen, Plan, Absicht, Vorhaben
- Student ist dabei Empfänger von (Wissens)Kommunikation
- Genetic Graph gibt Beziehung der Stoffgebiete untereinander in fortschreitender Reihenfolge an $\Rightarrow$ Lernkurve
- Theories of Bugs helfen, Fehlverhalten festzustellen.

TK: Benutzer-Modell

- Arten von Theories of bugs:
  - **Enumerativ:** Katalog / Bibliothek von Beschreibungen, Regeln bzw. “Mal-Rules”, ggf. inkl. Klassifizierung
  - **Rekonstruktiv:** Erkennen von Fehlerklassen anhand allgemeiner Beschreibungen; Daten-getriebenes “bottom-up” Verfahren, oft durch domänenspezifische Sprache beschrieben
  - **Generativ:** Fehlerklassen nicht nur erkennen, sondern auf falsch Gelerntes zurückführen inkl. Erklärung was falsch gemacht wurde; dient als Feedback, Umsetzung “has never been tried” [Wenger, 1987]

TK: Benutzer-Modell

- Arten von Theories of bugs (Forts.):
  - Auswahl geschieht in Abhängigkeit vom Stoffgebiet und dem angestrebten Schwierigkeitsgrad, ggf. auch Kombination mehrere Verfahren. Testen gegen Kataloge bieten bessere Ergebnisse anstatt hergeleiteter Methoden, da einfacher zu implementieren, keine Parametrisierung nötig und es könenn auch ungewöhnliche Fälle abgedeckt werden.

    $\Rightarrow$ VUlab: checks gegen Ist-Zustand und Eingaben

TK: Benutzer-Modell

- Viewpoints:
  - Fehler / falsches Wissen von zwei Seiten sehen:
    1. Manifestation in falschen Aktionen und Aussagen
    2. Ansatzpunkt um bestehendes Falschwissen zu korrigieren
  - Auftreten bei unterschiedlichen Lösungsansätzen, Probleme können auf mehrere Arten betrachtet werden, zudem können gleiche Entscheidungen unterschiedliche Ursachen besitzen
  - Background viewpoints: Lerner besitzen individuellen Lernkontext und Erfahrungsschatz, daher automatisch unterschiedlicher Background pro Lerner

TK: Benutzer-Modell

- Viewpoints (Forts.):
  - Composite Viewpoints: Aufgrund unterschiedlicher Blickwinkel können sich Viewpoints überlappen oder widersprechen. Einfaches Mischen oft nicht sinnvoll/richtig, Entscheidungen können erst durch den Gesamtüberblick (Singular!) bestimmt werden.
  - Idealerweise sollten verschiedene Blickwinkel zur Diagnose erkannt und zur Problemanalyse auch alternative Blickwinkel dargeboten werden.

TK: Benutzer-Modell

- Diagnose: Grundlage des Feedbacks
  - Drei Aufgaben der Diagnose:
    1. **Inference:** Deterministisches(!) Ableiten von semantischen Ereignissen aus Syntaktischen Ereignissen
    2. **Interpretation:** Bestimmen von Vorgängen durch den Kontext in dem sie geschehen, durch Viewpoints und Goals. Versuch den Studenten zu verstehen bevor man ihm hilft.
    3. **Classification:** Charakterisierung bzw. Evaluation von Beobachtungen und Ziehen von Schlüssen basierend auf den gemachten Erwartungen (setzt Kennen des “Plans” voraus)

TK: Benutzer-Modell

- Diagnose (Forts.):
  - Diagnose geschiehe über Kommunikation, dabei Kommunikationskanal mit begrenzter Bandbreite (z.B. Tastatur, Maus)
  - Trotz des begrenzten Kommunikationskanals sind qualitative und quantitative Daten erfaßbar, Granularität eher ausschlaggebend
  - Ideal: Granularität entspricht der Granularität des compiliertem Wissens zur optimalen Bewertung
  - Semantik von Zwischenstufen: unterscheiden sich durch Endergebnis, und können durchaus legal sein (z.B. halb konfiguriertes System)
  - Behaviouristische Diagnose: post-hoc und on-line, beide im VUlab eingesetzt (bzw. geplant)

TK: Benutzer-Modell

- Diagnose (Forts.):
  - Epistemisce Diagnose besteht aus drei Phasen:
    1. **Direct assignment of credit and blame:** Feststellen welches Wissen benutzt bzw. nicht benutzt wurde
    2. **Structural consistency:** Korrelation der Meisterung ähnlicher Gebiete; auch ohne vollständigem Modell des Themengebiets machbar
    3. **Longitudinal consistency:** Balance zwischen schneller Anpassung und zu vorzeitigem Unterbrechen des Studenten, oft empirisch bestimmt

TK: Benutzer-Modell

- Diagnose (Forts.):
  - Noise: Oft fehlinterpretierte epistemische Überlappungen
    - Durch inkonsistentes Verhalten des Studenten
    - Durch Uneindeutigkeit des diagnostischen Prozesses, z.B. aufgrund des zu “engem” Kommunikationskanals
    - Durch Unzulänglichkeiten des Kommunikationsmodells in bestimmten Situationen
    - Bei Feststellen von vermehrtem “Rauschen” kann ggf. ein Wechsel des Viewpoints Abhilfe schaffen

TK: Benutzer-Modell

- Diagnose (Forts.):
  - Erhebung diagnostischer Daten: aktiv, passiv oder interaktiv
    - **Passive** Datenerhebung beeinträchtigt Student nicht zusätzlich, Daten sind jedoch auch auf Äußerungen des Studenten beschränkt, was zu eingeschränkter Schlußfolgemöglichkeit führt
    - **Aktive** Datenerhebung erlaubt, Hypothesen zu prüfen und weitere Daten anzufordern um zwischen unterschiedlichen, konkurrierenden Modellen unterscheiden zu können
    - **Interaktive** Dialoge erlauben, Entscheidungen, Annahmen und Plänezu hinterfragen; Probleme (z.B. falsche Annahmen) können früh erkannt und korrigiert werden

TK: Benutzer-Modell

- Diagnose (Forts.):
  - Feedback
    - Zweck: 1) An den Studenten, um ihn über Stand und Fortschritt seines Wissens zu Informieren, und 2) damit das System selbst weiss ob seine didaktischen Mittel richtig eingesetzt werden
    - Der Student steigert damit seine Kontrollüberzeugung
    - Zu unterschiedlichen Zeiten gegeben: sofort, verzögert, nach der Übung oder “on demand”
    - Neben “gelerntem” und “ungelerntem” Wissen kann auch noch entschieden werden ob Feedback über schwellwertiges Wissen gegeben wird
    - Personalisiertes Feedback auf Basis des Benutzer-Modells

TK: Benutzer-Interface

- Benutzer-Interface ist Teil der Kommunikation zwischen Benutzer und System
- UI soll Lernprozeß unterstützen und kognitive Belastung des Benutzers möglichst gering halten
- Idealerweise internale und externale Wissensdarstellung:
  - **Internale Darstellung:** Hilft beim Abgleich zwischem dem was gelernt werden soll und was (vermutlich) wirklich gelernt wurde ($\rightarrow$ User Model)
  - **Externe Darstellung:** Vermittelt Wissen, stark an Hardware/Technik gebunden, unterstützt Problemlösung

TK: Benutzer-Interface

- Aufteilung des Übungsablaufs in kleine Teilaufgaben erlaubt flexibleren Übungsablauf $\Rightarrow$ Learning Management System
- Erweiterungen für Tutorium und Adaption sollen explorativen Charakter des Systems nicht beeinträchtigen
- UI besteht im VUlab aus zwei Teilen: Übungstext darstellen und Explorative Schnittstelle zu Übungssystemen
- Interaktion soll nicht noch durch eine natürlichsprachliche Komponente erweitert werden: kein weiterer Kommunikationskanal, zudem zu komplex zu realisieren

TK: Benutzer-Interface

- Neben Zustandsüberprüfung des Systems sollte zusätzlich das Verhalten des Benutzers überwacht werden (können). Bei komplexen Systemen ist dies nicht direkt aneinander gekoppelt!
- Feedback kann aus vielen Teilkomponenten kommen, sollte im User Interface konsistent dargestellt werden
- Auswahl und Darstellung der Hilfe: vergleiche RSS-Filter

Theorie zu Benutzeradaption

Benutzeradaption

- Ziel: System soll den Benutzer unterstützen, und dadurch seine kognitive Belastung senken
- Im Gegensatz zu Aufgaben-unterstützenden Assistenten soll bei Lehrsystemen die Aufgabe nicht für den Studenten gelöst werden, sondern sein Lernverhalten unterstützt werden
- Gruppen-Teaching: Zwar Lehren von einzelnen Studenten, diese sind jedoch Teil einer Studiengruppe, und Adaption kann heterogene Studenten-Struktur ausgleichen, z.B. bzgl. Hintergrundwissen, Motivation, Fähigkeiten
- Grundlage: Studenten-Modell der tutoriellen Komponente bzw. des Intelligenten Tutoriellen Systems

Benutzeradaption

- Hier nicht betrachtete Themen: Lehre in Gruppen, Accessability, Privacy
- Komponenten für Adaption:
  - Kontext-sensitive Hilfe
  - Navigationshilfe
  - Adaptiver Hypertext
  - Personalisierter News-Filter
- Bestimmen von Kontext: Für bestimmte Situation charakteristische Information
- Im aktuellen VUlab: Analyse der Situation am Übungsende, für Feedback an den Studenten

Benutzeradaption

- System soll Anpassungen nicht einfach hinter dem Rücken des Benutzers ausführen, sondern diese darlegen und erklären, um ein Gefühl der Hilflosigkeit beim Student zu verhindern
- Modellierungstechniken: Erkennen von Benutzern durch ihre Ausgangssituation ($\Rightarrow$ Stereotype, Clustering) und eingehen auf ihre Pläne
- Stereotype: Annahme von generellen Verhaltensmustern aufgrund weniger gesammelter Daten. Einfach, aber unsicher!
- Clustering: Nicht mehr komplettes User-Modell abgeleitet, sondern nur noch Teilgebiete, von denen angenommen wird, daß sich User auf ihnen gleich verhalten

Benutzeradaption

- Plan-Erkennung: Aufnahme der Interaktion des Users mit dem System, Analyse und Ableiten von Schlüssen über den User, sein Wissen, Pläne, Verhalten - höhere Sicherheit als durch Stereotype, aber auch mit mehr Daten(erhebung) verbunden
- Mehreren Datenquellen zur Erstellung des Benutzermodells können durch “Agenten” abgebildet werden, deren Daten dann zusammengeführt werden
- Alternative bzw. Einschränkung der Plan-Erkennung: Vorgabe einer bestimmten Aufgabe, v.a. bei Lehrsystemen einfach zu realisieren, und komplexe Plan-Erkennung nicht in vollem Umfang nötig

Benutzeradaption

- Adaptive Achsen:
  - Longitudinale Anpassung bzgl. Fortschrittsgeschwindigkeit, Qualität und Häufigkeit von Feedback, etc.
  - Weitere Beispiele: Sprachstil, Umfang der Hilfe, Anwenden unterschiedlicher Methoden, Level des Diskurses, Backtracking, Fehlertoleranz
  - Ggf. nach Interaktions-Stil: Frage & Antwort, Menüauswahl, Kommandosprache
  - Kann pro Skill bzw. pro Domäne geschehen
  - Anpassung sollte mit hinreichender Sicherheit gemacht werden

Überblick: Tutoriellen und adaptiven Komponenten im Virtuellen Unix Labor

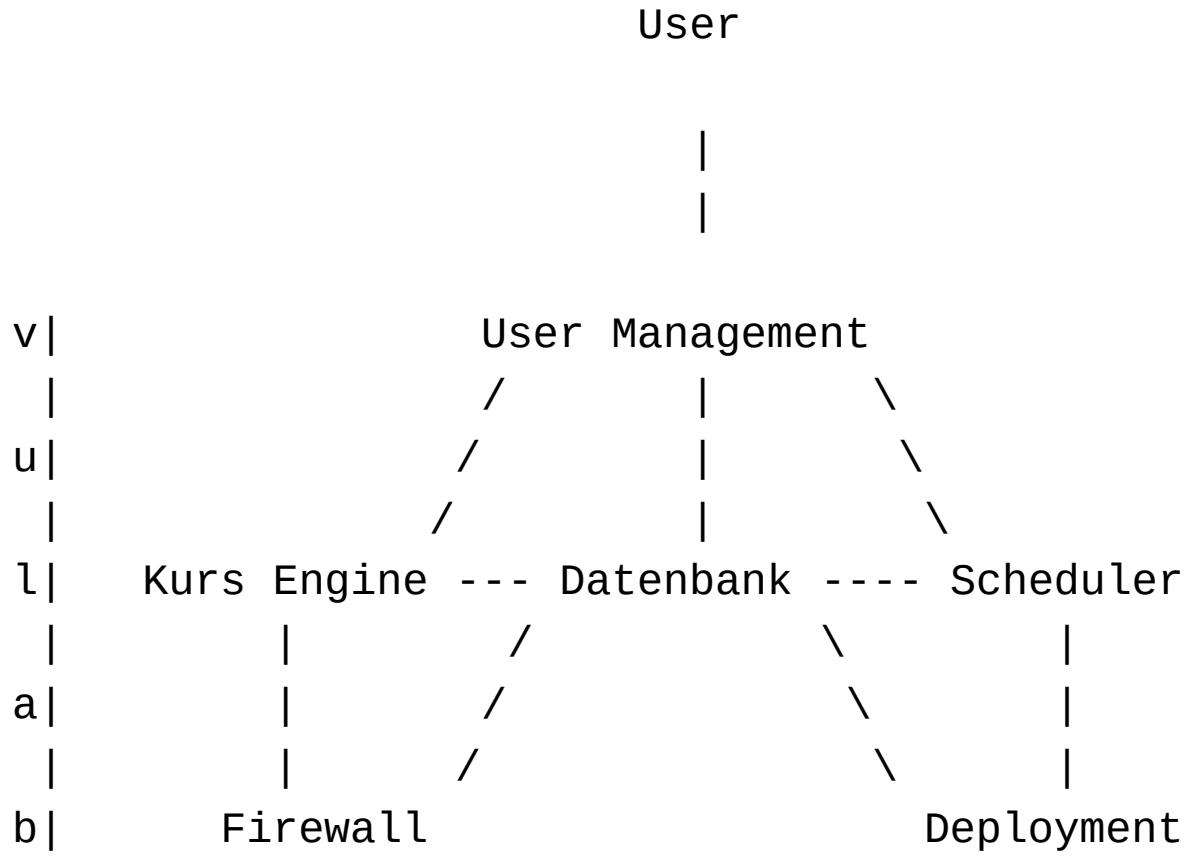
Ziele

- System soll Studenten während Übung unterstützen
- Nachbilden des Lehrers
- Architektur-Definition für tutorielle und adaptive Komponenten für den Bereich Systemadministration,
- Spezifikation für Umsetzung im Virtual Unix Lab

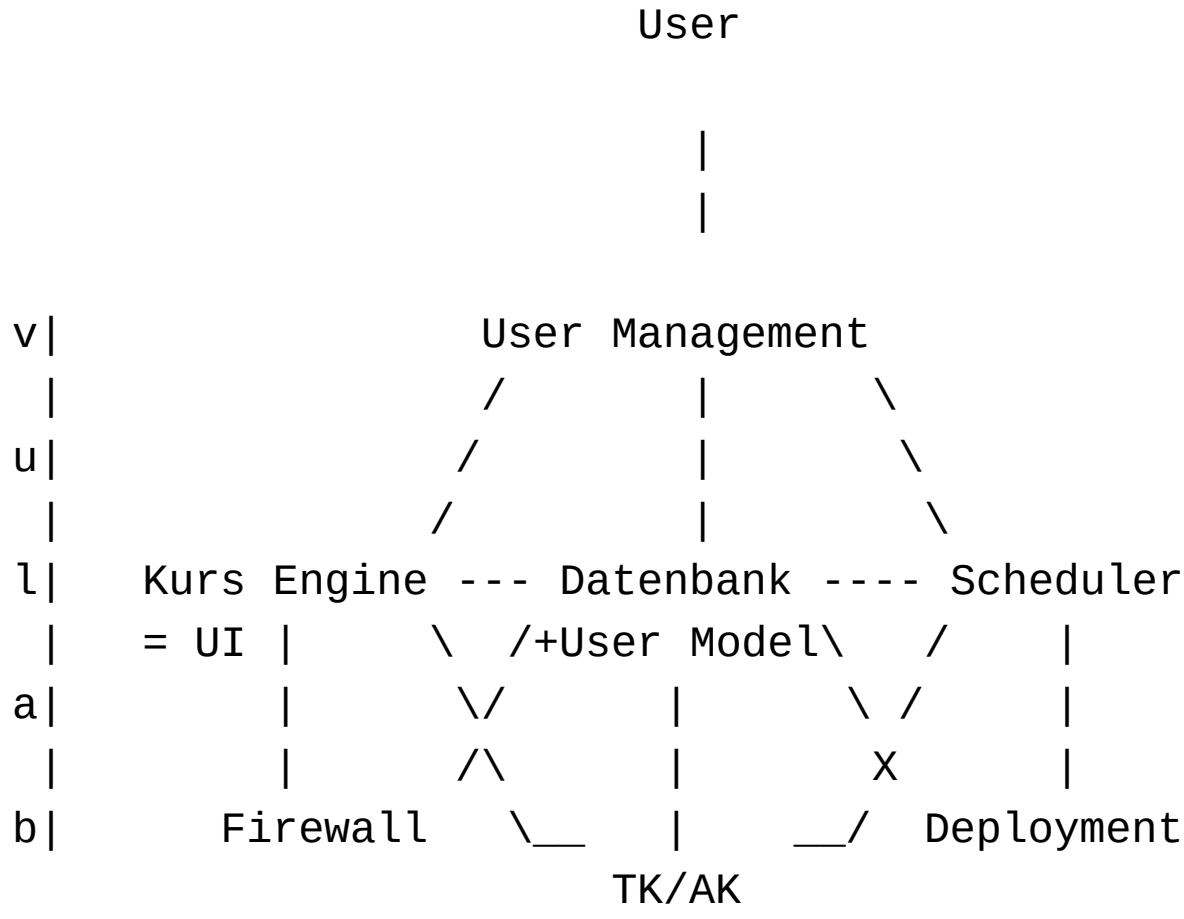
Methodik

- Inhaltlicher Bezug zur Theorie und den verschiedenen Modellen eines ITS
- Designansatz: Iterativ wie bereits bei der Ergebnisverifikation. Beginn mit tutorieller Komponente, dann aufbauend Benutzeradaption.
- Einbindung in bestehende Architektur des Virtual Unix Labs:

Architektur VUlab jetzt



Architektur VUlab mit TK+AK

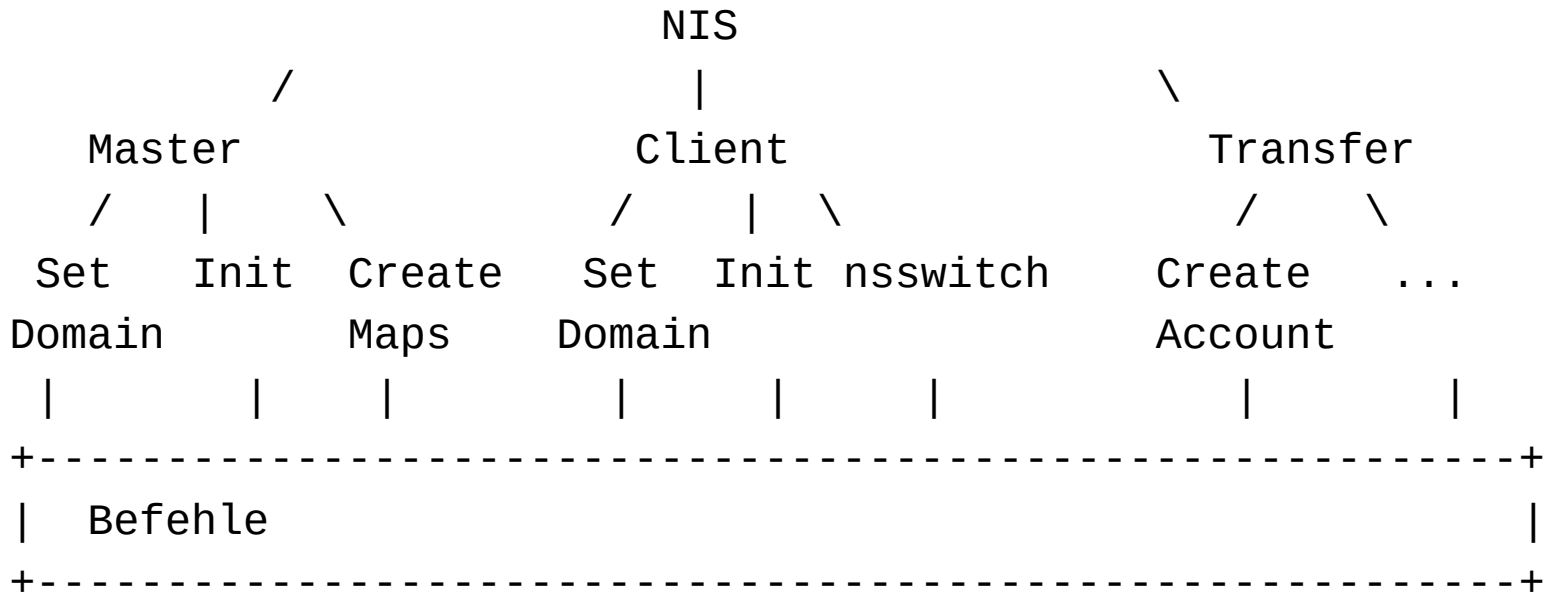


Architektur VUlab mit TK+AK

- Design entspricht den AVANTI-System von Kobsa et al
- TK/AK enthält Domänenmodell und Pädagogisches Modell
- Kommunikation zwischen Komponenten geschieht über SQL bzw. definierter DSL und deren Interfaces und Primitiven
- Theoretisch Wiederverwendung existierender Komponenten: BGP-MS, USCSH
- Ziel ist adaptives, tutorielles System, kein vollständiges Learning Management System

Domänen-Modell

- Bestehendes Script zur Vorlesung, bestehende Übungen zu den Themen Network File System und Network Information System
- Analyse des zugehörigen Stoffes inkl. Teilzielen und Wissen für Transferfragen. Beispiel:



Pädagogisches Modell

- Pädagogisches Modell: Overlay-Architektur mit “direct assignment of credit and blame”; KI- und linguistischer Ansatz zu aufwändig
- Ergebnis-Checks prüfen Zustand des Systems und damit die Auswirkungen der Anstrengungen des Studenten
- “Bugs” / Mal-Features werden nicht anhand einer (für den Bereich Systemadministration nicht existierenden!) Theory of Bugs festgemacht, sondern auf Basis der Stoffzerlegung
- “False positives” erkennen fehlerhafte / destruktive Schritte des Studenten

Studenten-Modell

- Im ersten Schritt unintelligenter Tutor, nach Nathan: “ITSs do too much thinking”
- Feedback wird jedoch gegeben. Aktuell (nur) am Ende der Übung, Ziel: öfters / on demand
- Tests des Systems und Feedback dazu erleichtert die Übung für den Studenten, da er den Fortschritt nicht selbst bewerten muß
- Ausbaufähigkeit für mehr Daten über Benutzer und deren Anwendung ist sichergestellt

User Interface

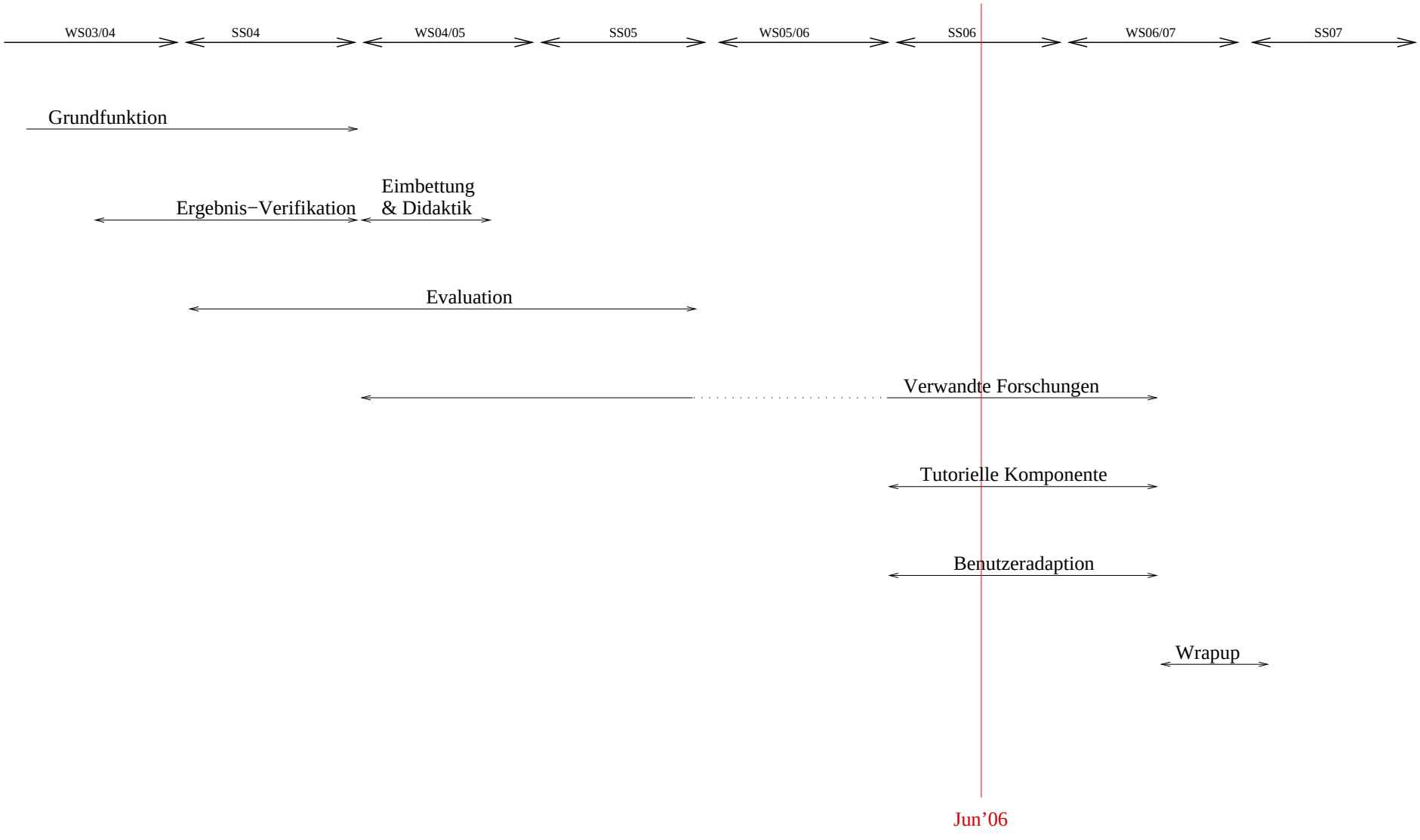
- Aktuell zwei Kommunikationskanäle:
  1. Gewinnen von Daten, aktuell nur über Checks
  2. Geben von Feedback am Übungsende
- Erweiterungen: Daten auch über direkte Benutzereingabe sammeln, nicht nur über deren Auswirkungen
- Feedback dann auch über erweiterte Daten
- Asynchrones und on-demand Feedback während Übung (vgl. Lehrer)
- Ggf. Auswahl aus vielen Informationskanälen und zusammenfügen im User Interface (vgl. RSS-Ansatz)
- Feedback sollte auch Benutzer-Stand und -Präferenzen berücksichtigen

Weiteres Vorgehen

Nächste Schritte

- Analyse der Übungsdomänen im Detail
- Theory of Bugs für Bereich Systemadministration?
- Aufstellung zu sammelnder / sammelbarer Daten für Benutzermodell
- Spezifikation Tutorielle Komponente mit periodischen Checks und Feedback
- Architektur zur Diagnose und Analyse der Benutzereingaben
- Architektur: Benutzeranpassung für Feedback
- Nachdenken über Aufbrechen der Übungsfolge in kleine Teilübungen

Zeitplan


WS03/04
SS04
WS04/05
SS05
WS05/06
SS06
WS06/07
SS07
Grundfunktion
Ergebnis−Verifikation
Eimbettung
& Didaktik
Evaluation
Verwandte Forschungen
Tutorielle Komponente
Benutzeradaption
Wrapup
Jun'06

Danke!

http://www.feyrer.de/vulab/

hubert@feyrer.de