TECHNISCHE UNIVERSITÄT ILMENAU

Fachgebiet Konstruktionstechnik
Fachgebiet Grundlagen der Elektrotechnik
Fachgebiet Medienkonzeption / Digitale Medien

Tagungsband

6. Workshop

Multimedia für Bildung und Wirtschaft

26. und 27. September 2002

ISSN1436-4492

Vera Yakimchuk, Hilke Garbe, Claus Möbus, Heinz-Jürgen Thole

Eine intelligente Problemlöseumgebung für die Grundlagen der Elektrotechnik

Abstract

Aufbauend auf umfangreichen Vorarbeiten [7,8,15] und basierend auf Aufgabensammlungen der Partner-Universitäten aus Dresden, Ilmenau und Magdeburg erstellt die Arbeitsgruppe von Prof. Möbus im Rahmen des Verbundprojekts mile (multimedia learning environments) [16] eine intelligente Problemlösungsumgebung für ein Themengebiet aus den Grundlagen der Elektrotechnik. Die Methodik der Formalen Begriffsanalyse [3] ermöglicht eine konzeptuelle Hierarchisierung der Aufgaben und Lösungen. Dieser Konzeptgraph steht den Lernenden beim Planen individueller Lernwege zur Verfügung. In der Diagnosekomponente werden Ziel-Mittel-Relationen [10] zur Elaboration der Planungsziele und zur symbolischen Berechnung der Aufgaben verwendet.
Diese wissensbasierten Komponenten bilden den Kern unserer Lernumgebung, die den Lernenden beim Lösen von Aufgaben adaptive Unterstützung nach dem ISP-DL-Ansatz [7,8] bietet.

IPSE und ISP-DL

Zielstellung des Projekts ist die Entwicklung einer wissensbasierten Selbstlernumgebung, die Studierende beim Lösen von Aufgaben zu ausgewählten Themen der Grundlagen der Elektrotechnik (GET) unterstützt.
IPSEs (Intelligent Problem Solving Environment) stellen eine spezielle Art der Intelligenten Tutor Systeme (ITS) [12] dar. Sie unterstützen den Lernenden beim aktiven Problemlösen innerhalb einer festgelegten Domäne. Adaptive Hilfestellung und Lösungshinweise basieren auf dem Formulieren von Hypothesen über die Korrektheit von Lösungsentwürfen und auf einem in das System integrierten Expertensystem (bzw. Orakel), das die Hypothesen der Lernenden analysieren, bewerten und weiterentwickeln kann. Das sonst in ITS übliche Tutormodell wird dabei durch eine Menge bzw. Sequenz von domänenspezifischen Aufgaben ersetzt. Ein Lernermodell entfällt, da das in IPSEs enthaltene generative Expertensystem einen großen Lösungsraum hat und Hilfen auf den Hypothesen der Lernenden basieren. Damit ist eine IPSE in der Lage, auf Anforderung adaptive, situationsbezogene Hilfen in Stocksituationen zu geben.
Die Entwicklung von IPSEs basiert auf der in der Arbeitsgruppe von Prof. Möbus entwickelten kognitionswissenschaftlich orientierten ISP-DL-Theorie (Impasse-Success-Problem-Solving-Driven-Learning) des Wissenserwerbs beim Problemlösen [7,8], die davon ausgeht, dass der Lernprozess und insbesondere Selbsterklärungen durch entdeckendes Lernen und selbständiges Formulieren von Hypothesen gefördert werden.
ISP-DL basiert auf den kognitionswissenschaftlichen Theorien von Newell [6,11] und Van Lehn [14] (impasse-driven-learning), Anderson [1,2] (success-driven-learning), sowie der motivationsorientierten Rubikontheorie von Heckhausen [5] und Gollwitzer [4] (problem solving phases). Ein Problemlöseprozess kann in die verschiedenen Problemlösephasen Abwägen, Planen, Ausführen und Bewerten eingeteilt werden.

Nach der ISP-DL-Theorie hat Lernen zwei wesentliche Aspekte. Wenn eine korrekte Lösung für ein Problem ohne Stocksituationen gefunden wurde, findet eine Wissensoptimierung statt. Dieser Prozess ist deduktiv, da das neue optimierte Wissen eine logische Konsequenz aus dem alten Wissen ist:

$$\text{Altwissen} \cup \text{Evidenz} \models \text{optimiertes Wissen}$$

Treten jedoch Stocksituationen beim Lösen einer Aufgabe auf, in denen das vorhandene Wissen zu ihrer Überwindung nicht ausreicht, so werden schwache Heuristiken eingesetzt wie nachfragen, sich Hilfe holen, Lösungsbeispiele studieren und elaborieren usw. Als Ergebnis wird das erforderliche Wissen induktiv erworben:

$$\text{Altwissen} \cup \text{Neuwissen} \models \text{Evidenz}$$

In diesem Fall können die eingesetzten Heuristiken als induktive Inferenz-Regeln betrachtet werden.

Eine IPSE für GET

Für unsere IPSE zu den Grundlagen der Elektrotechnik sind als Akteure zwei Benutzergruppen relevant: die Autoren und die Lernenden (Abbildung 1).

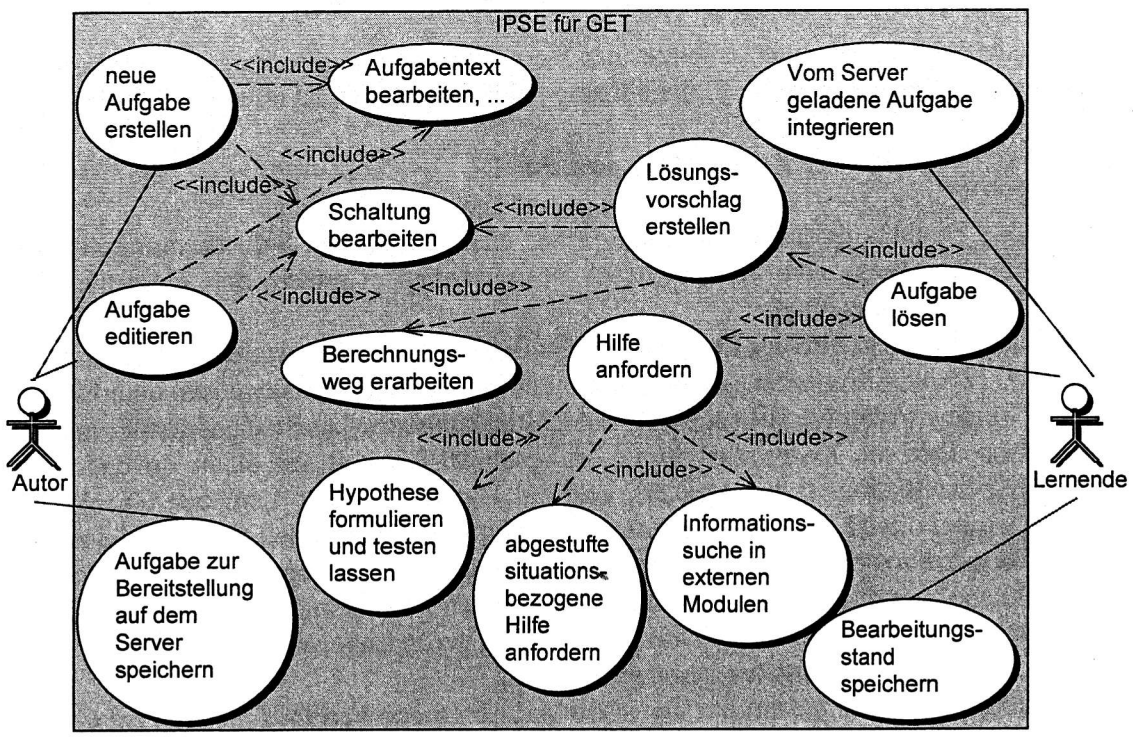

Abbildung 1. *Use Case* Diagramm der IPSE

Für jede dieser Benutzergruppen wird das System entsprechende Funktionalitäten zur Verfügung stellen. Den Autoren wird die Möglichkeit gegeben, die Aufgabensammlung zu erstellen und zu erweitern. Dies beinhaltet u.a. das Entwerfen einer Schaltung und das Formulieren des Aufgabentextes. Die Lernenden werden ausgewählte Aufgaben bearbeiten können, indem sie innerhalb eines graphischen Editors Schaltungsentwürfe konstruieren, Lösungswege planen und protokollieren sowie Hypothesen formulieren können. Das System wird in der Lage sein, diese

Hypothesen mit Hilfe der wissensbasierten Diagnose zu überprüfen. Dabei wird erkannt werden können, ob der Lösungsentwurf des Lernenden bereits die Lösung der Aufgabe darstellt, oder ob er sich zu einer korrekten Lösung vervollständigen lässt. Daraufhin wird das System differenzierte, situationsbezogene Rückmeldungen liefern können. Die Entscheidung über den Detaillierungsgrad der Hilfe wird bei dem Lernenden liegen. Zur Beantwortung von Hilfenanfragen, die sich nicht direkt auf die Aufgabe, sondern auf allgemeine Konzepte der Elektrotechnik beziehen, sind Verlinkungen auf die webbasierten Lernmodule der Projektpartner vorgesehen.

Systemarchitektur

Um eine spätere online-Benutzung des Systems zu ermöglichen, wird die Benutzungsoberfläche in Java realisiert (Abbildung 2, links oben). Sie bietet eine Arbeitsumgebung, die u.a. die Navigation in der Aufgabensammlung in Form eines Konzeptgraphen (s.u.), einen Schaltungseditor, einen Formeleditor und Protokoll- und Planungskomponenten zur Verfügung stellt.
Zur Speicherung der Aufgaben und des Bearbeitungsstandes werden XML-Formate entworfen (Abbildung 2, links unten).
Für die Implementierung der Diagnosekomponente wurde Prolog als Programmiersprache gewählt (Abbildung 3, rechts). Prolog liefert einen Inferenzmechanismus der mit Prädikaten arbeitet, die sowohl zum Generieren von Ergebnissen als auch zum Parsen von Eingaben genutzt werden können. Prolog-Prädikate, die die Wissensbasis bilden, sind dynamisch erweiterbar und veränderbar.
Die Regeln und Fakten des Domänenwissens müssen vom knowledge engineer in Kooperation mit Fachexperten implementiert werden. Prolog hat sich auch bei der Entwicklung von anderen IPSEs (Absynt [9], TAT [13], PULSE [7], MSAFE u.a.[15]) bewährt.
Der LPA Intelligence Server [18] bietet eine Schnittstelle für die Kommunikation zwischen Java und Prolog.

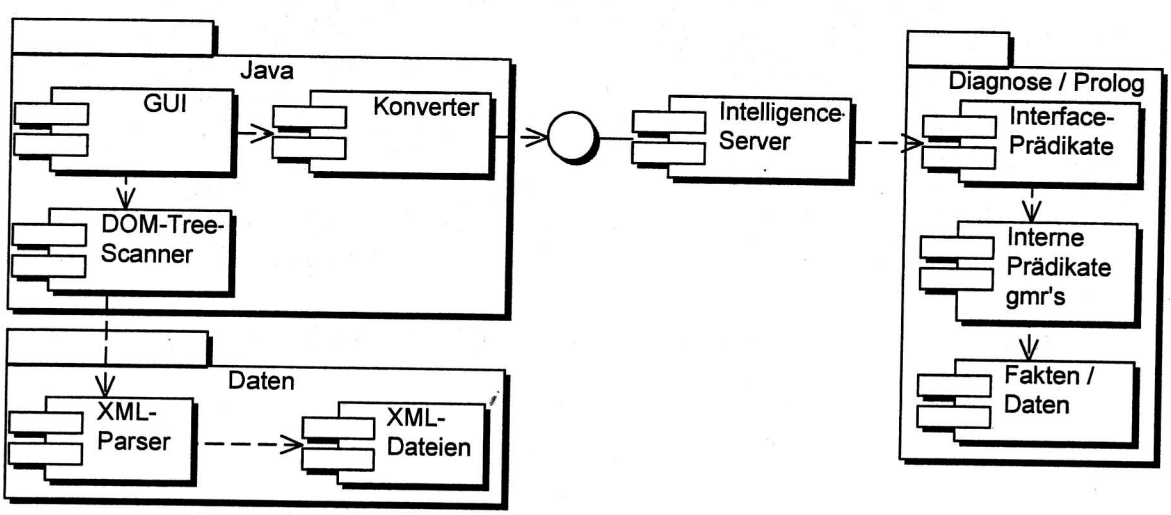

Abbildung 2. Komponentendiagramm der IPSE

Hierarchisierung der Aufgaben mit Formaler Begriffsanalyse

Die Grundlage für die Aufgabensammlung unserer IPSE bilden die Aufgaben der Partner-Universitäten TU Dresden, TU Ilmenau, Uni Magdeburg.
Die Anwendung der Methode der Formalen Begriffsanalyse [3] ermöglicht eine konzeptuelle Hierarchisierung der Aufgaben. Objekte (=Aufgaben und Teilaufgaben) werden nach charakteristischen Merkmalen (=elektrotechnische Gesetzmäßigkeiten) hierarchisiert. Die resultierende partielle Ordnung wird Begriffsverband genannt.

Die Abbildung 3 stellt beispielhaft einen Begriffsverband für drei Aufgaben dar. Aus den Aufgaben (AS1-AS3) wurden mögliche Teilaufgaben (A1-A10) definiert. Diesen Aufgaben und Teilaufgaben werden die zu ihrer Lösung notwendigen elektrotechnischen Gesetzmäßigkeiten (F1-F14) in einer Kreuztabelle (Abbildung 3, links) zugeordnet. Mit Hilfe mathematischer Verfahren lassen sich aus dieser Tabelle Begriffe berechnen, die sich in Form eines Liniendiagramms (oder Konzeptgraphen) darstellen lassen (Abbildung 3, rechts). Dieser Graph repräsentiert die inhaltlichen Abhängigkeiten der Aufgaben. Es lässt sich z.B. erkennen, dass die Teilaufgaben A1 und A2 Voraussetzungen für die Aufgabe AS1 sind und ihre Lösung auf den Gesetzmäßigkeiten F2, F3, F4 beruht.
Die Darstellung der Aufgabenhierarchien in Form eines Konzeptgraphen ermöglicht Orientierung und Navigation in der Aufgabensammlung und zeigt die inhaltlichen Abhängigkeiten der Aufgaben auf.
Der Konzeptgraph erleichtert dem Autor eine Überprüfung der Aufgabensammlung auf Vollständigkeit. So deutet ein Knoten ohne Aufgabe darauf hin, dass eventuell das Erstellen einer Aufgabe zu diesem Knoten sinnvoll wäre.
Der Lernende kann bei der Auswahl einer individuellen Aufgabensequenz unterstützt werden. Hat der Lernende z.B. Schwierigkeiten beim Lösen der Aufgabe AS3, weiß das System, welche Aufgaben als Hilfe angeboten werden können: AS1, A4, A5, A6 und welche elektrotechnischen Gesetzmäßigkeiten zum Verständnis notwendig sind: F2-F14.

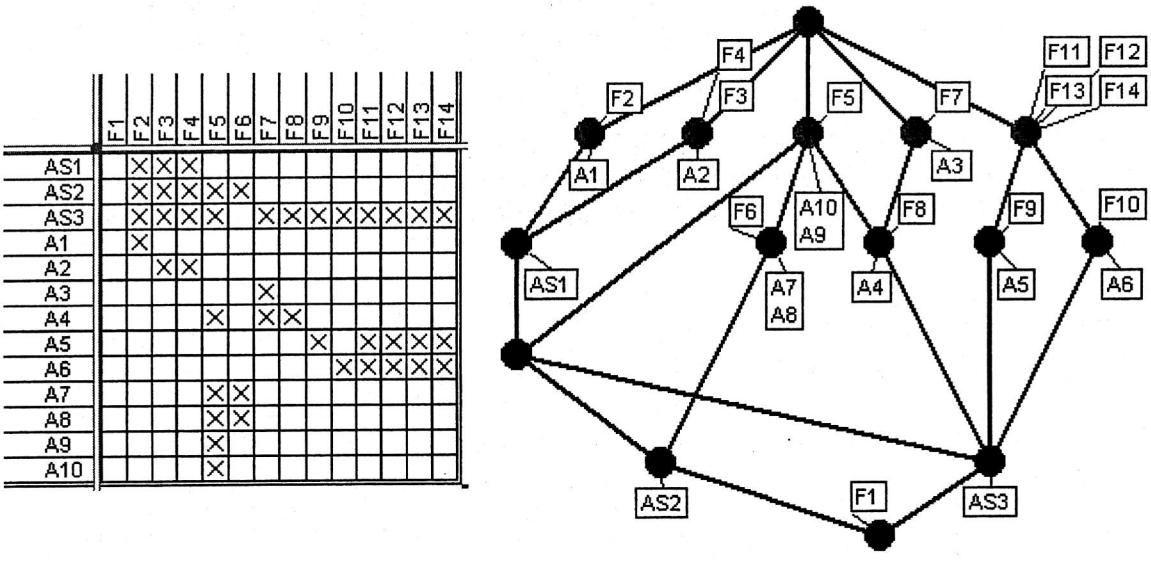

Abbildung 3. Formaler Kontext und Liniendiagramm eines Begriffverbandes (erstellt mit Anaconda und Toscana [17]).

Intelligente Diagnosekomponente mit GMR

GMR bezeichnet eine Menge von Regeln im Ziel-Mittel-Relationsformat (engl.: goals-means-relations), die eine Baumgrammatik bilden. Die Mittel variieren mit den unterschiedlichen Domänen, es können u.a. Implementations-Teile, chemische Elemente (inkl. Bindungen) oder Berechnungen sein. Dank Anwendung der GMR [10] zum Aufbau der Diagnosekomponente wird das System in der Lage sein, Hypothesen der Lernenden zu prüfen und Lösungsentwürfe ggfs. zu vervollständigen.
Abbildung 4 zeigt beispielhaft die zur Berechnung des Ersatzwiderstandes zweier parallel geschalteter Widerstände notwendigen Regeln.

```
gmr(rRepParallel(Goal1, Goal2), Mean) :- gmr(div(1, gTotal(Goal1, Goal2)), Mean).
gmr(gTotal(Goal1, Goal2), Mean) :- gmr(plus(g(Goal1), g(Goal2)), Mean).
gmr(g(Goal), Mean) :- gmr(div(1, Goal), Mean).

gmr(div(Goal1, Goal2), /(Mean1, Mean2)) :- gmr(Goal1, Mean1), gmr(Goal2, Mean2).
gmr(plus(Goal1, Goal2), +(Mean1, Mean2)) :- gmr(Goal1, Mean1), gmr(Goal2, Mean2).
gmr(plus(Goal1, Goal2), Mean) :- one gmr(plus(Goal2, Goal1), Mean).

gmr(A, A) :- atom(A).
gmr(N, N) :- number(N).
```

Abbildung 4. Berechnung eines Ersatzwiderstandes in der GMR-Notation

Die Prolog-Repräsentation von GMR-Regeln hat die Form: Regelkopf :- Regelkörper. Der Kopf enthält ein Ziel-Berechnungs-Paar, der Regelkörper enthält ein Ziel-Berechnungs-Paar oder eine Konjunktion von mehreren solchen Paaren. Das Ziel ist der erste und die Berechnung der zweite Parameter einer GMR. Die Regeln repräsentieren das Wissen in einer sehr feinkörnigen Struktur. Jede Regel enthält lediglich einen Planungs- bzw. Berechnungsschritt. Durch die rekursive Aufruf-Struktur können mit diesen Mikro-Regeln auch komplexe Probleme bearbeitet werden. Mit Hilfe der GMR ist sowohl das Generieren von Lösungsvorschlägen als auch das Parsen von Hypothesen möglich. Durch das Backtracking von Prolog kann mit den GMR eine Vielzahl von Lösungen zu einer Aufgabe generiert werden. So können z.B. unterschiedliche Lösungswege über verschiedene elektrotechnische Gesetzmäßigkeiten vorgeschlagen werden, aber auch mathematisch korrekte Umformungen der Lösung erkannt werden.

Einsatzgebiete

Die Lernumgebung kann sowohl im Direktstudium, als auch zum Selbstlernen eingesetzt werden. Weiterhin besteht die Möglichkeit, das System zur Vorbereitung auf Seminare und zur Vertiefung des Vorlesungsstoffes zu nutzen. Schließlich sind die Erklärungen beim Problemlösen mit kommentierten Lösungswegen auch für Prüfungsvorbereitungen eine hilfreiche Unterstützung. Nach Fertigstellung des ersten Prototyps wird eine umfassende Erprobung der IPSE im laufenden Studiumsbetrieb an den Partner-Universitäten stattfinden.

Literatur- bzw. Quellenhinweise:

[1] ANDERSON, J.R., Knowledge Compilation: The General Learning Mechanism. In: R.S. Michalski, J.G. Carbonell, T.M. Mitchell, Machine Learning II. Kaufman, 1986, 289-310.
[2] ANDERSON, J.R., A Theory of the Origins of Human Knowledge, Artificial Intelligence, 1989, 40, 313-351.
[3] GANTER, B., WILLE, R., Formale Begriffsanalyse: mathematische Grundlagen, Springer-Verlag 1996.
[4] GOLLWITZER, P.M., Action Phases and Mind-Sets, in: E.T. HIGGINS & R.M. SORRENTINO (eds), Handbook of Motivation and Cognition: Foundations of Social Behavior, 1990, Vol. 2, 53-92.
[5] HECKHAUSEN, H., Motivation und Handeln, Heidelberg: Springer, 1989 (second ed.).
[6] LAIRD,J.E, ROSENBLOOM, P.S., NEWELL, A., SOAR: An Architecture for General Intelligence, Artificial Intelligence, 1987, 33, 1-64.
[7] MÖBUS, C., Towards an Epistemology on Intelligent Problem Solving Environments: The Hypothesis Testing Approach, in P. BRNA, A. PAIVA, J. SELF (eds), Proceedings of EuroAIED 96, European Conference on Artificial Intelligence in Education, Lisbon, Portugal, 1996.
[8] MÖBUS, C., Towards an Epistemology on Intelligent Problem Solving Environments: The Hypothesis Testing Approach, in J. GREER (ed), Proceedings of AI-ED 95, World Conference on Artificial Intelligence and Education, Washington, DC, Association for the Advancement of Computing in Education (AACE), 1995, S. 138 – 145.
[9] MÖBUS, C., THOLE, H.J., Interactive Support for Planning Visual Programs in the Problem Solving Monitor ABSYNT: Giving Feedback to User Hypotheses on the Basis of a Goals-Means-Relation, in: D.H. NORRIE, H.-W. SIX (eds), Computer Assisted Learning. Proceedings of the 3rd International Conference on Computer-Assisted Learning ICCAL 90, Hagen, F.R.Germany, Lecture Notes in Computer Science, Vol. 438, Heidelberg: Springer, 1990, S. 36-49.
[10] MÖBUS, C., THOLE, H.-J., SCHRÖDER, O., Interactive Support of Planning in a Functional, Visual Programming Language, in P. BRNA, S. OHLSSON, H. PAIN (eds), Proceedings AI-ED 93, World Conference on Artificial Intelligence in Education, Edinburgh, Scotland, 1993.
[11] NEWELL, A., Unified Theories of Cognition, Cambridge, Mass.: Harvard University Press, 1990.
[12]SCHRÖDER, O., Intelligente tutorielle Systeme / Intelligente computerunterstützte Instruktion, in G. STRUBE, B. BECKER, C. FREKSA, U. HAHN, K. OPWIS, G. PALM (Hg), Wörterbuch der Kognitionswissenschaft, Stuttgart: Klett-Cotta, 1996.
[13]THOLE, H.-J., MÖBUS, C., SCHRÖDER, O., Domain Knowledge Structure, Knowledge Representation and Hypotheses Testing, in: B. Boulay, R. Mizoguchi (eds.): Artificial Intelligence in Education, Amsterdam: IOS Press, 1997.
[14] VanLEHN, K., Toward a Theory of Impasse-Driven Learning, in H. MANDL, A. LESGOLD (eds), Learning Issues for Intelligent Tutoring Systems, Berlin: Springer, 1988, 19-41.
[15] http://lls.informatik.uni-oldenburg.de
[16] http://www.tu-ilmenau.de/mile
[17] http://www.mathematik.tu-darmstadt.de/ags/ag1/Software/software_de.html
[18] http://www.lpa.co.uk

Autorenangaben:

Dipl.-Ing. Vera Yakimchuk
Cand.-Inf. Hilke Garbe
Prof. Dr. Claus Möbus
Heinz-Jürgen Thole
OFFIS, Escherweg 2
26121 Oldenburg
Tel.: (0441) 798-2379
Fax: (0441) 798-2196
E-mail: {Yakimchuk, Garbe, Moebus, Thole}@informatik.uni-oldenburg.de

Vera Yakimchuk, Hilke Garbe, Claus Möbus, Heinz-Jürgen Thole,
Jan-Patrick Osterloh, Lars Weber

Eine intelligente Problemlöseumgebung für die Grundlagen der Elektrotechnik

Zielstellung des mileET-Projektes [1] ist die Entwicklung einer wissensbasierten Selbstlernumgebung, die Studierende beim Lösen von Aufgaben zu ausgewählten Themen der Grundlagen der Elektrotechnik (GET) unterstützt [2].

Um eine spätere online-Benutzung des Systems zu ermöglichen, wird die Benutzungsoberfläche in Java [3] realisiert. Sie bietet eine Arbeitsumgebung, die zur Zeit einen Schaltungseditor (Abbildung 1) und einen Formeleditor (Abbildung 2) zur Verfügung stellt.
Zur Speicherung der Aufgaben und des Bearbeitungsstandes werden XML-Formate [4] entworfen. Abbildung 3 zeigt beispielhaft eine XML-Datei, die eine Schaltung mit einem Widerstand enthält.
Die intelligente Diagnosekomponente wird mit Hilfe von Prolog [5] implementiert. Sie ermöglicht u.a. adaptive Hilfestellungen, so dass der Lernende beim Lösen der Aufgaben unterstützt wird.
Der LPA Intelligence Server [5] bietet eine Schnittstelle für die Kommunikation zwischen Java und Prolog.

Schaltungseditor

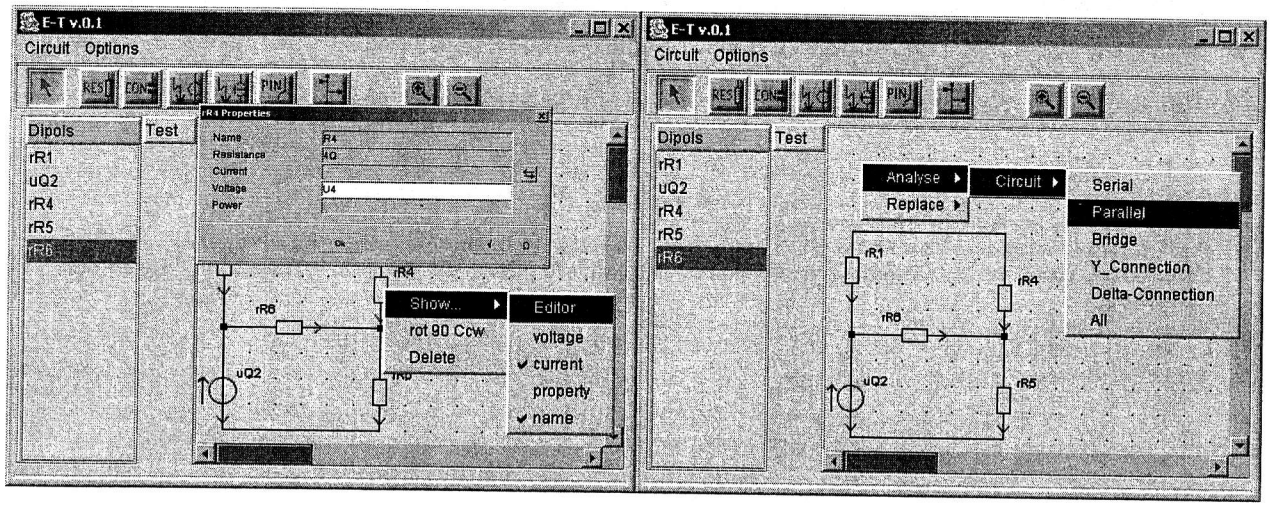

Abbildung (1): Schaltungseditor (Entwurf)

Formeleditor

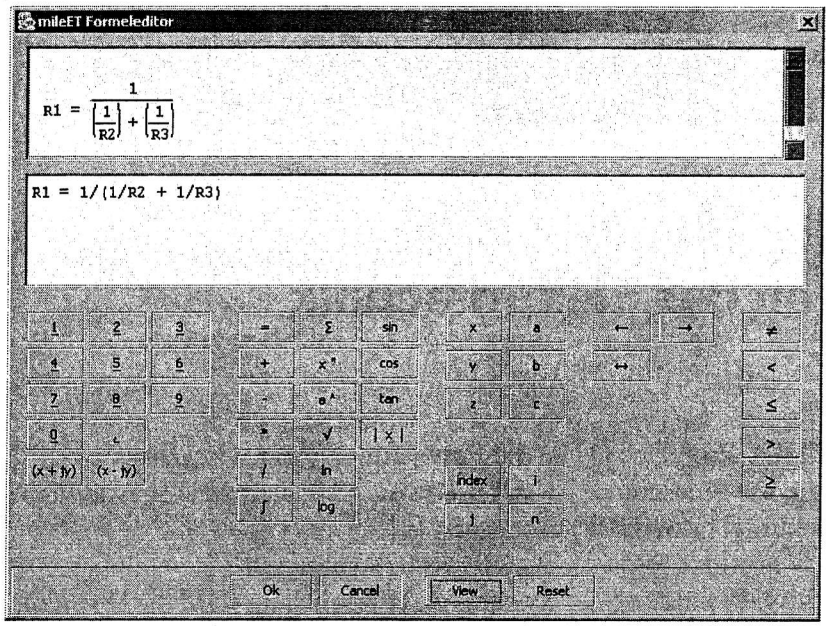

Abbildung (2): Formeleditor (Entwurf)

XML-Beispiel

```
- <circuit idcounter="4" snap="20">
  - <component x="280.0" y="260.0" inverted="false" direction="vertikal" id="2">
      <type>resistor</type>
      <name>rR2</name>
      <point nr="1" offset="20" flip="false">p1</point>
      <point nr="2" offset="0" flip="false">p2</point>
      <property type_long="Name" type_short="" direction="No Orientation">rR2</property>
      <property type_long="Resistance" type_short="R" direction="No Orientation">R2</property>
      <property type_long="Current" type_short="I" direction="N1 --> N2">I2</property>
      <property type_long="Voltage" type_short="U" direction="N1 --> N2">U2</property>
      <property type_long="Power" type_short="P" direction="No Orientation">P2</property>
    </component>
</circuit>
```

Abbildung 1: Ausschnitt aus der XML-Datei einer Schaltung

Literatur

[1] http://lls.informatik.uni-oldenburg.de/projekte/et/index.html

[2] Vera Yakimchuk, Hilke Garbe, Claus Möbus, Heinz-Jürgen Thole, Eine intelligente Problemlöseumgebung für die Grundlagen der Elektrotechnik, Vortrag und Veröffentlichung zum 6. Workshop „Multimedia für Bildung und Wirtschaft“ TU Ilmenau 26.-27.09.2002 (in diesem Heft)

[3] http://www.sun.com

[4] http://www.w3.org/XML/

[5] http://www.lpa.co.uk