16. Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen

Modellierung und Simulation von intelligenten Gebäudesystemen mit attributierten Petri-Netzen

Bernd Eichenauer, IBE Simulation Engineering GmbH
Klaus Scherer, Fraunhofer-IMS und inHaus-Zentrum Duisburg

Abstract: Es werden drei allgemeine Forderungen an eine Methodik zur Modellierung realer Prozesse angegeben und untersucht, inwieweit man diese Forderungen mit den zwei gängigsten Modellierungsmethoden erfüllen kann. Danach wird die Modellierungsprache MSL (Modeling and Simulation Language) skizziert und am Heizungsmodell für ein intelligentes Haus demonstriert.

1. Die Modellierungsmethode

1.1 Anforderungen

Bei der Planung von kommerziellen und technischen Systemen werden zunehmend Simulatoren eingesetzt, um auch das geforderte dynamische Verhalten solcher Systeme, d.h. die ordnungsgemäße Wirkungsweise der Prozesse und Funktionen, schon vor ihrer Realisierung zu überprüfen. Für diese Überprüfung ist zunächst ein ausführbares Modell des geplanten Systems zu erstellen, in dem neben den statischen auch alle für das Zielsystem geforderten dynamischen Eigenschaften aufzunehmen sind. Speziell sind bei der Modellierung von intelligenten Haussystemen Kombinationen der Informationsflüsse, Informationsverarbeitungsschritte und der Architektur des Datennetzwerks mit den physikalischen Prozessen der Gebäude und den sonstigen Prozessen, wie z.B. den Materialflüssen für die Ver- und Entsorgung, zu betrachten.

Wie die zahlreichen derzeit in Verwendung befindlichen Spezifikationssysteme für reale Systeme zeigen, gibt es sehr unterschiedliche Auffassungen darüber, wie Spezifikationen zu erstellen sind und was sie leisten sollen. Deshalb zunächst einige grundsätzlichen Gedanken über Spezifikationen und über die damit erreichbaren Ziele.

Spezifikationen werden erstellt, um den Systementwurf zu dokumentieren. Die Dokumentation ist oft das einzige Bindeglied zwischen verschiedenen Entwickler-Generationen, die ein System weiterentwickeln oder auf den neuesten Stand bringen müssen. Für diese und eine zweite wichtige Aufgabe, nämlich das Vermeiden von Planungsfehlern, werden seit den 70-iger Jahren zunehmend Computer eingesetzt, deren Unterstützungsleistung allerdings vornehmlich von den Möglichkeiten und der Performance der verfügbaren Systeme, d.h. von der jeweils verfügbaren Rechenleistung geprägt war.

Leider muß man feststellen, daß die heute gebräuchlichen allgemeinen Spezifikationssysteme für reale Systeme kaum mehr als Dokumentationshilfsmittel sind. Sie sind z.T. Abkömmlinge der in den 80-iger und 90-iger Jahren populären sog. CASE-Tools. Diese erlauben zwar häufig halbformale Systemspezifikationen mit Pseudocode und mit zahlreichen graphischen Darstellungsmöglichkeiten, prüfen aber höchstens die Konsistenz der Spezifikation, nicht deren Inhalt.

Wenn man den Computer als Planungshilfsmittel für kommerzielle und technische Systeme zur Vermeidung von Planungsfehlern einsetzen will, so muß man sich also zunächst darüber klar werden, wo und wie der Computer bei der Planung helfen kann. Daraus resultieren dann die Eingaben, die der Computer für diese Unterstützungsleistung benötigt.

Es ist ein altbewährtes Prinzip, physikalische und technische Systeme in statische und dynamische Anteile aufzuteilen, man denke beispielsweise an die Aufteilung in der Mechanik oder Elektrotechnik. Bei der Spezifikation realer Systeme wird die statische Systemstruktur und werden die Systembausteine durch den Planungsingenieur festgelegt. Der Computer kann hier, abgesehen von Spezialsystemen, im allgemeinen nur sehr wenig helfen, weil der Systemaufbau im wesentlichen durch die Erfahrung der Ingenieure und durch die Kenntnis der jeweils verfügbaren Bausteine geprägt ist. Dagegen kann der Computer sehr gut für die Prüfung und Optimierung der dynamischen Vorgaben eingesetzt werden und ist hier eine echte Hilfe, insbesondere dann, wenn in dem zu entwicklenden System zahlreiche parallele Aktivitäten zu betrachten sind. Die Beurteilung der dynamischen Sachverhalte und die optimale Auslegung der Bausteine macht dem Systementwickler normalerweise erheblich mehr Mühe als die Beurteilung der statischen Systemanteile. Sehr häufig sind die hohen Kosten für Nacharbeiten nach der Fertigstellung eines kommerziellen oder technischen Systems auf die mangelhafte Auslegung bzw. Planung der Prozesse zurückzuführen.

Für die Systembeschreibung, in der auch die dynamischen Anteile formal erfasst werden können, sind heute zwei Verfahren bekannt, nämlich die algebraische Beschreibung und die Nachbildung durch Modellierung. Die algebraische Methode beschreibt das System formal, basiert auf der Beweistheorie, ist wenig benutzerfreundlich und erfordert eine ausgedehnte mathematische Vorbildung. Sie war bisher sehr wenig erfolgreich und kann bisher auch nur sehr eingeschränkt eingesetzt werden. Dagegen erfordert die zweite Methode, bei der ein sog. Simulationsmodell des in Planung befindlichen Systems zu erstellen ist, erheblich weniger mathematische Vorkenntnisse und kann nicht nur von Spezialisten, sondern auch von Anwendern in angemessener Zeit erlernt werden.

Ein Modell kann nach dem Gesagten nur dann als echtes Planungshilfsmittel für die Auslegung und Optimierung realer Systeme eingesetzt werden, wenn es nicht nur den statischen Systemaufbau beschreibt, sondern sich auch dynamisch genau so verhält wie das reale System. Mit anderen Worten: **ein Modell für die Systemplanung muß ablauffähig sein**. Die Ausführung eines Modells wird gemeinhin als Simulation bezeichnet, ein ausführbares Modell heißt Simulationsmodell.

Neben dieser Forderungen gibt es zwei weitere Forderungen, die grundsätzlich an eine Methode zur Modellierung realer Systeme zu stellen sind. Die erste davon ist naheliegend und bedarf wohl keiner weiteren Erläuterung. **Die Methode muß es ermöglichen, ein reales System vollständig zu beschreiben, d.h. alle für die Dokumentation und Planung erforderlichen statischen und dynamischen Angaben für den Systemaufbau müssen darstellbar sein.**

Bei der nächsten Forderung handelt es sich um einen ingenieurmäßigen Gesichtspunkt. Damit ein Modell verständlich und in der praktischen Arbeit verwendet werden kann, beispielsweise um das dynamische Verhalten eines technischen Prozesses zu verstehen und zu verifizieren, sollte ein möglichst einfacher Zusammenhang zwischen Modell und Realität bestehen. Kann der Anwender die Realität visuell auch im Modell wiedererkennen, so lässt sich das Modell erheblich einfacher verwenden und verifizieren. Es kann darüber hinaus auch leichter zum Experimentieren verwendet werden, um während der Entwicklung den Entwurf zu verändern und zu optimieren. **Damit eine adäquate, anwendungs-bezogene Modellierung möglich ist, muß die Methode es gestatten, die Modelle in naher Anlehnung an die Realität aufzubauen.** Diese Forderung wird bei der Bereitstellung von Modellierungsmethoden leider selten berücksichtigt.

Im folgenden wird festgestellt, inwieweit die beiden heute üblichen Methoden für die Systemmodellierung, die objekt-orientierten Methode und die Petri-Netz-Methode, diese drei Forderungen erfüllen. Danach wird eine kombinierte Methode skizziert, mit der alle drei Forderungen erfüllt werden können.

1.2 Objekt-orientierte Methode

Wegen der Vielzahl von Veröffentlichungen (z.B. [1-5]) zum Thema Modellierung könnte man den Eindruck gewinnen, dass die Methodenauswahl kein grundsätzliches Problem darstellt. Beispielsweise zählt die objektorientierte Vorgehensweise heute unstrittig zu den erfolgreichsten Methoden, die in unzähligen Programmiervorhaben eingesetzt wird. Deshalb wird man auch bei der Modellierung der realen Systeme zunächst an einen rein objektorientierten Entwurf denken. Damit kann man jedenfalls die zweite der obigen Forderung (vollständige Beschreibung) gut befriedigen.

Man stellt jedoch sehr bald fest, dass diese Methode, die bei der Programmierung, d.h. bei der Modellierung erdachter Zusammenhänge, erfolgreich eingesetzt wird, bei der Modellierung von realen zeitabhängigen Systemen wesentliche Mängel aufzeigt. Da nur Botschaften zwischen Objekten betrachtet werden, fehlt oft der Bezug zur Realität, d.h. das Verhalten, z.B. die Bewegung und Veränderung der realen Objekte, geht aus dem Modell nicht ohne weiteres hervor. Deshalb sind die Abläufe, die man in der Realität wahrnimmt, im Modell schlecht nachzuvollziehen, häufig sogar kaum wiederzuerkennen. Die am Markt befindlichen objekt-orientierten Modellierungstools, wie z.B. UML (siehe z.B. [3-5]), sind an die klassischen CASE-Tools angelehnt, modellieren unter Zuhilfenahme zahlreicher Graphiken und sind meist nicht ablauffähig. Allgemeine Ergänzungen verschiedener Hersteller für die Ausführung von objekt-orientierten Modellen beschränken sich meist darauf, die gerade in Ausführung befindlichen Botschaften namentlich aufzulisten, was im allgemeinen wenig hilfreich ist.

1.3 Petri-Netze

Auch für die Erfüllung der beiden weiteren Forderung gibt es zahlreiche Ansätze (z.B. [6-9]), die zum größten Teil auf der Theorie der Petri-Netze basieren. Man versucht hier durch geeignete Gestaltung des Netzes die Topologie des realen Systems nachzubilden. Allerdings führt die algorithmische Verarbeitung im Netz und die Kommunikation mit dem Anwender in klassischen und farbigen Petri-Netzen häufig zu aufgeblähten Netzen und verhindert eine kompakte Modellierung. Auch die Zuordnung von realen Objekten, die in den Prozessen verarbeitet werden, zu Marken lässt sich nicht durchsichtig darstellen. Beispielsweise kann man in vielen klassischen und farbigen Petri-Netzen ein reales Objekt nicht generell durch eine einzelne Marke repräsentieren, weil Marken nicht alle jeweils relevanten Eigenschaften der realen Objekte zuordenbar sind. Auch verhindern Konzepte wie die Wertigkeit eines Konnektors, die unter anderem für der Realisierung von Berechnungen, von Prioritäten und von unscharfen Werten eingesetzt wird, eine solche Interpretationsmöglichkeit.

1.4 Die MSL-Methode

Es ist danach klar, dass die bisher in der Literatur propagierten Modellierungsmethoden gute Ansätze darstellen, aber insgesamt noch keine befriedigende Lösung für die Modellierung realer Prozesse bieten. Die objekt-orientierte Methode stellt ausreichende Möglichkeiten für die strukturierte Formulierung von algorithmischen Zusammenhängen bereit, während die Abläufe, die z.B. in einem technischen Prozess stattfinden, nicht visuell nachvollzogen werden können. Mit Petri-Netzen ist es dagegen möglich, die realen Abläufe zeitgerecht und animierbar darzustellen. Es fehlen aber unter anderem die algorithmischen Hilfsmittel für Bearbeitung der innerhalb der Prozesse anfallenden Daten.

Aufbauend auf den beiden genannten Methoden wurde deshalb von der IBE Simulation Engineering GmbH vor einigen Jahren die Modellierungssprache MSL (Modeling and Simulation Language) definiert und in der Folge weiter ausgebaut, mit der alle drei oben genannten Forderungen erfüllbar sind. MSL ist als Teil des Simulator-Entwicklungs-System PACE [10] implementiert und und wird seit etwa zehn Jahren in zahlreichen Vorhaben erfolgreich eingesetzt (siehe z.B. [11-17]). Um das System realitätsnah zu modellieren, enthält MSL

- Sprachelemente zur Beschreibung von Prozessen,
- Sprachelemente zur Darstellung der Objekte, die verarbeitet werden und
- Sprachelemente zur Beschreibung der Verarbeitung innerhalb der Prozesse.

Grundlegende Sprachelemente zur Beschreibung von Prozessen kann man von den Petri-Netzen übernehmen. In MSL werden die Verarbeitungsschritte mit verallgemeinerten Stellen, Transitionen und Konnektoren beschrieben. Die Verallgemeinerung besteht darin, daß diese Petri-Netz-Elemente mit zahlreichen Attributen versehen werden können, die den detaillierten Ablauf der Prozesse bestimmen. Hinzu kommen zwei neue MSL-Elemente für die hierarchische Strukturierung eines Modells, nämlich Module und Kanäle. Module enthalten beliebige wiederverwendbare Teilnetze, die über Schnittstellen (Stellen, Kanäle) in ein beliebiges Netz integriert werden. Kanäle sind wie Stellen passive Netzelemente, mit denen mehrfache Verbindungen zwischen Modulen zusammengefasst werden.

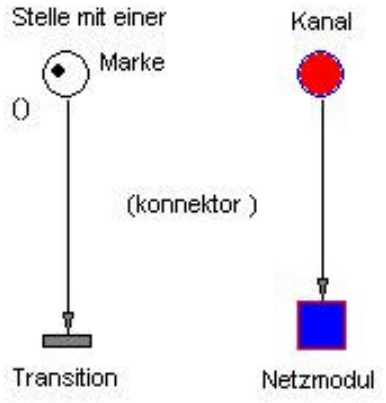

Abb. 1: Netzelemente in MSL

Auch die Objektdarstellung wurde an die Vorgaben in Petri-Netzen angelehnt. Objekte in MSL sind passive, zeitunabhängige, physikalische oder logische Größen (Bauteile, Formulare, Autos, Daten usw.), die im Modell als Einheit behandelt werden können. Sie werden jeweils durch eine Marke dargestellt, die durch das Netz wandert und die charakteristischen Eigenschaften des realen Objekts als Attribute trägt. Bearbeitungsvorgänge an Objekten werden in den Transitionen vorgenommen und verändern die charakteristischen Größen.

Für das Attributieren der Netzelemente (sog. Inskribieren) wird die objekt-orientierte Programmiersprache Smalltalk-80 verwendet. Sie ermöglicht eine anwender-freundliche Programmierung bei der Anpassung von Netzelementen an die jeweilige Aufgabe. Smalltalk wird für alle Texteinfügungen für Netz verwendet, also für Namen, Marken-Attribute, Konnektor-Attribute, Extra-Codes, Transitions-Codes, usw.

Eine weitergehende Darstellung von MSL würde den vorliegenden Rahmen überschreiten. Deshalb werden einige der Konstrukte im folgenden bei der Darstellung des inHaus-Modells erläutert.

2. MSL-Beispiel eines Heizungssystems für das intelligente Haus

2.1 Aufgabe des Modells

Das verteilte Heizungssystem (HeatingControl) regelt die Raumtemperatur im "intelligenten Haus" nach Vorgaben der Bewohner über ein PDA oder durch direkte Eingaben am Hauscomputer (ResidentialServer). Das vorliegende aus Platzgründen etwas vereinfachte Beispiel soll die Temperaturregelung im Wohnzimmer und Schlafzimmer beschreiben.

Alle vorzusehenden Geräte sind über ein Netz miteinander verbunden. Die Kommunikation über das Netz soll im Modell funktionell dargestellt werden.[1]

2.2 Das Modellfenster "HeatingController"

Das Fenster "HeatingController" stellt die oberste Ebene, den Hauptmodul, des hierarchisch aufgebauten MSL-Modells dar und zeigt die physikalisch vorliegenden Strukturelemente des Heizungssystems. Um das Netz verständlicher zu machen, wurden die Standardsymbole für Module durch geeignete Graphiken ersetzt.

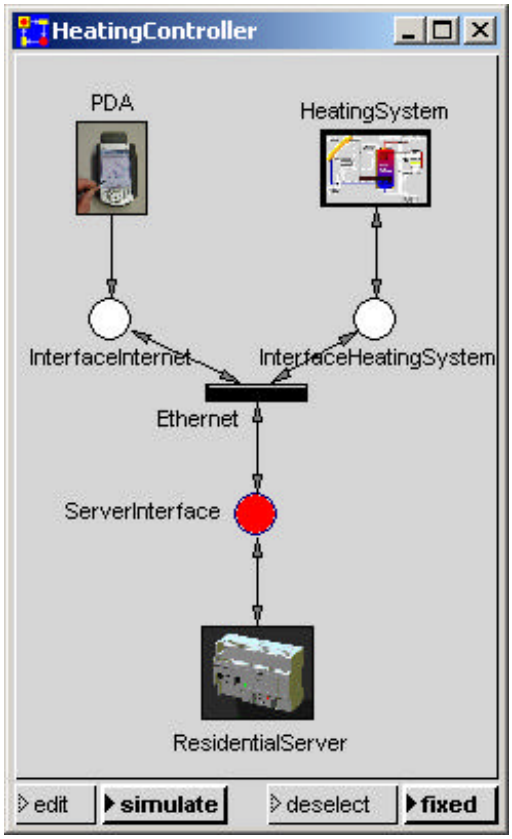

Abb. 2: Oberste Modellebene

Man erkennt die vier Module, aus denen das Heizungssystem aufgebaut ist bzw. die vom Heizungssystem mitverwendet werden:

- ResidentialServer
  Der ResidentialServer sorgt unter anderem dafür, dass die Zimmertemperatur im Haus zeitgerecht die von den Bewohnern vorgegebene Temperatur annimmt. Er nimmt Anforderungen der Bewohner entgegen und sorgt für deren zeitgerechte Erfüllung durch Versenden von Anweisungen an das HeatingSystem.

- HeatingSystem
  Das HeatingSystem nimmt Anweisungen des ResidentialServer zur Einstellung der Zimmertemperatur entgegen und sorgt für die entsprechende Temperatureinstellung.

- PDA
  Das Gerät PDA wird von den Bewohnern verwendet, um dem ResidentialServer die Vorgaben über die Zimmertemperatur mitzuteilen.

- Ethernet
  Die Kommunikation zwischen den drei genannten Bausteinen des Heizungssystem erfolgt über das hausinterne Datennetz.

2.3 Der Modul "Ethernet"

Über den Modul "Ethernet" verkehren alle weiteren Bausteine des Heizungssystems miteinander. Er soll deshalb zuerst verfeinert werden.

Eine Anweisungen für einen Baustein des Heizungssystems besteht aus vier Argumenten: (rec send op data). Darin bezeichnet rec den Empfänger (receiver), send den Absender (sender), op den Anweisungscode (operation) und mit data sind ggf. zu übertragende Daten gemeint, die für die Ausführung der Anweisung erforderlich sind.

Man erkennt links und rechts oben die beiden Stellen "InterfaceInternet" und "InterfaceHeatingSystem" die schon im Hauptmodul vorkamen. Sie sind hier schwächer gezeichnet, weil sie nicht im Modul "Ethernet" sondern in dem hierarchisch darüber liegenden Modul "Hea-

[1] In einer späteren Ausbaustufe des Modells soll auch die verteilte Architektur des Datennetzes abgebildet werden.

tingController" vereinbart sind. Zusammen mit den beiden unten liegenden schwach gezeichneten Stellen, die den Kanal "ServerInterface" bilden, stellen sie die Schnittstellen des Moduls "Ethernet" dar.

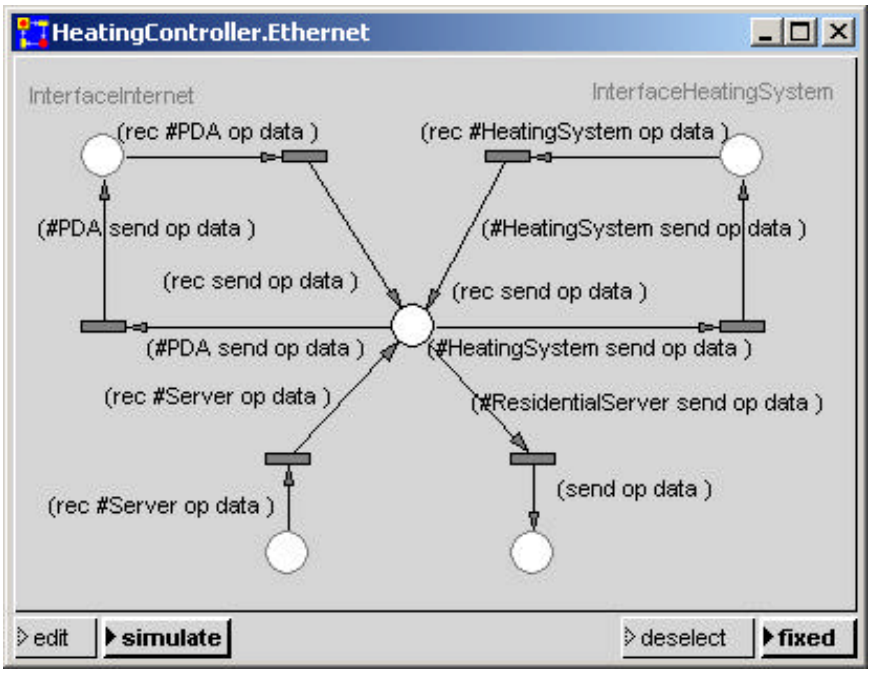

Abb. 3: Der Modul "Ethernet"

Konnektoren geben nicht nur die Richtung an, in der MSL-Objekte (Marken) laufen, sondern wählen auch den jeweils gewünschten Weg aus, wenn mehrere Alternativen vorliegen. Letzteres erfolgt über die Attributierung der Konnektoren. Nur wenn die Attribute einer Marke mit den Attributen eines Konnektors übereinstimmen, darf die Marke passieren.

Alle Konnektoren zwischen den Netzelementen sind nach dem oben beschriebenen Anweisungsformat attributiert. Damit eine einlaufende Anweisung richtig weitergeleitet wird, ist der erste Parameter eine sog. Konnektor-Konstante, die das Gerät bezeichnet, das die Anweisung ausführen soll. Die übrigen Attribute der Konnektoren geben Konnektor-Variablen an, können also beliebige Daten aufnehmen.

2.4 Der Hauscomputer (ResidentialServer)

Der ResidentialServer nimmt Anweisungen des PDA entgegen, die festlegen, zu welchen Zeiten die Zimmertemperatur bestimmte Werte annehmen soll. Er weist dann das Heizungssystem zeitgerecht an, die geforderte Temperatur einzustellen.

Für die Bearbeitung der Tabellen, in denen die Zeiten und zugeordneten Temperaturen gespeichert werden, sind in dem vorliegenden Modell fünf Anweisungen vorgesehen:

- Insert a Time
- Delete aTime
- Read Time Tables
- Show Time Tables
- Init Time Tables

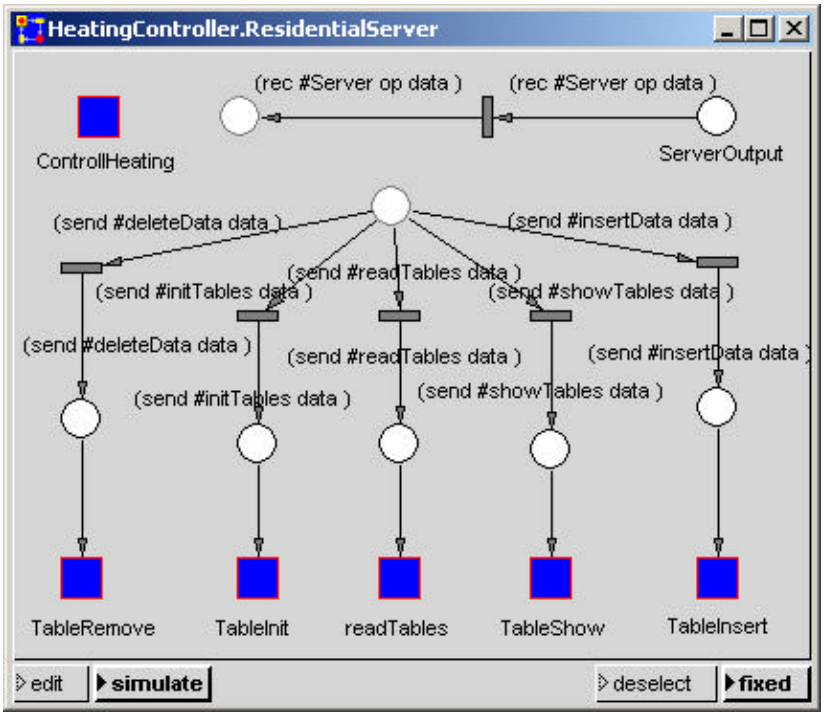

Abb. 4: Der ResidentialServer

Die Anweisungen werden mit den fünf Untermodulen "TableInsert, "TableRemove", "readTables", "TableShow" und "TableInit" implementiert. Die Module sind alle einfach und ähnlich aufgebaut. Hier wird nur der Modul "TableInsert" verfeinert (Abb.5).

Die einlaufenden Daten werden in der oberen Transition den Konnektor-Variablen "day", "room", "clock" und "temp" zugewiesen, die dann über die Konnektoren zu der unteren Transition transportiert werden. Der unter der unteren Transition angegebene, ihr zugeordnete Aktionscode verarbeitet diese Daten und erweitert die Zeittafeln entsprechend.

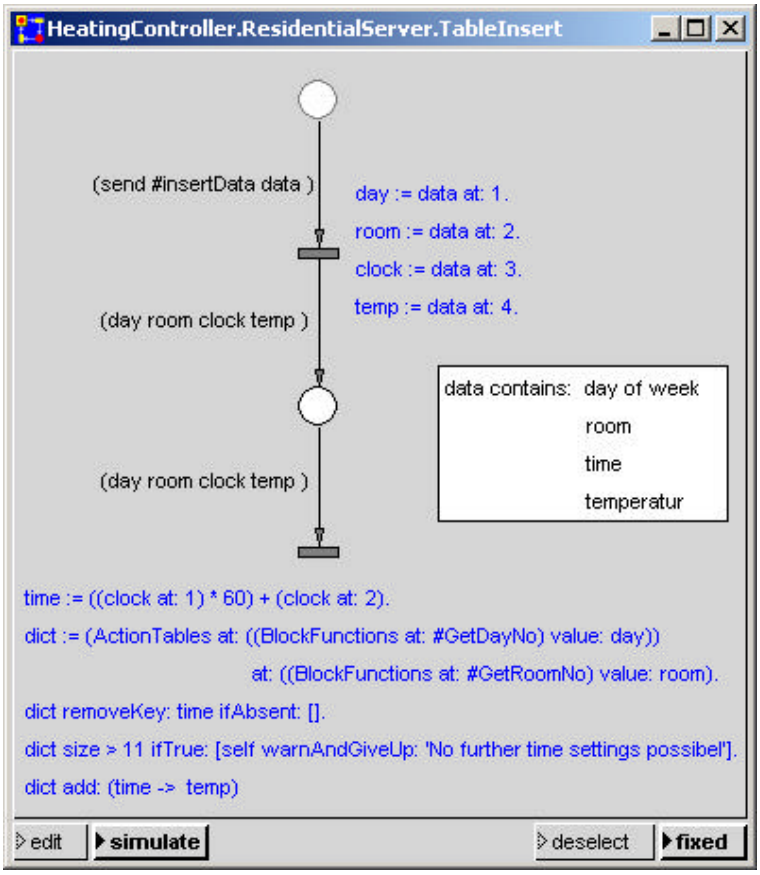

Abb. 5: Erweitern der Zeittafel

Der Modul zeigt, wie in MSL Transitionen mit Aktionscode an vorgegebene Aufgaben angepasst werden. Neben dem Aktionscode können Transitionen noch mit zwei weiteren Codes attributiert werden. Der Bedingungscode verhindert die Ausführung der weiteren Transitionscodes (das Schalten der Transition), bis eine vorgegebene Bedingung erfüllt ist, der Verzögerungscode verzögert die Ausführung um eine vorgegebene Zeitspanne.

Eine weitere Eigenschaft von MSL zeigt Abb. 4. Der Modul "ControlHeating", der hier aus Platzgründen nicht näher beschrieben wird, untersucht jede Minute, ob eine Zeit für das Ändern von Zimmertemperaturen erreicht wurde. Ist das der Fall, so wird für jede Änderungsanweisung eine Marke mit den entsprechenden Daten in die Stelle "ServerOutput" gelegt, die danach über das Netz zum "HeatingSystem" transportiert wird. Während in klassischen Petri-Netzen nur synchrone Ereignisse stattfinden, ermöglicht MSL auf die Weise auch die Modellierung asynchroner Ereignisse.

2.5 Das Heizungssystem

Das Heizungssystem ist in Abb. 6 dargestellt. Es besteht aus den beiden Radiatoren "LvRRadiator" und "SiRRadiator" im Wohn- und Schlafzimmer, deren Ankopplung an das Hausnetz und dem Boiler.

Für die Einstellung der Raumtemperaturen wurde ein einfaches lineares Modell verwendet, welches auf den Temperatur-Differenzen zwischen Raumtemperatur und Radiator-Temperatur und zwischen Raumtemperatur und Außentemperatur beruht. In einer zukünftigen Version kann das Verfahren anhand der im inHaus gemessenen Wärmeflüsse weiter präzisiert werden.

Die Radiatortemperatur wird aus der aktuellen Zimmertemperatur "acttemp", der Solltemperatur "solltemp", der Boilertemperatur "boilertemp" und der Ventilstellung am Radiator "ventilstellung" nach der folgenden Formel berechnet:

```
radiatortemp := boilertemp * (solltemp > acttemp
        ifTrue: [ventilstellung := ventilstellung +
                        (ventilstellung <= 0.9 ifTrue: [0.1 * (1 - (acttemp / solltemp))]
                                                ifFalse: [0])]
        ifFalse: [ventilstellung := ventilstellung –
                        (ventilstellung >= 0.1 ifTrue: [0.1 * (1 - (solltemp / acttemp))]
                                                ifFalse: [0])]).
```

Die neue aktuelle Raumtemperatur acttemp ergibt sich aus:

```
acttemp := acttemp + ( K2 * (outsidetemp - acttemp))
                   + ( K1 * (radiatortemp - acttemp)).
```

Die Faktoren K1 und K2 sind Kennwerte für die Wärmeübergänge.

Der Boiler stellt die Wassertemperatur des Heizungssystems nach einer Charakteristik ein, die graphisch vorgegeben wird. Abb. 7 zeigt eine solche Charakteristik. Die Abszisse gibt die Außentemperatur, die Ordinate die gewünschte Wassertemperatur im Heizungssystem (Boiler) an.

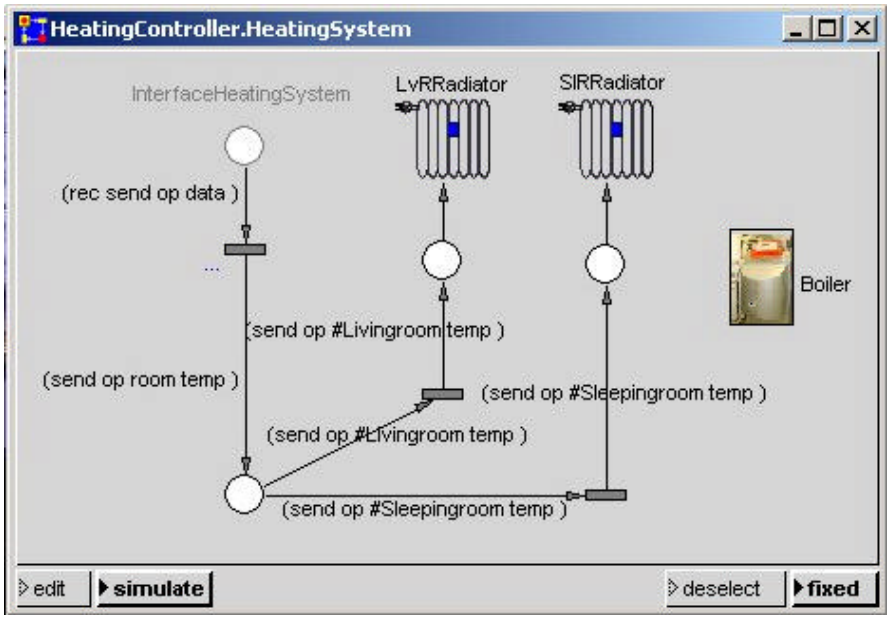

Abb. 6: Das Heizungssystem

3. Die Simulationsumgebung

Die Verwendung der objekt-orientierten Programmiersprache Smalltalk als Inskriptionssprache bietet zahlreiche Vorteile. Einer dieser Vorteile ist die umfangreiche Smalltalk-Standard-Bibliothek von Methoden, die direkt aus einem MSL-Programm aufrufbar sind. Eine große Anzahl von weiteren Methoden wurde im Rahmen der MSL-Implementierung in dem Simulator-Entwicklungs-System PACE bereitgestellt. Darunter sind insbesondere die Methoden für die graphischen Ein/Ausgabe hervorzuheben, mit denen attraktive Simulatoroberflächen erstellt werden können.

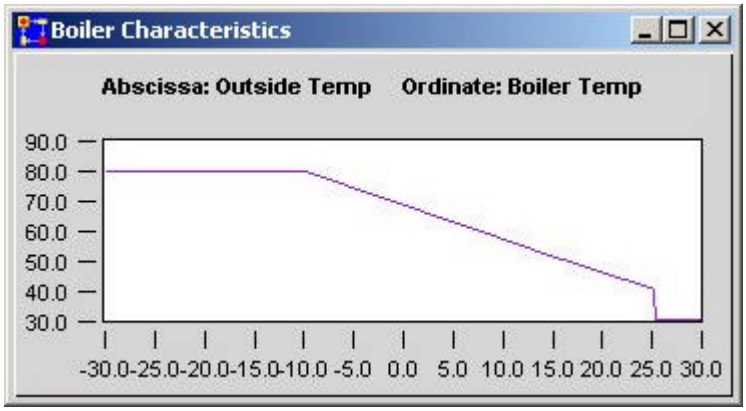

Abb. 7: Boilertemperatur als Funktion der Außentemperatur

Abb. 8 zeigt die mit PACE, Version 5 hergestellte Oberfläche eines Simulators für das beschriebene Heizungssystem. Speziell wird die Szene "Livingroom" angezeigt, die mit dem "Select Scenery" Fenster rechts oben ausgewählt wurde.

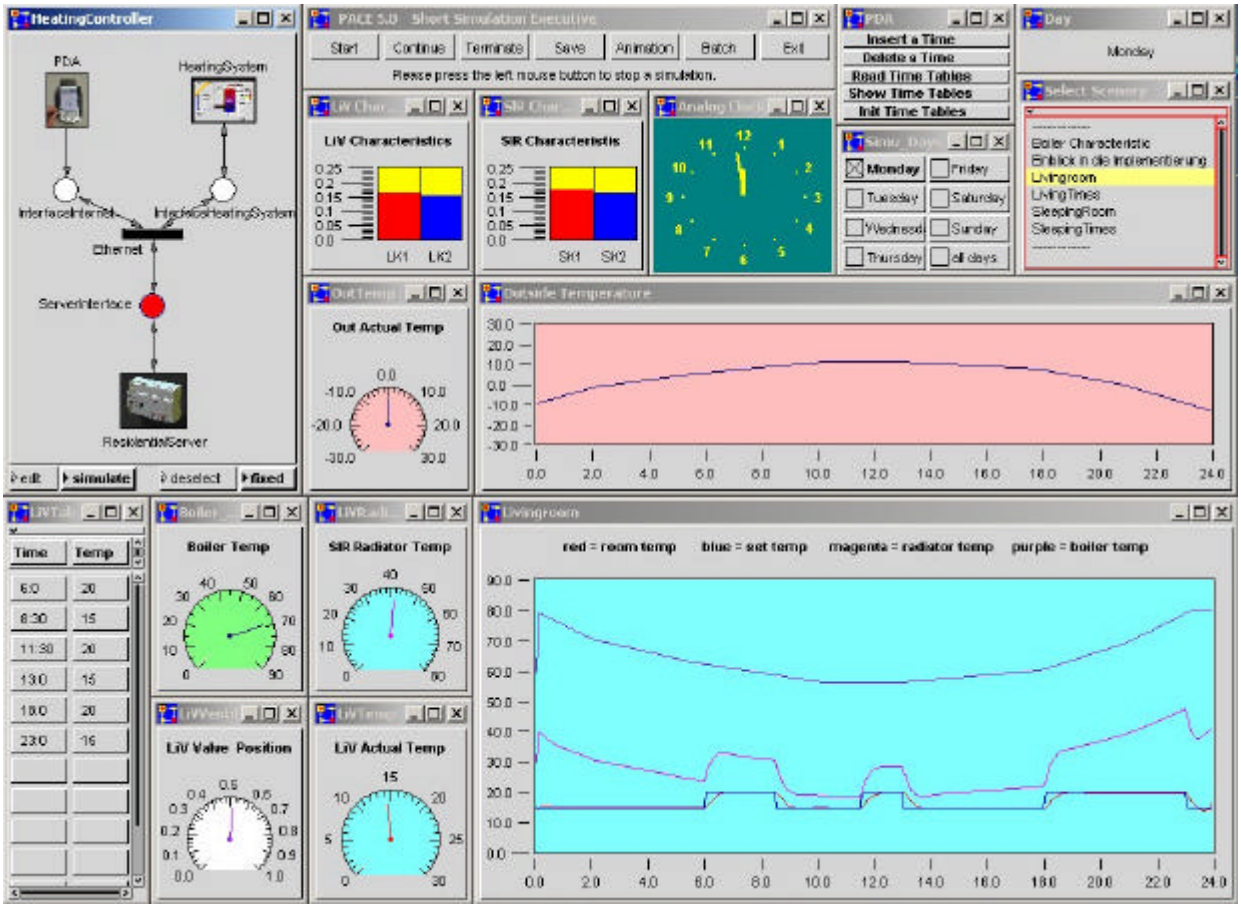

Abb. 8: Simulatoroberfläche mit Instrumenten und Temperaturkurven für das Wohnzimmer

Man erkennt die Benutzeroberfläche des PDA mit den früher beschriebenen Funktionen. Beim Drücken einer Taste öffnet sich ein Fenster für die Eingabe weiterer Daten. Beispielsweise öffnet sich nach Drücken der Taste "Insert a Time" das in Abb. 9 dargestellt Eingabefenster.

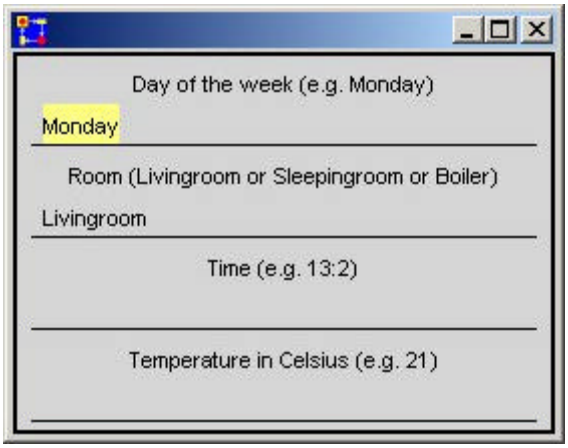

Abb. 9: Eingabefenster

Mit den mehrfachen Balken-Schiebereglern "LiV Characteristics" und "SiR Characteristics" können die in Abschnitt 2 beschriebenen Kennwerte K1 und K2 für das Wohnzimmer und das Schlafzimmer manuell eingestellt werden.

Das Verhalten des Heizungssystems in Abhängigkeit von der Außentemperatur geht aus den Zeigerinstrumenten und den Kurvenfenstern hervor. Mit der Kurve im Fenster "Outside Temperature", die manuell eingegeben wird, kann das Verhalten des Heizungssystems unter verschiedenen Belastungen untersucht werden. Dargestellt werden die Zimmertemperatur (rot), die Vorgabetemperatur (blau), die Radiatortemperatur (magenta-rot) und die Boilertemperatur (purpur). In Abb. 10 ist das Verhalten des Heizungssystems nochmals für einen extremen Verlauf der Außentemperatur dargestellt.

4. Schlussbemerkung

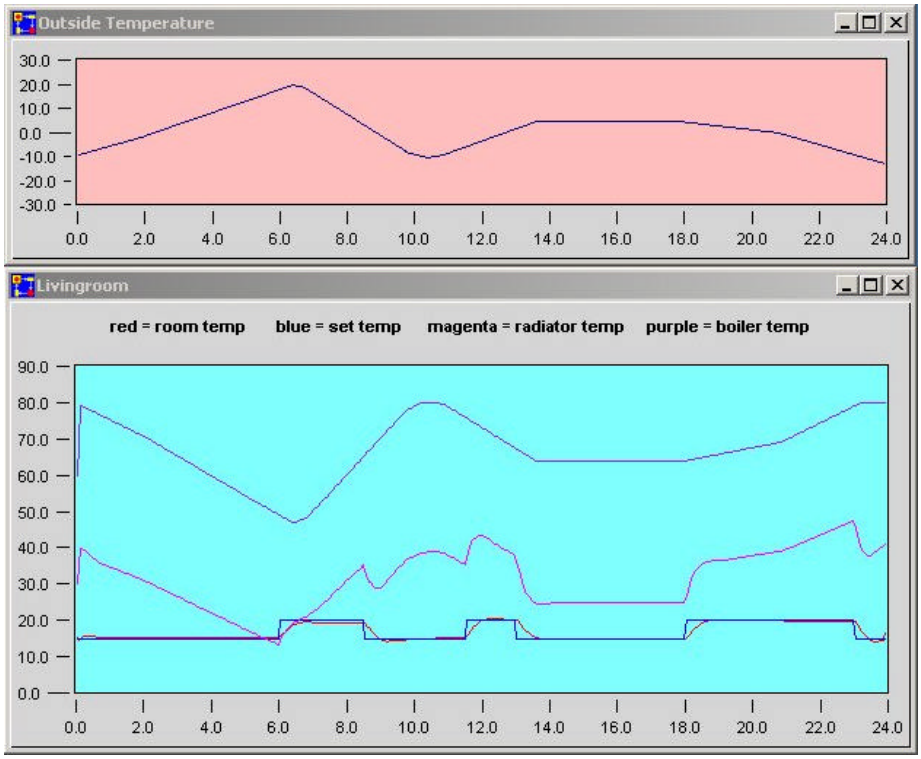

Abb. 10: Verhalten des Heizungssystems unter extremen Bedingungen

Wie das Beispiel erkennen lässt, können mit MSL beliebige diskrete Prozesse bis zu jeder gewünschten Tiefe mit angemessenem Aufwand modelliert und die Modelle unter Verwendung der Simulationsumgebung PACE ausgeführt werden. Mit auf zukünftigen Untersuchungen der inHaus-Physik basierenden Erweiterungen des Modells wird es möglich sein, Abläufe und die Ressourcenverwendung im intelligenten Haus schon in der Planungsphase eines Projekts genau zu studieren und mit weiteren, hier nicht beschriebenen Möglichkeiten von PACE zu optimieren.

Literatur

[1] S. Shlaer, S.J.Mellor: Object-oriented Systems Analysis – Modeling the World in Data, Yourdon Press, 1988, ISBN 0-13-629023-X

[2] J. Robertson / S. Robertson: Vollständige Systemanalyse, Hanser Verlag München Wien, 1996, ISBN 3-446-18115-6

[3] J. Rumbaugh, I. Jacobson, G. Booch: The Unified Modeling Language Reference Manual, Addison-Wesley-Longman, 1999, ISBN 0-201-30998-X

[4] R. Burkhardt: UML – Unified Modeling Language: Objektorientierte Modellierung für die Praxis, Addison-Wesley-Longman, 1999, ISBN 3-8273-1407-0

[5] P. Forbrig: Objektorientierte Softwareentwicklung mit UML, Fachbuchverlag Leipzig, 2001, ISBN 3-446-21975-7

[6] D. Abel: Petri-Netze für Ingenieure – Modellbildung und Analyse diskret gesteuerter Systeme, Springer Verlag Berlin Heidelberg 1990, ISBN 3-540-51814-2

[7] B. Baumgarten: Petri-Netze – Grundlagen und Anwendungen, BI-Wiss.-Verl., 1990, ISBN 3-411-14291-X

[8] E. Schnieder (Hrsg.): Petrinetze in der Automatisierungstechnik, Oldenburg Verlag München, 1992, ISBN 3-486-22045-4

[9] K. Jensen, Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Vol. 1, Springer Verlag, 1992, ISBN 3-540-55597-8

[10] PACE 5.0 Benutzerhandbuch, IBE Simulation Engineering GmbH, 2002, www.ibepace.com

[11] G. Feistl: Findet das Nadelöhr – Simulation von Verpackungsprozessen, neue verpackung 4, 1998, S. 32-34

[12] U. Dietel, F. Bennemann: Reihenfolgebildung von Gießaufträgen bei einem sächsischen Metallhersteller, in: M. Rabe, B. Hellingrath: Handlungsanleitung Simulation in

Produktion und Logistik – Ein Leitfaden mit Beispielen für kleinere und mittlere Unternehmen, SCS International, 2001, ISBN 1-56555-226-1
[13] U. Dietel, H.-J. Hanisch: Simulation des Bereichs Wärmebehandlung in der Porzellanherstellung, in: M. Rabe, B. Hellingrath: Handlungsanleitung Simulation in Produktion und Logistik – Ein Leitfaden mit Beispielen für kleinere und mittlere Unternehmen, SCS International, 2001, ISBN 1-56555-226-1
[14] V. Franz: Techniken der Simulation in der Praxis, BFT 5, 2001, S. 20 - 23
[15] M. Enkelmann: Simulation in der Betonsteinproduktion, BFT 5, 2001, S. 24 - 28
[16] S.M.O. Fabricius, E. Badreddin: Stochastic Petri Net Modeling for Availability and Maintainability Analysis, Proceedings of 14th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management (COMADEM), 2001, Manchester, UK
[17] C. Böhnlein: Modellierung des Bullwhip-Effekts mit Hilfe höherer Petri-Netze, wisu, 31 (2002) 8-9, S. 1124-1127