

Relationen zwischen Modellen, Daten und Methoden - Ein Konzept für ein IT-System im Umweltbereich -

Jochen Wittmann¹

Abstract

Nach einer Analyse bestehender Softwaresysteme, die zeigt, daß sich diese Systeme für die Behandlung von Modellen im Umweltbereich nur bedingt eignen, wird ein neuartiges Architekturkonzept vorgestellt.

Der Ansatz beruht auf einer einheitlichen Behandlung der verwendeten Modelle, Daten und Methoden. Als Daten werden in diesem Rahmen sowohl raum- als auch zeitbezogene Datenmengen erlaubt. Methoden können in Form von Modulen des neukonzipierten IT-Systems, aber auch als externe Programme vorliegen.

Die Grundidee des Systems besteht darin, die Beziehungen zwischen Modellen, Daten und Methoden zu erfassen und bei den jeweiligen Informationen als Verweise zu vermerken. Auf diese Weise entsteht ein Beziehungsgeflecht, in dem Modellbeschreibung, Modelldaten und Systemdaten direkt miteinander verknüpft sind. Die einheitliche Betrachtung von Modellen, Daten und Methoden als Objekte des Systems erlaubt eine saubere Behandlung von zeit- und raumbezogenen Prozessen und eine Integration von GIS- und Simulationsfunktionalität.

Dieses Beziehungsgeflecht kann traversiert werden, um Benutzeranfragen nach zukünftigen Systemzuständen automatisch zu beantworten. Dazu, wie auch zum Aufbau des Beziehungsgeflechtes, wird eine formalsprachliche Schnittstelle angeboten.

1 Problemanalyse

Das aktuelle Bild im Bereich der Softwareunterstützung für Umweltmodelle gestaltet sich unübersichtlich: Datenbanksysteme verwalten die in großem Umfang gesammelten Systemdaten, Geographische Informationssysteme konzentrieren sich auf die Verarbeitung von raumbezogenen Daten und Simulationssysteme sowie Statistiksysteme helfen bei der Entwicklung und Abarbeitung von dynamischen bzw. statistischen Modellen.

Die Funktionalität, die diese Systeme bieten, soll nun den Anforderungen gegenübergestellt werden, die der Anwender bei der Bearbeitung von Problemen aus dem

¹ Universität Rostock, Lehrstuhl Modellierung und Simulation, Fachbereich Informatik, D-18051 Rostock, email: wittmann@informatik.uni-rostock.de

Umweltbereich hat. Eine Übersicht über das Anwendungsspektrum bieten z.B. die Sammelbände von (Goodchild et al. 1996) oder von (Bascompte/Sole 1998). Hier sind insbesondere zu nennen:

- große Datenbestände, die häufig mit sehr großem Aufwand gesammelt werden mußten
- Inhomogenität der Daten
- Raum- und Zeitbezug der Daten
- Verwendung anspruchsvoller Verfahren zur Simulation, Visualisierung und der statistischen Analyse.

Es ist festzustellen, daß kommerziell verfügbare Softwaresysteme diesen Charakteristika von Umweltmodellen jeweils nur zum Teil gerecht werden. Eine präzise Analyse bietet das University Consortium for Geographic Information Science in seinem White-Paper (UCGIS 1997). Bezüglich der verwendeten Software läßt sich konstatieren:

Datenbanksysteme haben erhebliche Schwierigkeiten mit dem Raum- und Zeitbezug. Temporale Datenbanken arbeiten mit den Begriffen Änderungszeit und Gültigkeitszeit und verwenden einen Zeitbegriff, der zu dem im Simulationsbereich üblichen vollständig inkompatibel ist (Becker 1996, Myrach 1997). Darüber hinaus sind mehrdimensionale Daten und komplexe nutzereigene Verfahren in Datenbanken nur schwer integrierbar.

Geographische Informationssysteme hingegen bewältigen den Raumbezug, zeigen jedoch ebenfalls Schwächen bei der Darstellung der Systemdynamik und der Abarbeitung von dynamischen Modellen. Der Modellbegriff in diesem Bereich beschränkt sich auf die Abbildung statischer Abhängigkeitsbeziehungen in Form von algebraischen Gleichungen (siehe die Benutzerhandbücher von ArcView (ESRI 1996) oder IdrisiW (Eastman 1997)).

Simulatoren kommen naturgemäß mit der Modellbeschreibung und deren Abarbeitung sehr gut zurecht, ihre Schwächen liegen in den Bereichen der Verarbeitung von raumbezogenen Daten und der strukturierten Speicherung von großen Datenmengen (Lütkepohl et al. 1993). Ein Ausweichen auf Ansätze, die mit einer Rasterung des Raumes im Sinne von zellularen Systemen arbeiten, bieten wegen der geringen Flexibilität der Modellbeschreibung ebenfalls keine überzeugende Alternative (z.B. Maxwell/Costanza 1997, Beins-Franke et al. 1995).

Demzufolge benötigt der Anwender zur Bearbeitung seiner Fragen in der Regel Funktionalitäten aus mindestens zwei der aufgezählten Typen von Softwaresystemen. Unter diesen Gegebenheiten bleiben ihm letztlich zur Bewältigung seiner Aufgabe nur die folgenden beiden Alternativen: Entweder er realisiert für seine jeweilige Aufgabenstellung ein speziell auf diese zugeschnittenes Softwaresystem oder aber er realisiert eine Kopplung der bestehenden Softwareprodukte (Richter et al. 1997).

Beide Varianten erweisen sich als wenig effizient bzw. sehr fehleranfällig: Der Speziallösung mangelt es an der notwendigen Änderungsfreundlichkeit und Flexibilität. Die Kopplungsvariante führt zu aufwendigen Transformationsfunktionen zur Datenkonversion und zu Konsistenzproblemen, da die erforderlichen Daten in der Regel in mindestens zwei Softwarekomponenten gespeichert sind.

In der Literatur finden sich nur sehr wenige Ansätze, die sich dieser softwaretechnologischen Problematik widmen und Vorschläge für eine geschlossene Softwarelösung versuchen (z.B. Fedra 1994, 1996 oder Conrad 1996). Hier reiht sich die vorliegende Arbeit ein.

2 Lösungsidee

Ansatzpunkt des neuen Vorschlags ist die Übertragung der Konzepte der objektorientierten Analyse auf den Bereich der Umweltmodellierung und der dort verwendeten Softwaresysteme.

Die Grundidee ist dabei die folgende: Daten, Modelle und Methoden werden als Objekte aufgefaßt, die das neuartige IT-System, das im übrigen auch intern entsprechend des objektorientierten Paradigmas strukturiert ist, verwaltet. Aus dieser Sicht ergeben sich drei wesentliche Vorteile:

1. Es existiert eine einheitliche, problemnahe Darstellung für Modelle, Daten und Methoden.
2. Es sind beliebige Beziehungen zwischen den Objekten darstellbar. Dieses Feature verwendet das System dank der unter 1. dargestellten einheitlichen Repräsentation auf zwei Ebenen:
 - Ebene der Modellbeschreibung:
Beziehungen, die im realen System gelten (z.B. 'ist Teil von', 'ist Vater', ...), können explizit ins Datenmodell übernommen werden.
 - Ebene der Methodenspezifikation
Beziehungen zwischen den Daten und den sie verarbeitenden Methoden (z.B. Anwendbarkeit von Methode M_i auf Datum j) sind auf gleicher Ebene (also ohne zusätzliche Metainformationen) darstellbar.
3. Zur Beantwortung von Benutzeranfragen lassen sich die Beziehungen anschließend automatisch auswerten.

Es wird gezeigt, daß die konsequente Eingabe von Daten, Modellen, Methoden und deren wechselseitigen Beziehungen gemäß der hier vorgeschlagenen Strukturierung einen sehr einfachen Zugriff in Form einer Anfragesprache erlaubt, bei dem fehlende Informationen gegebenenfalls durch einen automatischen Methodenaufruf generiert werden können.

Die folgenden Abschnitte stellen die verschiedenen Komponenten der Systemarchitektur dar und folgen dabei den Arbeitsschritten im Verlauf einer Simulationsstudie. Dabei werden die in Abbildung 1 gezeigten Strukturen sukzessive aufgebaut.

2.1 Modellbeschreibung

Die Beschreibung des Modells erfolgt in mehreren Schritten. Zunächst werden sämtliche relevanten Modellgrößen zusammengestellt. Wie bei Simulationssprachen üblich werden die Bestimmungsgleichungen für Zustandsgrößen und algebraische Größen angegeben. Intern werden dabei die funktionalen Abhängigkeiten zwischen den Modellgrößen vermerkt, indem die Relation 'ist funktional abhängig von' aufgebaut wird. Die folgende Tabelle zeigt ein Beispiel:

Eingegebenes Gleichungssystem	Relationen
$z_j := a + b$	z_j ist abhängig von a z_j ist abhängig von b
$a := b^2 - 2z$	a ist abhängig von b a ist abhängig von z
$b := 5$	—

Im zweiten Schritt faßt der Anwender die Modellgrößen und Gleichungen zu Gruppen zusammen. Eine Gruppe von Attributen beschreibt die Eigenschaften eines in der realen Welt existierenden Objektes. Der Name des Objektes wird direkt ins Modell übernommen und dient als Identifikator für die Gruppe der zugeordneten Modellgrößen. Die zugeordneten Modellgrößen entsprechen damit Eigenschaften dieser Objekte.

Im Softwarekonzept werden die Modellgrößen daher Attribute genannt, wohingegen die bei der Modellierung eingeführten Objekte Reale-Welt-Objekte heißen sollen, um die Analogie zum realen System zu betonen. Intern wird die Zugehörigkeit von Attributen zu Objekten durch die Relation 'ist Attribut von' gespeichert und verwaltet.

Bis zu dieser Stelle entspricht die Vorgehensweise der konventionellen Modellierung, wenn man davon absieht, daß die funktionalen Abhängigkeiten zwischen Attributen in konventionellen Systemen nicht explizit gespeichert werden. Der folgende Abschnitt allerdings geht über die Möglichkeiten bestehender Systeme hinaus und integriert Konzepte des objektorientierten Designs in die Beschreibung von Simulationsmodellen.

Hat der Nutzer nämlich Attribute und Reale-Welt-Objekte definiert, erlaubt es das System, beliebige Beziehungen zwischen den Objekten abzubilden. Definiert werden diese Beziehungen durch die Angabe eines Namens für die Relation und die

Menge der zum betrachteten Objekt in Beziehung stehenden anderen Reale-Welt-Objekte. Auf diese Weise lassen sich beliebige Abhängigkeiten zwischen den Objekten modellieren, auch solche, die bei der Verwendung von konventionellen modular-hierarchischen Modellierungskonzepten (z.B. Simplex in (Schmidt 1995) oder (Grützner 1995)) nicht realisierbar sind. Diese Systeme beschränken sich nämlich auf die Modellierung von Input- und Outputrelationen sowie die Relationen zum Aufbau der Komponentenhierarchie, während der neue Ansatz beliebige, nutzerdefinierbare Relationen (z.B. 'ist Vater von', 'wohnt in', usw.) erlaubt.

Betrachtet man die bisher beschriebenen Modellierungsmöglichkeiten, so sieht man, daß das Konzept bezüglich der Art der Attribute keinerlei Beschränkungen unterworfen ist. Insbesondere ist die Angabe von Ortskoordinaten als Objekt-Attribut möglich. Die zeitliche Dimension kommt durch die Angabe von Differentialgleichungen ins Spiel, die als Erzeugungsregeln für zukünftige Modellzustände interpretiert werden können. Sie stellen neben den Beziehungen, die aus algebraischen Gleichungen entstehen, eine zweite Beziehungsklasse auf der Ebene der Attribute.

Bei näherer Betrachtung erkennt man, daß die so beschriebenen Reale-Welt-Objekte eine Klassenbeschreibung bzw. ein generisches Modell darstellen. Will man real existierende Objekte modellieren, müssen die Attribute und Relationen mit konkreten Werten bzw. mit Identifikatoren für die Objekte belegt werden.

In der Klassenbeschreibung der Reale-Welt-Objekte werden also die Attribute und die Beziehungen zwischen den Attributen in Form von Gleichungen beschrieben, während die potentiell möglichen Beziehungen zwischen den Objekten durch vom Anwender frei wählbare Relationen repräsentiert sind. Eine Ausprägung einer solchen Klasse enthält dagegen konkrete Werte für die Attribute und die Identifikatoren für die mit dem Objekt tatsächlich in Relation stehender Objekte.

2.2 Meßdaten und Ergebnisdaten

Zur Modellvalidierung, aber auch zur Auswertung von Modellexperimenten muß das IT-System eine Vielzahl von Daten verwalten. Die Struktur dieser Daten erweist sich als unabhängig vom speziellen Modell und der jeweiligen Anwendung. Daher stellt das System vorgefertigte Klassen zur Aufnahme der Daten zur Verfügung. Im Bereich der Daten beschränken sich die Aktionen der Nutzers damit auf das Erzeugen von Ausprägungen der im folgenden aufgezählten sogenannten Datenobjekte:

1. Tabellen zur Aufnahme von beliebigen, tabellarisch strukturierten Datenmengen
2. Zeitreihen zur Aufnahme von Tabellen, deren eine Dimension die Zeit darstellt
3. Layer zur Aufnahme von raumbezogenen Datenmengen analog der Repräsentation im GIS (Raster- und/oder Vektordaten)
4. Landkarten (Maps) zur Beschreibung der Komposition von verschiedenen Layern.

Alle diese Klassen für Datenobjekte zeichnen sich dadurch aus, daß sie aus einer Menge von Headerinformationen bestehen, auf die die eigentlichen Einträge mit den Nutzdaten folgen.

Die Besonderheit des vorgeschlagenen Systemkonzeptes besteht nun darin, daß die Einträge in diesen Datenobjekten mit den Objekten und Attributen aus dem Bereich der Reale-Welt-Objekte eng verknüpft werden.

Beim Erzeugen eines Datenobjektes wird vom Anwender verlangt, daß er nicht nur textuell beschreibt, welche Informationen das neue Datenobjekt enthält, sondern daß er Verweise auf die entsprechenden Attribute bzw. Objekte im Bereich der Reale-Welt-Objekte setzt. Auf diese Weise ist stets eine eindeutige Zuordnung der Daten zu den entsprechenden Modellgrößen gewährleistet. Folgt man diesen Verweisen in umgekehrter Richtung, so lassen sich zu einem Attribut sehr leicht sämtliche im System gespeicherte Daten ermitteln.

Ein ähnlicher Mechanismus verhindert die mehrfache Speicherung eines Datums und damit auch die eingangs erwähnten Konsistenzprobleme. Als Einträge in den Datenobjekten sind nämlich nicht nur Werte selbst, sondern auch Verweise auf Werte erlaubt. Diese Verweise können sowohl auf Reale-Welt-Objekte und deren Attribute weisen als auch auf andere Datenobjekte.

Beispiel:

Datenobjekt Map mit einem Datenobjekt Layer, das die Populationszahl einer Region beinhaltet, die wiederum als Attributwert im Bereich der Realen-Welt-Objekte gespeichert ist.

Drückt man diese Zusammenhänge als Relationen aus, so kann die Relation 'enthält Daten für' zwischen Datenobjekten untereinander sowie zwischen Datenobjekten und Reale-Welt-Objekten bzw. deren Attributen bestehen. Hervorzuheben ist, daß sämtliche Daten, die das System verwaltet, direkt an die entsprechenden Objekte des Modells gebunden sind, eine Vorgehensweise, die die Einführung von Metainformation überflüssig macht.

Werden die Daten als Meßdaten ins System eingebracht, sind die Relationen vom Anwender anzugeben. Anders sieht es aus, wenn die Daten von im System integrierten Methoden generiert werden. Wie solche Methoden zu spezifizieren sind, damit die Relationen automatisch eingetragen werden können, zeigt der folgende Abschnitt.

2.3 Methoden

Die Funktionalität des Softwaresystems steckt in dessen Methoden. Im neuen Systemkonzept sollen diese Methoden zur Unterscheidung von dem je nach Kontext unterschiedlich belegten Methodenbegriff 'Solver' genannt werden.

Das System unterscheidet dabei nicht, ob die Funktionalitäten dem Bereich der Simulationstechnik, der Geographischen Informationssysteme oder der Analysewerkzeuge zuzuordnen sind. Es setzt lediglich zweierlei voraus:

1. Die Schnittstelle der Methoden ist unter Verwendung von Objektklassen zu spezifizieren, die dem System bekannt sind. Für die Input-Seite heißt dies, daß die Angabe von Real-World-Objekt(-klassen) oder Datenobjekt(-klassen) möglich ist. Als Output sind lediglich die oben genannten vier Datenobjekt(-klassen) erlaubt.

Beispiel:

Eine Solver-Klasse Euler-Integration erzeugt zu allen Attributen vom Typ Zustandsgröße einer dem Solver übergebenen Objektklasse eine Zeitreihe zwischen den Zeitpunkten $TStart$ und $TEnd$.

2. Bei der Methodenspezifikation muß angegeben sein, in welcher Relation die Output-Objekte zu den Input-Objekten stehen.

Beispiel:

Für das Input-Objekt wird das ein Datenobjekt Zeitreihe erzeugt mit Werten für die Zeitpunkte $TStart + k * TStep$, wobei $TStart$, k , $TStep$ ebenfalls Input-Objekte darstellen. Damit wird eine Beziehungskette zwischen Input, Solver und Output-Objekten aufgebaut, die das System explizit speichert.

Eine Instanz einer Solver-Klasse, ein Solver, entspricht dann einem Aufruf der Methode mit einer konkreten Parametrisierung. Dabei werden die in der Schnittstellenbeschreibung angegebenen Datenobjekte als Ergebnisse erzeugt.

Für die Einbindung von Solvoren und Solver-Klassen in das System ist nun wesentlich, daß die Beziehungen zu Input- und Output-Objekten bzw. Objektklassen explizit vermerkt werden. Angelegt werden die Relationen sowohl auf Klassen- als auch auf Ausprägungsebene:

1. zwischen Reale-Welt-Objekt(-Klassen) und Solver(-Klassen): 'ist Input für'
2. zwischen Attributen bzw. Attributwerten und Solver(-Klassen): 'ist Input für'
3. zwischen Datenobjekt(-Klassen) und Solver(-Klassen): 'ist Output von'

Damit ergibt sich folgendes Bild für das Beziehungsgeflecht zwischen Daten, Modellen und Methoden (siehe Abbildung 1). Das Bild macht zweierlei deutlich:

1. Der Bereich der klassischen Modellierung mit Modellgleichungen und modular-hierarchisch kombinierbaren Modellkomponenten ist durch Attribute (Modellgrößen), Beziehungen zwischen Attributen (Modellgleichungen), Beziehungen zwischen Attributen und Reale-Welt-Objekten (Zuordnung von Modellgrößen zu Modellkomponenten) sowie durch die Beziehungen zwischen Reale-Welt-Objekten untereinander (Komponentenhierarchie) abgebildet.

Die vorgeschlagene Struktur zwingt den Modellierer zu einer strengen Unterscheidung zwischen generischem Modell (Reale-Welt-Objekte und Attribute auf

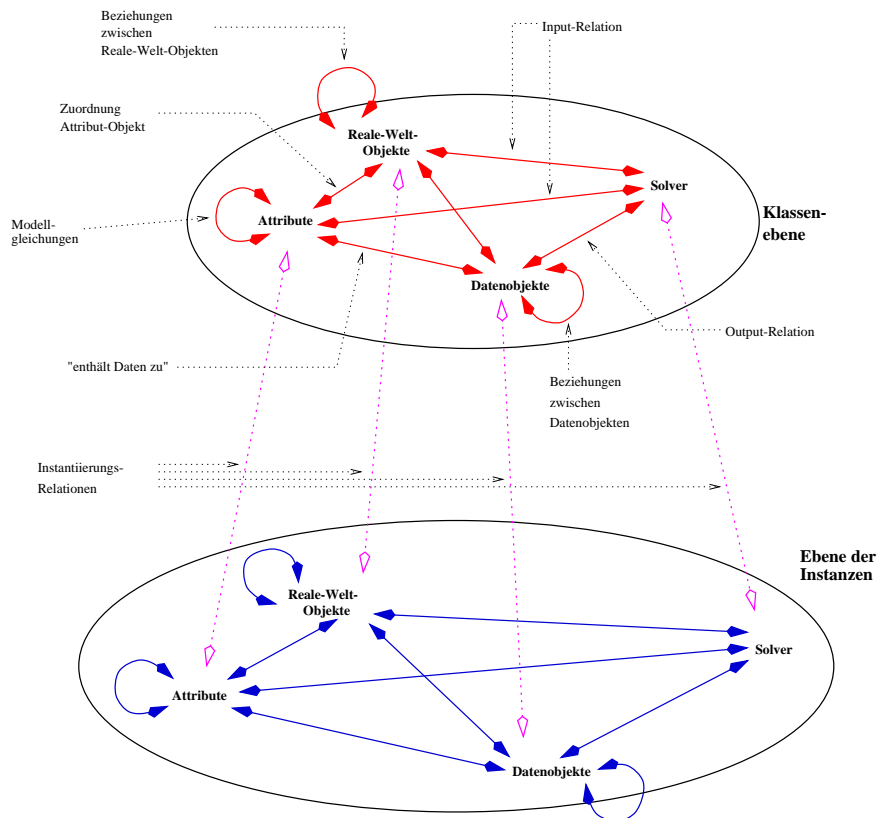


Abbildung 1
Die Struktur des Informationssystems

Klassenebene) und dem vollständig parametrisierten Modell, das einer Instanz mit entsprechender Wertebelegung für die Attribute entspricht.

- Die Möglichkeiten der objektorientierten Modellierung sind integriert, indem die Beziehungen zwischen Reale-Welt-Objekten nicht nur auf Input-, Outputbeziehungen und die Beschreibung einer Komponentenhierarchie (Sub- bzw. Superkomponentenbeziehungen) beschränkt bleiben, sondern daß beliebige Relationen (z.B. 'ist Vater von') explizit modelliert werden können.
- Anders als andere Ansätze zur objektorientierten Modellierung von Umweltsystemen bezieht dieser Ansatz neben der Modellierung des realen Systems (Attribute und Reale-Welt-Objekte) auch die Beziehungen zu Meß- und Experimentdaten und den beteiligten Methoden mit ein.

Unter diesem Aspekt betrachtet wird nicht allein das Modell des zu untersuchenden Wirklichkeitsausschnittes dargestellt, sondern auch dessen Einbindung in das Softwaresystem. Dies gelingt, indem auch die Komponenten des Softwaresystems mit objektorientierten Konzepten beschrieben werden und somit Relationen zwischen Modellobjekten, Datenobjekten und Solverobjekten explizit angegeben, verwaltet und -wie zu zeigen sein wird- auch ausgewertet werden können.

3 Benutzerschnittstelle

Es bleibt nun zu klären, wie die Benutzerschnittstelle für das vorgeschlagene System zu gestalten ist. Diese Aufgabe gliedert sich in die beiden Teilprobleme:

- die Modifikation des Beziehungsgeflechtes durch Einfügen, Ändern und Löschen von Objektklassen, Objekten und Relationen.
- die Nutzungsphase des Beziehungsgeflechtes, d.h. die Art und Weise, in der der Anwender Informationen anfragt und Antworten vom System erhält.

Der erste Punkt soll an dieser Stelle nur angedeutet werden: Der Anwender gibt die neuen Klassen bzw. Objekte über eine Sprachschnittstelle analog einer Modellbeschreibungssprache ein. Die Eingabe wird interpretiert und die entsprechenden Klassen und Objekte werden angelegt.

Es ist darauf hinzuweisen, daß das zunächst sehr unübersichtlich erscheinende Beziehungsgeflecht auf diese Weise sukzessive aufgebaut werden kann, und die vom Anwender anzugebenden Relationen sich stets nur auf die direkt mit dem neuen Objekt in Beziehung stehenden Objekte beziehen. Die hohe Komplexität entsteht erst durch die Vielzahl der Relationen und deren Verschachtelung. Die Beherrschung einer auf diese Weise entstehenden Komplexität ist allerdings sehr leicht durch rekursiv arbeitende Algorithmen zur Verwaltung der jeweiligen Beziehungsklassen möglich.

Der zweite Punkt zeigt die Vorteile des Systemkonzeptes und soll hier näher erläutert werden. Die Grundidee ist dabei die folgende: Der Anwender interessiert sich für den Wert eines bestimmten Objektes oder Attributes. Dies spezifiziert er, indem er die Hierarchie der Realen-Welt-Objekte traversiert und an der entsprechenden Stelle eine Anfrage initiiert. Zur Beantwortung einer solchen Anfrage durch das System sind die folgenden Fälle zu unterscheiden:

1. Der Attributwert ist, wie in einer konventionellen Datenbank, direkt beim Attribut des Objektes vermerkt. Der Wert kann also direkt zurückgeliefert werden.
2. Ähnlich verläuft die Anfrage nach einem zukünftigen Attributwert. Im Bereich der Reale-Welt-Objekte sind die zum aktuellen Zeitpunkt gültigen Werte verzeichnet. In den Datenobjekten sind sämtliche darüber hinausgehende Daten zu dem Attribut gesammelt wie zum Beispiel alternative Messungen der Systemgröße, in früheren Simulationsläufen ermittelte Daten, usw. Stimmt die nachge-

fragte Zeit nun mit der aktuellen nicht überein, so werden die Relationen hin zu diesen Datenobjekten (vgl. Abbildung 1) verfolgt und dort nachgesehen, ob der gesuchte Wert im System bereits vorliegt. Auf diese Weise greift der Anwender auf sämtliche, in früheren Experimenten ermittelte Information zu, ohne deren Einordnung in die Speicherstrukturen (z.B. Experimentname, Meßreihe, Name der Datendatei, usw.) kennen zu müssen.

3. Handelt es sich um raumbezogene Attribute, so erfolgt die Anfrage in der Regel auf Klassenebene. Beispielsweise nach dem Attributwert Populationsdichte von allen Ausprägungen der Reale-Welt-Objektklasse 'Stadt'. Über diese Ebene kann der Benutzer sämtliche Ausprägungen einer Klasse ansprechen, das System verfolgt die Instantiierungsrelation auf die Ausprägungsebene, verfolgt die Relation vom gewählten Attribut hin zu einem Datenobjekt zur Darstellung und ist so in der Lage, eine Karte mit der räumlichen Verteilung der Attributwerte zurückzuliefern.

In diesem Fall zeigen sich die Vorzüge der expliziten Verwaltung der Relationen zwischen den Objekten: Die angesprochene Klasse trägt Verweise auf ihre Ausprägungen, von den Ausprägungen führen Relationen zu den Datenobjekten. Findet sich nun ein Datenobjekt, das Werte für das gesuchte Attribut enthält, und findet sich zweitens eine Referenz von der Reale-Welt-Objekt-Klasse zur Datenobjekt-Klasse, die sicherstellt, daß die Werte dieses Attributes sinnvoll dargestellt werden können, so kann ein neues Datenobjekt vom Typ Layer instantiiert werden, das mit den auf dem ersten Weg gefundenen Werten parametrisiert wird und die Werte des gewünschten Attributes anzeigt.

4. Der vierte Fall tritt dann in Kraft, wenn die vorangegangenen drei Suchschritte ohne Erfolg verlaufen sind, das gesuchte Datum im System also (noch) nicht vorliegt. Unter Ausnutzung der Relationen wird dann nämlich versucht, einen Solver bzw. eine Solverfolge zu finden und auszuführen, die den gesuchten Wert als Output generiert und als Datenobjekt ablegt. Dies kann weitestgehend automatisch erfolgen; im folgenden ist ein entsprechender Algorithmus skizziert:
 - (a) Zunächst folgt man der Relation, die zwischen Objekt bzw. Attribut und der entsprechenden Klasse besteht, und begibt sich auf Klassenebene. Man kennt nun den Typ des gesuchten Wertes (z.B. Zustandsgröße).
 - (b) Im zweiten Schritt sucht man bei den Solverklassen eine Klasse, die ein Datenobjekt erzeugt, das Werte des unter (a) gefundenen Typs enthalten kann.
 - (c) Anschließend sammelt man sämtliche Inputs der in (b) gefundenen Solverklasse, indem man die Input-Relation zurückverfolgt zu den Objektklassen und Attributen und untersucht ...
 - (d) durch Verfolgen der Instantiierungsrelation, ob die benötigten Objekte und Attributwerte auf Instanzen-Ebene vorliegen. An dieser Stelle kann der

Suchalgorithmus für nicht gefundene Objekte bzw. Attributwerte rekursiv aufgerufen werden.

- (e) Liegen die notwendigen Inputs vor, wird die gefundene Solverklasse mit ihnen parametrisiert, ein Solverobjekt erzeugt und dessen Ausführung veranlaßt. Als Ergebnis wird ein Datenobjekt mit dem ursprünglich angeforderten Wert generiert und vom Solver automatisch die entsprechende Relation zwischen Datenobjekt und Reale-Welt-Objekt eingetragen. Damit ist das gesuchte Datum für die Anfrage (vergleiche Punkt (2) der Anfragebearbeitung) zugänglich.

Wie man sieht, veranlaßt dieser Algorithmus automatisch die Generierung des gesuchten Wertes. Lediglich an zwei Punkten können interaktive Eingriffe notwendig werden:

- Das System findet alternative Solverklassen, die zum Erfolg führen: In diesem Fall erscheint es sinnvoll, dem Anwender die Auswahl anzutragen (außer es läßt sich eine Solverklasse als sinnvoller Default-Wert eintragen).
- Nicht alle Werte, die zur Ausführung benötigt werden, können gefunden werden: Die fehlenden Werte (z.B. für Methodenparameter bzw. fehlende Anfangszustände) müssen dann interaktiv nachgefordert werden.

4 Zusammenfassung und Ausblick

Im folgenden sollen die wesentlichen Charakteristika des vorgeschlagenen Softwarekonzeptes thesenhaft zusammengefaßt werden:

- Der Ansatz vereinigt die Möglichkeiten konventioneller Modellbeschreibungssprachen (Definitionsgleichungen, modular-hierarchische Modellkomponenten) mit den Ausdrucksmöglichkeiten der objektorientierten Modellierung (beliebige Relationen zwischen Modellobjekten).
- Bei dieser Sicht ist der Raum- und Zeitbezug der Attribute eines Objektes kein die Modellierungsmöglichkeiten einschränkendes Hindernis. Sämtliche Attribute können in Raum und Zeit differenziert werden. Auf der Spezifikationsebene ist dieses Problem beim vorgeschlagenen Ansatz gelöst. Es bleibt hier auf die Implementierung der Speicherung der Datenobjekte beschränkt (persistente Speicherung mehrdimensionaler Daten).
- Das Objektmodell beschränkt sich nicht auf den Bereich der Modellierung des realen Systems, sondern erstreckt sich auch auf eine Repräsentation der Meß- und Experimentdaten und eine Spezifikation der Methodenschnittstelle. Auf diese Weise wird es möglich, die Beziehungen zwischen Modellen, Daten und Methoden explizit anzugeben und bei Anfragen auszuwerten.

- Aufgrund der modularen, objektorientierten Architektur ist das System leicht erweiterbar bezüglich seiner Daten- und Modellobjekte, aber auch bezüglich seiner Methoden, den Solvern. Auf diese Weise ist der breite, für Umweltprobleme notwendige Funktionsumfang sukzessive aufzubauen. Mehrfachimplementierungen von Systemfunktionalitäten in Datenbank, Geographischem Informationssystem und Simulationssystem werden vermieden.
- Modellerweiterungen stellen Ergänzungen des bestehenden Beziehungsgeflechtes der Realen-Welt-Objekte dar. Entweder werden weitere Objekte eingefügt oder bestehende Objekte werden um weitere Attribute erweitert. Das gesamte zuvor bereits gesammelte Wissen wird durch Verknüpfung der hinzukommenden Objekte bzw. Attribute mit den bereits bestehenden unmittelbar nutzbar. Damit entfallen Mehrfachimplementierungen für Varianten von Modellen für denselben Prozeß des realen Systems (einfache Realisierung von Modellen mit unterschiedlichem Detaillierungsgrad, Skalenproblematik bei geographischen Modellen).
- Alternative Modellstrukturen (z.B. Modellbeschreibung in Raster- bzw. Vektorrepräsentation) können nebeneinander in die Objekthierarchie eingebaut und wahlweise verwendet werden. Die entsprechenden Objekte verfügen dann über Attribute, die der Repräsentation in der einen bzw. der anderen Darstellung entsprechen. Die Beziehungen zwischen den alternativen Darstellungen ist über algebraische Gleichungen spezifiziert, die bei Bedarf die entsprechenden Umrechnungen erlauben und so eine doppelte Abspeicherung der Attributwerte überflüssig machen.
- Die Interpretation der Experimentdaten erfolgt durch direkten Bezug zum Modell mit seinen Attributen auf der einen und zu den Solvern mit ihren Parametern auf der anderen Seite. Zusätzliche Metainformationen, die die Herkunft und/oder die Interpretation der Daten betreffen, sind damit überflüssig.
- Die explizite Verwaltung der Relationen ermöglicht eine intelligente Auswertung der gespeicherten Information, gegebenenfalls sogar die automatische Wahl eines geeigneten Darstellungsverfahrens oder eine automatische Erzeugung der gesuchten Daten.

Erweiterungen des Konzeptes sollen aus Platzgründen ebenfalls nur skizziert werden:

- Im Mehrbenutzerbetrieb ist es vorstellbar, auf einem gemeinsamen Grundstock von Daten, Modellen und Methoden zu arbeiten, den sich der jeweilige Anwender individuell ergänzen kann. Für jede Objektklasse bzw. jedes Objekt können dazu Nutzungsrechte (z.B. angelehnt an die UNIX-Dateibehandlung) vergeben werden, die eine gemeinsame Nutzung der Informationen ohne Konsistenzprobleme und eine für jeden Nutzer individuelle Abspeicherung des Datenmaterials erlauben.

- Analoges gilt für die verteilte Speicherung der einzelnen Objekte. Unter Anwendung der Konzepte verteilter, objektorientierter Datenbanksysteme ist es denkbar, die Objekte (Modelle, Daten und Methoden) verteilt zu speichern. Dem Anwender würde dies konzeptuell verborgen bleiben, allerdings ist wegen der hochgradigen Vernetzung der Objekte über die Relationen mit erheblichen Laufzeitverlängerungen zu rechnen.
- In ähnlicher Weise können auch Daten und Programme eingebunden werden, die nicht im vom System vorgegebenen Format für Solver bzw. Datenobjekte vorliegen. Die entsprechenden Informationen können damit nicht direkt in das Relationengeflecht integriert werden. Indirekt läßt sich eine Integration jedoch sehr einfach realisieren: Man legt die entsprechenden Datenobjekte bzw. Solverobjekte im System an und verbindet diese wie gewohnt über die Angabe der Relationen mit den bereits gespeicherten Objekten. Bezüglich der Funktionalität nutzt man die Tatsache der objektorientierten Strukturierung des gesamten Systems aus und definiert für die externen Objekte die entsprechenden Methoden über. So lassen sich Transformationsfunktionen und Filter elegant einbauen, die Daten und Methoden in beliebigen Fremdformaten dem System in verständlicher Form verfügbar machen.

Es bleibt, die Realisierbarkeit des Konzeptes zu betrachten und den aktuellen Stand der Arbeiten zu beschreiben. Ein erheblicher Einwand sind mit Sicherheit die zu erwartenden Laufzeiten. Angaben über gemessene Rechenzeiten lassen sich derzeit noch nicht machen, da sämtliche Systemkomponenten derzeit erst in Form der Spezifikation und eines Implementierungskonzeptes vorliegen. Eine Umsetzung wird derzeit begonnen.

Allerdings läßt sich auch in diesem Stadium sagen, daß sich die Laufzeiten ausschließlich auf das Antwortverhalten bei Anfragen beziehen, nicht aber auf die Bearbeitung der Modellstruktur bzw. das Einfügen von Datenobjekten und Solvern. Andererseits ist als Pluspunkt für den Ansatz zu konstatieren, daß das Systemkonzept eine völlig neue Arbeitsweise beim Umgang von Modellen vorschlägt, die dem Anwender mit der automatischen Selektion eines Darstellungsverfahrens und der automatischen Generierung von Informationen einen bislang nicht gekannten Bedienungskomfort bietet. Weiterhin wird sich das Suchproblem durch Optimierung des Suchalgorithmus (es handelt sich im wesentlichen um UND/ODER-Bäume, deren Behandlung z.B. in (Niemann 1983) beschrieben ist) und durch vom Benutzer vorgebbare Einschränkung seiner Komplexität drastisch reduzieren lassen. Im Hinblick auf den Bedienungskomfort und die Möglichkeiten, die durch die einheitliche Behandlung von Daten, Modellen und Methoden entstehen, sollte der neue Entwurf prototypisch realisiert und einem eingehenden Test unterzogen werden.

Ein gleitender Übergang von bestehenden Modellen zum neuen Systemkonzept wird dabei durch die Möglichkeiten, der Einbindung bestehender Module, wie sie oben beschrieben wurden, die Akzeptanz bei den Nutzern erhöhen.

Literatur

- Bascompte, J., Sole, R.V. (Hrsg.) (1998): Modeling Spatiotemporal Dynamics in Ecology, Berlin
- Becker, L. et al. (1996): Temporal Support for Geo-Data in Object-Oriented Databases, in: Wagner, R.R., Thoma, H. (Hrsg.): Database and Expert Systems Applications 7th International Conference, DEXA'96, Zürich, Proceedings, Berlin, S. 79-93
- Beins-Franke, A. et al. (1995): Anbindung objektbezogener Modellier- und Analysewerkzeuge an Geographische Informationssysteme – Beispiele für die Modellierung ökologische Prozesse, in: Buziek, G. (Ed.): GIS in Forschung und Praxis, Stuttgart, S. 209-221
- Conrad, R. (1996): Konzepte GIS-basierter Visualisierung gekoppelter Umweltsimulationsmodelle, in: Lessing, H., Lipeck, U.W. (Hrsg.): Informatik für den Umweltschutz. 10. Symposium, Hannover 1996, Marburg, S. 397-407
- Eastman, R. (Ed.) (1997): Idrisi for Windows – User's Guide, Clark University, Worcester
- Environmental Systems Research Institute Inc. ESRI (Ed.) (1996): ArcView GIS – The Geographic Information System for Everyone, Readlands/CA
- Fedra, K. (1994): Model-based Environmental Information and Decision Support Systems, in: Hilty, L.M. et al. (Eds.): Informatik für den Umweltschutz, 1994, Marburg, S. 37-58
- Fedra, K. (1996): Distributed Models and Embedded GIS: Integration Strategies and Case Studies, in: Goodchild, M.F., Steyaert, L.T., Parks, B.O. (Eds.): GIS and Environmental Modeling: Progress and Research Issues, Fort Collins, S. 413-417
- Goodchild, M.F. et al. (Eds.) (1996): GIS and Environmental Modeling: Progress and Research Issues, Fort Collins
- Grützner, R. (1995): Environmental Modeling and Simulation – Applications and Future Requirements, ISES
- Lütkepohl, S. et al. (1993): Datenbankkonzepte für die Simulation: Eine prototypische Datenbankkomponente für das Simulationspaket DESMO in Modula-2, in: Sydow, A. (Hrsg.): Simulationstechnik, 8. Symp. in Berlin, Sept. 1993, Braunschweig, S. 199-202
- Maxwell, T., Costanza, R. (1997): An Open Geographic Modeling Environment, in: Simulation, März, S. 175-185
- Myrach, T. (1997): SQL2: Der Konsens über eine temporale Datenbanksprache, in: Informatik-Spektrum, Nr. 20, S. 143-150
- Niemann, H. (1983): Klassifikation von Mustern, Berlin
- Richter, O. et al. (1997): Kopplung Geographische Informationssysteme (GIS) mit ökologischen Modellen im Naturschutzmanagement, in: Kratz, R., Suhling, F. (Eds.): Geographische Informationssysteme im Naturschutz, Magdeburg, S. 5-29
- Schmidt, M. (1995): Stoffstromanalysen als Basis für ein Umweltmanagementsystem im produzierenden Gewerbe, in: Haasis, H.-D. et al. (Hrsg.): Umweltinformationssysteme in der Produktion, Marburg, S. 67-80
- UCGIS (1997): Research Priorities for Geographic Information Science (white paper and summary paper), Technical report, University Consortium for Geographic Information Science, <http://www.ucgis.org>