

# Flexibler Entwurf von Gebäudesimulatoren<sup>1</sup>

Jan Peter Riegel

riegel@informatik.uni-kl.de

FB Informatik, Universität Kaiserslautern, AG VLSI Entwurf und Architektur

Postfach 3049, 67663 Kaiserslautern

## Kurzfassung

Diese Arbeit präsentiert einen Ansatz zum Entwurf von Simulatoren basierend auf Software-Engineering-Methoden. Das Ziel ist es, mit möglichst wenig Aufwand und ohne viel Spezialwissen die Entwicklung eines auf die jeweilige Situation angepassten Simulators zu ermöglichen.

## 1 Einleitung

Der Entwurf von Simulatoren erfordert üblicherweise ein fundiertes Wissen über die physikalischen Grundlagen des zu simulierenden Systems. Zudem muss viel mathematisches Wissen vorhanden sein, um einen “guten” Simulator zu implementieren. Die Modellierung wird durch Simulationssprachen (z. B. Modelica) und Simulator-Baukästen (z. B. Mathematika/Simulink) unterstützt. Dabei entstehen dann aber typischerweise monolithische Simulatoren, die nicht, oder nur schwer, mit anderen Systemen gekoppelt werden können. Ein Software-Engineering basierter Ansatz zur Simulator-Erstellung bietet zwar jeden erdenkbaren Freiraum, beansprucht allerdings auch wesentlich mehr Zeit und erfordert zusätzliches Modellierungswissen.

Hier soll nun ein neuer Modellierungsansatz vorgestellt werden, der es auch Nicht-Spezialisten erlaubt, gute und flexibel einsetzbare Simulatoren zu erstellen. Diese Simulatoren sollen eingesetzt werden, um Steuerungssysteme schon in sehr frühen Entwurfsphasen testen zu können. Dabei können Simulator und Steuerungssystem parallel entworfen werden, sodass bereits auf sehr abstrakter Ebene das Gesamtsystem getestet werden kann.

“Modellierung” bedeutet in diesem Zusammenhang den Entwurf und die Realisierung einer Simulationssoftware. Die mathematische Modellierung der Realität (bzw. der Physik) ist ein Teil davon, die aber zu großen Teilen durch fertige Bausteine vorgegeben werden kann.

Das Software-Engineering beschränkt sich hier auf eine feste und eingeschränkte Anwendungsdomäne. Dadurch kann das Domänenwissen bereits vorverarbeitet und aufgeschrieben werden, sodass es nicht für jeden Simulator neu erarbeitet werden muss. Es entsteht ein Entwurfshandbuch, das den Benutzer durch die Modellierung führt und ihm Entwurfsentscheidungen erleichtert. Die von uns bisher betrachteten Simulatoren stammen aus der Domäne der Gebäudesimulation, jedoch lässt sich der Ansatz auch leicht auf andere

---

<sup>1</sup>Erschienen in “Simulationstechnik - ASIM 2002”, Hrsg. D. Tavangarian und R. Grützner, Seiten 576-578, Gruner Druck GmbH, Erlangen, 2002

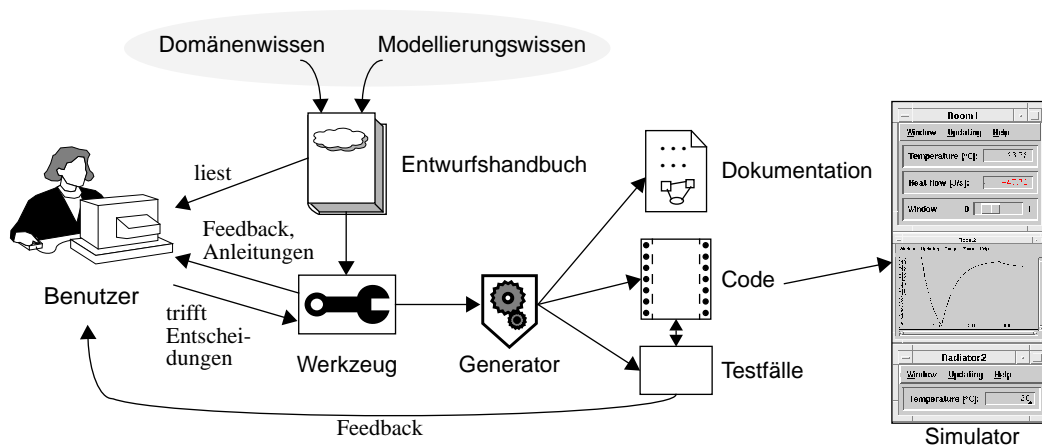


Abbildung 1: Modellierung eines Simulators

## 2 Entwurfshandbuch

Sämtliches relevante Domänen- und Modellierungswissen wird in strukturierter Form in einem Entwurfshandbuch niedergeschrieben. Der Benutzer kann dieses lesen und eine maschinenlesbare Form des Handbuchs wird direkt in einem Modellierungswerkzeug verwendet. Ein Generator erzeugt aus den vom Benutzer eingegebenen Modellen Dokumentation und lauffähigen Quellcode für einen Simulator (siehe Abb. 1).

Die einzelnen Einträge im Entwurfshandbuch sind in Pattern-Form (vgl. [1]) notiert. Dabei werden Teilprobleme (z. B. "Wärmeleitung" oder "Echtzeit-Simulation") zunächst informell beschrieben und dann durch formale Parameter und Constraints formalisiert. Zur Lösung der Teilprobleme werden die formalen Parameter an Teile des aktuellen Simulator-Modells gebunden. Alle notwendigen Bindungspunkte und die Teilschritte, die zur Lösung des Problems notwendig sind, sind dabei im Entwurfshandbuch beschrieben. Beispielsweise muss zur Simulation der Wärmeleitung in einem Gebäude zunächst die Wärmesenke (z. B. ein Raum oder eine schwere Wand) festgelegt werden. Danach werden die Wärmequellen (Wände, Radiatoren etc.) ausgesucht und zuletzt werden die notwendigen Parameter (Wärmekapazität, Raumvolumen usw.) eingestellt. Aus diesen Parameter-Bindungen werden Diagramme und Code generiert, die die Lösung des Problems widerspiegeln (siehe auch [3]). Formalisierte Einschränkungen (Constraints) innerhalb des Entwurfshandbuchs erlauben nur sinnvolle Modellierungsschritte. Zusätzlich werden Testfälle beschrieben, die die abschließende Überprüfung des Simulators erleichtern. Mit diesen Hilfsmitteln ist es möglich, einen kompletten Gebäudesimulator in der Größenordnung von einem Tag zu entwerfen.

Die Erstellung eines Simulators besteht aus einer Reihe von Entwurfsentscheidungen, die eine anfänglich grobe und ungenaue Idee schrittweise verfeinern und konkretisieren. Am Anfang steht dabei eine umgangssprachliche Problembeschreibung, die das zu lösende (Simulations-)Problem darstellt. Auf Grund dieser Problembeschreibung kann nun der Entwickler geeignete Pattern aus einem Entwurfshandbuch auswählen und anschließend umsetzen.

Am Anfang des Entwurfs dienen allgemeinere Pattern (z. B. "Simulation" oder "Gebäudestruktur") dazu, die grobe Simulatorarchitektur festzulegen. Später werden sie dann durch speziellere Pattern verfeinert ("Simulation"  $\Rightarrow$  "Schedulingstrategie", "Ein-/Ausgabe", simulierte Effekte, ...).

Jedes Pattern für sich beinhaltet informelle Beschreibungen über seinen Sinn und Zweck und zeigt den Benutzern (gestützt durch ein entsprechendes Werkzeug), wie es umgesetzt werden kann. Der eigentliche Simulator wird dann später (zu ca. 90%) von Codegeneratoren erzeugt. Neben den Pattern besteht auch jederzeit die Möglichkeit, das aktuelle Simulatormodell durch handgeschriebenen Code oder durch externe Komponenten zu erweitern. So können selten benötigte Effekte, die noch nicht in das Entwurfshandbuch aufgenommen wurden, realisiert werden. Da alle Schnittstellen des Simulators bekannt sind, ist die Integration mit anderen Software-Komponenten sehr einfach.

### 3 Zusammenfassung

Die großen Vorteile der hier vorgestellten Methode liegen in ihrer Flexibilität und in der Benutzerführung. Ähnlich wie z. B. bei Bibliotheken von Modelica-Komponenten wird der Simulator aus vorgefertigten (aber noch sehr anpassbaren) Teilen zusammen gesetzt. Dabei sind die Dokumentation und das Umsetzungsvorgehen dieser Teile besonders wichtig. Anders als bei reinen Simulationsbibliotheken bleibt hier allerdings die komplette Mächtigkeit des Software-Engineerings erhalten. Auch "exotische" Effekte wie Personenverhalten, zufällige Ereignisse oder die Simulation von Fehlern können problemlos integriert werden. Als Nachteil ist der hohe Aufwand zur Erstellung des Entwurfshandbuches zu erwähnen, der sich erst bei hoher Wiederverwendung der einzelnen Pattern rechnet.

### Literatur

- [1] *Gamma, E., Helm, R., Johnson, R., Vlissides, J.:* Design Patterns. Addison-Wesley, 1995.
- [2] *Schütze, M., Riegel, J. P., Zimmermann, G.:* PSiGene - A Pattern-Based Component Generator for Building Simulation. Journal Theory and Practice of Object Systems (TAPOS), Vol. 5, No. 2, (1999), S. 83-95.
- [3] *Riegel, J. P., Kaesling, C., Schütze, M.:* Modeling Software Architecture Using Domain-Specific Patterns. First Working IFIP Conference on Software Architecture (WICSA 1), Kluwer Academic Publishers, 1999.