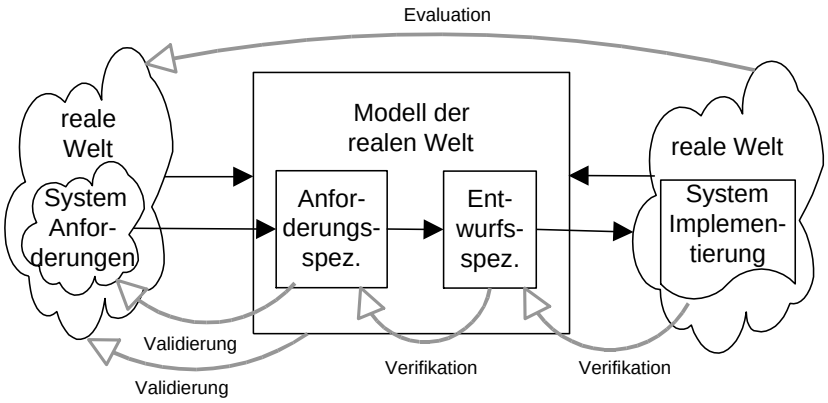
Validierung und Verifikation von Prozessmodellen

Jörg Desel

Lehrstuhl für Angewandte Informatik
Katholische Universität Eichstätt
Ostenstraße 28, 85072 Eichstätt
joerg.desel@ku-eichstaett.de

Die modellbasierte Systementwicklung kann nur dann zu anforderungsgerechten Systemen führen, wenn die zugrundeliegenden Modelle die tatsächlichen Anforderungen widerspiegeln. Zu den Anforderungen gehört die existierende oder geplante Systemumgebung (die reale Welt) sowie das gewünschte Verhalten des Systems in dieser Umgebung. Die prozessorientiere Systementwicklung gibt sowohl für die Umgebung als auch für das Verhalten Prozesse und Prozesseigenschaften vor. In diesem Tutorium wird gezeigt, wie durch alternierende Anwendung von Validierungs- und Verifikationsschritten ein korrektes und valides Gesamtmodell der zu unterstützenden Prozesse erzeugt wird, das als formale Spezifikation für das zu implementierende System verwendet werden kann. Der Ansatz verwendet verschiedene Typen von Petrinetzen.

Im folgenden Bild wir die unterschiedliche Rolle von Validierung und Verifikation dargestellt. Dabei ist *Validierung* stets als Gegenstück zu *Formalisierung* zu verstehen, wie auch *Verifikation* mit *Spezifikation* ein derartiges begriffliches Paar bildet. Die Evaluation des Systems betrifft die modellunabhängige Bewertung des implementierten Systems (siehe für eine ausführlichere Diskussion [De02]).



Das Diagramm macht deutlich, dass Formalisierung und Validierung an verschiedenen Stellen auftreten; zum einen bei der Modellierung eines gegebenen Systems (der Umgebung bzw. der realen Welt), zum anderen bei der Modellierung des zu implementierenden Systems. Entsprechend spielt die Spezifikation und Verifikation sowohl bei dem Schritt von den Anforderungen zum Entwurf eine Rolle als auch bei dem Schritt vom Entwurf zur Implementierung. In dem hier dargestellten Ansatz werden Verifikationsverfahren außerdem bei der Validierung von Anforderungen eingesetzt.

Um formale Verfahren bei Validierung und Verifikation einsetzen zu können, beziehen wir uns auf formale Prozessmodelle, gegeben durch höhere Petrinetze. Diese haben eine graphische Darstellung, sind mathematisch definiert und sind ausführbar (siehe die Diskussion in [DJ01]). Die graphische Darstellung unterstützt das intuitive Verständnis eines Petrinetz-Modells, unabhängig von der genauen Petrinetz-Variante (insofern sind Petrinetze eine frühe UML für Prozessmodelle). Die Ausführbarkeit erlaubt die Konstruktion von Simulatoren. Die mathematische Grundlage ist Voraussetzung für Analyseverfahren.

Im Projekt VIP[1] wurde ein simulationsbasierter Ansatz zur Validierung und Verifikation von Prozessmodellen entwickelt, der auf halbgeordneten Abläufen von (höheren) Petrinetz-Modellen von Systemen beruht [De00]. In folgenden wird dieser Ansatz skizziert.

Die Grundannahme des Ansatzes ist, dass der Entwickler eine genaue Vorstellung von gültigen Abläufen eines (existierenden oder zu konstruierenden) Systems besitzt. Ein Ablauf beschreibt Aktivitäten und ihren kausalen Zusammenhang, der durch die jeweiligen Vor- und Nachbedingungen gegeben ist. Jeder Ablauf lässt sich durch eine halbgeordnete Struktur bzw. durch ein spezielles Petrinetz, dessen Elemente Aktivitäten und Bedingungen repräsentieren, darstellen [DR98]. Zusätzlich gehören zur Systemspezifikation Eigenschaften, die für alle Abläufe erfüllt sein müssen (die also durch eine Linear-Time Logik ausgedrückt werden können). Diese werden in einer Petrinetz-angelehnten Syntax formuliert.

Bei dem Entwicklungsprozess eines Petrinetz-Modells werden durch Simulation Ablaufnetze generiert und visualisiert. Dies macht dem Entwickler deutlich, welches Systemverhalten er mit seinem Modell spezifiziert hat. Die generierten Abläufe können bezüglich der zusätzlichen logischen Spezifikationen automatisch untersucht werden (an dieser Stelle werden Verifikationsverfahren beim Validierungsprozess eingesetzt). Dies erlaubt die Validierung der Anforderungen; es wird genau deutlich, welche Abläufe durch eine Anforderung ausgeschlossen werden und welche nicht.

Dieses Vorgehen kann zur schrittweisen Implementierung weiterer Spezifikationen wiederholt eingesetzt werden, bis schließlich eine Entwurfsspezifikation entstanden ist, die sowohl das Verhalten des (validierten) Umgebungsmodells respektiert als auch allen (validierten) Anforderungen entspricht. Die Implementierung einer Spezifikation erfolgt dabei halbautomatisch, erfordert aber im allgemeinen jeweils einen anschließenden Verifikationsschritt. Zudem können Spezifikationen nicht in beliebiger Reihenfolge implementiert werden, um frühere Spezifikationen durch Systemveränderungen nicht zu verletzen.

Zur Verifikation von Sicherheitseigenschaften bieten sich klassische Petrinetz-Techniken an, wie sie für elementare Netze z.B. in [De98a, De98b] beschrieben werden. Für Le-

[1] *Verifikation von Informationssystemen durch Auswertung halbgeordneter Prozesse*, gefördert durch die DFG

bendigkeitseigenschaften werden andere Verfahren benötigt, wie z.B. die für Petrinetze adaptierte Proof-Graph Methode [DK98, DK01].

Erweiterungen des Ansatzes betreffen einerseits eine Validierung unter Berücksichtigung von zeitlichen Aspekten und von Kosten der unterstützten Abläufe [DE00a] und andererseits die Berücksichtigung flexibler Ablaufbeschreibungen im Rahmen von Workflow-Systemen [DE00b].

In der abschliessenden Literatursammlung sind aus Platzgründen nur Publikationen des Autors aufgeführt, die im Tutorium angesprochen werden. Grundlagen und weitere Literaturangaben findet man in diesen Arbeiten.

Literatur

[De98a] Jörg Desel. *Petrinetze, lineare Algebra und lineare Programmierung*. Teubner-Texte zur Informatik Band 26. Teubner, Stuttgart 1998

[De98b] Jörg Desel. Basic Linear Algebraic Techniques for Place/Transition Nets. In: Reisig, W., Rozenberg, G. (eds.): *Lectures on Petri Nets I: Basic Models*, Lecture Notes in Computer Science, Vol. 1491. Springer-Verlag, Heidelberg (1998) 257–308

[DK98] Jörg Desel und Ekkart Kindler. Proving correctness of distributed algorithms - a case study. In: Proceedings of the 1998 International Conference on Application of Concurrency to System Design (CSD'98), Fukushima, Japan, März 1998, pp.177-186, IEEE Computer Society Press (1998)

[DR98] Jörg Desel und Wolfgang Reisig. Place/Transition Petri Nets. In: Reisig, W., Rozenberg, G. (eds.): *Lectures on Petri Nets I: Basic Models*, Lecture Notes in Computer Science, Vol. 1491. Springer-Verlag, Heidelberg (1998) 122–173

[De00] Jörg Desel. Validation of Process Models by Construction of Process Nets. In: van der Aalst, W., Desel, J., Oberweis, A. (eds.): *Business Process Management*, Lecture Notes in Computer Science, Vol. 1806. Springer-Verlag, Heidelberg (2000) 110–128

[DE00a] Jörg Desel und Thomas Erwin. Modeling, Simulation and Analysis of Business Processes. In: van der Aalst, W., Desel, J., Oberweis, A. (eds.): *Business Process Management*, Lecture Notes in Computer Science, Vol. 1806. Springer-Verlag, Heidelberg (2000) 129–141

[DE00b] Jörg Desel und Thomas Erwin. Hybrid Specifications: Looking at Workflows From a run-time Perspective. *International Journal of Computer System Science & Engineering* Vol. 15 No. 5 (2000) 291–302

[DJ01] Jörg Desel und Gabriel Juhás. What is a Petri Net? – Informal Answers for the Informed Reader. In: Ehrig, H., Juhás, G., Padberg, J., Rozenberg, G. (eds.): *Unifying Petri Nets*, Lecture Notes in Computer Science, Vol. 2128. Springer-Verlag, Heidelberg (2001) 1–25

[DK01] Jörg Desel und Ekkart Kindler. Petri Nets and Components – Extending the DAWN Approach. In: Moldt, D. (ed.): *Workshop on Modelling of Objects, Components. and Agents*, Aarhus, Denmark, DAIMI PB–553 (2001) 21–36

[De02] Jörg Desel. Model Validation – A Theoretical Issue? In: Esparza, E., Lakos, C. (eds.): *Application and Theory of Petri Nets 2002*, Lecture Notes in Computer Science, Vol. 2360. Springer-Verlag, Heidelberg (2002) 23–43