Zur Integration von Struktur- und Verhaltensmodellierung mit OCL

Lars Hamann, Martin Gogolla, Oliver Hofrichter
AG Datenbanksysteme
Universität Bremen
Postfach 330440
28334 Bremen
(lhamann|gogolla|hofrichter)@informatik.uni-bremen.de

Das werkzeuggestützte Validieren von in der Unified Modeling Language (UML) spezifizierten Modellen ist zu einem wichtigen Forschungszweig in der modellgetriebenen Softwareentwicklung geworden. Durch eine frühzeitige Validierung sollen Designfehler bereits zu Beginn des Entwicklungsprozesses aufgedeckt werden, um spätere aufwändige Korrekturen zu vermeiden. Herkömmliche Werkzeuge in diesem Kontext verwenden UML-Klassendiagramme in Verbindung mit textuellen Einschränkungen, die in der Object Constraint Language (OCL) definiert sind. Eher unbeachtet sind weitergehende Modellierungselemente wie zum Beispiel Zustandsautomaten, die zwar von der UML bereitgestellt werden, aber in OCL-basierten Validierungswerkzeugen bisher kaum Einzug gefunden haben.

Der hier zusammengefasste Beitrag [HHG12b] zeigt, wie die in der UML vorhandenen Protokolzustandsautomaten (Protocol State Machines; PSMs) die Modellierungs- und Validierungsmöglichkeiten erweitern. Alle vorgestellten Konzepte sind in einem UML/OCL-Werkzeug [USE, GBR07], das international in der Praxis sowie zu Forschungsprojekten und in der Lehre eingesetzt wird, umgesetzt.

Während Klassendiagramme dazu verwendet werden, um die Struktur eines Systems z. B. mit Hilfe von Klassen und Assoziationen zu beschreiben, können Zustandsautomaten dazu verwendet werden das Verhalten zu beschreiben. PSMs verfolgen dabei einen rein deklarativen Ansatz, indem sie gültige Reihenfolgen von Operationsaufrufen für eine Klasse festlegen. Sie beschreiben also ein oder mehrere Protokolle einer Klasse. Den PSMs gegenüber stehen Behavioral State Machines, die mit Hilfe einer Action Language zustandsverändernde Aktionen in einem Objekt ausführen können.

Unser Beitrag befasst sich mit Protokollzustandsautomaten, da diese unter anderem durch ihre Zustandsfolgen im Gegensatz zu Vor- und Nachbedingungen von Operationen mehr als nur einen Zustandswechsel berücksichtigen. Es wird dargestellt, wie PSMs während des Designprozesses genutzt werden können, um gültige Programmabläufe zu spezifizieren und an welchen Stellen OCL diesen Vorgang unterstützen kann. So können z. B. für Transitionen Vorbedingungen (Guards) in OCL definiert werden. Diese Guards erlauben es, dass ein Aufruf einer Operation zu unterschiedlichen Zuständen führen kann. Die auszuführende Transition kann hierbei anhand des Wahrheitswerts der einzelnen Guards gewählt werden.

Diese und weitere Funktionen der Automaten werden dabei ebenso ausführlich beschrieben, wie das Laufzeitverhalten während der Modellausführung. Das gesamte Vorgehen wird anhand von Beispielen erläutert, die zeigen, wie Designfehler frühzeitig entdeckt werden können. Die Möglichkeiten der Fehleranalyse mit dem vorgestellten UML/OCL-Werkzeug durch unterschiedliche Sichten auf das Modell und auf den aktuellen Systemzustand, wie z. B. Objektdiagramme, Sequenzdiagramme und Zustandsautomaten, die stets synchronisiert sind, werden ebenfalls gezeigt.

Zusätzlich werden besondere Funktionen, die im Kontext der Laufzeitverifikation von Implementierungen benötigt werden diskutiert. So erlaubt z. B. der in [HHG12a] beschriebene Ansatz zur Laufzeitverifikation, dass die Verifikation einer Anwendung zu einem beliebigen Zeitpunkt der Ausführung starten kann. Dies führt dazu, dass die bisher erfolgten Zustandsübergänge nicht beobachtet werden konnten. Dadurch sind die jeweiligen Automateninstanzen nicht mit der Anwendung synchronisiert. Um eine nachträgliche Synchronisation zu ermöglichen, zeigen wir einen Ansatz, der die für einzelne Zustände definierten Invarianten (State Invariants) verwendet; sind diese geeignet definiert, kann auch ohne Wissen über die vorher auf einer Instanz aufgerufenen Operationen der jeweilige zustand eines Automaten ermittelt werden.

Literatur

[GBR07] Martin Gogolla, Fabian Büttner und Mark Richters. USE: A UML-Based Specification Environment for Validating UML and OCL. *Science of Computer Programming*, 69:27–34, 2007.

[HHG12a] Lars Hamann, Oliver Hofrichter und Martin Gogolla. OCL-Based Runtime Monitoring of Applications with Protocol State Machines. In Antonio Vallecillo, Juha-Pekka Tolvanen, Ekkart Kindler, Harald Störrle und Dimitrios S. Kolovos, Hrsg., *ECMFA*, Jgg. 7349 of *Lecture Notes in Computer Science*, Seiten 384–399. Springer, 2012.

[HHG12b] Lars Hamann, Oliver Hofrichter und Martin Gogolla. On Integrating Structure and Behavior Modeling with OCL. In Robert B. France, Jürgen Kazmeier, Ruth Breu und Colin Atkinson, Hrsg., *Model Driven Engineering Languages and Systems - 15th International ACM/IEEE Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings*, Jgg. 7590 of *Lecture Notes in Computer Science*, Seiten 235–251. Springer, 2012.

[USE] A UML-based Specification Environment. http://sourceforge.net/projects/useocl/.