Vorwort

Softwaresysteme sind heutzutage in der Regel komplexe Produkte, für deren erfolgreiche Produktion der Einsatz ingenieurmäßiger Techniken unerlässlich ist. Diese nun mittlerweile mehr als 30 Jahre alte und häufig zitierte Erkenntnis hat dazu geführt, dass in den letzten drei Jahrzehnten innerhalb der Informatik im Gebiet des Software Engineering intensiv an Sprachen, Methoden und Werkzeugen zur Unterstützung des Softwareerstellungsprozesses gearbeitet wird. Trotz großer Fortschritte hierbei muss allerdings festgestellt werden, dass im Vergleich zu anderen, durchgängig viel älteren Ingenieursdisziplinen noch viele Fragen unbeantwortet sind und immer neue Fragestellungen auftauchen.

So macht auch ein oberflächlicher Vergleich zum Beispiel mit dem Gebiet des Bauwesens schnell deutlich, dass dort internatonale Standards eingesetzt werden, um Modelle von Gebäuden zu erstellen, die Modelle zu analysieren und anschließend die Modelle in Bauten zu realisieren. Hierbei sind dann auch die Rollen- und Aufgabenverteilungen allgemein akzeptiert, so dass etwa Berufsgruppen wie Architekten, Statiker sowie Spezialisten für den Tief- und Hochbau existieren.

Eine derartige modellbasierte Vorgehensweise wird zunehmend auch in der Softwareentwicklung favorisiert. Dies bedeutet insbesondere, dass in den letzten Jahren international versucht wird, eine allgemein akzeptierte Modellierungssprache festzulegen, so dass etwa wie im Bauwesen, von einem Software-Architekten erstellte Modelle von einem „Software-Statiker“ analysiert werden können, bevor sie von Spezialisten für die Realisierung, also Programmierern in ausführbare Programme umgesetzt werden.

Diese standardisierte Modellierungssprache ist die Unified Modeling Language, die in einem schrittweisen Prozess durch ein international besetztes Konsortium stetig weiterentwickelt wird. Aufgrund der vielfältigen Interessenlagen im Standardisierungsprozess ist mit der aktuellen Version 2.0 der UML eine Sprachfamilie entstanden, deren Umfang, semantische Fundierung und methodische Verwendung noch viele Fragen offen lässt.

Diesem Problem hat sich Herr Rumpe in den letzten Jahren in seinen wissenschaftlichen und praktischen Arbeiten gewidmet, deren Ergebnisse er nun in zwei Büchern einer breiten Leserschaft zugänglich macht. Hierbei hat Herr Rumpe das methodische Vorgehen in den Vordergrund gestellt. Im Einklang mit der heutigen Erkenntnis, dass leichtgewichtige, agile Entwicklungsprozesse insbesondere in kleineren und mittleren Entwicklungsprojekten große Vorteile bieten, hat Herr Rumpe Techniken für einen agilen Entwicklungsprozess entwickelt. Auf dieser Basis hat er dann eine geeignete Modellierungssprache definiert, in dem er ein so genanntes Sprachprofil für die UML definiert hat. In diesem Sprachprofil UML/P hat er die UML geeignet abgespeckt und an einigen Stellen so abgerundet, dass nun eine handhabbare Version der UML insbesondere für einen agilen Entwicklungsprozess vorliegt.

Herr Rumpe hat diese Sprache UML/P ausführlich in dem diesem Buch vorangehenden Buch „Modellierung mit UML“ erläutert. Das Buch bietet eine wesentliche Grundlage für das hier vorliegende Buch, deren Inhalt allerdings auch in diesem Buch noch einmal kurz zusammengefasst wird. Das hier vorliegende Buch mit dem Titel „Agile Modellierung mit UML“ widmet sich nun in erster Linie dem methodischen Umgang mit der UML/P.

Hierbei behandelt Herr Rumpe drei Kernthemen einer modellbasierten Softwareentwicklung. Dies sind

- die Codegenerierung, also der automatisierte Übergang vom Modell zu einem ausführbaren Programm,
- das systematische Testen von Programmen mithilfe einer modellbasierten, strukturierten Festlegung von Testfällen sowie
- die Weiterentwicklung von Modellen durch den Einsatz von Transformations- und Refactoring-Techniken.

Alle drei Kernthemen werden von Herrn Rumpe zunächst systematisch aufgearbeitet und die zugrunde liegenden Begriffe und Techniken werden eingeführt. Darauf aufbauend stellt er dann jeweils seinen Ansatz auf der Basis der Sprache UML/P vor. Diese Zweiteilung und klare Trennung zwischen Grundlagen und Anwendungen machen die Darstellung außerordentlich gut verständlich und bieten dem Leser auch die Möglichkeit, diese Erkenntnisse unmittelbar auf andere modellbasierte Ansätze und Sprachen zu übertragen.

Insgesamt hat dieses Buch einen großen Nutzen sowohl für den Praktiker in der Softwareentwicklung, für die akademische Ausbildung im Fachgebiet Softwaretechnik als auch für die Forschung im Bereich der modellbasierten Entwicklung der Software. Der Praktiker lernt, wie er mit modernen modellbasierten Techniken die Produktion von Code verbessern und damit die Qualität erheblich steigern kann. Dem Studierenden werden sowohl wichtige wissenschaftliche Grundlagen als auch unmittelbare Anwendungen der dargestellten grundlegenden Techniken vermittelt. Dem Wissenschaftler bietet das Buch einen umfassenden Überblick über den heutigen Stand der Forschung in den drei Kernthemen des Buchs.

Das Buch stellt somit einen wichtigen Meilenstein in der Entwicklung von Konzepten und Techniken für eine modellbasierte und ingenieurmäßige Softwareentwicklung dar und bietet somit auch die Grundlage für weitere Arbeiten in der Zukunft. So werden praktische Erfahrungen mit dem Umgang der Konzepte ihre Tragbarkeit validieren. Wissenschaftliche, konzeptionelle Arbeiten werden insbesondere das Thema der Modelltransformation etwa auf der Basis von Graphtransformationen genauer erforschen und darüber hinaus das Gebiet der Modellanalyse im Sinne einer Modellstatik vertiefen.

Ein derartiges vertieftes Verständnis der Informatik-Methoden im Bereich der modellbasierten Softwareentwicklung ist eine entscheidende Voraussetzung für eine erfolgreiche Kopplung mit anderen ingenieurmäßigen Methoden etwa im Bereich von eingebetteten Systemen oder im Bereich von intelligenten, benutzungsfreundlichen Produkten. Die Domänenunabhängigkeit der Sprache UML/P bietet auch hier noch viele Möglichkeiten.

Gregor Engels

Paderborn im September 2004