2 UML – Unified Modeling Language

2.1 Entwicklung der Sprache

UML (Unified Modeling Language) ist nach Aussage ihrer Entwickler eine Sprache zur Spezifikation, Visualisierung, Konstruktion und Dokumentation von Software. Ursprünglich aus den Ideen der drei Gurus Booch, Jacobson und Rumbaugh zusammengestellt, hat sie sich durch die Beteiligung wichtiger Industriepartner mehr und mehr als wirklicher Standard für die Modellierung objektorientierter Spezifikationen etabliert.

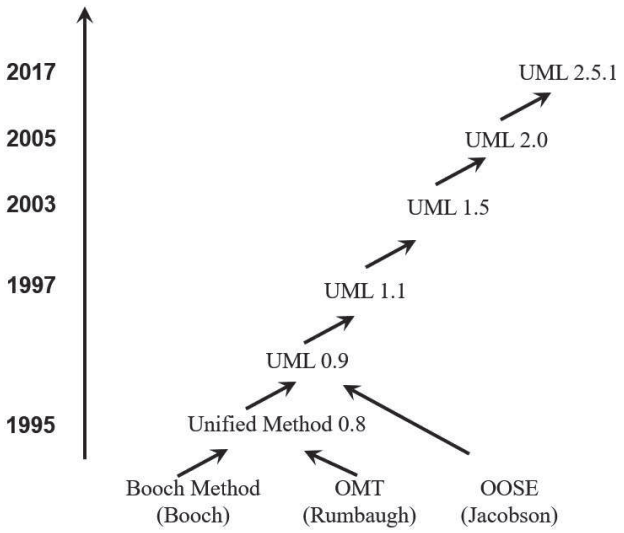

Bild 2.1 Überblick zur Geschichte von UML

Die Entwicklung von UML kann bis 1994 zurückverfolgt werden, als Booch und Rumbaugh begannen, ihre Entwicklungsmethoden zusammenzuführen. Beide Ansätze waren zu dieser Zeit weltweit schon stark verbreitet, wobei die Stärken von Booch in der Modularisierung und im Entwurf lagen, während die Stärken Rumbaughs besonders bei der objektorientierten Modellierung bestanden. Den in beiden Ansätzen gemeinsamen Konzepten galt

es, eine einheitliche Notation zu geben. Eine erste Version der „Unified Method“ wurde im Oktober 1995 veröffentlicht. Zu dieser Zeit wurde die Zweiergruppe durch Ivar Jacobson erweitert, der mit seinem OOSE (Object-Oriented Software Engineering) den Gesichtspunkt des Anwendungsfalls in die Methodendiskussion einbrachte.

Gemeinsam legten sie 1996 eine Spezifikation von UML in der Version 0.9 vor, mit der sie sich wohl auch den Namen die „drei Amigos“ verdient haben, unter dem sie heute oft zitiert werden.

Mit dem gemeinsamen Bericht von 1996 sahen auch viele Unternehmen und Organisationen in einer einheitlichen Modellierungssprache UML eine strategische Basis für ihre Arbeit. Auf einen Aufruf der Object Management Group (OMG), einer Vereinigung von Unternehmen mit Bemühungen zur Standardisierung in der Softwareentwicklung, zur Einreichung von Veränderungsvorschlägen gab es große Resonanz. Daraufhin etablierte die Firma Rational, bei der die „drei Amigos“ tätig waren, ein Konsortium zur Definition der Sprache UML.

Dieses Konsortium umfasste bedeutende Firmen wie DEC, HP, IBM, Microsoft, Oracle und Unisys. Es modifizierte den Sprachvorschlag und erarbeitete Berichte zur genauen Definition von UML. Im Januar 1997 reichten dann weitere Firmen einen Vorschlag bei der OMG ein. Diese Gruppe wurde in das Konsortium aufgenommen, und ihre Vorschläge führten zur Sprachentwicklung UML 1.1. Die weiteren Revisionen der Unterlagen erfolgten bis zur Spezifikation von UML 1.5 schrittweise. Danach wurde eine vollständige Überarbeitung der Sprache vorgenommen und UML 2.0 entwickelt. Mittlerweile gibt es bereits Version 2.5.1. Die letzten Änderungen waren hauptsächlich Präzisierungen und nicht mehr so umfangreich. Die Version 2.5.1 wurde genau 20 Jahre nach der ersten Standardisierung von UML eingereicht und bis 2023 noch nicht weiter verändert.

Das immer größere Interesse an UML hat auch seine Nachteile. Immer mehr unterschiedliche Ansichten beeinflussten die Diskussion über die Entwicklung der Sprache. So hat es mehrere Jahre gedauert, bevor der Standard durch alle Gremien bestätigt wurde. Die Firma Rational ist in der Zwischenzeit durch IBM aufgekauft worden.

UML ist nicht abgeschlossen, sondern offen für neue Konzepte. Die Sprache stellt in sich bereits Erweiterungsmöglichkeiten zur Verfügung. Zukünftige Entwicklungen sind damit integrierbar. Bei der OMG wurde auch eine „Revision Task Force“ (RTF) installiert, die Ansprechpartner für die Öffentlichkeit ist, um Fehler in den Spezifikationen zu beheben.

Welche anderen Konzepte und Methoden hauptsächlich beim Entwurf der Grundprinzipien von UML eine Rolle spielten, ist aus Bild 2.2 zu entnehmen, die aus einer Präsentation von Grady Booch selbst stammt.

Diese Abbildung stellt vereinfacht Einflussfaktoren auf die Entwicklung von UML dar. Dabei spielen die Arbeiten von Booch [2.3], Jacobson [2.17] und Rumbaugh [2.24] eine herausragende Rolle. Sie sind in der Abbildung links mit grauen Pfeilen dargestellt. Natürlich basieren die Konzepte von UML auf weiteren Arbeiten wie beispielsweise der Idee der Kapselung nach Parnas oder anderen objektorientierten Ansätzen. Diese sind aus Übersichtlichkeitsgründen nicht dargestellt.

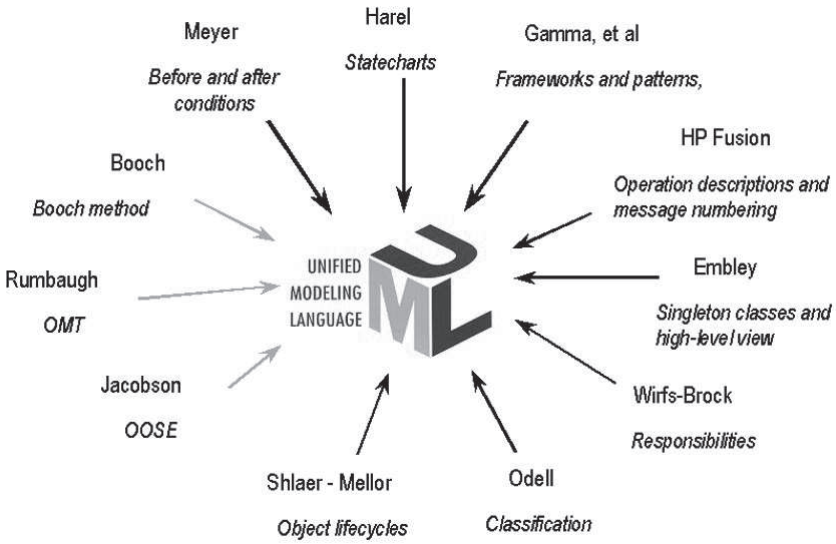

Bild 2.2 Einflüsse auf die Entstehung von UML (nach Booch)

Einige Konzepte wie die *Statecharts* (Zustandsdiagramme) nach Harel [2.14] sind nur leicht modifiziert in UML übernommen worden. Sie dienen zur Spezifikation dynamischer Vorgänge und sind daher unter anderem für die Spezifikation der Lebenszyklen von Objekten geeignet. In diesem Buch werden sie deshalb in zwei Abschnitten ausführlich behandelt. Über die Grundlagen finden sich einige Ausführungen bereits in Abschnitt 1.2.3. Mit den entsprechenden Sprachelementen der UML befasst sich Abschnitt 2.5.1.

Um die Semantik der Zustandsdiagramme zu verdeutlichen, wurden auch einige Animationen erstellt, die auf der Webseite zu diesem Buch (im Zusatzmaterial auf www.plus.hanser-fachbuch.de) bereitgestellt werden. Sie können herangezogen werden, um das Verständnis für die Spezifikationen zu erhöhen.

Eigene Sprachelemente wurden auch für Design Patterns (Entwurfsmuster) in UML vorgesehen. Sie werden in Abschnitt 2.3.9 erläutert. Die Anwendung von Patterns wird dann später noch in Kapitel 3 diskutiert. Entwurfsmuster spielen bei der Softwareentwicklung eine immer wichtigere Rolle. Sie ermöglichen eine neue Art der Wiederverwendung. Dem Leser und der Leserin wird das Studium der entsprechenden Literatur (z. B. [2.13]) unabhängig von der Nutzung von UML ans Herz gelegt. In Bünnig et al. [2.4] wird eine Programmiersprache auf der Basis von Design Patterns vorgeschlagen. Dazu findet man mehr auf der Webseite zu diesem Buch.

Vor- und Nachbedingungen hat Meyer sehr erfolgreich in den Entwurf seiner Programmiersprache Eiffel [2.20] integriert. Damit hat er einen wichtigen Beitrag zur Etablierung einer sicheren Softwareentwicklung geleistet. Das Konzept basiert auf den Arbeiten von Hoare [2.15]. Dieser hat sich sehr intensiv mit dem Nachweis der Korrektheit von Programmen beschäftigt. Zusicherungen spielen dabei eine wichtige Rolle. Der Nachweis der Korrektheit

der Nachbedingungen aus der Korrektheit der Vorbedingungen wird zum Beweis der Korrektheit von Programmen genutzt. Die Möglichkeit der Notation von Zusicherungen ist in UML integriert. Ein Nachweis ihrer Korrektheit ist allerdings nicht möglich, da die einzelnen Sprachelemente nicht ausreichend formalisiert sind.

Auf weitere Einflussfaktoren soll hier nicht weiter explizit eingegangen werden. Der Leser und die Leserin seien auf die entsprechende Literatur verwiesen, um den Grad des Einflusses auf die Sprachgestaltung von UML selbst zu beurteilen (z. B. [2.8], [2.19], [2.20]).

Die eine kompakte Darstellung der UML findet man in [2.1], [2.2] und [2.9].

Der folgende Abschnitt widmet sich zunächst dem anwendungsfallorientierten Ansatz und den Modellen, die dafür bei der Analyse eines Problems zur Verfügung stehen. Danach werden Diagramme vorgestellt, die zur Beschreibung statischer Zusammenhänge geeignet sind, bevor sich ein weiterer Abschnitt mit der Spezifikation dynamischer Eigenschaften beschäftigt.

2.2 Anwendungsfallmodelle

Softwareentwicklung beginnt bei der Anforderungsanalyse mit dem Ziel, die wirklichen Bedürfnisse der Anwender und Auftraggeber zu ermitteln. Basierend auf den intuitiven Hauptzielen eines Projektes sind präzise Spezifikationen zu entwickeln. Dieser Prozess kann nur unter Einbeziehung der Anwender und in einer gut strukturierten Form erfolgreich gemeistert werden. Dabei hat die *Analyse der Anwendungsfälle* eine entscheidende Bedeutung.

Diese Analyse wird nicht nur bei der objektorientierten Softwareentwicklung hervorgehoben. Bei der strukturierten Analyse ist das Kontextdiagramm Ausgangspunkt der Betrachtungen. Dabei wird die Kommunikation zwischen Umgebung und System in Form von Datenflüssen zu Datenquellen und Datensenken modelliert.

Bei der Analyse betrieblicher Zusammenhänge greift man auf die Geschäftsprozesse zurück, die Ziele eines Unternehmens unterstützen.

Mit dem Aufkommen der objektorientierten Analysemethoden [2.3] wurde der funktionale Aspekt der Anwendungen zunächst etwas in den Hintergrund gerückt. Es war ein Verdienst von Jacobson [2.17], den Anwendungsfallaspekt in die Welt der objektorientierten Modellierung zu integrieren. Der Versuch der Integration von Datenflüssen in die Spezifikationsmöglichkeiten von OMT durch Rumbaugh [2.24] war zunächst nicht von diesem anhaltenden Erfolg geprägt. Das Buch über die Spezifikationssprache OMT war zwar ein weltweiter Bestseller, die Datenflüsse fanden aber nicht den Weg in die UML.

Definition 2.1: Szenario

Ein Szenario ist eine spezifische Folge von Aktionen, die zur Verdeutlichung des Verhaltens eines Systems dient.

Szenarien sind Ausgangspunkt jeglicher Spezifikation, die menschliche Handlungen beschreiben. Sie sind die Beispiele aus dem Anwendungsbereich, die als Ausgangspunkt für eine Abstraktion dienen. Szenarien können konkret oder abstrakt sein.

Definition 2.2: Konkretes Szenario

Ein konkretes Szenario ist eine spezifische Folge von Aktionen, die von konkreten Akteuren (Personen oder Systemen) unter konkreten Randbedingungen durchgeführt werden.

So ist beispielsweise die Buchung einer Reise für einen konkreten Touristen Felix mit dem konkreten Preis von 1.000 Euro an den Zielort Berlin mit dem Anreisedatum 3. Mai 2023 und weiteren konkreten Randbedingungen ein konkretes Szenario. Ein solches Szenario ist die Basis für spätere Testfälle des entwickelten Softwaresystems. Es entspricht genau einem Durchlauf durch dieses System. Erfolgt die Beschreibung der Folge von Aktionen etwas weniger konkret durch allgemeine Akteure wie beispielsweise „ein Tourist“, dann spricht man von abstrakten Szenarien.

Definition 2.3: Abstraktes Szenario

Ein abstraktes Szenario ist eine spezifische Folge von Aktionen, die von Akteuren (im Sinne von Rolle) unter abstrakten Randbedingungen durchgeführt werden.

Abstrakte Szenarien können Zusicherungen (Constraints) enthalten, die für konkrete Szenarien exakte Werte annehmen. Um eine Reise zu buchen, muss ein Tourist sein Reiseziel und sein Reisedatum angeben. Danach erhält er ein Angebot, das er innerhalb von drei Tagen bezahlen muss, um die Reise zu buchen.

Aus einer Reihe solcher abstrakten oder konkreten Szenarien kann eine allgemeinere (noch abstraktere) Beschreibung abgeleitet werden. Dabei sind alle Eventualitäten und Besonderheiten zu berücksichtigen. Für das dargestellte Beispiel bedeutet dies, dass beschrieben wird, wie sich das Reisebüro verhält, wenn dem Touristen der Preis des Angebotes zu hoch ist.

Szenarien können dazu genutzt werden, um das Anwendungsgebiet zu charakterisieren und damit für alle Beteiligten erschließbar zu machen. Umgekehrt können sie auch zur Verdeutlichung der abstrakten Beschreibung dienen. Sie stellen in diesem Fall ein konkretes Beispiel dafür dar, was mit einer abstrakten Spezifikation gemeint ist. Eine allgemeine Beschreibung wird durch Szenarien erläutert und damit verständlicher. So kann zur Verdeutlichung der allgemeinen Beschreibung, was bei der Buchung einer Reise alles zu beachten ist, das konkrete Beispiel der Buchung einer speziellen Reise beitragen.

Es ist offensichtlich, dass zwischen allgemeiner Beschreibung und den abstrakten und konkreten Szenarien keine Widersprüche existieren dürfen.

Im Folgenden ist mit Szenario immer ein abstraktes Szenario gemeint. Nur bei einem konkreten Szenario wird dies explizit hervorgehoben.