10 Referenzschemata im Reverse Engineering

Andreas Winter
Universität Koblenz-Landau, Institut für Softwaretechnik, D-56016 Koblenz,
winter@uni-koblenz.de

10.1 Motivation

Die Entwicklung und Weiterentwicklung von Softwaresystemen erfordert umfangreiche und detaillierte Kenntnis unterschiedlichster Aspekte des Systems. Unabhängig, ob Softwaresysteme nach klassischen oder agilen Vorgehensmodellen entwickelt werden, wird das Verstehen vorhandener Softwaresysteme zur immer zentraleren Aktivität der Software-Entwicklung. Zur Unterstützung dieser Aktivität wurden in den letzten Jahren vielfältige Reverse-Engineering-Techniken und Werkzeuge entwickelt. Reverse-Engineering Werkzeuge basieren in der Regel auf einer Repository-basierten Architektur [3]. *Extaktoren* übertragen die Problem-relevanten Fakten aus Quelltexten in ein *Repository*. Analysewerkzeuge (*Abstraktoren*) untersuchen, verändern oder ergänzen diese Daten. Analyseergebnisse werden mit Hilfe geeigneter Visualisierungswerkzeuge (*Visualisierer*) präsentiert.

Die Synchronisierung dieser Einzelkomponenten erfolgt über das gemeinsame Repository. Dessen Struktur legt fest, welche Fakten über das Softwaresystem bereitgestellt werden und welche Analysen hierauf ausgeführt werden können. Jeder Problemkontext, der durch das jeweilige Software-Entwicklungsproblem, die hierbei zu betrachtenden Programmier- und Modellierungssprachen, sowie die jeweils eingesetzten Reverse-Engineering-Techniken geprägt ist, erfordert hierbei jeweils *individuell definierte Repository-Strukturen*.

Die Definition solcher Repository-Strukturen durch *Schemata* und der hierauf definierten Dienste ist eine wesentliche Aufgabe zur Entwicklung von Reverse-Engineering-Werkzeugen. Aufgrund der vielfältigen Aspekte, die bei der Entwicklung dieser Schemata zu beachten sind, ist sowohl die Entwicklung solcher *Problem-angemessener Schemata* als auch die Entwicklung passender Extraktoren zeitaufwändig und teuer.

Abhilfe schafft eine Sammlung geeigneter *Referenz-Schemata*, die jeweils für einzelne Problembereiche etablierte und bewährte Schemata anbieten.

10.2 Standard- und Referenz-Schemata

Standards beschreiben allgemein akzeptierte und festgeschriebene Lösungen für bestimmte Probleme. Eine Sammlung von *Standard-Schemata* im Reverse Engineering bietet eine fest definierte Menge vollständig definierter Schemata für *alle* Reverse-Engineering Probleme. Eine solche Sammlung muss hierbei die unterschiedlichesten Programmiersprachen aus verschiedenen Programmierkulturen auf diversen Granularitäts-Ebenen und unterschiedlichste Notationen zur Visualisierung berücksichtigen. Aufgrund der Heterogenität der Reverse-Engineering Probleme und der fehlenden Flexibilität von Standards scheint die Entwicklung einer solchen allumfassenden Standard-Schema-Sammlung ein unlösbares Unterfangen.

Referenz-Schemata bieten einerseits die benötigte Stabilität, indem sie die charakteristischen Eigenschaften einer Klasse von Schemata definieren. Andererseits bieten sie auch die nötig Flexibilität, indem sie durch wenige, einfache Anpassungen (Löschen, Spezialisieren und Ergänzen von Konzepten) an konkrete Anforderungen eines Anwendungsbereichs angepasst werden können. Referenz-Schemata sind folglich genereller aber auch unvollständiger als Standard-Schemata.

Zentrale Dokumente der Software-Entwicklung sind der Programm-Quelltext und diverse visuelle Modelle. Diese Artefakte bestimmen auch die grundlegende Struktur der *Referenz-Schemata im Reengineering*. Abstraktoren basieren i. Allg. auf Repräsentationen des Quelltexts in unterschiedlicher Granularität. Visualisierer setzen i. Allg. auf Strukturen auf, die durch die zur Visualisierung verwendeten Notationen determiniert ist. In beiden Bereichen lassen sich diverse *elementare Aspekte* identifizieren, die in Schemata für unterschiedliche Problembereiche in leicht modifizierter Form berücksichtigt werden. Die charakteristischen Eigenschaften dieser elementaren Aspekte werden durch *Schema-Moleküle* beschrieben. Referenz-Schemata für einzelne Problembereiche sind aus diesen Schema-Molekülen und vorhandenen Referenzschemata zusammengesetzt.

10.3 Programmiersprachen-Sicht

Quelltext ist oft die einzige verlässliche Informationsquelle zu einem bestehenden Softwaresystem. Im Reverse-Engineering sind die Strukturen dieser Softwaresysteme zu erkennen und zu visualisieren.

Reverse-Engineering-Werkzeuge benötigen Repräsentationen der unterschiedlichsten Programmiersprachen aus verschiedenen Programmierkulturen auf unterschiedlichen Abstraktionsebenen. Schemata dieser Repräsentationen können entlang wesentlicher Sprachmerkmale klassifiziert werden. Je nach verwendeter Programmiersprache sind hierzu u. A. *Aufrufstrukturen*, *Ausdrücke*, *Enthaltenseinsstrukturen (Include)*, *Funktions/Methoden-Deklarationen*, *Klassen-Definitionen*, *Kontrollstrukturen und Anweisungen*, *Nebenläufigkeit*, *Objekt/Variablen-Deklaration*, *Präprozessor-Anweisungen*, *Templates*, *Typ-Definitionen* zu berücksichtigen. Abhängig von der, der Programmiersprache zugrunde liegende Programmierkultur, sind diese Kernaspekte durch Schema-Moleküle in unterschiedlichen Ausprägungen zu beschreiben. Beispielsweise erfordert die Modellierung der Kontrollstrukturen in Cobol, Lisp, oder Pascal unterschiedliche Ausprägungen. Referenzschemata zur Darstellung dieses Aspekts für C++ oder Java sind dagegen weitgehend ähnlich.

Referenz-Schemata für einzelne Problembereiche setzen sich dann aus geeignet angepassten Schema-Molekülen für die jeweils genutzten Aspekte in ihren jeweiligen Ausprägungen zusammen. So besteht z. B. das Columbus C++ Schema [2], das als Ausgangspunkt für ein C++ Referenz-Schema genutzt werden kann, aus Komponenten zur Modellierung von Ausdrücken (expr), Klassendefinitionen (struc), Kontrollstrukturen und Anweisungen (stmt), Templates (templ), und Typdefinitionen (Type).

10.4 Modellierungssprachen-Sicht

Informationen über Softwaresysteme werden sowohl zur initialen Erstellung als auch während der Weiterentwicklung durch visuelle Modellierungsmitteln beschrieben.

Ausgehend von geeigneten Repräsentationen der vorliegenden Quelltexte, werden im Reverse Engineering Softwarezusammenhänge z. B. durch Klassendiagramme, Datenfluss-Diagramme, Aktivitätsdiagramme, oder Sequenzdiagramme visualisiert. Diese visuellen Beschreibungsmittel stellen unterschiedliche dynamische und statische Aspekte von Software-Systemen in den Mittelpunkt: u. A. *Datenflüsse*, *Klassendefinitionen*, *Klasseninteraktionen*, *Kontrollflüsse*, *Nachrichtenaustausche*, *Objektinteraktionen*, *Systemzustände und Übergänge*. Diese Kernaspekte visueller Modellierungsmittel sind ebenfalls durch Schema-Moleküle zu beschreiben. Geeignetes Zusammenfassen, evtl. angepasster Schema-Moleküle der jeweils benötigten Modellierungsaspekte, liefert Referenz-Schemata konkreter Modellierungssprachen (also *Referenz-Metaschemata*). Entlang gemeinsam dargestellter Konzepte bzw. Aspekte lassen sich diese Referenz-Metaschemata zu integrierten Referenz-Metaschemata gruppieren [4]. Je nach Implementierungsnähe überlappen sich diese Schema-Moleküle mit den Schema-Molekülen aus Sicht der Programmiersprachen (vgl. den Aspekt *Klassendefinition*).

10.5 Abgeleitete Referenz-Schemata

Referenz-Schemata bieten einerseits mit Schema-Molekülen aus Programmier- und Modellierungsprachen-Sicht bzw. mit jeweils sprach-spezifischen Schemata eine Sammlung bewährter Schemata zur Lösung klassischer Reverse-Engineering-Probleme. Sie bieten andererseits aber auch eine Bausteinsammlung, aus der, durch geeignetes Anpassen und Zusammenfassen einzelner Bausteine, weitere Schemata abgeleitet werden können.

So lassen sich beispielsweise Schemata zur Beschreibung von *Software-Architekturen* entlang bekannter Architekturstile [1] durch Variantenbildung und Komposition aus Schema-Molekülen und Referenz-Schemata ableiten: Schemata zur Beschreibung von Software-Architekturen nach dem *Pipe-Filer-Style* erfordern Konzepte zur Modellierung von Prozessen (Filter), die Daten verändern, und Datenflüssen (Pipe), die den Datenaustausch zwischen Prozessen beschreiben. Ein solches Schema entspricht einem stark vereinfachen Metaschema zur Modellierung mit Datenfluss-Diagrammen, und lässt sich leicht aus einem Schema-Molekül zur Beschreibung von Datenflüssen ableiten. Ist eine breite Verwendbarkeit eines solchen Schemas erwiesen, kann dieses auch in die Sammlung der Referenz-Schemata übernommen werden.

10.6 Zusammenfassung

Referenz-Schemata im Reverse Engineering liefern vorgefertigte Standardlösungen für wiederkehrende Reverse-Engineering-Probleme, bestehend aus Datenstrukturen und den hierzu passenden Extraktor- und Analysediensten. Referenz-Schemata beschreiben charakteristische Eigenschaften dieser Lösungen und sind durch wenige Änderungen an weitere Reverse-Engineering-Probleme anpassbar.

Die Bereitstellung eines solchen Referenz-Schema-Werkzeugkastens erfordert neben der weiteren Entwicklung einzelner Schema-Moleküle und Referenz-Schemata, auch die Entwicklung einer Methodik zur Verwendung dieser Schemata. Diese umfasst u. A. die Bereitstellung von Operationen zur Anpassung und Integration von Schemata und deren Übertragung auf entsprechende Instanz-Daten.

Literaturverzeichnis

[1] L. Bras, P. Clements, R. Kazman. *Software Architecture in Practice*. Addison-Wesley, Boston, 1998.

[2] R. Ferenc, F. Magyar, Á. Beszédes, Á. Kiss, M. Tarkiainen. Columbus - Tool for Reverse Engineering Large Object Oriented Software Systems. In *Proceedings SPLST 2001, Szeged, Hungary*, pp 16–27. June 2001.

[3] C. Lange, H. Sneed, A. Winter. Comparing Graph-based Program Comprehension Tools to Relational Database-based Tools. in 9th International Workshop on Program comprehension. IEEE, Los Alamitos, pp 209–218. 2001.

[4] A. Winter. *Referenz-Metaschemata für visuelle Modellierungssprachen*. DUV, Wiesbaden, 2000.