



Software-Tomography

Software auf dem Prüfstand

Software-Tomography ist eine neuartige Technologie zur Analyse objektorientierter Softwaresysteme. Mit dem Sotograph steht dafür ein Softwareanalyse-Werkzeug zur Verfügung. Er durchleuchtet Software-Systeme und zeigt ihre inneren Strukturen auf.

Bei der Diskussion über die Qualität eines IT-Systems ist üblicherweise von folgenden Aspekten die Rede: Korrektes funktionales Verhalten, Ausfallsicherheit, Datensicherheit, Benutzerfreundlichkeit, Performance. Diese Liste lässt sich problemlos um weitere Qualitätsaspekte verlängern, die alle eines gemeinsam haben: Es wird das Verhalten des IT-Systems nach außen betrachtet. Die Qualitätsanforderungen werden durch geeignete Tests überprüft und sichergestellt.

Neben diesen wichtigen, traditionellen Qualitätskriterien rückt zunehmend ein anderer Qualitätsbegriff ins Bewusstsein der IT-Verantwortlichen: Die innere Qualität eines IT-Systems. Man spricht auch von struktureller oder tech-

nischer Softwarequalität. Diese blickt auf die innerer Beschaffenheit einer Software und wirft Fragen auf wie etwa:

- Mit welchem Aufwand ist die Software wartbar?
- Wie komplex ist ihre innere Struktur?
- Gibt es eine Softwarearchitektur, die sich im Quelltext wiederfinden lässt?
- Wo sind Architekturvorgaben / Designregeln bei der Implementierung nicht eingehalten?
- Können Systemteile ersetzt werden?
- Werden Technologie-Standards verwendet?
- Wie verständlich ist ein Quelltext für eine externe Person?

- Wie groß ist der Einarbeitungsaufwand für einen neuen Software-Ingenieur?
- Sind Module in einem anderen Kontext wiederverwendbar?

Die Antworten auf diese Fragen haben einen entscheidenden Einfluss auf die Lebensdauer eines Softwaresystems und die Kosten, die dafür entstehen. Bei der inneren Qualität ist insbesondere ihre Auswirkung auf die Wartungsphase von Interesse. Diese macht in den meisten Fällen mehr als 80% der Lebensdauer aus.

Zahlreiche IT-Manager haben leidvolle Erfahrungen gemacht mit schlecht wartbaren IT-Systemen. Sie werden von Ihren Software-Entwicklern mit Aussagen konfrontiert, wie: „Diese Applikation fassen wir am besten nicht an. Nach jeder Änderung brauchen wir Wochen, bis das Ganze wieder stabil läuft.“ Oder: „Diese Software sollten wir wegwerfen und alles neu implementieren.“ Oft sind aufwändige Reengineering-Arbeiten erforderlich, um diese Situation abzumildern. Dafür hat sich auch schon der Begriff Software-Sanierung etabliert.

Aus diesen Erfahrungen erhält die Forderung Nachdruck, Software von Anfang so zu schreiben, dass sie langfristig gut wartbar und erweiterbar bleibt. Eine solche „Long-life“-Eigenschaft bekommt man allerdings nicht geschenkt. Analog zur Entropie in der Physik haben Softwaresysteme die Tendenz während der Wartung und Weiterentwicklung zu degenerieren. Dieser Tendenz muss man entgegenwirken, wenn man ein Softwaresystem über längere Zeit ökonomisch warten und weiterentwickeln will. Für den damit verbundenen Aufwand wird man mit einem niedrigeren und vor allem kalkulierbaren Gesamtaufwand in den Wartungs- und Erweiterungsphasen belohnt.

Mehr und mehr IT-Abteilungen greifen das Thema „Innere Softwarequalität“ auf. In der praktischen Umsetzung erweist sich die Herstellung und Überwachung von technischer Qualität allerdings als problematisch. Wie soll beispielsweise ein Softwarearchitekt bei einem Projekt mit fünf oder mehr Softwareentwicklern sicherstellen, dass die vorgegebene Architektur im Quelltext vollständig berücksichtigt wird. Selbst

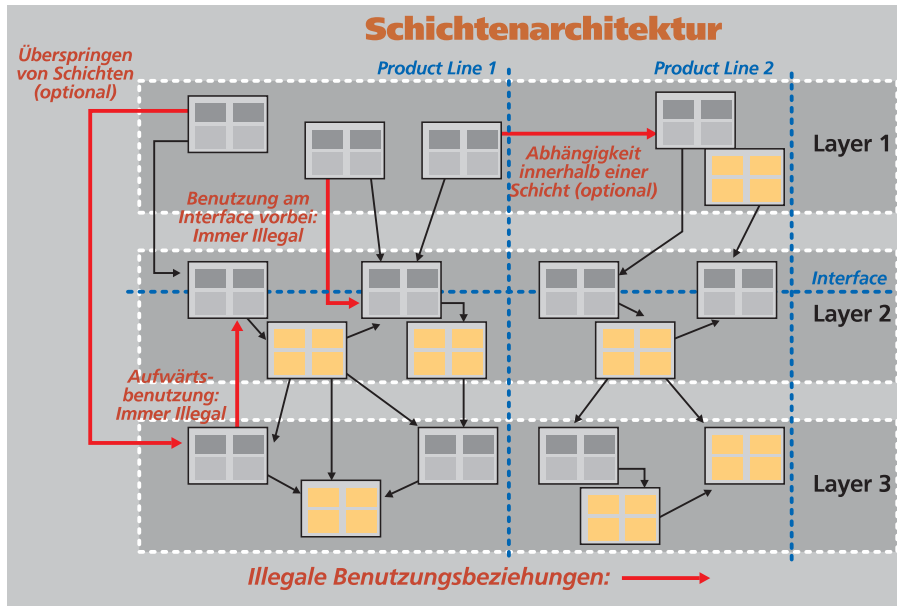


Bild 1: Schematische Darstellung einer Schichtenarchitektur zwischen (gelb markierten) Subsystemen. Die roten Linien deuten architekturverletzende Abhängigkeitsbeziehungen an, die der Sotograph automatisch suchen und anzeigen kann.

messen“. Für große Systeme ist es hilfreich, eine zusätzliche Abstraktionsebene zu definieren. Das Tool ermöglicht es, Packages in Subsysteme zusammenzufassen und diesen eine Architektur zuzuordnen. Damit wird eine Verbindung hergestellt zwischen den Artefakten der Implementierung und den Modellen auf Architekturebene. Der Sotograph kann dann automatisch prüfen, an welchen Stellen im Quelltext diese vorgegebene Sollarchitektur verletzt wird (Bild 1 und Bild 2).

Eine weitere Methode zur strukturellen Untersuchung von Softwaresystemen ist die Betrachtung von zyklischen Abhängigkeiten auf Makro- und Mikroebene. Fallstudien haben ergeben, dass es einen direkten Zusammenhang zwischen der Anzahl der Zyklen und der Wartbarkeit eines Systems gibt. Je mehr Zyklen ein System in sich trägt, um so aufwändiger sind die Wartungsarbeiten.

wenn er die Zeit hätte, alle Codeänderungen und Programmerweiterungen regelmäßig manuell nachzulesen, würde es ihm nicht gelingen, neben den wichtigen technischen Details auch noch alle Aspekte der Architektur in Betracht zu ziehen.

Hier setzt der Software-Tomograph (Sotograph) mit seiner neuartigen Technologie zur Analyse von objektorientierten Softwaresystemen an. Das Produkt ist eine Softwareanalyseumgebung, die ein Softwaresystem durchleuchtet und seine inneren Strukturen aufzeigt.

Das Werkzeug beinhaltet eine relationale Datenbank, in der die gesamten Strukturinformationen eines Systems abgelegt werden. Dies sind alle Artefakte (Attribute, Methoden, Klassen, Dateien, Packages) sowie ihre Abhängigkeitsbeziehungen (Vererbung, Aufruf, Assoziation, Typverwendung, Attributzugriffe). Diese Datenbasis kann dann für umfangreiche Analysen und graphische Darstellungen genutzt werden. Um einen besseren Überblick zu gewinnen, können die genannten Benutzungsbeziehungen nicht nur auf Methoden- und Klassenebene sondern auch auf höheren Abstraktionsebenen, nämlich zwischen Files oder Packages, betrachtet werden.

Mit Hilfe dieser Technik lassen sich auch sehr große Softwaresysteme mit mehreren Millionen Code-Zeilen „ver-

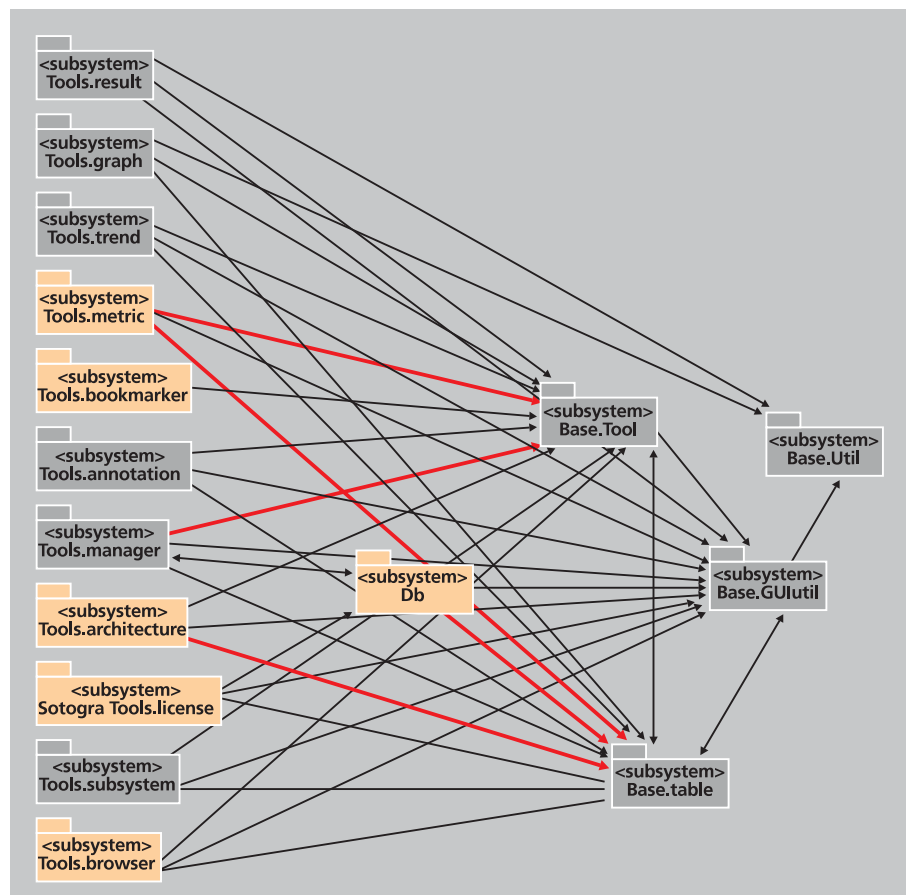


Bild 2: Graphische Darstellung der Vererbungsbeziehungen auf Subsystemebene für eine Beispielapplikation. Für die farbige markierten Subsysteme wurden bei der Architekturprüfung illegale Referenzen gefunden. Die Dicke der Linien spiegelt die Anzahl der Vererbungsbeziehungen wider. Durch einen Mausklick auf eine Verbindungslinie kann die Detailliste der zugrundeliegenden Vererbungsbeziehungen angezeigt werden. Das Bild wurde mit dem Sotographen erzeugt.

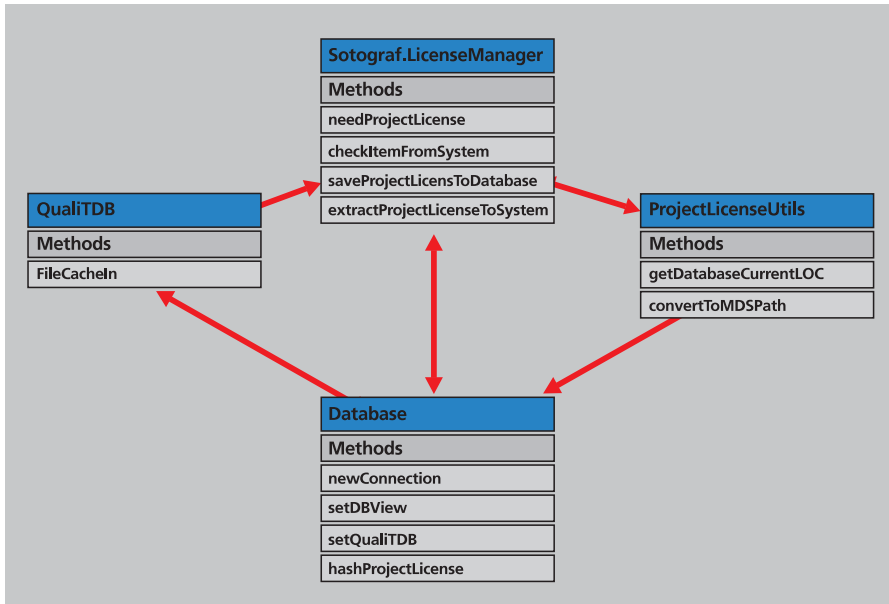


Bild 3: 4stufiger Zyklus auf Klassenebene. Die Pfeile stehen für Abhängigkeitsbeziehungen beliebiger Art. Zusätzlich sind die Methoden eingeblendet, die an dem Zyklus beteiligt sind (beispielsweise durch Aufruf oder Typverwendung). Das Bild wurde mit dem Sotographen erzeugt.

Dabei können sich globale Zyklen deutlich negativer auswirken als solche, die auf einen lokalen Teilbereich begrenzt sind. Dieses Ergebnis ist nicht überraschend. Eine zyklische Struktur ist schlechter verständlich, da es keine klare Zuordnung von Aufgaben und Rollen gibt. Einzelne Artefakte innerhalb des Zyklus können nicht so einfach durch neue ersetzt werden. Darüber hinaus können zyklisch gekoppelte Systemelemente nicht mehr einzeln getestet werden. Es muss immer das gesamte Teilsystem, das den Zyklus enthält, in den Test.

Der Sotograph kann die Zyklen in einem System finden und detailliert darstellen. Dies liefert wertvolle Informationen für Reengineering-Tätigkeiten. Manuell ist es fast nicht möglich mehrstufige Zyklen in einem großen System zu identifizieren (Bild 3).

Der Sotograph hat ein integriertes Metrikwerkzeug, das eine Vielzahl von vordefinierten Metrikwerten ermitteln kann. Dabei können diese Werte für größere Einheiten wie Packages oder Subsysteme aggregiert werden. Metriken sind ein gutes Hilfsmittel, um Aussagen über die Komplexität und die Qualität eines Systems treffen zu können. Sie liefern wichtige Hinweise auf potentiell problematische Codestellen

und können so auch als Indikator dienen, um gezieltere Code-Reviews oder Tests durchzuführen (Bild 4).

Die hier exemplarisch beschriebenen Analysearten liefern relevante Informationen zur Beurteilung eines Softwaresystems. Aufgrund der potentiell großen Informationsmenge bietet der Sotograph mehrstufige Filter- und Visuali-

sierungskonzepte. Damit wird die Interpretation der Informationen zwar erleichtert – eine gewisse Zeit kann dies trotzdem in Anspruch nehmen. Deshalb ermöglicht der Sotograph nicht nur die Analyse für einen Stand des Softwaresystems, sondern auch für eine Reihe von Versionsständen. Nach einer Erstanalyse genügt es dann, die Unterschiede zum letzten Stand zu untersuchen; das heißt es müssen etwa nur noch neu hinzugekommene Architekturverletzungen, Zyklen oder erhöhte Metrikwerte betrachtet werden. So ist es für einen Qualitätsbeauftragten, Architekten oder technischen Projektleiter mit geringem Zeitaufwand (ca. 30 Minuten wöchentlich) möglich, potentielle Probleme zu erkennen und frühzeitig gegenzusteuern.

Fazit

Mit seiner Kombination aus trendfähigen Analyse und Visualisierungswerkzeugen kann der Sotograph sowohl bei Neuimplementierungen wie auch bei Softwaresanierungsaktivitäten, Aufwandsabschätzungen, Code-Assessments oder Abnahmeverfahren von zugelieferter Software eingesetzt werden. Mit ihm wird es möglich, die technische Softwarequalität mit vernünftigem Aufwand kontinuierlich zu überwachen und zu verbessern.

Thomas Schoen
schoen@software-tomography.com

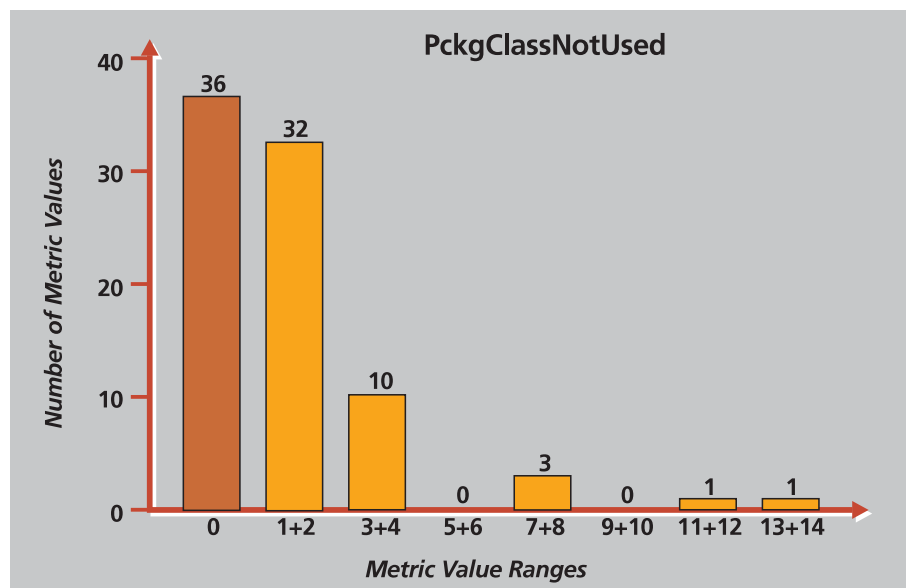


Bild 4: Dieses Chart zeigt die Werteverteilung der Beispielapplikation für die Metrik PckgClassNotUsed (Anzahl der Klassen in einem Package, die nicht verwendet werden). Die roten Balken signalisieren Metrikwerte außerhalb des erlaubten Wertebereichs. Das Bild wurde mit dem Sotographen erzeugt.