Herstellen der Beziehung von Testfällen zu Codekomponenten mittels statischer und dynamischer Analyse

Martin Hasitschka (martin.hasitschka@sds.at) und Harry Sneed (harry.sneed@sds.at), Software Data Services

In Software, die nach dem Cleanroom Approach entwickelt wird, werden die Testfälle und der Code parallel zueinander auf Basis der Spezifikationskomponenten entwickelt. Daher sind nur die Beziehungen zwischen Spezifikationskomponente und Testfall einerseits und Spezifikationskomponente und Codekomponente andererseits bekannt.
Ist die Software einmal in Produktion, dann stellt sich vor Zwischenlieferungen eines Teils der Codekomponenten die Frage, welche Testfälle zu ihnen gehören, denn sie haben bei der Testplanung absoluten Vorrang. Verschiedene Zugänge, sie zu beantworten, wurden im Projekt GEOS (Global Entity Online System) beschritten. Dabei handelt es sich um ein Wertpapierabwicklungssystem auf einer mehrstufigen Client-Server Architektur. Es ist z.T. in C++ und z.T. in C implementiert. Das Codevolumen beträgt ca. 6,7 Mio LOC.

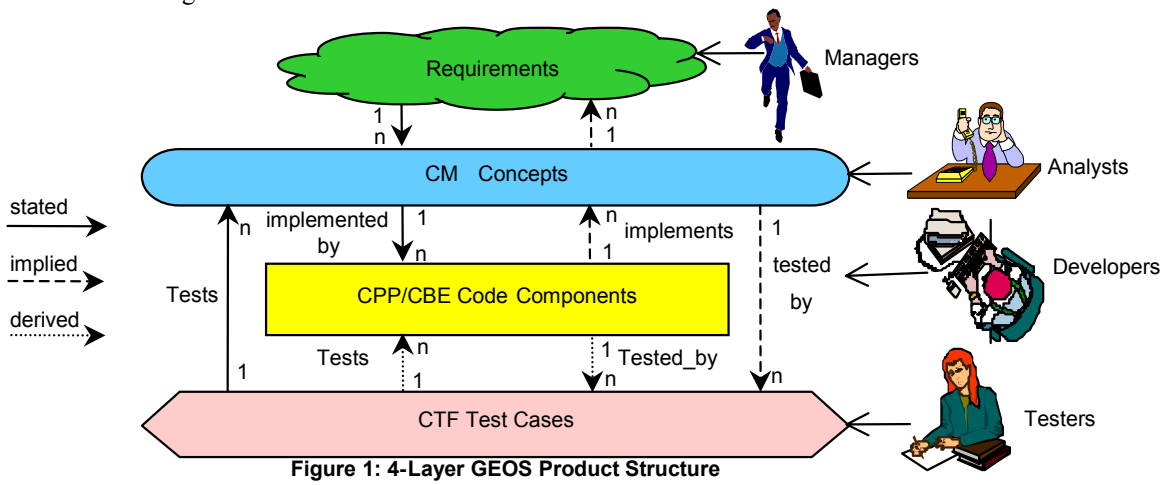

Figure 1: 4-Layer GEOS Product Structure

GEOS ist auf vier semantischen Ebenen beschrieben, die den vier Hauptphasen der Softwareentwicklung der SDS entsprechen:

- Anforderungsebene
- Konzeptebene (Systemspezifikation)
- Codeebene und
- Testebene

Jede Ebene wird von einem separaten Team betreut. Für die vorliegende Arbeit sind nur die letzten drei interessant.
Eine Konzeptkomponente besteht aus Geschäftsregeln, Pseudocode, Parameterlisten und natürlichsprachigen Beschreibungen. Zu jeder Konzeptkomponente ist die Codekomponente bekannt, die ihre Geschäftslogik implementiert. Da aber jede Codekomponente beliebig viele andere aufrufen kann, gibt es in Wahrheit eine m:n Beziehung zwischen Konzeptkomponenten und Codekomponenten.

Für GEOS sind ca. 80000 Testfälle definiert. Zu jedem werden die Konzeptkomponenten, die dadurch betroffen sind, gewartet. Auch hier gibt es eine m:n-Beziehung. Die Testfallerstellung erfolgt z.T. manuell, z.T. automatisch aus den Konzepten.

Die Kernfrage ist: **Welche Testfälle testen welche Module?**
Zwei Lösungsansätze wurden untersucht:

1. Statische Analyse
2. Dynamische Analyse

Statische Analyse

Die statische Analyse baut auf der *Implementiert*-Beziehung zwischen Konzeptkomponenten und Codekomponenten und der *Testet*-Beziehung zwischen Konzeptkomponenten und Testfällen auf.
Gibt es zu einer Codekomponente X eine Konzeptkomponente, deren Geschäftslogik sie implementiert, dann besteht das Testfallset für X aus allen Testfällen, die diese Konzeptkomponente betreten.
Oft existiert aber keine direkte Beziehung zwischen X und irgendeiner Konzeptkomponente. Da aber durch die Änderung an X sich auch das Verhalten aller seiner Aufrufer ändert, werden auch die dazu gehörigen Konzeptkomponenten für die Ermittlung des Testfallsets herangezogen.
Der statische Zugang ist schneller und billiger, hat aber zwei Nachteile:

1. Die Beziehungen werden z.T. händisch gewartet und sind daher unvollständig dokumentiert
2. Die Aufrufer einer Codekomponente lassen sich wegen des dynamischen Bindens nur begrenzt durch statische Analyse ermitteln

Dynamische Analyse

Die dynamische Analyse basiert auf einer Instrumentierung des Source Codes. Jedes Codemodul wird so verändert, dass es seine Aufrufzeit protokolliert. Auch die Testfälle werden so verändert, dass sie die Beginn- und Endezeitpunkte ihrer Durchführung protokollieren. Die beiden Protokolle werden abgeglichen. Alle Codemodule, die zwischen dem Beginn- und Endezeitpunkt eines Testfalls betreten wurden, werden diesem zugerechnet.