GI-2.1.7. TAV AK-TOOP

Protokoll des Treffs am 29.10.99 in Stuttgart

Anwesende

Stefan Bisanz, Uni Bremen / Verified Systems Int.
Michael Brunner, TU Darmstadt
Falk Fraikin, TU Darmstadt
Uwe Hehn, 3Soft
Marija Jovanovic, TU Wien / Anecon
Thomas Leonhardt, TU Darmstadt
Sven Nordhoff, DTK
Peter Rüppel, GEBIT
Andreas Spillner, Hochschule Bremen
Mario Winter, FernUniversität Hagen

Ablauf

$11^{\underline{30}}$ - $11^{\underline{50}}$ Uhr	Begrüßung, Vorstellung der Teilnehmer
$11^{\underline{50}}$ - $12^{\underline{50}}$ Uhr	Diskussion: Xtreme Programming und Testen
$12^{\underline{50}}$ - $13^{\underline{30}}$ Uhr	Mittag
$13^{\underline{30}}$ - $15^{\underline{20}}$ Uhr	Diskussion und Modellbildung: OO-Testbarkeit
$15^{\underline{20}}$ - $16^{\underline{00}}$ Uhr	Diskussion: Weiteres Vorgehen des TOOP

Themen

Diskussion: Xtreme Programming und Testen

Hinter dem Schlagwort „Xtreme Programming“ verbergen sich eine Reihe von bekannten Techniken der Programmierung (im Kleinen):

- Iterative, inkrementelle Entwicklung
- Pair Programming
- Code Sharing
- Refactoring
- Unit Testing

Allem zugrunde liegt die Attitüde „Make it simple“, so dass in kleinen Teams (8-12 Personen) ohne großartige Modellbildung schrittweise von einfachen Programmen als Teillösungen hin zu komplexeren Programmen als endgültige Lösungen einer Entwicklungsaufgabe gearbeitet wird. Hierbei werden vor der Programmierung einer Operation ausführbare Testfälle quasi als Spezifikation der auszuprogrammierenden Operation entwickelt. Diese werden nach der Implementation ausgeführt und vor bzw. nach jeder Änderung an einer Operation fortgeschrieben bzw. erneut ausgeführt. Durch die Paarbildung (ein Entwickler gibt Code ein bzw. ändert, der zweite schaut ihm über die Schulter und kontrolliert; dabei erfolgt ständig ein Wissensaustausch) sind informelle Code-Reviews sozusagen in das Xtreme Programming „eingebaut“.

Im Rahmen unserer Diskussion ergaben sich folgende Fragestellungen, die weiter verfolgt werden sollen:

- Wer testet die Tests? Einer der hauptsächlichen Kritikpunkte ist die informelle Testfallerstellung durch den Entwickler, bei der ja der Test gleichzeitig die Spezifikation z.B. einer Operation ist. Wogegen testet man also, und was sind die Testendekriterien?
- Wie lassen sich die Testfälle des Xtreme Programming z.B. im Integrationstest wiederverwenden? Reicht die einfache, z.B. in [BeckGamma99] beschriebene Spezifikationstechnik für Testfälle aus, oder sollte sie erweitert werden (but always remember: keep it simple...)?
- Wie lässt sich Xtreme Programming in allgemeine Vorgehensmodelle einbetten?

Quellen:

[BeckGamma99] Kent Beck, Erich Gamma: Test Infected - Programmers Love Writing Tests.
http://members.pingnet.ch/gamma/junit.htm

Unter

http://www.extremeprogramming.org/index.html und
http://www.xprogramming.com/

findet man viele Info's über XP.

Fragen zu XP und QA sind unter

http://www.xprogramming.com/qa/xp_q_and_a_QA.htm

diskutiert.

Diskussion und Modellbildung: OO-Testbarkeit

Nach ausführlicher Diskussion über Einflussfaktoren etc. in Bezug auf die Testbarkeit von (objektorientierter) Software einigten wir uns auf folgende Arbeitsdefinition:

> Testbarkeit gibt die zu erwartenden Kosten an, mit denen ein Prüfling bzgl. vorgegebener Qualitätsanforderungen geprüft werden kann.

Ein rudimentäres Modell hierzu ist in Abb. 1 dargestellt. Eingaben sind der Prüfling und die Qualitätsanforderungen, Ausgabe die „Testbarkeit" des Prüflings bzgl. der Qualitätsanforderungen. Im Prinzip wird jede Qualitätsanforderung über Testendekriterien für ein oder mehrere Testverfahren operationalisiert, deren zu erwartende Kosten für den Prüfling durch Metriken ermittelt werden. Durch eine Gewichtung der Metriken wird dann ein „Testbarkeitsindex" T berechnet.

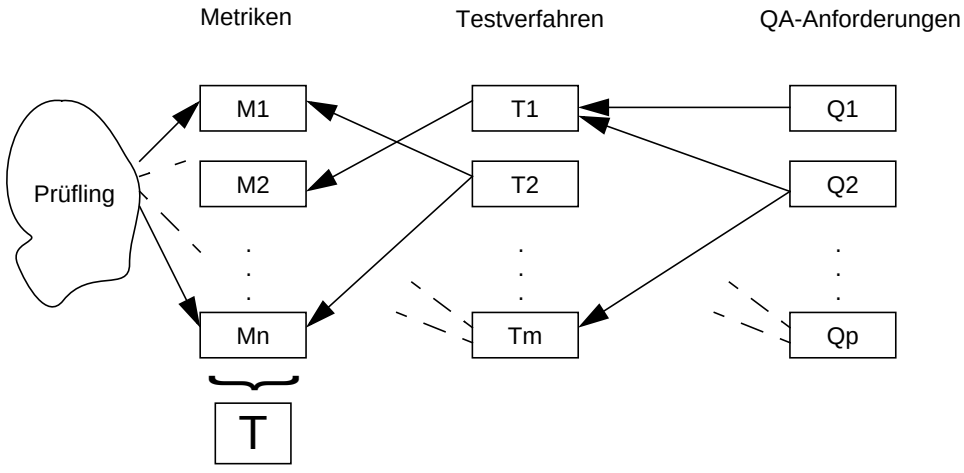

Abbildung 1: Einfaches Testbarkeitsmodell

Da sich Stefan Jungmayer mit diesem Thema beschäftigt, soll sein auf der EuroSTAR 99 in Bar-

celona publiziertes Papier über Testbarkeitsreviews [Jungmayer99] an alle Mitwirkenden verschickt werden (ist bereits geschehen). Bis zum nächsten Treff des AK soll per eMail der Testbarkeitsbegriff für OO-Software weiter diskutiert und anhand einiger Artefakte der OO-Entwicklung konkretisiert werden (z.B. für bestimmte UML-Diagramme, Programmiersprachen etc.).

Quellen:

[Jungmayer99] S. Jungmayr: Reviewing Software Artifacts for Testability. Proceedings of the EuroSTAR ’99, Barcelona, November 10th - 12th, 1999.

Weiteres Vorgehen des TOOP

Wir wollen den Fragebogen für Werkzeughersteller noch einmal publizieren. Dazu werden im wesentlichen die Anmerkungen zu den einzelnen Fragen erweitert und überarbeitet, um eine größere Verständlichkeit zu erzielen. Die Promotion des Fragebogens wollen wir wieder in die Hände von Stephanie Ulrich (Journale) und Stefan Jungmayr (Internet) legen. Der Fragebogen wurde von Andreas Spillner auch anderen europäischen Test-SIG’s bekanntgemacht.

Wie bereits oben gesagt soll der Testbarkeitsbegriff für OO-Software weiter diskutiert und anhand einiger Artefakte der OO-Entwicklung konkretisiert werden

Nächster Treff

Das nächste Treffen des AK-TOOP findet im Rahmen des 15. Treffen der Fachgruppe am 18.-19. Mai 2000 bei der GMD in Sankt Augustin bei Bonn statt. Themenvorschläge (neben OO-Testbarkeit) werden erbeten an Mario Winter.