Gedächtnisprotokoll mündliche Diplomprüfung Informatik Vertiefung / Softwaretechnik / Nagl (i3)

Gebiet:	Vertiefung / Softwaretechnik	*Prüfer*:	Nagl
Datum:	Oktober 2006	*Dauer*:	50 Minuten
Note:	1,3		

Vorlesungen

Nagl: *Einführung in die Softwaretechnik* (SWT), *Programmieren im Großen* (PiG), *Ada*.
Westfechtel: *Management von Softwareentwicklungsprozessen* (MSP), *Spezifikationssprachen* (SL).

Materialien

- Folien der fünf Vorlesungen.
- Kimm, Koch, Simonsmeyer, Tontsch: *Einführung in Software Engineering*. De Gruyter ; Berlin, New York ; 1979. ISBN 3-11-007836-8.
- Nagl: *Softwaretechnik: Methodisches Programmieren im Großen*. Springer ; Berlin [u. a.] ; 1990. ISBN 3-540-52705-2.
- Nagl: *Softwaretechnik mit Ada 95*. Vieweg ; Braunschweig, Wiesbaden ; 1999. ISBN 3-528-05583-9.
- Prüfungsprotokolle und wichtige Hinweise (PKs Tips), Zusammenfassungen und Übersichtstabellen, können alle in einem Ordner am Lehrstuhl i3 kopiert werden (Mitarbeiter in Raum 4306 fragen).[1]

Die drei genannten Bücher sind alle nah an den entsprechenden Vorlesungen, Nagls eigene Bücher fast 1:1 (das Buch von Kimm et al. deckt allerdings einige Kapitel der Vorlesung SWT gar nicht ab). Sie sind mit ihren zusätzlichen Erläuterungen besonders dann hilfreich, wenn man sich wenig oder keine Anmerkungen in seine Folienausdrucke geschrieben hat.

Bei den zwei kleinen Westfechtel-Vorlesungen hatte ich ein ungutes Gefühl, weil sie in Protokollen fast nie vorkamen. Ich habe sie nur ganz oberflächlich gelernt und das nicht bereut. Wie in den wenigen anderen Protokollen angedeutet, in denen sie vorkommen, wurden sie nur am Ende kurz angerissen.

Die Übungen zu den Vorlesungen habe ich im Rahmen der Vorbereitung für die Prüfung nicht mehr durchgelesen. Ich hatte sie aber teilweise besucht.

Vorwort

Die Prüfung war so, wie sie bereits in vielen anderen Protokollen beschrieben wurde. Nagl ist ein entspannter und ruhiger Prüfer, er gibt lange Einführungserklärungen zur Frage, er hilft bei

[1] evtl. muß man ein Geldpfand hinterlegen, bis man ein eigenes Protokoll einreicht

Problemen, bricht Antworten im Satz ab, wenn er sieht, dass der Prüfling verstanden hat, worum's geht. Er selbst hat auch kein Problem, wenn man ihn unterbricht, um eigenes Wissen einzubringen.

Wichtig sind ihm vorlesungsübergreifende Konzepte und die Beziehung von allem zur Architektur. Auswendig gelernte Definitionen prüft er wenn überhaupt nur in SWT, und sobald er merkt, dass diese Grundlagen beim Geprüften da sind, geht er über zu den Konzepten. Fragen wie „Nennen sie alle Konsistenzregeln für Module“ (sind immerhin über 20) stellt er wohl nur, wenn sonst nichts klappt.

Prüfungsverlauf

Softwaretechnik

- Qualitätssicherung – die acht Kriterien für Prüftechniken aufgezählt und kurz erläutert.
- Für die Prüftechnik Test diese Kriterien durchgespielt (schöne Tabelle dazu im Lehrstuhlordner, siehe oben bei Materialien).
- Kurz den Integrationstest angerissen (nur Variante Big Bang: Testtreiber oben, einzelner Modul, Teststummel unten).
- Modultest: Whitebox, Flußgraph – für ein gegebenes Beispiel (Schleife mit 100 Iterationen, darin nur eine if-then-else-Anweisung) ein Ablaufdiagramm aufgezeichnet. Dann die Anzahl möglicher Varianten ermitteln. Dabei habe ich ein paar Probleme, mit Hilfe von Nagl geht es dann. Es gibt drei Varianten, (a) immer in den then-Teil verzweigen, (b) immer in den else-Teil verzweigen, (c) eine beliebige Mischung von beiden. Verteilung der Fälle abschätzen: (a) und (b) je genau ein Fall, (c) alle anderen Fälle, also bei 100 Iterationen mit je zwei Möglichkeiten 2^{100} (er fragt nach, wieviel das ist; ich sage etwa 10^{30}, was er nicht hört oder was ihm nicht gefällt[2]) minus die zwei Fälle aus (a) und (b).

Programmieren im Großen

Die Einstiegsfrage zielt auf Objektorientierung und eingebettete Systeme ab, wo ist bei der Verbindung eine Problem? Da ich nicht weiß, was er meint, nenne ich seinen Standardeinwand gegen OOP (man erzeugt einen großen Baum von allgemein benutzbaren Klassen, was Nagls Architektursicht widerspricht, die auch Enthaltensein und Lokalität an geeigneten Stellen fordert). Das ist es aber nicht. Ich erwähne vorsichtig, dass z.B. Java schon ziemlich lange für eingebettete Systeme benutzt wird, insofern wüßte ich nicht, wo da ein allgemeines Problem läge. Es läuft wohl doch darauf hinaus, dass OO-Systeme mehr Laufzeitressourcen benötigen. Das steht offenbar in Extra-Folien zur Diskussion zu OOP, die er mit den Studenten in der Vorlesung PiG stets führt, wenn OOP gerade durchgenommen wird. Außerdem: rein funktionale Anteile im Code lassen sich schlecht umsetzen (Nagl betrachtet Klassen als Hilfsmittel zur Datenabstraktion). Evtl. auch problematisch: bei OO läßt sich der Kontrollfluß wegen der dynamischen Bindugn schlecht nachvollziehen. Beides dürften allerdings allgemeine OO-Kritikpunkte sein.

Standardfrage nach dem integrativen Ansatz, der in der Vorlesung besprochen wurde, was wird da integriert? Ich halte einen relativ ausführlichen Kurzvortrag zu Modulen (Baustein, logisch unabhängig, Exportschnittstelle plus Rumpf, usw.), Modularten (funktional: aktiv, Transformation / Steuerung, kein Gedächtnis; Datenabstraktion: passiv, Gedächtnis), Teilsysteme (schachtelt Module und evtl. andere Teilsysteme, Information Hiding, zur Arbeitsteilung geeignet). An der Stelle reicht es ihm, obwohl natürlich noch einige Dinge fehlen. Ich hätte noch erwähnen soll, dass der Ansatz deswegen *integrativ* heißt, weil Konzepte unterschiedlicher Herkunft vereinigt werden.

Wie ist das Verhältnis zwischen Objektorientierung und Generizität? Man kann Generizität mit OO simulieren (Parameter in eigenen ADT stecken), was aber in der Praxis nicht empfehlenswert

[2] sein Vorgehen: 2^{10} = etwa 1000, das zehnmal mit sich selbst multiplizieren; da $1000^{10} = 10^{30}$ meinen wir dasselbe, was ich auch sage

ist. Habe ich so aus dem PiG- oder Ada-Buch. War nicht ganz das, was er hören wollte, ist aber als Antwort in Ordnung. Ich nenne eine Liste mit dem Elemententyp als generischem Parameter als Beispiel für eine Anwendung. Sein Lieblingsbeispiel zum Thema OOP + Generizität ist eine Kollektion, deren Eintragstyp eine Oberklasse einer OO-Klassenhierarchie ist. Man kann dann Objekte verschiedener abgeleiteter Klassen in der Kollektion unterbringen.

Da eben schon das Stichwort Java fiel, soll ich mal kurz durchspielen, wie die genannten Konzepte (Module, Modularten, Teilsysteme) durch Java unterstützt werden. Das ist unerwartet, aber bitte: Module als Klassen, Modularten (funktional: emulieren durch Verzicht auf Datenfelder und mit statischen Klassen, ADT/ADO: normal mit über Methoden gekapselte Datenfelder), Teilsysteme (es gibt anonyme bzw. innere Klassen zur Schachtelung und zum Verstecken nach außen hin; zum Ausdrücken der Gemeinsamkeiten der beteiligten Klassen gibt es den Namensraum über die Packages). Er wollte als Fazit hören, dass man vieles simulieren kann, aber keine saubere 1:1-Übersetzung möglich sei.

Ada[3]

- Standardfrage zu geschützten Objekten bzw. Typen – wozu wurden diese in Ada 95 eingeführt? Weil ein eigener Prozess für einen Datenpuffer zuviel Aufwand ist. Eine normale Liste kann man nicht benutzen, weil der gegenseitige Ausschluß nicht gewährleistet ist.
- Genaue Frage weiß ich nicht mehr. Ada-Maschine stellt beliebig vielen Lesern und einem Schreiber Ressourcen bei geschützten Objekten zur Verfügung. Wichtig bei eingebetteten Systemen, die geschützte Objekte nutzen: der Schreiber muß zum Zug kommen, das kann (muß?) man über einen eigenen Kontrollprozess erreichen, der den Schreiber gegenüber den Lesern mit geeigneten Barrieren bevorzugt (Buch S. 388f).

MSP

Einstiegsfrage weiß ich nicht mehr, ich nenne die drei Arbeitsbereiche Prozesse, Aktivitäten und Ressourcen und dafür jeweils Beispiele. Ich weiß nicht mehr, ob der Begriff Konfiguration fällt, aber wir schwenken um zu Nagls Sicht auf Konfigurationen und Dynamik (SWT Kapitel 3). Ich erläutere kurz Beispiele, wie sich die Rückgriffdynamik auswirken kann. Er erzählt dann noch etwas darüber, wie die Prozesse nicht vorherbestimmbar sind und sich während der Ausführung ändern, was glaube ich der Evolutionsdynamik während eines Rückgriffs entspricht. Der Begriff fällt allerdings nicht.

SL

Hier nur die Frage „Was haben Sie in der Vorlesung gemacht?“ Dazu hatte ich mir einen Kurzvortrag zurechtgelegt, den ich abspule. Am Anfang des Entwicklungsprozesses (RE, PiG) in der Praxis oft informelle Dokumente, deswegen Mißverständnisse, was der Kunde will. Spezifikationssprachen helfen, weil sie formal definiert sind und schnell abprüfbare Ergebnisse bringen (sie sind ausführbar, oder über Rapid Prototyping, oder Codegenerierung). Problematisch sind die hohen Anforderungen an die Entwickler, und zur Kommunikation mit Kunden sind sie auch eher ungeeignet. Nagl hört nur stumm zu.

Er fragt nach der Uhrzeit, 50 Minuten sind vorbei, wir sind fertig. Nach kurzer Besprechung mit dem Beisitzer in meiner Abwesenheit Note 1,3. Fazit: Ich hatte eine Mischung aus Standard- und Exotenfragen, letztere konnte ich nie direkt, aber größtenteils auf Umwegen beantworten. Viel Erfolg bei Eurer Prüfung, und erstellt bitte auch ein Protokoll!

[3] Der Ada-Teil war sehr kurz, ich bin froh, nicht die Syntax für die zig Sprachmittel auswendig gelernt zu haben (im Buch Kapitel 3 und 4). Wie erwartet beschränkt sich alles auf Nebenläufigkeit und typische Anwendungsmuster in Ada (Kapitel 6). Kapitel 5 ist aber ebenfalls wichtig. Apropos Buch, Vorsicht, was den Umfang angeht. Sind zwar nur (!) 500 Seiten, aber in Minischrift und fast nur Text. Also genügend Zeit einplanen.