Opdracht 1: Softwaremetrieken

INTRODUCTIE

Dit is de eerste practicumopdracht van de cursus Software evolution. Het doel van deze opdracht is om een onderhoudbaarheidsmodel te realiseren met behulp van een aantal softwaremetrieken. Met dit model worden twee softwaresystemen, beide geschreven in Java en van verschillende omvang, geanalyseerd op hun onderhoudbaarheid.

LEERDOELEN

Na het maken van deze opdracht, wordt verwacht dat u

- de kwaliteit en structuur van een bestaand softwaresysteem te analyseren door het extraheren van feiten, en hier conclusies uit te trekken
- een afweging te maken tussen de voor- en nadelen van softwaremetrieken bij het bepalen van de productkwaliteit
- softwaremetrieken toe te passen op een bestaand systeem.

Studeeraanwijzing

Deze opdracht wordt individueel gemaakt; op verzoek en na goedkeuring van de examinator mag er worden samengewerkt in een tweetal. Lees vooraf het artikel over het onderhoudbaarheidsmodel grondig door en maak enkele van de oefeningen met Rascal om de taal te leren kennen. Op Studienet staat een FAQ voor het gebruiken van Rascal.

Software

In deze opdracht gebruiken we de domeinspecifieke programmeertaal Rascal. Informatie over de installatie van Rascal in de Eclipse-omgeving is te vinden op de Rascal-webpagina: http://www.rascal-mpl.org/.

Inleveren

Stuur uw uitwerking op per e-mail naar de examinator van de cursus. Vermeld daarbij duidelijk uw naam, uw studentnummer en het versienummer van de opdracht. Een inzending bestaat uit de Rascal code en een verslag (pdf of Word document). Maak tijdig een afspraak met de examinator om uw opdracht mondeling toe te lichten. De afspraak kan al voor het inleveren worden gemaakt; wel moeten code en verslag minimaal 24 uur voor de afspraak in het bezit zijn van de examinator.

Opdracht 1: Softwaremetrieken

In de eerste practicumopdracht bij de cursus Software evolution worden softwaremetrieken bestudeerd. Softwaremetrieken worden bijvoorbeeld door de Software Improvement Group (SIG, http://www.sig.eu/) gebruikt om snel een overzicht te krijgen van de kwaliteit van een softwaresysteem, en om mogelijke probleemgebieden te identificeren die lastig te onderhouden zijn. Enkele interessante vragen over het gebruiken van metrieken zijn:

1. Welke metrieken gebruik je?
2. Hoe worden deze metrieken uitgerekend?
3. Hoe goed geven de metrieken weer wat je werkelijk wilt weten over een systeem en hoe bepaal je dat?
4. Hoe kunnen de bovenstaande punten verder worden verbeterd?

Het onderhoudbaarheidsmodel van het SIG geeft een antwoord op de eerste vraag. Meer informatie over het model is te vinden in het artikel:

> I. Heitlager, T. Kuipers, and J. Visser. A practical model for measuring maintainability. In *Proceedings of the 6th International Conference on Quality of Information and Communications Technology*, QUATIC '07, pages 30–39, Washington, DC, USA, 2007. IEEE Computer Society.

De tweede vraag ('Hoe worden deze metrieken uitgerekend?') is het onderwerp van deze practicumopdracht. De overgebleven vragen zouden interessant kunnen zijn voor een vervolgonderzoek.

Opdracht
Schrijf een Rascalprogramma dat het onderhoudbaarheidsmodel van het SIG implementeert. Met dit programma moeten systemen die zijn geschreven in Java geanalyseerd kunnen worden. De metriek over unit test coverage is optioneel. Het implementeren van de andere vier metrieken is verplicht:
– volume
– complexiteit per eenheid
– duplicatie
– grootte per eenheid.

De Javasystemen waarop de metrieken moeten worden toegepast zijn:
– SmallSQL: dit is een systeem van beperkte omvang. Het bepalen van de metrieken voor dit systeem is een minimale eis om de opdracht te halen. De broncode is te downloaden vanaf SourceForge[1]:
  http://sourceforge.net/projects/smallsql/
– HyperSQL Database Engine (HSQLDB): dit is een omvangrijker systeem. Het bepalen van metrieken voor dit systeem is geen vereiste maar geeft wel een betere beoordeling. De code is ook te downloaden vanaf SourceForge:
  http://sourceforge.net/projects/hsqldb/

Verslag
Bij het uitwerken van deze opdracht schrijft u een kort verslag met daarin de resultaten van de onderhoudbaarheids-analyse. Dit verslag zal het uitgangspunt zijn van de mondelinge nabespreking. In het verslag moeten tenminste de volgende punten aan bod komen:
– *Aannames.* Welke aannames heeft u gemaakt bij het implementeren van de metrieken, bijvoorbeeld vanwege onduidelijkheden in de specificatie? In welke mate beïnvloedt dit de uitkomsten? Welke principes heeft u gehanteerd bij de implementatie?
– *Uitkomsten.* De uitkomsten van de metrieken op de softwaresystemen. Dit kan bijvoorbeeld de uitvoer van het programma zijn.

[1] Download de code via het grijze tabblad 'Files' en kies de laatste versie.

– *Validiteit.* Hoe nauwkeurig zijn de gevonden uitkomsten, en op welke manier is de juistheid gevalideerd?
– *Interpretatie.* Hoe moeten de uitkomsten worden geïnterpreteerd wat betreft de onderhoudbaarheid van het bestudeerde systeem? Wat zijn de scores op systeemniveau van de vier onderhoudbaarheids-subkarakteristieken? Zijn er risicogebieden aan te wijzen?

Hou als richtlijn voor de lengte van het verslag twee pagina's aan gevuld met tekst, exclusief getoonde programmauitvoer, figuren en tabellen.

Aanwijzing
Voor het berekenen van een aantal metrieken moeten eerst alle Javamethoden uit een project worden ingelezen. Met Rascal's M3 bibliotheek kan dit gemakkelijk worden gedaan. Het onderstaande codefragment laat zien hoe alle methoden van een project kunnen worden geprint:

```
module ToonMethoden

import lang::java::jdt::m3::Core;
import IO;

public void toonMethoden(loc project) {
  M3 model = createM3FromEclipseProject(project);
  for (loc l <- methods(model)) {
    str s = readFile(l);
    println("=== <l> ===\n<s>");
  }
}
```

Met de methode `methods` worden alle methodedeclaraties uit een `M3` model teruggegeven. Het resultaat is een verzameling locaties. Zo'n locatie kan vervolgens als parameter worden meegegeven aan `readFile` die slechts een gedeelte van een bestand zal inlezen. Een locatie kan ook worden omgezet naar een abstracte syntax boom (zie de module `lang::java::jdt::m3::AST`). Probeer de werking van de methode bijvoorbeeld eens uit met de volgende aanroep:
`toonMethoden(|project://JabberPoint|);.`

Beoordeling
Deze opdracht wordt beoordeeld door de uitkomsten van het onderhoudbaarheidsmodel en de Rascalcode te laten zien en deze toe te lichten in een korte interactieve sessie. Bij de beoordeling wordt op de volgende punten gelet:
– Hoe volledig is de implementatie van het SIG model? Een onvolledige uitwerking van het model kan voldoende zijn, maar hier wordt rekening mee gehouden bij het toekennen van een cijfer. Het bepalen van de unit test coverage is een optioneel onderdeel.
– Zijn de resultaten gelijk aan die van het SIG? Eventuele verschillen moeten kunnen worden uitgelegd.
– Hoe simpel is de implementatie? Bonuspunten worden toegekend voor elegante, simpele, begrijpbare programma's.
– Er wordt niet gelet op efficiëntie.

Zie ook de eisen en criteria bij de opdrachten die op Studienet zijn te vinden onder 'Studietaken'.