Análise, Modelação e Teste

Métodos Formais em Engenharia de Software

22 de Julho de 2009

1. Considere o modelo do sistema de informação do mestrado de informática em anexo:
   (a) Qual o objectivo dos invariantes `A`, `B`, `C`, e `D`?
   (b) Especifique o invariante `NotasGrupos`, cujo objectivo é garantir que todos os elementos do mesmo grupo têm a mesma nota.
   (c) Defina asserções e predicados para verificar a correcção e consistência da operação `LancaNota` em relação ao invariante definido.
   (d) Corrija a especificação da operação `LancaNota` por forma a garantir a sua correcção e consistência.
2. As ferramentas JML exploradas durante o curso foram *Runtime Assertion Checking* e *Extended Static Checking*. Explique, com base na sua experiência, quando é que é mais efectiva a utilização de uma ou de outra.
3. Especifique com anotações JML a função `sndmax` que, recebendo um *array* de inteiros:
   - se este tiver dois ou mais elementos, retorna o segundo maior elemento do array;
   - se contiver um único elemento, retorna esse elemento;
   - noutro caso lança a excepção `NoElem`.
4. A actividade de teste é um dos recursos importantes na produção de software fiável.
   (a) Indique quais são, no seu entender, as vantagens de integrar o *unit-testing* com a utilização do JML.
   (b) Baseando-se na sua experiência, aponte as principais vantagens e incovenientes de se utilizar o `jml-unit` para concretizar essa integração. Sugira, se for caso disso, possíveis alternativas.
5. Considere a seguinte função func que recebe três booleanos, `A`, `B` e `C`:

```
1: long func(bool A, bool B, bool C)  {
2:        if (A) return B;
3:        if (B \/ ~C)  {
4:                // do something
5:                return C;
6:        }
7:        if (~B) {
8:                // do something
9:                return ~C;
10:        }
11: }
```

   (a) Defina um conjunto mínimo de casos de teste para cobrir todas as instruções, condições e decisões. Justifique a cobertura.
   (b) Os casos de teste da alínea anterior garantem cobertura modificada de condições e decisões (MC/DC)? Justifique.

```
sig Aluno {}

sig Grupo {
  membros : some Aluno
}

abstract sig UCE {}
one sig MFES, CSSI, SD, EA, ... extends UCE {}

sig Nota {}

sig MI {
  inscritos : UCE -> Aluno,
  grupos : UCE -> Grupo,
  notas : UCE -> Aluno -> Nota
}

pred A [m : MI] {
  all a : Aluno | #(m.inscritos.a) < 3
}

pred B [m : MI] {
  all u : UCE | m.notas[u].Nota in m.inscritos[u]
  all u : UCE | m.grupos[u].membros in m.inscritos[u]
}

pred C [m : MI] {
  all u : UCE, a : m.inscritos[u] | lone m.notas[u][a]
}

pred D [m : MI] {
  all u : UCE, a : m.inscritos[u] | one (m.grupos[u] & membros.a)
}

pred NotasGrupos [m : MI] {
  ...
}

pred Inv [m : MI] {
  A[m] && B[m] && C[m] && D[m] && NotasGrupos[m]
}

run Inv for 3 but 1 MI

pred LancaNota [m,m' : MI, u : UCE, a : Aluno, n : Nota] {
  // PRE
  a in m.inscritos[u]
  // POS
  m'.inscritos = m.inscritos
  m'.grupos = m.grupos
  m'.notas = m.notas + (u -> a -> n)
}
```