Computação Fiável

Departamento de Informática
Universidade da Beira Interior

Ano lectivo 2012/2013

Esta página no formato pdf

1 Novidades

- Como colocar uma dúvida ao regente da Unidade Curricular?
  1. Comparecer nas aulas e colocá-la directamente ao regente
  2. Comparecer no horário de atendimento do regente e colocá-la directamente

3. enviar um email ao regente (desousaUUU@UUUdi.ubi.pt, (retire os UUU)) com o assunto "CF: XXXX"em que XXX é o título da dúvida em questão. Qualquer outro formato no assunto arrisca condenar o email ao esquecimento.

- Inscrição em turmas práticas: via site dos serviços académicos.
- Os alunos com estatuto de *trabalhador estudante* são convidados a dirigir-se ao regente para discutir dos critérios de avaliação.

Conteúdo

2 Docentes

Simão Melo de Sousa (regente) - Gabinete 3.17 - Laboratório 6.10 - Bloco VI

3 Objectivos

- Perceber e dominar o ciclo de vida do desenvolvimento de sistemas informáticos baseado em Métodos Formais.
- Conhecer os métodos formais existentes, saber quando devem ser aplicados e quais são os mais adequados em cada caso.
- Aplicar os Métodos Formais de especificação e verificação no desenvolvimento de sistemas informáticos.

4 Competências por adquirir

Os alunos deverão

- saber construir e especificar formalmente um sistema informático, comprovar a correcção desta última e
- estar preparados para abordar as fases de prototipagem rápida e produção de implementações comprovadamente fiáveis

5 Programa

- Introdução: problemática, contexto, história e lugar dos Métodos Formais na Engenharia Informática e na Eng. de Software.
- Especificar, Modelar e Analisar SIs: Especificação formal, máquina abstracta de estados, lógica de Hoare. Semântica operacional, semântica denotacional
- Especificar e Demonstrar Propriedades de SIs: verificação de modelo, sistemas de prova automática, sistemas de ajuda a prova.
- Especificar e Derivar Implementações: extracção de programas, refinamento

- Especificar e Transformar: interpretação abstracta.

6 Critérios de Avaliação

A avaliação avaliará a aprendizagem teóricas e prática dos conceitos introduzidos. Como tal, esta será constituida por **provas escrita** e por **resoluções de exercícios práticos**.

Fraudes A equipa docente gostaria de realçar que qualquer tipo de fraude em qualquer dos itens desta disciplina implica a reprovação automática do aluno faltoso, podendo ainda vir a ser alvo de processo disciplinar. Listamos a seguir as diferentes componentes da avaliação.

6.1 Componente Ensino/Aprendizagem Prática

- A avaliação da Componente Ensino/Aprendizagem Prática mede em termos práticos a aquisição dos conceitos expostos. Como tal é baseada na realização de **três exercícios** entregue à equipa docente. Alguns exercícios poderão ter uma parte opcional que, se realizada, poderá valorizar a nota do exercício.

- A *Nota da Componente Prática* (NCP, 20 valores) é a média das notas dos exercícios.

6.2 Componente Ensino/Aprendizagem Teórica

- Esta componente mede em termos teóricos a aquisição dos conceitos expostos. Como tal é baseada na realização de **provas escritas** agendadas no fim de cada capítulo exposto.

- Da média destas provas resulta a *Nota da Componente Teórica* (NCT, 20 valores).

6.3 Admissão e Avaliação por Exame

- Mínimos: são seguidas as disposições aprovadas pelo Conselho Científico da Universidade aplicadas individualmente as duas componentes

de avaliação. É admitido a exame quem tiver ambas as notas NCP e NCT acima dos mínimos (i.e. $NCP \geq 6$ e $NCT \geq 6$).

- A Nota da Prova Escrita do exame substituirá a Nota da Componente Teórica. No final, para obter aprovação a disciplina, esta nota terá de ser maior do que 6. Ou seja:
- A nota final (NF) é calculada pela fórmula:

$$NF = if\ (NCT \geq 6) \wedge\ then\ \ \frac{NCT + NCP}{2}\ else\ \ Reprovado$$

7 Datas Importantes

- Entrega do enunciado do primeiro exercício: inicio de Outubro
- Entrega do primeiro exercício: Primeira semana de Novembro
- Entrega do enunciado do segundo exercício: Primeira semana de Novembro
- Entrega do segundo exercício: segunda semana de Dezembro
- Entrega do enunciado do terceiro exercício: segunda semana de Dezembro
- Entrega do terceiro exercício: última semana de aulas
- Exame Época 1 : conferir nos SA.
- Exame Época 2 : conferir nos SA.
- Exame Época Especial : conferir nos SA.

8 Software

Proof Assistants : COQ

Design by contract - Deductive Program Verification : Principalmente why3, mas também Frama-C

Model checking: Uppaal
E se houver tempo
Atelier B

9 Material Pedagógico e Referências Bibliográficas

- Apontamentos apresentados e disponibilizados nas aulas.
  - Slides do Capítulo 1 do livro; aqui.
  - Slides do Capítulo 2 do livro : aqui.
    em português, e em conjunto com o Capítulo 1: aqui.
  - Slides do Capítulo 3 do livro : aqui e aqui.
  - Slides do Capítulo "Definições indutivas"aqui.
  - Slides do Capítulo COQ: aqui. aqui e aqui.
  - Slides do Capítulo Verificação de Modelos - UPPAAL: aqui.
  - Slides do Capítulo Lógica de Hoare: aqui (principal) et aqui(segundário).
  - Exercícios:
    Ficha DN-COQ
    Ficha indução
    Ficha COQ
    Exercício COQ por resolver (variante do exercício 13 da Ficha DN-COQ)
    Ficha Lógica de Hoare/why3
    Ficha UPPAAL
    Arquivo com rascunhos de resolução dos exercícios uppaal
    Arquivo com enunciados de provas dos anos anteriores
- Livro de Apoio [2] (web-site : aqui)
- Referências principais [16, 6, 14, 20, 21, 22, 12, 15]
- Referências online segundárias (why3):

Proofs of Programs at the Master Parisien de Recherche en Informatique

Deductive Program Verification with Why3 (Tallinn, Estonia, 2013)

- Referências online segundárias (COQ) :

  Tutorial de COQ online - nível básico

  Tutorial de COQ - nível básico/intermédio (Coq in a Hurry)

  Outro Tutorial de COQ - nível básico/intermédio

  Tutorial de COQ - nível intermédio (Tutorial on Recursive Types in Coq)

  Certified Programming with Dependent Types, Adam Chlipala

  Software Foundations by Benjamin C. Pierce, Chris Casinghino, Michael Greenberg, Vilhelm Sjöberg and Brent Yorgey

  uma formação COQ

  Alguns apontadores pedagógicos sobre COQ

- Referências segundárias [3, 4, 25, 13, 24, 11, 7, 8, 9, 18, 19, 5, 10, 26, 1, 23, 17]

10 Horário

Tipo de aula	**Horário**	**Sala**
Teórica	Sexta-Feira das 14h00 às 16h00	6.13
Prática	Sexta-Feira das 16h00 às 18h00	6.13

11 Atendimento

Horário
Sexta das 11h00 às 13h00

ou por mail (medida anti spam, retire os UUU): desousaUUU@UUUdi.ubi.pt.

Referências

[1] J.-R. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[2] J.B. Almeida, M.J. Frade, J.S. Pinto, and S. Melo de Sousa. *Rigorous Software Development, An Introduction to Program Verification*, volume 103 of *Undergraduate Topics in Computer Science*. Springer-Verlag, first edition, 307 p. 52 illus. edition, 2011.

[3] H. P. Barendregt. *The Lambda Calculus, its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, revised edition, 1984.

[4] H.P. Barendregt. Lambda calculi with types. In S. Abramsky, Dov M. Gabbay, and T.S.E Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–310. Oxford University Press, New York, 1992.

[5] Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, and Philippe Schnoebelen. *Systems and Software Verification. Model-Checking Techniques and Tools*. Springer, 2001.

[6] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Serie. Springer Verlag, 2004. `http://www-sop.inria.fr/lemme/Yves.Bertot/coqart.html`

[7] D. Bjorner. *Software Engineering 1 : Abstraction And Modelling*. Springer Verlag, 2005.

[8] D. Bjorner. *Software Engineering 2: Specification Of Systems And Languages*. Springer Verlag, 2005.

[9] D. Bjorner. *Software Engineering 3: Domains, Requirements, And Software Design*. Springer Verlag, 2005.

[10] E.M. Clarke, O. Grumberg, and D Peled. *Model Checking*. MIT Press, 2000.

[11] J. Cooke. *Constructing correct software*. Formal approaches to computing and information technology. Springer Verlag, 2005.

[12] E. Gimenez. A tutorial on recursive types in Coq. `http://coq.inria.fr/doc-eng.html`

[13] Chris Hankin. *Lambda Calculi: A Guide for Computer Scientists*, volume 3 of *Graduate Texts in Computer Science*. Clarendon Press, Oxford, 1994.

[14] K. Lano and H. Haugthon. *Specification in B: An Introduction using the B Toolkit*. Imperial College Press, 1996.

[15] David Makinson. *Sets, Logic and Maths for Computing*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[16] J-F. Monin. *Understanding Formal Methods*. Springer Verlag, 2002. Translation editor M. Hinchey.

[17] H. R. Nielson, F. Nielson, and C. L. Hankin. *Principles of Program Analysis*. Springer-Verlag, 1999.

[18] B. C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

[19] B. C. Pierce, editor. *Advanced Topics in Types and Programming Languages*. MIT Press, 2005.

[20] S. Schneider. *The B-Method: An Introduction*. Cornerstones of Computing series. Palgrave, 2001.

[21] Coq Development Team. Reference manual of the Coq theorem prover. `http://coq.inria.fr/doc-eng.html`

[22] Coq Development Team. A tutorial of the Coq theorem prover. `http://coq.inria.fr/doc-eng.html`

[23] R. D. Tennent. *Specifying Software. A Hands-On Introduction*. Cambridge University Press, 2002.

[24] A. S. Troelstra and H. Schwichtenberg. *Basic proof theory*. Cambridge University Press, New York, NY, USA, 1996.

[25] D. van Dalen. *Logic and Structure*. Springer Verlag, Berlin, Germany, 1983.

[26] G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. Foundations of Computing series. MIT Press, Cambridge, Massachusetts, February 1993.

Enviar comentários e dúvidas para (retire os UUU) : desousaUUU@UUUdi.ubi.pt