

# Computação Fiável (cod. 11480)

Departamento de Informática  
Universidade da Beira Interior

Ano lectivo 2015/2016



Figura 1: as seen on <http://dilbert.com/2011-02-03/>

Esta página no formato pdf

## 1 Novidades

- Primeira versão da página. Encontrará aqui as novidades associadas à disciplina de **Computação Fiável**. A sua consulta regular é necessária ao bom funcionamento da Unidade Curricular.
- 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

<b>1</b>	<b>Novidades</b>	<b>1</b>
<b>2</b>	<b>Docentes</b>	<b>3</b>
<b>3</b>	<b>Objectivos</b>	<b>3</b>
<b>4</b>	<b>Competências por adquirir</b>	<b>4</b>
<b>5</b>	<b>Programa</b>	<b>4</b>
<b>6</b>	<b>Material de Apoio, Pedagógico e Referências Bibliográficas</b>	<b>7</b>
<b>7</b>	<b>Critérios de Avaliação</b>	<b>9</b>
7.1	Componente Ensino/Aprendizagem Prática . . . . .	9
7.2	Componente Ensino/Aprendizagem Teórica . . . . .	9
7.3	Admissão e Avaliação por Exame . . . . .	10
<b>8</b>	<b>Datas Importantes</b>	<b>10</b>
<b>9</b>	<b>Horário</b>	<b>10</b>
<b>10</b>	<b>Atendimento</b>	<b>10</b>

## 2 Docentes

Simão Melo de Sousa (regente) - Gabinete 3.17 - Laboratório Release/QuiVVer (6.25) - 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.

## Contexto da Aprendizagem

Esta UC visa instruir os seus alunos sobre os conceitos, técnicas, ferramentas e aplicações destas à construção de software fiável e seguro, de programas comprovadamente correctos. Uma introdução a cada família de técnicas é dada mas nem todas são exploradas com todo o detalhe. A abordagem aqui explorada em profundidade introduz as famílias de técnicas que permitam obter um perfil comportamental expressivo dos programas por análise, raciocínio e demonstração. Por natureza este escrutínio *não é automático*. No entanto, e este é o foco desta UC, este consegue ser suportado e sistematizado computacionalmente, permite uma expressividade e uma granulosidade impár quando comparado com outros métodos.

Neste sentido esta UC é complementar da UC de *Desenho de Linguagens de Programação e de Compiladores -DLPC (link)*. Ambas estudam a essência das linguagens de programação, dos seus programas, introduzem técnicas relacionadas.

Na UC DLPC procura-se construir linguagens de programação (e respectivos compiladores) para que esses permitam a expressão ou síntese de programas expressivos, eficiente e bem comportados. Os métodos estudados permitam *calcular* estes programas com as garantias de eficiência e de correcção (por exemplo, relativamente ao código fonte).

Assim a *safety*, correcção, eficiência, expressividade análise comportamental obtidas por desenho, por cálculo, em tempo de compilação, *automaticamente*.

## Agradecimentos

Esta UC toma apoio sobre o texto principal [2] com co-autoria de Maria João Frade, José Bacelar Almeida, Jorge Sousa Pinto e do próprio regente. Como tal, muito do material pedagógico utilizado foi preparados pelos co-autores, em particular na cobertura feita nesta UC, pela Maria João Frade. O regente agradece-os assim por tê-lo autorizado à reutilização destes slides.

## Contexto e parcerias industriais/académicas

Esta UC contou na sua organização e leccionação com vários intervenientes industriais e académicos que citamos aqui (figura 2) como indicador da relevância e abertura desta UC ao meio tecnológico no qual evolui e o seu compromisso firme e reconhecido em potenciar os seus alunos junto desta.

Estes intervenientes influenciaram, participaram na definição da componente prática, propuseram extensões a esta componente na forma de estágios, teses de mestrado, ou contratações. Estas parcerias justificaram a dinâmica escolhida e impressa na exposição teorica e prática da matéria, colaboraram em termos de investigação com a equipa docente em temáticas abordadas nesta UC o que resultou numa exposição que se pretendeu mais esclarecida.

## 4 Competências por adquirir

Os alunos deverão:

- saber construir e especificar formalmente um sistema informático, comprovar a correcção desta última;
- saber planejar e aplicar as fases de prototipagem rápida e produção de implementações comprovadamente fiáveis.

## 5 Programa

Os tópicos seguintes serão abordados.



Figura 2: parceiros e intervenientes (ordem alfabética)

- 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: transformações de especificações/programas, interpretação abstracta.

As aulas são organizadas da seguinte forma:

- Aula 1 - Métodos formais em engenharia de software
- Aula 2 - Lógica, teoria das ordens e álgebra universal - revisões
- Aula 3 - Semântica operacional para a verificação
- Aula 4 - Semântica denotacional, de traços
- Aula 5 - Semântica axiomática, Lógica de Hoare, WPC e VCGEN
- Aula 6 - Why3: Bases da verificação deductiva em Why3, variáveis *ghost*, funções, mapas, vectores
- Aula 7 - Why3: Estruturas de dados, funções *Lemma*, excepções
- Aula 8 - Why3: *Aliasing* e programas com apontadores
- Aula 9 - Why3: Arte e engenho da verificação deductiva
- Aula 10 - Lambda Cálculo, Teoria dos Tipos e Isomorfismo de Curry-Howard
- Aula 11 - COQ - parte 1
- Aula 12 - COQ - parte 2

- Aula 13 - COQ - parte 3
- Aula 14 - Lógica temporal (temporizada), verificação de modelos temporizados,
- Aula 15 - Verificação de modelos com UPPAAL

## 6 Material de Apoio, Pedagógico e Referências Bibliográficas

- Apontamentos apresentados (e disponibilizados) nas aulas.

Encontrará aqui ([link](#)) um repositório de apontadores avulsos relacionado com o funcionamento da UC em edições anteriores.

### Ferramentas

A listagem seguinte apresenta o software usado nas aulas

- A linguagem de programação OCaml ([link](#))
- Why3 - Where Programs Meet Provers ([link](#))
- The COQ Proof Assistant ([link](#))
- UPPAAL - Timed Temporal Model Checking ([link](#))
- (opcional - se houver tempo) Atelier B ([link](#))

### UCs semelhantes

Apresenta-se aqui apontadores para 4 unidades curriculares que serviram de modelo a UC aqui definida (e cuja exploração se recomenda):

- Verificação Formal de Software
- Proof of Programs
- Semantics and applications to verification
- Proof assistants

## Recursos Suplementares

- Why3 :
  - Proofs of Programs at the Master Parisien de Recherche en Informatique
  - Deductive Program Verification with Why3 (Tallinn, Estonia, 2013)
- COQ:
  - uma formação COQ
  - Alguns apontadores pedagógicos sobre 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
  - COQ + Programming Language Theory: Software Foundations by Benjamin C. Pierce, Chris Casinghino, Michael Greenberg, Wilhelm Sjöberg and Brent Yorgey

## Referências Principais

- Livro de Apoio [2] (web-site : aqui)
- As referências seguintes são complementares ou abordam alguns dos tópicos de forma mais pormenorizada. São assim de leitura secundária para exposição da matéria nesta aula (mas aconselhada para uma aprendizagem mais aprofundada).
  - Lógica, Indução, Álgebra Universal:[1, 4, 17, 26, 10, 13, 25]
  - Model Checking: [8, 12]
  - Lambda Calculo, Teoria de Tipos, Reescrita:[24, 5, 6, 7, 14, 15]



- COQ:[9, 23, 11]
- Fundamentos das Linguagens de Programação, Semântica:[27, 19, 21, 22, 23, 20, 18, 3, 16]

## 7 Critérios de Avaliação

A avaliação avaliará a aprendizagem teóricas e prática dos conceitos introduzidos. Como tal, esta será constituída 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.

### 7.1 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.

### 7.2 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).

### 7.3 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:

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

## 8 Datas Importantes

- Exame Época 1 : conferir nos SA.
- Exame Época 2 : conferir nos SA.
- Exame Época Especial : conferir nos SA.

## 9 Horário

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

## 10 Atendimento

Horário
Terça das 11h00 às 13h00

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

## Referências

- [1] Peter Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.7, pages 739–782. North-Holland, Amsterdam, 1977.
- [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] Andrew W. Appel, Robert Dockins, Aquinas Hobor, Lennart Beringer, Josiah Dodds, Gordon Stewart, Sandrine Blazy, and Xavier Leroy. *Program Logics for Certified Compilers*. Cambridge University Press, New York, NY, USA, 2014.
- [4] A. Arnold and I. Guessarian. *Mathematics for Computer Science*. Prentice-Hall, 1996.
- [5] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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>

- [10] S. Burris and H. Sankappanavar. *A Course in Universal Algebra*. Springer Verlag, 1981. <http://www.cs.uu.nl/people/franka/ref>
- [11] Adam Chlipala. *Certified Programming with Dependent Types - A Pragmatic Introduction to the Coq Proof Assistant*. MIT Press, 2013.
- [12] E.M. Clarke, O. Grumberg, and D Peled. *Model Checking*. MIT Press, 2000.
- [13] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order: Second Edition*. Cambridge University Press, 2002.
- [14] J-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1988.
- [15] Chris Hankin. *Lambda Calculi: A Guide for Computer Scientists*, volume 3 of *Graduate Texts in Computer Science*. Clarendon Press, Oxford, 1994.
- [16] Robert Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, New York, NY, USA, 2012.
- [17] David Makinson. *Sets, Logic and Maths for Computing*. Springer Publishing Company, Incorporated, 1 edition, 2008.
- [18] John C. Mitchell. *Concepts in programming languages*. Cambridge University Press, 2003.
- [19] H. R. Nielson, F. Nielson, and C. L. Hankin. *Principles of Program Analysis*. Springer-Verlag, 1999.
- [20] Hanne Riis Nielson and Flemming Nielson. *Semantics with Applications: An Appetizer (Undergraduate Topics in Computer Science)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [21] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, 2002.
- [22] Benjamin C. Pierce, editor. *Advanced Topics in Types and Programming Languages*. MIT Press, 2005.

- [23] Benjamin C. Pierce, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hrițcu, Vilhelm Sjöberg, and Brent Yorgey. *Software Foundations*. Electronic textbook, 2015.
- [24] I. Poernomo, J. Crossley, and M. Wirsing. *Adapting Proofs-as-Programs, The Curry–Howard Protocol*. Monographs in Computer Science. Springer Verlag, 2005.
- [25] A. S. Troelstra and H. Schwichtenberg. *Basic proof theory*. Cambridge University Press, New York, NY, USA, 1996.
- [26] D. van Dalen. *Logic and Structure*. Springer Verlag, Berlin, Germany, 1983.
- [27] 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](mailto:desousaUUU@UUUdi.ubi.pt)