Métodos de Desenvolvimento de Software (MDS) 2016/2017

Vasco Amaral
vma@fct.unl.pt

Aula1

Introdução

Agradecimentos

- Regentes de MDS de anos lectivos anteriores
- Vasco Amaral
- Miguel Goulão
- Ana Moreira
- João Araújo

- **Nota prévia:** vários dos materiais a usar em MDS nomeadamente a maioria dos slides a usar nas aulas teóricas, são versões revistas e adaptadas de materiais produzidos pelos anteriores regentes, que gentilmente os cederam e de material original.

Lecturers

Vasco Amaral

João Araújo

Ankica Barisic

Schedule

	2ª	3ª	4ª	5ª	6ª	Sábado
8:00 – 9:00						
9:00 – 10:00						
10:00 – 11:00		**MDS** t.1 Ed 2: 128/Ed.II	**MDS** t.1 Ed 2: 128/Ed.II			
11:00 – 12:00		**MDS** p.3 Ed 2: Lab 121/Ed.II	**MDS** p.4 Ed 2: Lab 122/Ed.II			
12:00 – 13:00						
13:00 – 14:00						
14:00 – 15:00	**MDS** p.1 Ed 2: Lab 122/Ed.II	**MDS** p.7 Ed 2: Lab 112/Ed.II	**MDS** p.5 Ed 2: Lab 121/Ed.II			
15:00 – 16:00						
16:00 – 17:00	**MDS** p.2 Ed 2: Lab 122/Ed.II	**MDS** p.6 Ed 2: Lab 121/Ed.II				
17:00 – 18:00						
18:00 – 19:00						
19:00 – 20:00						
20:00 – 21:00						
21:00 – 22:00						
22:00 – 23:00						
23:00 – 24:00						
	2ª	3ª	4ª	5ª	6ª	Sábado

Avaliação

100%= NPx0,3+NTx0,7

- NP Nota de um Projecto obrigatório, com nota >= 10 valores (Frequência), vale 30%
- NT Nota de dois testes sem consulta, vale 35%+35%= 70%

.Data a confirmar com CCMIEI do 1º Teste – 9ª semana

.Data a confirmar com CCMIEI do 2º Teste – 15ª semana

Avaliação - Avaliação “Assíduidade”

100%= NPx0,3+NTx0,6+NQx0,1

- NQ Nota dos 6 melhores em 7 “Quizzes”
  - Realizados nas aulas práticas
  - perguntas de avaliações de anos anteriores e outro material de estudo identificado pelo docente nas aulas

A componente NQ só é válida se houver registo de presenças > 20 nas aulas teóricas (não contam as semanas de avaliações).

Avaliação: Projeto

- Trabalho em grupo
- Terá a forma de um projeto,**elaborado fora da sala de aula**, em **grupos de 3 alunos** e com recurso a uma ferramenta
- O trabalho será enriquecido incrementalmente **ao longo de todo o semestre**
-
- Discussão: Semana 16 de Dezembro (durante o horários dos laboratórios e teóricas a agendar oportunamente)
-
- **Não se aceitam trabalhos fora dos prazos estabelecidos**

Avaliação - Frequências anteriores

Serão consideradas frequências até duas edições anteriores da cadeira.

Os alunos poderão fazer a componente de testes ou exame de recurso.

Neste caso:

NF = NT

Melhoria

NF = Nota de Exame de Recurso

Trabalhadores Estudantes

Devem de comunicar o seu estatuto (e que pretendem fazer uso deste)

Têm de realizar provas (testes, projectos e exames)

Atenção

–A não realização de qualquer teste (excepto Quizzes) implica nota 0 (zero) nesse elemento de avaliação.
–Os “quizzes”(mini-testes) e testes são sem consulta.
–Qualquer tipo de fraude detectada no teste ou exame implica a impossibilidade de fazer a cadeira no ano letivo corrente, mesmo em época de recurso ou época especial, perdendo-se a frequência eventualmente adquirida nesta ou noutras edições da cadeira.

Avaliação:Melhorias

- As melhorias de nota apenas se realizam por exame,sendo a nota final igual à nota do exame.

Avaliação

- Melhoria de nota
- Os alunos que desejarem fazer melhoria de nota devem:
  - inscrever-se nos serviços académicos
  - fazer o exame de recurso
  - a nota final é a nota do exame de recurso

Página da cadeira

- Página da cadeira no Moodle
- URL:
  - https://moodle.fct.unl.pt/course/view.php?id=4450
- Inscreva-se
  - Toda a informação sobre a cadeira estará aí disponível

Bibliografia [1/2]

- **SoftwareEngineering, Ian Sommerville,** Addison-Wesley; 8th, 9th or 10th edition

Bibliografia [2/2]

- **UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Ed.), Jim Arlow, Ila Neustadt, Addison-Wesley, ISBN 0-321-32127-8, 2005.**
- The Unified Modeling Language Reference Manual, G. Booch, J. Rumbaugh & I. Jacobson, Addison-Wesley, 2004.
- http://www.uml.org/
- The Object Constraint Language: Getting Your Models Ready for MDA (2nd Ed.), Anneke Kleppe, Jos Warmer, Addison-Wesley, ISBN 0321179366, 2003.

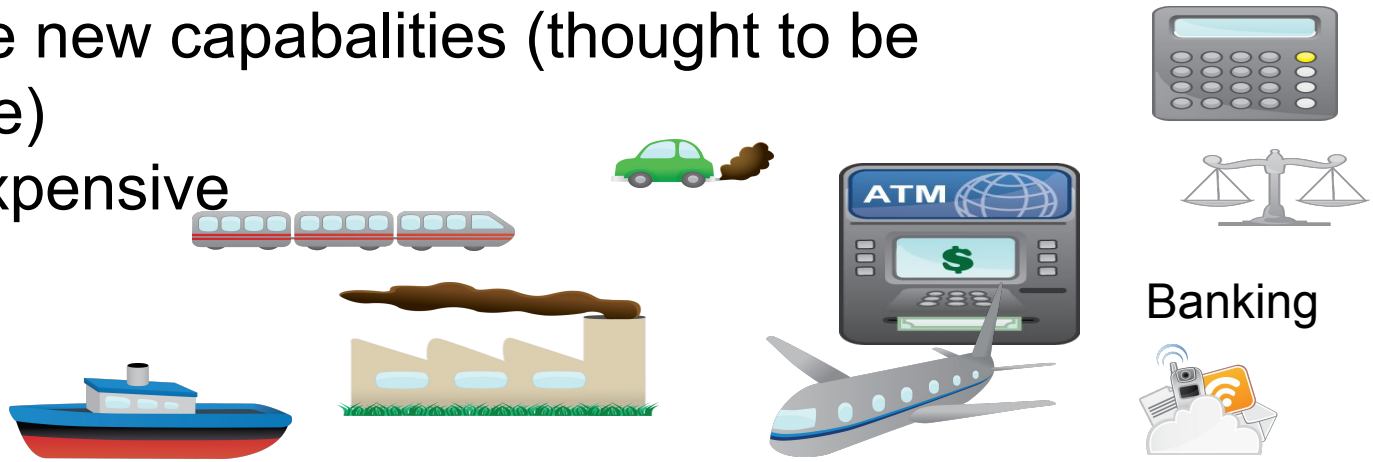
“And now something different...”

Motivation: why are we here?

Read Chapter 1 of the Book “Software Engineering”,10th edition, by Jan Sommerville

Modern society can not cope without software systems

- Immersion of Software in all areas of human activity
  - Economy, Banking, Health, Aeronautics, Space Industtry, R&D, etc…
- The great majority of electrical systems rely on computers and control software
- Systems are getting larger and more complex
  - need to be delivered more quickly
  - must have new capabalities (thought to be impossible)
  - be less expensive
  - reliable



Politically under focus

According to USA Presidential Information Technology Advisory Committee (PITAC) Report 1999:

“Software is the new physical infraestructure of the information age”

Politically under focus

European Software Strategy a NESSI Position paper 2008

From a societal point of view:

- flexibility, intelligence and security to all complex systems and equipment that support and control the key infrastructures of our society

From a economic point of view:

- Increases productivity and competitiveness in all business activities

From a technology point of view

- Traditional slip hardware/software will disappear with blurred frontier between the computer, the network and the application

European Software Strategy a NESSI Position paper 2008

Software is no longer just an IT issue, it is a key asset, opportunity of growth factor for all actors of the economy.

Software drives the rise of global production systems across many sectors, software itself is emerging as the first global production system

The Crisis

General Motivation…let us blame the (Software) crisis

Software Crisis

The term "**software crisis**" was coined by F. L. Bauer at the first NATO Software Engineering Conference in 1968

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful!

To put it quite bluntly: as long as there were no machines, programming was no problem at all;

when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

– Edsger Dijkstra, The Humble Programmer (EWD340), Communications of the ACM

"No Silver Bullet – Essence and Accidents of Software Engineering", Fred Brooks, 1986

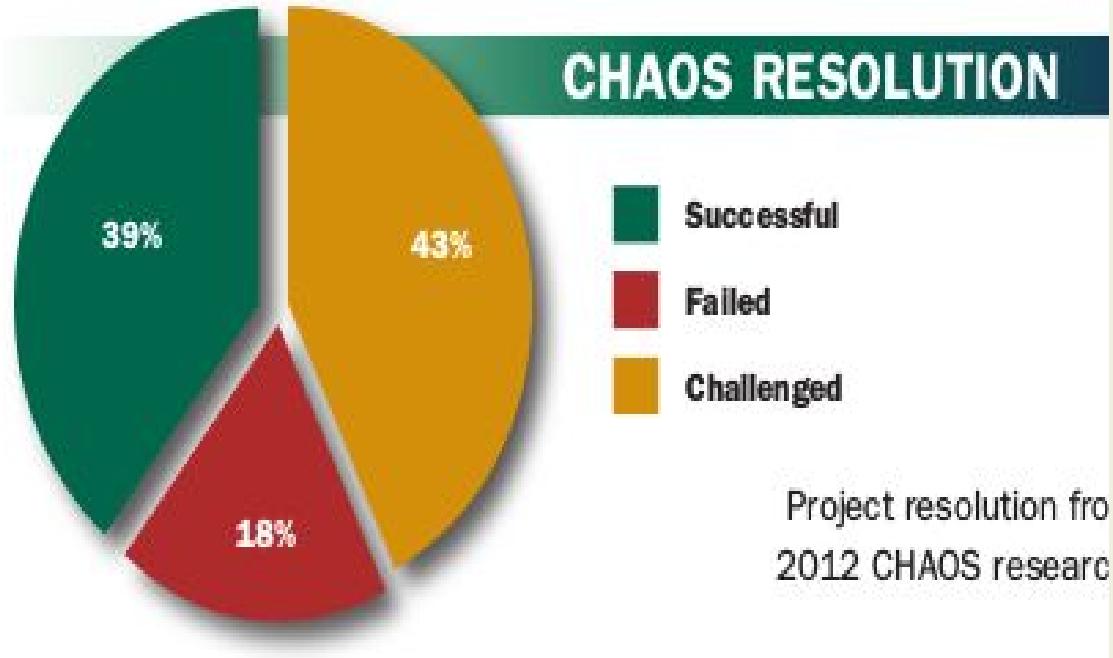
Accidental Complexity
non essential to the problem solved.

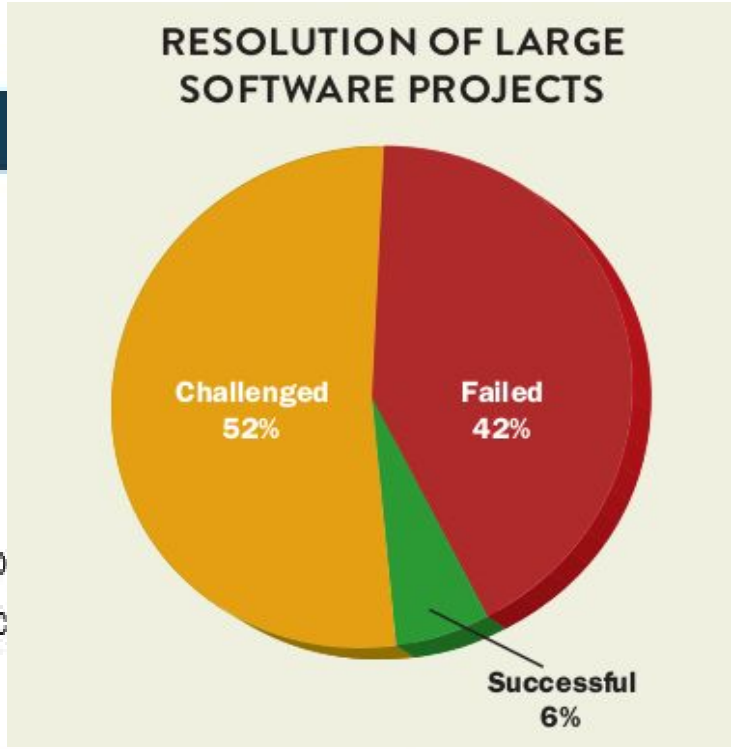
Essential complexity
is inherent and unavoidable

Silver Bullets
High-Level Languages Advancments
Expert Systems
Automatic Programming
Graphical Programming
Program Verification
Environment and tools
Workstations

Silver Bullets for the Essence
Buy versus build
Requirements refinement and rapid prototyping
Incremental development
Great Designers

CHAOS Manifesto – Standish group, 2013





RESOLUTION

	2004	2006	2008	2010	2012
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

Project resolution results from CHAOS research for years 2004 to 2012.

www.sol.pt/noticia/114673

Sociedade

Falha no sistema informático Citius interrompe mega julgamento

Partilhar no Facebook | Partilhar no Twitter | Partilhar no Google+

08/09/2014 15:48:23 850 Visitas

Foto: Shutterstock

Why do projects fail?

- Size of the projects (when compared to previous ones)
- Development cost explodes due to lack of productivity
- Lack of efficiency of the communication channels on big teams
- Key personal has left
- Fail to understand requirements
- Projects are delivered with lack of quality
- Introduction of recent technology badly understood
- Other

Kinds of Software

Stand-alone
Interactive transaction-based applications
Embedded Control Systems
Batch processing systems
Entertainment Systems
Systems for modeling and simulation
Data Collection and analysis systems
Systems of Systems

Desenvolver Software(Grady Booch)

Professional Software Development

Software Engineering is intended to support professional software development rather than individual programming

Software is more than just code

- Software !=Code
- Code
- Specification
- Configuration
- Manuals (system and/or user)
  - Libraries
  - Support websites

Not just for yourself (amateur) but for other people to use and engineers to change or maintain

Software engineers

Concerned with developing software trustworthy products that cope with increasing diversity, short time-to-market needs:

- Generic - standalone developed by an organization and sold in the open market
- Customized - commissioned and developed for a particular customer.

Software engineers - Moral and Ethics

Your job involves wider responsibilities that go beyond the scope of technical skills.

You must behave morally and ethically responsible way to be respected as professional engineer.

Software engineers - Moral and Ethics

IEEE/AACM Code of Ethics
http://www.sqa.org.uk/e-learning/ProfIssues03CD/page_04.htm

Software Engineering

Engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance in a cost-effective way.

Software Engineering Methods

- Structured approach for developing high quality software that is efficient in a cost effective manner.

Essential attributes of good Software

- Maintainability
- Security, reliability and safety
- Efficiency
- Usability

Challenges for Software Engineering

- Heterogeneity
- shorter time-to-market
- Product trustworthiness
- Deal with legacy systems

Software Engineering “vs.” Computer Science

- Computer Science focuses on theory and fundamentals
- Software Engineering is concerned with practicalities of developing and delivering useful software