Resilient Computing Curriculum Draft – ReSIST NoE Deliverable D16

L. Simoncini, J-C. Laprie, K. Kanoun, J-C. Fabre, H. Waeselynck, I. Majzik, A. Pataricza, R. Bloomfield, L. Strigini, N. Suri, M. Dacier, G. Urvoy- Keller, C. Lac, C. Cachin, M. Raynal, P. Palanque, M. Correia, P. Verissimo, C. Bernardeschi, A. Bondavalli, F. Giandomenico, G. Lami, H. Pfeifer, U. Voges, R. Jimenez-Peris, W. Ahrendt, J. Karlsson

DI–FCUL TR–07–27

November 2007

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

Technical reports are available at `http://www.di.fc.ul.pt/tech-reports`. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

ReSIST: Resilience for Survivability in IST

A European Network of Excellence

Contract Number: 026764

Deliverable D16: Resilient computing curriculum draft

Report Preparation Date: June 2007
Classification: Public

Contract Start Date: 1st January 2006
Contract Duration: 36 months
Project Co-ordinator: LAAS-CNRS
Partners: Budapest University of Technology and Economics
City University, London
Technische Universität Darmstadt
Deep Blue Srl
Institut Eurécom
France Telecom Recherche et Développement
IBM Research GmbH
Université de Rennes 1 – IRISA
Université de Toulouse III – IRIT
Vytautas Magnus University, Kaunas
Fundação da Faculdade de Ciencias da Universidade de Lisboa
University of Newcastle upon Tyne
Università di Pisa
QinetiQ Limited
Università degli studi di Roma "La Sapienza"
Universität Ulm
University of Southampton

Deliverable D16:
Resilient Computing Curriculum Draft

Co-ordinator: Luca Simoncini

Contributors in ReSIST: Jean-Claude Laprie, Karama Kanoun, Jean-Charles Fabre, Helene Waeselynck, Istvan Majzik, Andras Pataricza, Robin Bloomfield, Lorenzo Strigini, Neeraj Suri, Marc Dacier, Guillame Urvoy-Keller, Chidung Lac, Christian Cachin, Michel Raynal, Philippe Palanque, Miguel Correia, Paulo Verissimo, Cinzia Bernardeschi, Andrea Bondavalli, Felicita Di Giandomenico, Giuseppe Lami, Luca Simoncini, Holger Pfeifer

External contributors: Udo Voges, Ricardo Jimenez-Peris, Wolfgang Ahrendt, Johan Karlsson

Comments: ReSIST T&D Committee, ReSIST EB Committee

Contents:

Introduction

The Bologna Declaration, signed on June 19, 1999 [1], and the subsequent documents [2], [3] have identified three big reform areas in higher education:

- Curricular reform: The three cycle system, competence based learning, flexible learning paths, recognition, mobility
- Governance reform: University autonomy, strategic partnerships, quality assurance
- Funding reform: Diversified university income, tuition fees, grants and loans, equity and access, EU funding

and have started the so-called Bologna process that is forecast to be completed in 2010. In particular [1], Universities and other institutions of higher education may:

- Profile their own curricula, in accordance with the emerging post-Bologna environment, in particular through the introduction of bachelor courses in systems where they have not traditionally existed, and **through the creation of master courses** meeting the needs of mobile postgraduate students from around the world;
- **Activate their networks in key areas such as joint curriculum development**, joint ventures overseas or worldwide mobility schemes;
- **Contribute individually and collectively** to the next steps in the process.

In 2003, the Computing Research Association, has identified the following Four Grand Challenges in Trustworthy Computing [5]:

- Challenge 1: Eliminate Epidemic Attacks by 2014
- Challenge 2: Enable Trusted Systems for Important Societal Applications
- Challenge 3: Develop Accurate Risk Analysis for Cybersecurity
- Challenge 4: Secure the Ubiquitous Computing Environments of the Future

identifying very strategic fields of activity necessary to fill gaps that limit the introduction of resilience in many complex computer-based critical applications.

Starting January 2006 and lasting three years, the EU has funded a NoE called ReSIST – Resilience for Survivability in IST [4], that collects 18 partners, among the most well-recognized groups in Europe expert in dependability, security and human factors with the following objectives:

- **Integration** of teams of researchers so that the fundamental topics concerning scalable resilient ubiquitous systems are addressed by *a critical mass* of co-operative, multi-disciplinary research.
- **Identification**, in an international context, of the key *research directions (both technical and socio-technical)* induced on the supporting ubiquitous systems by the requirement for trust and confidence in AmI.
- **Production of significant research results** *(concepts, models, policies, algorithms, mechanisms)* that pave the way for scalable resilient ubiquitous systems.
- **Promotion and propagation of a resilience culture in university curricula** and in engineering best practices.

ReSIST has therefore identified in Work Package 3 -Training and Dissemination an activity towards the preparation of a MSc curriculum in Resilient Computing as properly providing a timely and necessary answer to requirements posed by EU.

This Deliverable presents the first version of the Curriculum in Resilient Computing, limited to the description of the syllabi for the first year (Semesters 1 and 2) and indicates the line and title for the curriculum in the second year (semesters 3 and 4) and propose it to the general discussion for improvements. The curriculum will be updated and completed in successive versions that will take advantage of a large open discussion inside and outside

ReSIST.

1. Activity in ReSIST towards the Curriculum

ReSIST has dedicated, during the first 18 months, a constant and continuous integrated effort towards the identification of the MSc Curriculum in Resilient Computing and towards the identification of the syllabi of the identified courses.
The first step of this activity was dedicated to internal interviews to all members in ReSIST for obtaining a snapshot of what is offered inside ReSIST as courses (of all kinds: university, short, industrial courses) so to have a possibly comprehensive view on the following: i) present expertise inside ReSIST, and ii) how to extend to the present curriculum (as first version) the gathered information. From this activity 54 forms have been collected. They are reported in Appendix A.
A second step, parallel to the first one, was a survey through searching the web, to identify relevant links to Universities and Organizations inside and outside ReSIST that offer Resilience-related courses, to extend the base of knowledge coming from step one. The output of this second step is reported in Appendix B.
The third step of this activity has been the organization in the frame of EDCC-6, Sixth European Dependable Computing Conference, held in Coimbra, Portugal on 18-20 October 2006, of a Panel Session on "Education in Dependable and Resilient Computing – Meeting the Needs of the Information Society", where panellists from Chalmers University, Polytechnic University of Madrid, University of Lisbon and University of Pisa discussed on the topics.
The fourth step was an Open T&D Meeting, held in London on January 31, 2007. At the meeting EWICS-TC7 was invited to coordinate efforts towards a commonly agreed curriculum. In that meeting the skeleton of the present curriculum was outlined, and was decided to call for a Joint ReSIST/EWICS TC-7 Workshop on Teaching Resilient Computing, held in Erlangen on May 2, 2007. The Final Program and attendance list are reported in Appendix C, while the set of presentations are publicly available on the ReSIST web site.
Finally, on May 3, 2007, an Open T&D Committee Meeting was held in Erlangen to consolidate the outputs of the Workshop.

2. Curriculum description

An MSc curriculum in Resilient Computing that covers years 4 and 5 of a University Master Degree is usually composed by a total of 120 ECTS [6], evenly divided in two years (60 ECTS each year). This is not a common characteristic among European Universities, but following the T&D Meeting that was held in London on January 31, 2007, it was agreed to take as working assumption that a MSc curriculum corresponds to 120 ECTS for an amount of lectures + labs hours in the range of 1000 hours. It was also agreed that the curriculum should target to provide knowledge to students so that they could be productive in industrial environments, having anyway a solid basic knowledge of the fundamentals and of the methods, techniques and tools related to resilient computing. As another working agreement, it was decided to work on an organization of the curriculum into 4 semesters, two for each year, each tentatively 30 ECTS worth: the 1st semester on Basics and Fundamentals, the 2nd on Methods, Techniques and Tools, the 3rd on Projects in cooperation with industry on specific application fields, and the 4th for Master's Thesis and Dissertation.

There are three phases in the curriculum. In the first phase (2 semesters, 60 credits), we introduce core knowledge and skills through modules in: system dependability and security; advanced information security; human factors engineering; distributed and fault-tolerant computing; and system validation, ranging from theoretical bases to methods,

techniques and tools.
In the second phase (3rd semester, 30 credits), we emphasize the practice of resilient computing, through modules in high-integrity software development and research skills, followed by a group project on the development and assessment of a real system in specific application domains.
The third phase of the curriculum (4th semester, 30 credits) is a six-month individual system development or research project, undertaken with personal supervision of one senior scientist, or in industry, and will be concluded with the preparation and discussion of a Master Thesis.

2.1 Curriculum Pre-requisites

A student who wants profitably enrol to the MSc Curriculum in Resilient Computing would take advantage from having a basic knowledge in the following fields:

- Discrete Mathematics
- Calculus
- Basic Computer and Network Architectures
- Programming and Data Structures
- Basics of Operating Systems
- Basics of Software Engineering
- Basics of Probability and Statistics

This basic knowledge has to be provided in the first phase of the higher education scheme of the Bologna process.

2.2 Curriculum aims

The aims of the curriculum are:

- To equip students with the skills and knowledge required to develop and assess secure and dependable computer-based systems
- To provide a qualification enhancing employment prospects in resilient computing
- To develop research skills
- To develop and improve key skills in written and oral communication and in teamwork
- To develop and improve skills in using the literature and information technology resources relevant to resilient computing
- To encourage the development of creativity skills
- To develop skills in critical assessment, analysis and storage of information
- To provide a curriculum which meets the requirements of appropriate professional bodies, thus providing a basis for further professional development and lifelong learning
- To address the relevant professional, legal and ethical issues relevant to the development, assessment and maintenance of resilient systems
- To provide an international perspective on developments in computer resilience.

2.3 Knowledge and understanding

A successful student will have gained and be able to demonstrate:

- Understanding of the theory underpinning dependability, security and resilience in computer-bases systems
- Knowledge of major and advanced techniques, methods and tools for assessing information security and system dependability
- Knowledge of the major and advanced fault tolerance techniques, methods and tools applicable in computer system design

- Understanding of the technologies for the design of trustworthy interactive systems, including human error assessment
- Understanding of the computer aided verification techniques relevant to security in distributed systems
- Understanding of the principles underlying high integrity software development using advanced static analysis and formal techniques
- Understanding of major professional, legal and ethical issues associated with work in secure and dependable computing systems
- Understanding of the international character of contemporary developments in security, dependability and resilience.

2.4 Professional skills

A successful student will:

- Be able to propose, conduct and write up an extended research project involving where appropriate, a literature review, problem specification, design, verification, implementation and analysis
- Be able to design, implement and validate new software for secure and dependable applications
- Be able to organize and take part in systematic analyses of existing systems
- Have expertise in the use and applicability of up-to-date software development tools
- Be able to assess the main human factors relevant to secure and dependable system operation
- Be able to apply the leading techniques for security in networks and Internet environments, including cryptography and public key infrastructures
- Be able to apply the major methods for assessing system resilience
- Be able to deploy fault tolerance appropriately in system design.

2.5 Key skills

A successful student will have:

- The ability to communicate orally in English in a professional context
- Written communication skills, including an appreciation of the role of peer review of papers, software, proposals and other research and development products
- Information literacy skills, including the ability to use computer-based resources for research in the professional literature and the capacity to undertake critical reviews
- The ability to work as part of a team, including group-based learning, research and development activity
- Creativity skills: recognizing and responding to opportunities for innovation
- Planning and organization skills.

3. Curriculum organization

The curriculum is structured in 4 semesters (tentatively 30 ECTS each) over two years, as in the following Tables. The number of ECTS is not absolute but indicative of the relative weight among the several courses. Courses worth 6 ECTS are taught in parallel, while there is an ordering between courses worth 3 ECTS:

1st Year

1st semester: Basics and Fundamentals (30 ECTS) Courses: • **Advanced Probability and Statistics** (6 ECTS) • **Cryptology and Information Security** (6 ECTS) • **Logic in Computer Science** (6 ECTS) • **Advanced Graph Theory** (3 ECTS) • **Human Factors, Human and Organisational Behaviour** (3 ECTS) • **Fundamentals of Real-Time Systems** (3 ECTS) • **Fundamentals of Dependability** (3 ECTS)	**2nd semester: Methods, Techniques and Tools (30 ECTS)** Courses: • **Computer Networks Security** (6 ECTS) • **Fault and Intrusion-Tolerant Distributed Systems and Algorithms** (6 ECTS) • **Dependability Evaluation of Computer Systems** (6 ECTS) • **Testing, Verification and Validation** (6 ECTS) • **Usability and User Centred Design for Dependable and Usable Socio-technical Systems** (6 ECTS)

1st Semester scheduling (time flows from left to right)

Advanced Probability and Statistics	
Cryptology and Information Security	
Logic in Computer Science	
Advanced Graph Theory	**Human Factors, Human and Organisational Behaviour**
Fundamentals of Real-Time Systems	**Fundamentals of Dependability**

2nd Semester scheduling (time flows from left to right)

Computer Networks Security
Fault and Intrusion-Tolerant Distributed Systems and Algorithms
Dependability Evaluation of Computer Systems
Testing, Verification and Validation
Usability and User Centred Design for Dependable and Usable Socio-technical Systems

2nd Year

3rd semester: Projects (in cooperation with industry on specific application fields) (30 ECTS)	**4th semester: Master's Thesis and Dissertation (30 ECTS)**
Courses (common to all application tracks) • **Management of Projects** (3 ECTS) • **Fault Tolerant Middleware- based Systems** (3 ECTS) • **Software Reliability Engineering** (3 ECTS) Application track: **Telecom.** Courses (specific for this track): • **Resilience of Protocols and Architecture** (3 ECTS) • **Resilience of Mobile Applications** (3 ECTS) Application track: **Safety critical Systems** Courses (specific for this track): • **Development Process and Standards for Safety critical Applications** (3 ECTS) • **Architectural Issues and Examples of Systems** (3 ECTS) Application track: **e-Business** Courses (specific for this track): • **Resilience of SOA and Web-based Applications** (3 ECTS) • **Damage Tolerance in Large scale Systems** (3 ECTS) Common to all Application tracks: • **Project in cooperation with Industry** (9 ECTS) • **Space for additional Courses** (6 ECTS)	• **Specific Courses and Seminars** (3 ECTS) • **Preparation and Presentation of the Thesis** (27 ECTS)

3rd Semester scheduling (time flows from left to right)

<table>
<tr><td rowspan="3">Management of Projects

Fault Tolerant Middleware- based Systems

Software Reliability Engineering</td><td>Appl. Track: Telecom.:
• Resilience of Protocols and Architecture
• Resilience of Mobile Applications</td></tr>
<tr><td>Appl. Track: Safety critical Systems:
• Development Process and Standards for Safety critical Applications
• Architectural Issues and Examples of Systems</td></tr>
<tr><td>Appl. Track: e-Business:
• Resilience of SOA and Web-based Applications
• Damage Tolerance in Large scale Systems</td></tr>
<tr><td colspan="2">Project in cooperation with Industry
Additional Courses</td></tr>
</table>

4th Semester scheduling (time flows from left to right)

<table>
<tr><td>Specific Courses and Seminars</td></tr>
<tr><td>Preparation and Presentation of the MSc Thesis</td></tr>
</table>

4. Courses syllabi

1ST Year – 1st Semester

4.1 Advanced Probability and Statistics (6 ECTS)

The purpose of this course is to provide a methodical advanced background of probability, stochastic processes, and statistics needed to the students to address modelling and assessment issues in resilient computing.

Contents

- Introduction
- Discrete Random Variables
- Continuous Random Variables
- Expectation
- Conditional Distribution and Expectation
- Bayesian probability and inference
- Stochastic Processes

Suggested readings:

K. S. Trivedi: **Probability and Statistics with Reliability, Queuing, and Computer Science Applications**, Second Edition, John Willey & Sons, 2002

Courseware examples and locations where taught:

Slides from the same author, available at http://www.ee.duke.edu/~kst/

4.2 Cryptology and Information Security (6 ECTS)

The purpose of this course is to give an up-to-date treatment of the principles, techniques, and algorithms of interest in information security and cryptography. Emphasis on fundamental concepts their practical applications.

Contents

- Introduction
- Information security and cryptology
- Access control and security policies
- One-way functions and pseudorandomness
- Hash functions
- Symmetric-key encryption
- Public-key encryption and digital signatures
- Authentication and identification protocols
- Key agreement, certificates, and public-key infrastructures
- Key management and trust management
- Anonymity and Privacy

Suggested readings

A. J. Menezes, P. C. van Oorschot and S. A. Vanstone: **Handbook of Applied Cryptography,** CRC Press, 1996.
N. Smart: **Cryptography, An Introduction**, McGraw-Hill, 2002.

Courseware examples and locations where taught

- ETH Zurich, Cryptography, Ueli Maurer, http://www.crypto.ethz.ch/teaching/lectures/Krypto06/
- KU Leuven, Cryptography and Network Security, Bart Preneel, http://www.kuleuven.ac.be/onderwijs/aanbod2006/syllabi/H0244BE.htm
- Univ. Bristol, Introduction to Cryptography, Nigel Smart, http://www.cs.bris.ac.uk/Teaching/Resources/COMS30124/

4.3 Logic in Computer Science (6 ECTS)

The goal of the course is to present the fundamental notions of logic that are important in computer science.

Contents

- Propositional Logic
  - Natural Deduction
  - Induction
  - Semantics
  - Normal Form
  - SAT solving
- Predicate Logic
  - Natural Deduction
  - Semantics
  - Undecidability
  - Expressivity
- Temporal Logics
  - Branching Time Logic
  - Linear Time Logic
  - Fixed-Point Characterisation
  - Repetition

Suggested readings:

The course is based mainly on the 3 first chapters of:
M. Huth and M. Ryan: **Logic in Computer Science**, Cambridge University Press

As additional reading, one can point to the hypertext-book by
V. Detlovs, K. Podnieks: **Introduction to Mathematical Logic**
http://www.ltn.lv/~podnieks/mlog/ml.htm

Courseware examples and locations where taught:

The course book's webpage, http://www.cs.bham.ac.uk/research/projects/lics/ offers several materials, among them an interactive tutor for each chapter.

Courses based on the Huth&Ryan book are given at many Universities, see:

- http://www.cs.bham.ac.uk/research/projects/lics/adoptions.html

Among them are Chalmers University (teacher Thierry Coquand), the University of Copenhagen (teachers Nils Andersen & Julia Lawall), and the University College London (teacher Jonathan P. Bowen). The web pages of these courses all offer additional material, like slide sets, exercises, and notes elaborating on the topics. See:

- http://www.cs.chalmers.se/Cs/Grundutb/Kurser/logcs/index.html
- http://www.diku.dk/undervisning/2004f/202/
- http://www.cs.ucl.ac.uk/staff/J.Bowen/GS03/

4.4 Human Factors, Human and Organisational Behaviour (3 ECTS)

The purpose of this course is to present human and organisational fundamental concepts and frameworks that influence and determine failures (catastrophic or not) in complex systems and hence the impact on the resilience of the socio-technical system.

Contents

- Cognitive processes for the description and the prediction of human understanding and information processing capabilities
- Human performance cognitive issues (learning, problem-solving)
- Human performance physiological issues (sensation, perception, motor skills, Fitts' law, steering law, …)
- Human error (concepts and classifications)
- Mode confusion and automation surprises
- Effects of trust and automation bias
- Organizational resilience and industrial risk

Suggested readings:

C. W. Johnson: **Failure in Safety-Critical Systems. A Handbook of Accident and Incident Reporting**. Available on-line at: http://www.dcs.gla.ac.uk/johnson/book October 2003. 2003. Glasgow, Scotland, University of Glasgow Press.

J. Reason: **Human Error**. 1990. Cambridge University Press.

J. Reason: **Managing the Risks of Organizational Accidents**, 1997, Aldershot, UK, Ashgate.

C. D. Wickens and J. G. Hollands: **Engineering Psychology and Human Performance**. 3rd edition, 1999, Prentice Hall.

Courseware examples and locations where taught:

- http://sigchi.org/cdg/index.html (gathering a large set of lectures on HCI and Human factors mainly in the US. This set of courses have been gathered and organised by the ACM Special Interest Group on HCI)
- http://liihs.irit.fr/palanque/Ps/MasterHM-IntroHCIPalanque.pdf

4.5 Mathematical Programming (3 ECTS)

The purpose of this advanced course is to present the aspects of graph theory beneficial for the design of resilient systems. "Advanced" means here that the basics of graph theory (such as paths, traversals, trees, maximum flow, planarity, basic graph NP-complete problems, etc.) are already known. So, "advanced" is here the "discovery" of concepts that have been recently (or not) introduced in computer science to model problems related to computability, efficiency, or fault-tolerance).

Graph theory has long become recognized as one of the more useful mathematical subjects for the computer science student to master. The approach which is natural in computer science is the algorithmic one; the interest is not so much in existence proofs or enumeration techniques, as it is in finding efficient algorithms for solving relevant problems, or alternatively showing evidence that no such algorithms exist.

Contents

- Connectivity and traversability: Bounded connectivity, regularity, Overlay networks
- Graph coloring and graph NP-complete problems
- Topological graph theory: Imbeddings, genus and maps
- Analytic graph theory: Random graphs, Ramsey graphs and the probabilistic approach
- Graph measurements: Domination,and tolerance graph
- Small-world networks (on grid and uniform topology, Kleinberg's distribution)

Suggested readings:

S. Even: **Graph Algorithms**, Computer Science Press, 1979, 249 pp.

J.L. Gross and J. Yellen (Eds.): **Handbook of graph theory**, CRC Press, 2003, 1192 pp.

Courseware examples and locations where taught:

- Excerpts of the book by Shimon Even can be downloaded at http://www.wisdom.weizmann.ac.il/~oded/even-alg.html

4.6 Fundamentals of Real-time Systems (3 ECTS)
The purpose of this course is to provide a large overview of fundamentals aspects of real-time system architectures and development. This covers scheduling techniques, scheduling analysis including WCET evaluation, design principles of distributed real-time embedded systems, programming distributed real-time applications. Fault tolerance aspects are also addressed, in particular regarding timing faults handling. Examples of real time executive layers are also presented.

Contents
- Introduction to basic concepts
- Reminder of operating systems basic notions
- Scheduling in real-time systems
- WCET analysis and evaluation
- Design principles of distributed RT applications
- Programming distributed RT systems
- Real-time executives and examples

Suggested readings:

F. Cottet, J. Delacroix, C. Kaiser, Z. Mammeri: **Scheduling in Real-Time Systems**, Wiley Eds, 2002, 266p. ISBN: 0-470-84766-2
G. Buttazzo: **Hard Real-Time Computing Systems**, Second Edition, Series: Real-Time Systems Series, Vol. 23, 2005, XIII, 425 p., ISBN: 978-0-387-23137-2Springer, 2005.
H. Kopetz: **Real-Time Systems: Design Principles for Distributed Embedded Applications**, Series: The Springer International Series in Engineering and Computer Science, Vol. 395, 1997, 356 p., ISBN: 978-0-7923-9894-3
A. Burns and A. J. Wellings: **Real-Time systems and programming languages**, 3rd ed., Addison Wesley, 2001, Pages 610 p., ISBN 0-201-40365-X

Courseware examples and locations where taught:
These are examples of places where parts of this course are taught, giving emphasis on some aspects of real-time systems.
- University of York (UK): scheduling and programming. See: http://www.cs.york.ac.uk/MSc/Modules/rts.html
- Scuola Superiore Santa Anna Pisa (Italy): scheduling and analysis. Courseware (in Italian) through this page: http://feanor.sssup.it/~giorgio/srt.html
- Universidad Politecnica de Madrid (Spain): real-time and applications. See: http://polaris.dit.upm.es/~jpuente/strl/guia.html
- University of Rennes (France): generic course on real-time. Courseware (in French) through this page: http://www.irisa.fr/caps/people/puaut/puaut.html

4.7 Fundamentals of Dependability (3 ECTS)

The purpose of this course is to give a structured introduction to the concepts of Dependability and Security and to the methods and techniques used for dependable and secure design of systems and for scaling to complex resilient systems. The course is composed of 8 chapters. The first chapter is devoted to the basic concepts and gives the corresponding definitions. The second chapter addresses the attributes of dependability. The third chapter addresses the threats to dependability. The four succeeding chapters address the means for dependability: fault prevention, fault removal, fault forecasting and fault tolerance. The last chapter brings together the previous chapters into the development of dependable systems.

Contents

- **Basic concepts and definitions.** Attributes of, means for, threats to dependability. Importance of dependability in current information infrastructures (examples of widely publicised failures, economic impact of failures)
- **Attributes of dependability.** Primary attributes: reliability, availability, safety, integrity, confidentiality, maintainability. Secondary attributes: robustness, survivability, accountability, authenticity, non-repudiability. Relationship between dependability and security, and between dependability and resilience.
- **Threats to dependability.** Faults, errors, failures. Definitions and classification. Failure pathology. Respective importance based on statistics (frequency, consequences) of the various classes (physical, development, interaction; accidental, malicious)
- **Fault prevention.** Hardening and shielding. Security policies and models. Authentication and authorisation.
- **Fault removal.** Basic verification approaches: reviews and inspections, abstract interpretation, theorem proving, model checking, symbolic execution, testing. Principles, possibilities and limitations of the approaches. The cost of verification. Fault removal in operation: corrective maintenance, including patches for vulnerabilities. Statistics on classes of faults uncovered during development and during operation
- **Fault forecasting.** Ordinal (or qualitative) and stochastic (or quantitative) evaluations. Stable vs. evolutive (growing, decreasing) dependability. Measures (or metrics) of dependability. Evaluation approaches: modelling (FMECA, reliability diagrams, fault trees), measurements. The case of safety-critical systems.
- **Fault tolerance.** Basic primitives: error detection (concurrent, pre-emptive), system recovery by error treatment (rollback, rollforward, compensation) and fault treatment (diagnosis, isolation, reconfiguration, reinitialization). Coding for error control; cryptography. Partitioning and confinement. Basic architectures for detection-and-recovery, and masking-and-recovery. Solid vs. elusive and transient faults. Distributed fault tolerant systems, and the consensus problem. The notion of coverage and its importance. Common-mode failures, failure independence and correlation. Examples of fault tolerant systems (Tandem families). Software fault tolerance. Intrusion detection and tolerance. Validation of fault tolerance.
- **Development of dependable systems.** Requirements and specifications. Development models (V, spiral, processes). Risk analysis and criticality definition. Development process control (maturity models) and evaluation. Relationship between a) criticalities, and b) approaches and methods; norm and standard examples (IEC 61508 on Electronic safety-related systems, DO 178 on Software in airborne systems, ISO 17799 on Security techniques, ISO 15408 on Evaluation criteria for IT security).

Suggested readings:

A. Avizienis, J-C. Laprie, B. Randell and C. Landwehr: **Basic Concepts and Taxonomy of Dependable and Secure Computing,** IEEE Trans. On Dependable and Secure Computing,

Vol.1, n.1, Jan.-March 2004
D. P. Siewiorek and R. Swartz: **Reliable Computer Systems, Design and Evaluation,** Third Edition, A K Peters, Ltd., 1998
J-C. Laprie et al.: **Guide de la sûreté de fonctionnement**, Cepaduès Editions, 1995 (in French)
N. Ferguson and B. Schneider: **Practical Cryptography,** John Wiley &Sons, 2003

Courseware examples and locations where taught:
Higher National School of Aeronautics and Space (ENSAE), Toulouse. Jean-Claude Laprie
Higher National School of Electronics, Informatics, and Radiocommunications of Bordeaux. Jean-Claude Laprie
Higher National School of Electrotechnology, Electronics, Informatics, Hydraulics and Telecommunications, Toulouse. Jean-Charles Fabre
These topics are covered in the MSc-level course on "Fault-tolerant design of computer systems" available at City University as a professional development short course. The slides are not publicly available.

1st Year – 2nd Semester

4.8 Computer Networks Security (6 ECTS)

The purpose of this course is to offer a broad overview not only of security building blocks but also of existing threats. The course is made of three distinct parts. In the first one, one considers preventive techniques that aim at preventing attacks from succeeding against computing systems. This includes i) the necessary cryptographic foundational material, existing security protection mechanisms at ii) the operating system, iii) at the network level, and iv) at the application level. In the second part, the existing threats against information systems are reviewed. The goal of this exercise is to explain, by means of concrete examples, the reasons why a purely preventive approach is unlikely to succeed in any reasonably complex network. In other words, one will always have to cope with remaining vulnerabilities inserted or activated during the operational life of the system. This leads to the third part that considers the various intrusion detection mechanisms that aim at informing security administrators that an attack is on going. The various paradigms and implementations, together with their weaknesses and drawbacks are reviewed

Contents

- Security services and policy.
- Basic information security concepts and mechanisms
- Network security mechanisms
- Operating system security mechanisms
- Application level security mechanisms:
- Information Security Threats
- Intrusion detection

Suggested readings:

C. Kaufman, R. Perlman, and M. Speciner: **Network Security: Private Communication in a Public World**, Second Edition, Prentice Hal

W. Stallings: **Cryptography and Network Security**, (4th Edition), Prentice Hall

J. F. Kurose, K. W. Ross, Pearson: **Computer Networking: A Top-Down Approach**, (4th Edition), chap. 7

W. R. Cheswick, S. M. Bellovin, and A. D. Rubi: **Firewalls and Internet Security: Repelling the Wily Hacker**, Second Edition, Addison Wesley

Courseware examples and locations where taught:

- Institut Eurecom, Security applications in networking and distributed systems. R. Molva http://www.eurecom.fr/util/coursdetail.fr.htm?id=23
- Institut Eurecom, Operational Network Security. M. Dacier. http://www.eurecom.fr/util/coursdetail.fr.htm?id=19
- Universidade de Lisboa, Faculdade de Ciências. Security. P. Verissimo

4.9 Fault and Intrusion-Tolerant Distributed Systems and Algorithms (6 ECTS)
The purpose of this course is to prepare the students to understand and design dependable and secure distributed algorithms. The emphasis is on fault-tolerant protocols that tolerate not only accidental faults but also malicious faults, which is a more recent research trend in dependability.

Contents
- Introduction to distributed algorithms
- System models: distributed computation, processes, communication, time assumptions, failure detection
- Quorum systems
- Shared memory, registers, and distributed storage
- Group communication
- Failure detectors
- Reliable broadcast
- Consensus algorithms
- Consensus variants: atomic broadcast, non-blocking-atomic commit, group membership
- Byzantine algorithms: failure detection, reliable broadcast, consensus, atomic broadcast
- Distributed cryptographic protocols

Suggested readings:
R. Guerraoui and L. Rodrigues: **Introduction to Reliable Distributed Programming,** Springer, 2006.
P. Verissimo and L. Rodrigues: **Distributed Systems for System Architects**, Kluwer, 2001
N. Lynch: **Distributed Algorithms**, Morgan Kaufmann, 1996.

Courseware examples and locations where taught:
- Slides related to the first book and the first part of the course: http://www.di.fc.ul.pt/~ler/irdp/teaching.htm
- Slides on security and fault-tolerance in distributed systems: http://www.zurich.ibm.com/~cca/sft06/

4.10 Dependability Evaluation of Computer Systems (6 ECTS)
The purpose of this course is to introduce the quantitative evaluation approaches that take aim at probabilistically estimating the adequacy of a computer system with respect to the performance and dependability-related requirements given in its specification.

Contents
Analytical model-based evaluation
- Combinatorial dependability models (fault tree, event tree, reliability block diagram, reliability graphs)
- State space based dependability models (Markov chains, Markov regenerative processes, higher level formalisms as Stochastic Petri Nets, Stochastic Reward Nets and their extensions)
- Model construction and solution approaches (model composition approaches, model decomposition and aggregation approaches)
- Failure independence and common mode failure
- Generation of dependability models by model transformation from engineering models (UML, AADL)

Simulation based evaluation
- Simulation techniques (Monte Carlo, discrete event, trace driven simulation)
- Coping with the rare event problem
- Network simulation models (e.g. impact of mobility and traffic)

Experimental evaluation
- Field measurements and statistical inference techniques,"software reliability growth modeling", prediction methods and tests of accuracy
- Fault injection techniques (software or hardware implemented fault injection)
- Experimental evaluation of security (identifying vulnerabilities and characterization of attacker's behaviour)
- Dependability benchmarking

4. Combining the different evaluation techniques

Suggested readings:
K. S. Trivedi: **Probability and Statistics with Reliability, Queuing, and Computer Science Applications,** Second Edition, John Willey & Sons, 2002
M. Lyu (Ed.): **Handbook of Software Reliability Engineering,** McGraw Hill, 1996
available on line: http://www.cse.cuhk.edu.hk/~lyu/book/reliability/

Courseware examples and locations where taught:
Slides from the same author, available at http://www.ee.duke.edu/~kst/

4.11 Testing, Verification and Validation (6 ECTS)

The purpose of this course is to understand the role of testing, verification and validation in the design and analysis of systems, and to provide an advanced background on related methods and tools.

The course presents formal models and specification techniques that can be used for verification and validation: model checking, deductive methods and static analysis.

The course also presents the fundamentals of software testing. It provides an understanding of testing problems, and covers the major test design techniques. Emphasis is put on the need for rigorous, semi-automated approaches.

Contents

- Introduction to formal methods and computer aided verification, theoretical and practical limitations, semi-formal and formal modelling (UML, transition systems, process algebras, Petri nets, logic, ...)
- Model checking
  - Industrial application of model checking
  - State explosion problem: Symbolic and SAT-based model checking
  - Model checking timed and hybrid systems
  - System verification with a model checker tool: a case study.
- Deductive methods
  - System verification with a theorem prover tool: logical background & a case study
- Static program analysis
  - Control flow program analysis
  - Approximate semantics of programs
  - Basic elements of abstract interpretation theory
- Software testing

- Fundamentals of testing: role of testing throughout the software life cycle, test selection and oracle problems, test integration strategy, classification of test methods.
- Usual structural & functional approaches: control and data flow criteria, predicate coverage, domain testing, model-based testing (e.g., from finite state machines, labelled transition systems), syntax testing.
- Mutation analysis: principle, examples of mutation operators.
- Probabilistic test approaches: uniform profile, operational profile, profiles based on structural and functional criteria.
- Test automation: classification of tools.

Suggested readings:

T. Kropf: **Introduction to Formal Hardware Verification**, Springer, 1999.

B. Berard, M. Bidoit, et al.: **System and Software Verification – Model-Checking Techniques and Tools**, Springer, 2001.

C. Hankin, F. Nielson, H. R. Nielson: **Principles of Program Analysis**, Springer, 1999.

B. Beizer: **Software Testing Techniques**, Van Nostrand Reinhold, 1990 (2nd edition)

R. D. Craig, S. P. Jaskiel: **Systematic Software Testing**, Artech House, 2002

A. Robinson, A. Voronkov (eds.): **Handbook of Automated Reasoning**, **Volume I**, North Holland, 2001

Courseware examples and locations where taught:

- "Computer-Aided Verification", Rajeev Alur at University of Pennsylvania, Philadelphia, USA, http://www.cis.upenn.edu/cis673/, Slides and draft textbook available
- "Verification of Reactive Systems" and "Deductive Verification of Reactive Systems",

Amir Pnueli at The Weizmann Institute of Science, Rehovot, Israel http://www.wisdom.weizmann.ac.il/~amir/Course01a/header.html and http://www.wisdom.weizmann.ac.il/~amir/Course02a/header.html, Slides available

- "Test and Verification" Emmanuel Fleury, Kim G. Larsen, Brian Nielsen, Arne Skou at Aalborg University, Denmark http://www.cs.auc.dk/~kgl/TOV04/Plan.html, Slides available
- " Theorem Proving and Model Checking in PVS " Edmund M. Clarke and Daniel Kroening at CMU, Pittsburgh, USA http://www.cs.cmu.edu/~emc/15-820A/ Slides available
- "System Validation" Theo C. Ruys at University of Twente http://fmt.cs.utwente.nl/courses/systemvalidation/ Slides available
- "Validation and Verification" J.P. Bowen at University College London http://www.cs.ucl.ac.uk/staff/J.Bowen/GS03/ Slides available
- "Abstract Interpretation" Patrick Cousot, Jerome Clarke Hunsaker at MIT http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www Slides available
- "Software testing " Paul Ammann and Jeff Offutt at George Mason University http://ise.gmu.edu/~pammann/637-sched.html Slides available

4.12 Usability and User Centred Design for Dependable and Usable Socio-technical Systems (6 ECTS)

The purpose of this course is to introduce the notion of usability of systems and to present User centred development processes that are targeting at usability.

Contents

- Introduction to usability (definition, principles & concepts)
- Ergonomic rules and design guidelines
- Work analysis and task analysis
- Usability evaluation
- User Centred Development processes (implication of users, prototyping approaches)

Suggested readings:

ISO 9241-11:1998 **Ergonomic requirements for office work with visual display terminals** (VDTs) -- Part 11: Guidance on usability
http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16883
J. Nielsen: **Usability Engineering**, Morgan Kaufmann, San Francisco, 1994.
D. Norman: **The design of everyday things**. Basic books, 3rd edition, 2002.

Courseware examples and locations where taught:

- http://sigchi.org/cdg/index.html (gathering a large set of lectures on HCI and Human factors mainly in the US. This set of courses have been gathered and organised by the ACM Special Interest Group on HCI)
- http://vip.cs.utsa.edu/classes/cs6693s2006/lectures/index.html

2nd Year – 3rd Semester

The syllabi for the courses in a box will be identified and detailed in subsequent versions of this document.

Common Courses

4.13 Management of Projects (3 ECTS)
The purpose of this course is understanding how to manage a project from initial specification to delivery on the field of systems with resilience requirements.

Contents
- Purpose of Project Management
- Project Management in System Engineering
- Project Management as a process
- Principal phases of the Project Management process:
    - Definition of the scope of the project
    - Feasibility evaluation and resources estimation
    - Identification of project interfaces
    - Responsibility allocation
    - Project planning
    - Project monitoring and risk management
    - Project deviation Management and impact analysis
    - Project data storage
- Techniques, methods and tools for Project Management with resilience requirements.

Suggested readings:
Howard Eisner **Essentials of Project and System Engineering Management**, Second Edition. John Wiley and Sons, 2002.
James Taylor **Managing Information Technology Projects,** AMACOM Div American Mgmt Assn 2003.
Mary Beth Chrissis, Mike Konrad, Sandy Shrum **CMMI Guidelines for Process Integration and Product Improvement**, SEI Series in Software Engineering, 2004.

Courseware examples and locations where taught:
TBD

4.14 Fault Tolerant Middleware-based Systems (3 ECTS)
The purpose of this course is ……..

Contents
(RTCORBA, FTCORBA, JAVA, EJB, J2EE, components framework, SOA, WS, reflection, /AOP/AOSD, …)
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

4.15 Software Reliability Engineering (3 ECTS)
The purpose of this course is ……..

Contents
TBD
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

Telecom Applications Track

4.16.1 Resilience of Protocols and Architecture (3 ECTS)
The purpose of this course is ……..

Contents
(Mobile, Internet, industrial networks, routing, …)
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

4.17.1 Resilience of Mobile Applications (3 ECTS)
The purpose of this course is ……..

Contents
TBD
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

4.18.1 Project in cooperation with Industry (9 ECTS)
The purpose of this project is ……..

4.19.1 Additional Course(s) (6 ECTS)
The purpose of these courses is ……..

Contents
TBD
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

Safety Critical Applications Track

4.16.2 Development Process and Standards for Safety critical Applications (3 ECTS)
The purpose of this course is ……..

Contents
(embedded C&C, aerospace, automotive, train, nuclear, DO, ARINC, IEC, AUTOSAR, CENELEC, …)
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

4.17.2 Architectural Issues and Examples of Systems (3 ECTS)
The purpose of this course is ……..

Contents
TBD
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

4.18.2 Project in cooperation with Industry (9 ECTS)
The purpose of this project is ……..

4.19.2 Additional Course(s) (6 ECTS)
The purpose of these courses is ……..

Contents
TBD
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

e-Business Applications Track

4.16.3 Resilience of SOA and Web-based Applications (3 ECTS)
The purpose of this course is ……..

Contents
(SOA, webservices, e-commerce, e-*, …)
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

4.17.3 Damage Tolerance in Large scale Systems (3 ECTS)
The purpose of this course is ……..

Contents
TBD
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

4.18.3 Project in cooperation with Industry (9 ECTS)
The purpose of this project is ……..

4.19.3 Additional Course(s) (6 ECTS)
The purpose of these courses is ……..

Contents
TBD
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

2nd Year – 4th Semester

The syllabi for the courses in a box will be identified and detailed in subsequent versions of this document.

4.20 Specific Courses and/or Seminars (3 ECTS)
The purpose of this course/seminar is ……..

Contents
TBD
Suggested readings:
TBD
Courseware examples and locations where taught:
TBD

4.21 MSc Thesis Preparation and Presentation (27 ECTS)

References in the text

[1] http://ec.europa.eu/education/policies/educ/bologna/bologna.pdf
[2] http://ec.europa.eu/education/policies/educ/bologna/bergen.pdf
[3] http://ec.europa.eu/education/policies/educ/bologna/report06.pdf
[4] http://www.resist-noe.org/
[5] http://www.cra.org/reports/trustworthy.computing.pdf
[6] http://ec.europa.eu/education/programmes/socrates/ects/index_en.html

References to all suggested readings

- K. S. Trivedi: **Probability and Statistics with Reliability, Queuing, and Computer Science Applications**, Second Edition, John Willey & Sons, 2002.
- M. Lyu (Ed.): **Handbook of Software Reliability Engineering,** McGraw Hill, 1996.
- J. Menezes, P. C. van Oorschot and S. A. Vanstone: **Handbook of Applied Cryptography,** CRC Press, 1996.
- N. Smart: **Cryptography, An Introduction**, McGraw-Hill, 2002.
- M. Huth and M. Ryan: **Logic in Computer Science**, Cambridge University Press
- V. Detlovs, K. Podnieks: **Introduction to Mathematical Logic** http://www.ltn.lv/~podnieks/mlog/ml.htm
- W. Johnson: **Failure in Safety-Critical Systems. A Handbook of Accident and Incident Reporting**. Available on-line at: http://www.dcs.gla.ac.uk/johnson/book October 2003. 2003. Glasgow, Scotland, University of Glasgow Press.
- J. Reason: **Human Error**. 1990. Cambridge University Press.
- J. Reason: **Managing the Risks of Organizational Accidents**, 1997, Aldershot, UK, Ashgate.
- D. Wickens and J. G. Hollands: **Engineering Psychology and Human Performance**. 3rd edition, 1999, Prentice Hall.
- F. S. Hillier, G. J. Lieberman: **Introduction to Operation Research**, 8th Edition, McGraw Hill, 2005
- D. Bertsimas and J. Tsitsiklis: **Introduction to Linear Optimization**, Athena Scientific, 1997
- M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali: **Linear Programming and Network Flows**, Wiley, 1990

- R.L. Rardin: **Optimization in Operations Research**, Prentice Hall, 1997
- W.L. Winston and M. Venkataramanan: **Introduction to Mathematical Programming**, Duxbury Press, 2002
- S. Even: **Graph Algorithms**, Computer Science Press, 1979, 249 pp.
- J.L. Gross and J. Yellen (Eds.): **Handbook of graph theory**, CRC Press, 2003, 1192 pp.
- F. Cottet, J. Delacroix, C. Kaiser, Z. Mammeri: **Scheduling in Real-Time Systems**, Wiley Eds, 2002, 266p. ISBN: 0-470-84766-2
- G. Buttazzo: **Hard Real-Time Computing Systems**, Second Edition, Series: Real-Time Systems Series, Vol. 23, 2005, XIII, 425 p., ISBN: 978-0-387-23137-2Springer, 2005.
- H. Kopetz: **Real-Time Systems: Design Principles for Distributed Embedded Applications**, Series: The Springer International Series in Engineering and Computer Science, Vol. 395, 1997, 356 p., ISBN: 978-0-7923-9894-3
- A. Burns and A. J. Wellings: **Real-Time systems and programming languages**, 3rd ed., Addison Wesley, 2001, Pages 610 p., ISBN 0-201-40365-X
- A. Avizienis, J-C. Laprie, B. Randell and C. Landwehr: **Basic Concepts and Taxonomy of Dependable and Secure Computing,** IEEE Trans. On Dependable and Secure Computing, Vol.1, n.1, Jan.-March 2004
- D. P. Siewiorek and R. Swartz: **Reliable Computer Systems, Design and Evaluation,** Third Edition, A K Peters, Ltd., 1998
- J-C. Laprie et al.: **Guide de la sûreté de fonctionnement**, Cepaduès Editions, 1995 (in French)
- N. Ferguson and B. Schneider: **Practical Cryptography,** John Wiley &Sons, 2003
- C. Kaufman, R. Perlman, and M. Speciner: **Network Security: Private Communication in a Public World**, Second Edition, Prentice Hal
- W. Stallings: **Cryptography and Network Security**, (4th Edition), Prentice Hall
- J. F. Kurose, K. W. Ross, Pearson: **Computer Networking: A Top-Down Approach**, (4th Edition), chap. 7
- W. R. Cheswick, S. M. Bellovin, and A. D. Rubi: **Firewalls and Internet Security: Repelling the Wily Hacker**, Second Edition, Addison Wesley
- R. Guerraoui and L. Rodrigues: **Introduction to Reliable Distributed Programming,** Springer, 2006.
- P. Verissimo and L. Rodrigues: **Distributed Systems for System Architects**,
- Kluwer, 2001
- N. Lynch: **Distributed Algorithms**, Morgan Kaufmann, 1996.
- T. Kropf: **Introduction to Formal Hardware Verification**, Springer, 1999.
- B. Berard, M. Bidoit, et al.: **System and Software Verification – Model-Checking Techniques and Tools**, Springer, 2001.
- C. Hankin, F. Nielson, H. R. Nielson: **Principles of Program Analysis**, Springer, 1999.
- B. Beizer: **Software Testing Techniques**, Van Nostrand Reinhold, 1990 (2nd edition)
- R. D. Craig, S. P. Jaskiel: **Systematic Software Testing**, Artech House, 2002
- A. Robinson, A. Voronkov (eds.): **Handbook of Automated Reasoning**, **Volume I**, North Holland, 2001
- ISO 9241-11:1998 **Ergonomic requirements for office work with visual display terminals** (VDTs) -- Part 11: Guidance on usability http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16883
- J. Nielsen: **Usability Engineering**, Morgan Kaufmann, San Francisco, 1994.
- D. Norman: **The design of everyday things**. Basic books, 3rd edition, 2002.

Appendixes

Appendix A – Courses in Dependability, Security and Human Factors taught at present from ReSIST partners

In this appendix 53 forms have been collected during the first year, through direct interviews to all persons members of ReSIST to have a snapshot of what is available from inside the NoE. They are accessible through the RKB at http://resist.ecs.soton.ac.uk/courseware/ also in a geographical form.

COURSES

Advanced seminars on Distributed Systems

Taught at: Universita degli studi di Roma, La Sapienza

Description: The course focuses on recent advances on distributed systems. A set of topic is selected and studied through the help of original papers and, practically, most known distributed system platforms are selected and analyzed.

Language(s) of the course: English

Select Author(s): Roberto Baldoni

Select Lecturer(s): Roberto Baldoni

Submitted by: Roberto Baldoni

Number of credits: 5

Total hours of lectures: 30

Total hours of labs: 20

Total hours of personal study: 18

Student Interaction Type:

Individual Class Participation

Group Project

Group Presentation

Group Laboratory Practice

Group Homework

Assessment methods:

Attendance

Pre-requisites:

Fundamentals of Operating Systems

Data Structures

Basic understanding of Operating Systems concepts

Basic knowledge of computer networking

Basic concepts of Dependable Computing

Knowledge of distributed systems

Infrastructure Required: Single PC

Course Objectives:

Support courseware used:

Computer-assisted Modelling and Analysis

Taught at: University of Ulm
Description:
Language(s) of the course: German
Select Author(s): Friedrich von Henke
Select Lecturer(s): Friedrich von Henke, Holger Pfeifer
Submitted by: Friedrich von Henke
Number of credits: 6
Total hours of lectures: 28
Total hours of labs: 28
Total hours of personal study: 28
Student Interaction Type:
Individual Laboratory Practice
Individual Homework
Lectures
Assessment methods:
Oral Examination
Pre-requisites:
Basic knowledge of logic
Infrastructure Required:
Single PC
Course Objectives:
Support courseware used:

Cryptographic Protocols and Formal Methods

Taught at: IBM Research GmbH

Description: For a long time, cryptography and formal methods provided two almost unrelated ways of proving security protocols: Cryptography provides precise and realistic definitions of the security of cryptographic primitives e.g., encryption and signatures, and protocols, e.g., secure channels or fair exchange. However, these definitions and corresponding proofs are long and difficult because many details are explicit, e.g., active attacks, error probabilities, and computational restrictions.

Formal methods, on the other hand, have well-defined protocol languages and provide tool support for proofs, such as model checking or theorem proving. This is particularly useful for distributed-system aspects of protocols, which are particularly tedious and error-prone if proved by hand.

However, almost all formal methods abstract from cryptography using so-called Dolev-Yao models. These models idealize cryptography as term algebras with fixed cancellation rules, and assume that knowledge that cannot be derived by these rules cannot be derived at all. For twenty years there was no justification for this abstraction.

These talks give an overview of recent advances of linking cryptography and formal methods. Particular emphasis will be on the cryptographic justification of a Dolev-Yao model under active attacks and for arbitrary protocol environment. We will also survey how cryptographic theorems for composition and property preservation enable this link and where actual tool-supported verification over proven abstractions stands.

Language(s) of the course: English

Select Author(s):Michael Backes, Birgit Pfitzmann

Select Lecturer(s): Birgit Pfitzmann

Submitted by: Birgit Pfitzmann

Number of credits: 0

Total hours of lectures: 6

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Lectures

Assessment methods:

Pre-requisites:

Basic knowledge of modern cryptography is helpful; graduate level

Infrastructure Required: None

Course Objectives: See "**Description**". This course is a summer-school type course, with the objective to introduce graduate students to a busy research field. It has also

been given by Michael Backes, University of Saarland, in various lengths, most recently in Estonia with added credits, assessments, etc.

Support courseware used:

Dependability 1: Reliable Distributed Systems

Taught at: Technische Universitat Darmstadt

Description: Development of concepts for design, assessment and testing for distributed embedded systems and software/OS's.

Language(s) of the course: English

Select Author(s): Neeraj Suri

Select Lecturer(s): Neeraj Suri

Submitted by: Brahim Ayari

Number of credits: 6

Total hours of lectures: 28

Total hours of labs: 28

Total hours of personal study: 42

Student Interaction Type:

Lectures

Assessment methods:

Oral Examination

Pre-requisites:

Basic understanding of Operating Systems concepts

Probability Theory

Infrastructure Required:

Course Objectives: Development of concepts for design, assessment and testing for distributed embedded systems and software/OS's.

Support courseware used:

http://resist.ecs.soton.ac.uk/courseware/resources/d41d8cd9

Freely Available: Yes

Copyright or Restrictions: Yes

Dependability evaluation of computer systems

Taught at:
ENAC
Institut National des Sciences Appliquees de Toulouse (INSA-T)
University of Toulouse III
Description: A master-level course on stochastic-based analytical techniques for the dependability evaluation of computer systems.
The course is structured as follows:

1. Introduction and basic concepts: overview of dependability concepts, importance and objectives of dependability evaluation in the system life cycle, ordinal and stochastic evaluation techniques
2. Probabilistic Metrics for dependability evaluation (R(t), A(t), MTTF, MTTR, MUT, MDT, etc.)
3. Evaluation methods: FMECA, Fault trees, Reliability Diagrams, Markov models
4. Examples and Case studies

Language(s) of the course: French
Select Author(s): Mohamed Kaaniche
Select Lecturer(s): Mohamed Kaaniche
Submitted by: Mohamed Kaaniche
Number of credits: 0
Total hours of lectures: 8
Total hours of labs: 2
Total hours of personal study: 0
Student Interaction Type:
Individual Class Participation
Assessment methods:
Written Examination
Pre-requisites:
Probability Theory
Infrastructure Required:
Course Objectives: To provide an introduction to the main metrics and methods for quantifying the dependability of computer based systems, using analytical evaluation techniques
Support courseware used:

Dependability of computer systems

Taught at:

Description: an introductory course to dependability of computer systems. It is composed of 6 chapters.

1) Basic concepts and definitions: attributes of, means for, threats to dependability. Importance of dependability in current information infrastructures (examples of widely publicized failures, economic impact of failures)
2) Threats to dependability: faults, errors, failures. Definitions and classification. Failure pathology. Respective importance based on statistics (frequency, consequences) of the various classes (physical, development, interaction; accidental, malicious)
3) Fault removal. Basic verification approaches: reviews and inspections, abstract interpretation, theorem proving, model checking, symbolic execution, testing. Principles, possibilities and limitations of the approaches. The cost of verification. Statistics on classes of faults uncovered during development and during operation.
4) Fault forecasting. Ordinal (or qualitative) and stochastic (or quantitative) evaluations. Stable vs. evolutive (growing, decreasing) dependability. Measures (or metrics) of dependability. Evaluation approaches: modelling (FMECA, reliability diagrams, fault trees), measurements. The case of critical systems.
5) Fault tolerance. Basic primitives: error detection (concurrent, preemptive), system recovery by error treatment (rollback, rollforward, compensation) and fault treatment (diagnosis, isolation, reconfiguration, reinitialization). Solid vs. elusive and transient faults. Basic architectures for detection-and-recovery, and masking-and-recovery. The notion of coverage and its importance. Distributed fault tolerant systems, and the consensus problem. Examples of fault tolerant systems (Tandem families). Software fault tolerance. Validation of fault tolerance.
6) Development of dependable systems. Risk analysis. Requirements and specifications. Development models (V, spiral, processes).

Language(s) of the course: English, French

Select Author(s): Jean-Claude Laprie

Select Lecturer(s): Jean Arlat, Jean-Charles Fabre

Submitted by Jean-Claude Laprie

Number of credits: 2

Total hours of lectures: 9

Total hours of labs: 0

Total hours of personal study: 9

Student Interaction Type:

Individual Class Participation

Assessment methods:

Written Examination

Pre-requisites:

Basic understanding of Operating Systems concepts

Knowledge of distributed systems

Probability Theory

Infrastructure Required:

Course Objectives: An introductory course to dependability of computer systems

Support courseware used:

Dependable Embedded Systems

Taught at: Technische Universitat Darmstadt
Description: This seminar course considers research in the area of making Embedded Systems Dependable. Covering Systems, Software/OS and Middleware, the course covers topics of design, assessment and testing of Dependable Embedded Systems
Language(s) of the course: English
Select Author(s): Neeraj Suri
Select Lecturer(s): Neeraj Suri
Submitted by: Brahim Ayari
Number of credits: 3
Total hours of lectures: 28
Total hours of labs: 0
Total hours of personal study: 28
Student Interaction Type:
Individual Presentation
Assessment methods:
Oral Examination
Pre-requisites:
Basic Knowledge of Data Structures
Infrastructure Required:
Course Objectives: Deepen understanding and knowledge of a particular field of research in the area of dependable embedded systems.
Support courseware used:
http://resist.ecs.soton.ac.uk/courseware/resources/d41d8cd9
Freely Available: Yes
Copyright or Restrictions: Yes

Dependable systems and architectures (Architetture di sistemi affidabili)

Taught at:

Ansaldo Segnalamento Ferroviario

Ericsson LAb Italy

Università degli Studi di Firenze

Description:

Language(s) of the course: English, Italian

Select Author(s): Andrea Bondavalli

Select Lecturer(s): Andrea Bondavalli, Felicita Di Giandomenico

Submitted by: Andrea Bondavalli

Number of credits: 0

Total hours of lectures: 16

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Individual Class Participation

Assessment methods:

Attendance

Oral Examination

Pre-requisites:

Infrastructure Required:

Course Objectives:

Support courseware used:

Design Rationale for Interactive Systems

Taught at

ENAC

Université Toulouse 1, Sciences Sociales

University of Toulouse III

Description: This course introduces issues and techniques for supporting rationale analysis, design and implementation of interactive systems

Language(s) of the course: French

Select Author(s): Palanque Philippe

Select Lecturer(s): Palanque Philippe

Submitted by: Philippe Palanque

Number of credits: 1

Total hours of lectures: 6

Total hours of labs: 3

Total hours of personal study: 9

Student Interaction Type:

Group Homework

Group Laboratory Practice

Lectures

Assessment methods:

Assessed Laboratories

Pre-requisites:

Basic knowledge of Human-Computer Interaction

Programming languages

State based modelling

Infrastructure Required:

Single PC

Windows XP

Course Objectives:

- Learn how to systematically consider various options while analysing, designing and implementing interactive systems.
- Learn the underlying concepts of rationalizing and structuring argumentation during the various phases of the development process of interactive systems.
- Lear how to organise models and designs in order to provide traceability of the analysis, design and implementation choices

Support courseware used:

Design Rationale Slides (in French)

Freely Available: Yes

Copyright or Restrictions: No

Design and Evaluation of Web Applications

Taught at

ENAC

University of Toulouse 1, Sciences Sociales

University of Toulouse III

Description:

- WWW history
- Diversity of Web application
- Usability and Accessibility in Web Design
- Development of Web Applications
- Model-based development of Web Applications
- Web architectural framework
- Web Standards and Technologies
- Architecture of Web applications
- Methods for Evaluation of Web applications

Language(s) of the course: English

Select Author(s):

Select Lecturer(s):

Submitted by:

Number of credits: 2

Total hours of lectures: 18

Total hours of labs: 12

Total hours of personal study: 18

Student Interaction Type:

Group Project

Individual Laboratory Practice

Assessment methods:

Practical Examination

Pre-requisites:

Basic Knowledge of Data Structures

Basic knowledge of Human-Computer Interaction

Basic knowledge of computer networking

Basic understanding of Operating Systems concepts

Fundamentals of Operating Systems

Knowledge of distributed systems

Software Engineering Fundamentals

State based modelling

Basic knowledge of computer architectures

Infrastructure Required:

Java programming environment

Linux

Networked PCs

Windows XP

Course Objectives:

- Provide basic background for the development of complex Web applications.
- Enable students to employ methods, tools and techniques for the design of Web application
- Present model-based approaches for design and specification of Web applications
- Discuss techniques for support the management of large Web site including evolution of content and structure
- Show evaluation techniques for testing the usability and feasibility of Web applications

Support courseware used:

Distributed Algorithms (MSc Computer Security & Resilience)

Taught at: University of Newcastle upon Tyne

Description: Distributed algorithms are the foundation on which system services are built. This module will cover core distributed algorithms by concentrating on three key attributes: processing and communication delays and component failures. Students will acquire skills in understanding distributed algorithms and protocols required for designing distributed systems in a modular way.

Outline Syllabus:

- Preliminaries: Synchronous and Asynchronous communication models, precedence relations, non-deterministic computations and execution configurations, basics on tree structures, and basics of cryptography.
- Fundamental Algorithms: Wave and Election Algorithms for trees, rings, and arbitrary topological structures. Example applications on Routing Algorithms and e-auction sites.
- Algorithms in e-Commerce: Fair Exchange Algorithms. On-line and Off-line algorithms. Contract Exchange Applications.
- Algorithms for Distributed Data Management: Database Commit Protocols: 2-phase and 3-phase protocols. The requirements and the limitations of commit protocols.
- Algorithms for Fault-Tolerant Distributed Computing: Replication Strategies. Reliable and Ordered Broadcasts, Reaching agreement and Consensus protocols.

Language(s) of the course: English

Select Author(s):

Select Lecturer(s):

Submitted by: John Fitzgerald

Number of credits: 5

Total hours of lectures: 12

Total hours of labs: 6

Total hours of personal study: 64

Student Interaction Type:

Individual Laboratory Practice

Lectures

Assessment methods:

Individual Coursework

Written Examination

Pre-requisites:

Data Structures

Knowledge of distributed systems

Infrastructure Required:

Course Objectives:

Aims:

Distributed algorithms are the foundation on which system services are built. The aim of the module is to cover core algorithms by concentrating on three key attributes that are very significant in building responsive applications: processing and communication delays and component failures.

Intended knowledge outcomes:

Students will have developed an understanding of:

- the role of essential services needed to build reliable and responsive applications
- the concept of feasibility: whether services can be implemented under specified restrictions under which feasibility can be attained.

Intended skills outcomes:

Students will acquire skills in:

- construction of distributed algorithms and protocols underpinning design of distributed systems
- reasoning about the properties of distributed algorithms and protocols

Support courseware used

Readings in Distributed Computing Systems/Eh0359-0 (IEEE Computer Society Press Tutorial)

Distributed Algorithms (The Morgan Kaufmann Series in Data Management Systems)

Introduction to Distributed Algorithms

Distributed Fault Tolerance

Taught at:

ENSICA

Université Toulouse 1, Sciences Sociales

Description:

3-hour part of a 20-hour 2nd year masters course on dependable computing. Subjects covered include distributed system models, distributed algorithms for agreement, clock synchronization, group communication, checkpointing and replication.

Language(s) of the course: English, French

Select Author(s): David Powell

Select Lecturer(s): David Powell

Submitted by: David Powell

Number of credits: 0

Total hours of lectures: 3

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Lectures

Assessment methods:

Written Examination

Pre-requisites:

Basic concepts of Dependable Computing

Infrastructure Required:

Course Objectives: To provide an introduction to software-implemented approaches to fault-tolerance.

Support courseware used:

Distributed Systems

Taught at: Universita degli studi di Roma, La Sapienza
Description:
Language(s) of the course: Italian
Select Author(s):
Select Lecturer(s):
Submitted by: Roberto Bonato
Number of credits: 5
Total hours of lectures: 35
Total hours of labs: 15
Total hours of personal study: 26
Student Interaction Type:
Lectures
Assessment methods:
Written Examination
Pre-requisites:
Basic understanding of Operating Systems concepts
Fundamentals of Operating Systems
Infrastructure Required: None
Course Objectives:
Support courseware used:

Distributed algorithms for fault-tolerant distributed systems

Taught at : Universite de Rennes 1

Description: The aim of this course is to make the student familiar with the fundamental results in fault-tolerant distributed computing: what can be done (i.e., what can be computed
and what can be efficiently computed) in failure-prone asynchronous distributed systems and what cannot.
To that end, the course explore the:
Fundamental problems with a particular accent on agreement problems, and mainly on the consensus problem. The course visits both synchronous and asynchronous systems in failure models of increasing severity: crash, general omission and Byzantine failures. Then, the construction of communication primitives with added values value is studied, mainly uniform reliable broadcast and atomic broadcast. Wait-free construction of asynchronous objects is also deeply investigated. Herlihy 's hierarchy is investigated and universal constructions are developed.

Language(s) of the course: English, French

Select Author(s): Michel Raynal

Select Lecturer(s): Michel Raynal

Submitted by: Michel Raynal

Number of credits: 4

Total hours of lectures: 20

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Group Homework

Group Project

Assessment methods:

Individual Coursework

Written Examination Open Book

Pre-requisites:

Fundamentals of Operating Systems

Infrastructure Required:

Course Objectives: At end of the course, the student has to be familiar with basic concepts, models, mechanisms and algorithms that underlie the basic foundations of fault-tolerant distributed computing.

Support courseware used:

Textbooks:

- "Distributed algorithms" by Nancy Lynch ISBN 1-55860-348-4
- Distributed Computing" by H. Attiya and J. Welch ISBN 0-471-45324-2

- Reliable distributed programming " by R. Guerraoui and L. Rodrigues ISBN 3-540-28845-7

E-Privacy: Privacy in the Electronic Society

Taught at: ETH Zurich

Description: Privacy issues have been the subjects of public debates since years. In particular, as new technologies are developed, they increasingly raise privacy concerns - the World Wide Web, wireless location-based services, and RFID chips are just a few examples. Thus, the need for privacy-aware policies, regulations, and techniques has been widely recognized by lawmakers, regulators, and the media. As a result, businesses are under pressure to draft privacy policies, chief privacy officers are becoming essential members of many organizations, and companies are taking pro-active steps to avoid the potential reputation damage of a privacy mistake. This course provides an in-depth look into privacy laws and regulations as well as into technologies for achieving privacy in an electronic world.

Language(s) of the course: English

Select Author(s): Günter Karjoth, Jan Camenisch

Select Lecturer(s):Günter Karjoth, Jan Camenisch

Submitted by Christian Cachin

Number of credits: 5

Total hours of lectures: 0

Total hours of labs: 30

Total hours of personal study: 30

Student Interaction Type:

Assessment methods:

Pre-requisites:

Infrastructure Required:

Course Objectives:

Support courseware used:

Experimental Assessment of Computer System Dependability

Taught at:
ENSICA
EPFL
University of Naples
University of Toulouse III
Tokyo Institute of Technology
Description: A master-level course on experimental assessment of dependability. Subjects covered include: dependability, fault tolerance, assessment, fault injection, fault load, workload, measurements, testing, evaluation, coverage estimation, fault injection techniques, examples of conducted evaluations, benchmarking
Language(s) of the course: English, French
Select Author(s): Jean Arlat
Select Lecturer(s): Jean Arlat, Jean-Charles Fabre
Submitted by: Jean Arlat
Number of credits: 0
Total hours of lectures: 10
Total hours of labs: 2
Total hours of personal study: 0
Student Interaction Type:
Individual Class Participation
Assessment methods:
Written Examination
Pre-requisites:
Basic concepts of Dependable Computing
Infrastructure Required:
Course Objectives: To provide insight on the role of fault-injection based controlled experiments and to describe techniques for designing, running and exploiting such experiments.
Support courseware used:

Fault Tolerance Via Diversity Against Design Faults: Design Principles and Reliability Assessment

Taught at: City University, London

Description:

Design faults account for a large part of failures in mature software-based products. Fault tolerance, employing redundant, diverse software component, has been used for many years as a defence. Though commonly associated with safety-critical real-time systems, its principles are applicable to all kinds of software designs. While the idea is attractive, the use of diversity has been vehemently attacked. Most practitioners for whom software fault tolerance could be a viable choice have little information for decisions about its use.

This half-day tutorial introduces both the methods of fault-tolerant software design, with examples from industrial practice and from research, and the problem of estimating the reliability gain from fault-tolerant design. This estimation is as difficult as for other software engineering methods. The tutorial explains the basic results so far, clarifies the terms of decisions about using software fault tolerance, and outlines recent progress in research. The tutorial is designed as an introduction to this topic for an audience with an experience in software engineering and an understanding of the basic concepts of software reliability.

Language(s) of the course: English

Select Author(s): Bev Littlewood, Lorenzo Strigini

Select Lecturer(s): Bev Littlewood, Lorenzo Strigini

Submitted by: Andrey Povyakalo

Number of credits: 0

Total hours of lectures: 4

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Lectures

Assessment methods:

Attendance

Pre-requisites:

Probability Theory

Software Engineering Fundamentals

Infrastructure Required: None

Course Objectives: The tutorial is designed as an introduction to the topic for an audience with an experience in software engineering and an understanding of the basic concepts of software reliability

Support courseware used:

Fault Tolerance in Distributed Object-Oriented Systems

Taught at: LAAS-CNRS

Description: This course covers some core aspects of fault tolerance in distributed systems and shows the role and the use of object-oriented concepts in the design and in the implementation of fault tolerant systems. Beyond basic principles of Dependability, and more precisely Fault Tolerant Computing in distributed systems, we focus on architectural concepts and implementation techniques for the development of object-oriented fault tolerant distributed systems. Object-oriented aspects are addressed at the language level, at the middleware level (CORBA) and using more advanced features like reflective computing. Several implementation approaches and examples of object-oriented fault tolerant systems are analyzed to discuss properties, limits and basic assumptions (Delta-4, Electra, Eternal, Arjuna, Maud, Garf, Friends).

Language(s) of the course: English

Select Author(s): Jean-Charles Fabre

Select Lecturer(s): Jean-Charles Fabre

Submitted by: Jean-Charles Fabre

Number of credits: 0

Total hours of lectures: 8

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Lectures

Assessment methods:

Attendance

Pre-requisites:

Basic concepts of Dependable Computing

Fundamentals of Operating Systems

Knowledge of distributed systems

Programming languages

Infrastructure Required:

Course Objectives: This course provides to computer system designers architectural concepts for the development of fault tolerant object oriented systems and illustrate them with well-known examples of systems and prototypes.

Support courseware used:

Fault-Tolerant Design of Computer Systems: An Introductory Course

Taught at: City University, London

Description: Companies place increasing reliance on computer systems for the very survival of their business; computer applications become ever more complex, yet they are often built from unreliable components, hardware or software.

Fault tolerance - design for surviving component failures- is becoming a necessity for a growing number of companies, far beyond its traditional application areas, like aerospace and telecommunications.

This course - organised as five one-day lectures that can be taken individually - addresses the needs of:

- IT and engineering managers who have to address new needs for dependability of their (or their customers') computer applications
- Software designers or system integrators who want an introduction to the problems found in designing for fault tolerance and to the range of design solutions.

Language(s) of the course: English

Select Author(s): Lorenzo Strigini

Select Lecturer(s): Lorenzo Strigini

Submitted by: Andrey Povyakalo

Number of credits: 0

Total hours of lectures: 25

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Lectures

Assessment methods:

Attendance

Pre-requisites:

Basic understanding of Operating Systems concepts

Software Engineering Fundamentals

Basic knowledge of computer architectures

Infrastructure Required: None

Course Objectives:

At the end of this course students should:

- understand the risk of computer failures and their peculiarities compared with other equipment failures;
- know the different advantages and limits of fault avoidance and fault tolerance techniques;

be aware of the threat from software defects and human operator error as well as from hardware failures;

- understand the basics of redundant design;
- know the different forms of redundancy and their applicability to different classes of dependability requirements;
- be able to choose among commercial platforms (fault-tolerant or non fault-tolerant) on the basis of dependability requirements;
- be able to specify the use of fault tolerance in the design of application software;
- understand the relevant factors in evaluating alternative system designs for a specific set of requirements;
- be aware of the subtle failure modes of "fault-tolerant" distributed systems, and the existing techniques for guarding against them;
- understand cost-dependability trade-offs and the limits of computer system dependability

Support courseware used:

Group Project (MSc Computer Security and Resilience)

Taught at: The University of Newcastle upon Tyne

Description: In this module you will have the opportunity to design, build and analyse a high integrity computing system with security or safety implications. You will have access to technical guidance form faculty members but will essentially work as a small independent team under your own initiative. This provides an opportunity to practice the technical skills developed in other modules, and to develop new skills of cooperative working and organisation.

Outline Syllabus:

1. Team Working: the basics of successful team working in systems analysis and development. Teams of 5-7 members will be selected by the faculty, mixing backgrounds, abilities and skills. Each team has a faculty member who can advise on the problem specification but will not generally intervene in the group. Choice group organisation is left to the group, subject to the constraints of the problem. At the end of the project, there will be an opportunity to debrief with faculty members and with other students and to share experience of good and bad practice in team working.
2. System Synthesis and Analysis: review of process models, organisation and potential sequencing of synthesis, evaluation and V&V activities for high integrity systems. Outline of professional, legal and ethical issues relevant to the subject area of the project. An introduction to the forms of trade-off that exist between technical aspects of security and resilience with other properties such as system liveness. Initial functional and non-functional (including trustworthiness) requirements will be supplied, as well as process and product constraints. Development will involve research, requirements elicitation, modelling and analysis which may involve information flow modelling, protocol design, deductive or inductive fault analysis. Selection of development methods and tools will be done by each group. In a separate evaluation phase teams will be asked to evaluate and find defects in other teams' implementations, giving an opportunity to practice defect detection skills.
3. Reporting: Oral and written reporting of technical progress; an introduction to presentation skills for technical presentations; structuring and writing of key documents and presentation of evidence for high integrity systems. The project constraints will define technical deliverables. Teams will be invited to present findings in short oral presentations at various stages in the project. Team members will keep personal logs and write a short final individual report outlining their contributions and the lessons that they have learned from the project in terms of their own continuing professional development needs.

Example Scenario:
The challenge of producing a trustworthy e-Voting system combines technical work on secrecy and encryption with human factors and the acceptability of the system in the voting booth. Teams will be asked to develop part of an e-Voting scheme that provides voter-verifiability coupled with ballot secrecy, integrity and accuracy. They will be required to analyse key properties of their system, including those derived form a human factors analysis. In the evaluation phase, teams will try to discover flaws in the implementations developed.
Language(s) of the course: English
Select Author(s): John Fitzgerald
Select Lecturer(s):
Submitted by: John Fitzgerald
Number of credits: 7
Total hours of lectures: 9
Total hours of labs: 75
Total hours of personal study: 66
Student Interaction Type:
Group Laboratory Practice
Group Presentation
Group Project
Assessment methods:
http://resist.ecs.soton.ac.uk/courseware/cc6f8724
Pre-requisites:
Infrastructure Required:
Course Objectives:
Aims:

- To gain and reflect on the experience of applying the techniques taught in preceding modules to the analysis and development of a high integrity system.
- To gain experience of working in groups and to design and implement software under time and resource constraints, practising relevant professional skills.

Intended knowledge outcomes:

- Appreciation of the challenges involved in planning a system development in which security, resilience and integrity are significant.
- Improved understanding of issues that relate to the planning and execution of a team-based software project.

Intended skills outcomes:
Ability to:

- Plan a series of development activities working within resource limitations
- Work cooperatively as a team in a competitive environment

- Select appropriate technology for an analysis or development task for a high integrity system, taking account of resilience requirements, professional, legal and ethical aspects
- Justify design decisions from among a range of alternatives

Support courseware used:

High Integrity Software Development (MSc Computer Security & Resilience)

Taught at: The University of Newcastle upon Tyne

Description: An increasing range of tools is available for the development of secure and dependable systems. This module introduces the programming principles that underpin systems that have predictable levels of dependability and shows how those principles are realised in the leading high-integrity programming systems. The module has a strong practical component.

Outline Syllabus:

1. Role of validation and verification in "V" and iterative life cycle models
2. Correctness & Security of Code:
    a. Correctness: notions of partial and total functional correctness versus weaker type correctness properties.
    b. Verifying correctness: limitations of dynamic testing, code verification, Correctness by Construction, refinement, integrated approaches (in outline)
    c. Domain-specific Properties: examples of critical properties from security, safety and human factors domains.
    d. Maintainability: requirements & design volatility, lessons from reliability growth modelling
3. Language subsets:
    a. Case Studies: MISRA-C (safety), Ravenscar (real-time systems)
    b. Design level: Safecharts
4. Modern Analytic Tools:
    a. Principles of Static Analysis: data flow & program slicing;
    b. Model checking in practice: SPIN
5. Integrating advanced analysis & model checking: Spec#, ESC/Java

Language(s) of the course: English

Select Author(s): Steve Riddle

Select Lecturer(s):

Submitted by; John Fitzgerald

Number of credits: 5

Total hours of lectures: 20

Total hours of labs: 10

Total hours of personal study: 70

Student Interaction Type:

Individual Class Participation

Individual Laboratory Practice

Individual Project

Individual Homework
Lectures
Assessment methods:
Individual Coursework
Pre-requisites:
Basic Knowledge of Data Structures
Basic concepts of Dependable Computing
Basic knowledge of computer networking
Basic understanding of Operating Systems concepts
Data Structures
Knowledge of distributed systems
Programming languages
Infrastructure Required:
Course Objectives:
Aims:
To introduce principles of high-integrity systems programming, use of restricted language sets and analysis techniques to develop dependable software
Intended knowledge outcomes:

- Knowledge and understanding of processes and techniques for development of high-integrity software
- Knowledge and understanding of pitfalls of "standard" programming languages and the motivation for safe subsets of programming languages
- Understanding of the role of techniques and tools to develop and analyse software

Intended skills outcomes:

- Ability to select and apply techniques for development of high-integrity software
- Ability to critically analyse software and explain safety implications
- Ability to justify design decisions to a high level of rigour as part of a dependability case

Support courseware used:

- High Integrity Software: The SPARK Approach to Safety and Security
- Safety Critical Computer Systems

How testing improves reliability: models for comparing testing methods and their combinations

Taught at: City University, London

Description: Testing can be used to assess how good software is, or to find faults and thus improve the software. This tutorial focuses on this second use of testing, and on a specific meaning of "improvement", i.e., "making software more reliable". There are many testing methods, and strong opinions on their relative merits, but empirical validation of these opinions is difficult and hard to generalize. Rather abstract models of how testing leads to removing faults can provide valuable insight about what we should expect from practical applications of testing, and what we should measure to guide our choices. Two practical problems will be addressed in which modelling helps understanding.

The first problem is choosing between testing methods in terms of the project risk that they imply, i.e., the risk of delivering a product of sub-standard quality. It turns out that "operational" testing appears to have greater strengths than claimed by its advocates: testing methods that appear superior to it on average may actually be much riskier than it is.

The second problem is how best to combine different methods in the V&V of a product. There is a dilemma between applying diverse methods to take advantage of their different strengths, and looking instead for one best method and concentrating all resources on applying that method alone. The models explain how to resolve this dilemma: when is it that diversity pays off even if it means using methods that, on their own, are inferior, and which measures are needed to support a decision.

Language(s) of the course: English

Select Author(s): Lorenzo Strigini

Select Lecturer(s): Lorenzo Strigini

Submitted by: Andrey Povyakalo

Number of credits: 0

Total hours of lectures: 4

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Lectures

Assessment methods:

Attendance

Pre-requisites:

Probability Theory

Software Engineering Fundamentals

Infrastructure Required: None

Course Objectives: The tutorial is designed as an introduction to the topic for an audience with an experience in software engineering and an understanding of the basic concepts of software reliability.

Support courseware used:

Human Factors Engineering (MSc Computer Security & Resilience)

Taught at: The University of Newcastle upon Tyne

Description: Interactions between humans and technical systems are an important source of potential errors. This module will introduce you to principles and techniques for analysing human factors in the design of computer-based systems. We discuss psychological aspects of human error, and the socio-technical factors. We also discuss and practice techniques for error assessment and user-centred design.

Outline Syllabus:

1. Human Performance and New Technology: technology, scenarios, tasks, errors and principles.
2. Cognitive Psychology and Human Error. An introduction to theories of Human Information processing and Human Error. An error classification exercise;
3. Workload and situation awareness and their measurement;
4. People in systems. An orientation to human factors in interactive safety critical systems, the role of ergonomic, organizational and socio-technical factors, the ironies of automation, the anatomy of an accident. Exercise on analysing an aircraft accident report;
5. Work Representation and Design. An introduction to Hierarchical Task Analysis and scenarios with examples;
6. Human Reliability Assessment. An introduction to the core features of human reliability assessment (HRA) with an emphasis on descriptive techniques such as THEA. A critique of quantification. HRA worked example and exercise;
7. User-centred design and evaluation methods.

Language(s) of the course: English
Select Author(s): Michael Harrison
Select Lecturer(s):
Submitted by: John Fitzgerald
Number of credits: 5
Total hours of lectures: 20
Total hours of labs: 10
Total hours of personal study: 60
Student Interaction Type:
Group Laboratory Practice
Individual Laboratory Practice
Individual Homework
Lectures
Assessment methods:
Individual Coursework

Written Examination

Pre-requisites:

Infrastructure Required:

Course Objectives Aims:

- To give students knowledge and experience of the concepts and techniques that can be used to support the design and assessment of dependable interactive systems.
- To develop an appreciation of ergonomic, psychological and socio-technical factors that are relevant to an understanding of these systems.

Intended knowledge outcomes:

- A knowledge of human factors theories and models and methods appropriate to complex systems;
- An understanding of human interface considerations that affect productive, safe and reliable operation; Appreciation of user-centred design and evaluation of interactive systems;
- An understanding of the role and pitfalls of quantitative human error assessment.

Intended skills outcomes:

- Ability to select and apply methods of human error assessment and usability evaluation.

Support courseware used:

Normal Accidents

The Design of Everyday Things

A Guide To Practical Human Reliability Assessment

Cognitive Reliability and Error Analysis Method (CREAM)

Imaging for Security Applications: Watermarking and Biometrics

Taught at: Institut Eurecom

Description:

- Brief history of information hiding
- Basic principles and techniques: still images, video and 3-D video objects
  Expected applications: owner authentication, content authentication, information embedding as communication with side information
- Evaluation and benchmarking
- Malicious attacks, bit rate limitation; counterfeiting marks; removal attacks, etc…
  Overview of different attempts to formalize watermarking

Language(s) of the course: English

Select Author(s): Jean-Luc Dugelay

Select Lecturer(s): Jean-Luc Dugelay

Submitted by: Guillaume Urvoy-Keller

Number of credits: 2

Total hours of lectures: 21

Total hours of labs: 6

Total hours of personal study: 21

Student Interaction Type:

Lectures

Assessment methods:

Written Examination

Pre-requisites:

Infrastructure Required:

Course Objectives:

Watermarking:

Watermarking allows owners or providers to hide an invisible and robust message inside a digital Multimedia document, mainly for security purposes such as owner or content authentication. There is a complex trade-off between the different parameters: capacity, visibility and robustness.

Biometrics:

The security fields use three different types of authentication: something you know, something you have, ore something you are – a biometric.

Common physical biometrics includes fingerprints, hand geometry and retina, iris or facial characteristics. Behavioural characters include signature, voice. Ultimately, the technologies could find their strongest role as intertwined and complementary pieces of a multifactor authentication system. In the future biometrics is seen playing a key role in enhancing security, residing in smart cards and supporting personalized Web

e-commerce services. Personalization through person authentication is also very appealing in the consumer product area. This course will focus on enabling technologies for Biometrics, wit a particular emphasis on person verification an authentication based on or widely using image/video processing.

Support courseware used

Biometrics: Personal Identification in Networked Society (The International Series in Engineering and Computer Science)

Information Security and Trust (MSc Computer Security & Resilience)

Taught at: The University of Newcastle upon Tyne

Description: Encryption alone does not guarantee security: we must understand the flow of information and the levels of trust that exist between individuals and organisations. This module investigates rigorous techniques for modelling and reasoning about trust, security policies and communication protocols. Established and new approaches, their advantages and limitations, are discussed and demonstrated.

Outline Syllabus:

1. Security properties: authentication, secrecy, integrity, etc.
2. Information flow models
3. Trust and trust models
4. Security policies and policy languages
5. Security kernels and enforceable policies
6. Role of formal techniques in the development and evaluation of secure systems
7. Security protocol specification languages.
8. Verification of security properties.
    a. State-based verification techniques
    b. BAN logic based approaches
    c. Theorem proving (rank functions).
9. Computer-aided verification of cryptographic protocols

Language(s) of the course: English

Select Author(s): Peter Ryan

Select Lecturer(s):

Submitted by: John Fitzgerald

Number of credits: 5

Total hours of lectures: 24

Total hours of labs: 12

Total hours of personal study: 64

Student Interaction Type:

Individual Class Participation

Individual Laboratory Practice

Individual Homework

Lectures

Assessment methods:

Individual Coursework

Written Examination

Pre-requisites

Basic concepts of Dependable Computing

Basic knowledge of computer networking

Programming languages

Infrastructure Required

Course Objectives

Aims: To cover formal techniques and computer aided verification topics relevant to those involved in the design and validation of information security systems.

Intended knowledge outcomes:

- An appreciation of the importance of information security requirements in modern distributed environments
- An understanding of the information security mechanisms and primitives, threats and counter-measures.
- An understanding of techniques to analyse designs against security requirements and threat environments.
- An understanding of the ways in which formal methods can be used to support rigorous specification and validation of a design.

Intended skills outcomes:

The ability to

- Design and validate designs of secure systems.
- Select and use techniques and computer-aided model checking tools for security protocols.

Support courseware used:

Information System Security

Taught at: LAAS-CNRS, Institut National des Sciences Appliquees de Toulouse (INSA-T)

Description:

Subjects covered include security properties; attack classification; defence techniques (cryptography, isolation, audit, intrusion detection, ...); security policies and models; authentication; protection and access control; security evaluation.

Language(s) of the course: French

Select Author(s): Yves Deswarte

Select Lecturer(s): Yves Deswarte

Submitted by: Yves Deswarte

Number of credits: 0

Total hours of lectures: 12

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Lectures

Assessment methods:

Written Examination

Pre-requisites:

Basic knowledge of computer networking

Basic understanding of Operating Systems concepts

Infrastructure Required:

Course Objectives: To provide computer system engineers/researchers with basics on security.

Support courseware used:

Interactive Systems Engineering

Taught at: University of Toulouse III

Description:

- Development processes of interactive systems
- Formal notations for dialogue **Description** of interactive systems (IS)
- Verification of properties of IS
- Interactive Systems testing
- From specification to implementation
- Development with Visual Basic prototyping tool

Language(s) of the course: French

Select Author(s): Palanque Philippe

Select Lecturer(s): Palanque Philippe

Submitted by: Philippe Palanque

Number of credits: 3

Total hours of lectures: 40

Total hours of labs: 15

Total hours of personal study: 20

Student Interaction Type:

Group Homework

Group Project

Lectures

Assessment methods:

Practical Examination

Pre-requisites:

Programming languages

State based modelling

Infrastructure Required:

Java programming environment

MicroSoft Visual Basic 6.0

Single PC

Windows XP

Course Objectives: Learn the specificities of the development of interactive Systems. These specificities are explained with respect to the development process, the notation and the implementation.

Basic introduction to Visual Basic 6.0 is given as it proves to be a very efficient tool for high fidelity prototyping of standard interactive systems.

Support courseware used:

Interactive Systems Engineering Slides (in French)

Freely Available: Yes
Copyright or Restrictions: Yes

Introduction to software testing

Taught at: University of Toulouse III, ENSICA

Description: The course involves 5 chapters:

1. Basic notions: testing in the software development process, testing levels, selection and oracle problems, classification of testing methods.
2. Structural testing: control flow, data flow
3. Functional testing: equivalence classes and boundary analysis, decision tables, finite state automata, IOLTS
4. Mutation analysis: principle, tools
5. Probabilistic generation of test data: uniform profile, operational profile, profiles based on structural and functional criteria

Language(s) of the course: French

Select Author(s): Helene Waeselynck

Select Lecturer(s): Helene Waeselynck

Submitted by: Helene Waeselynck

Number of credits: 0

Total hours of lectures: 7

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Individual Class Participation

Assessment methods:

Written Examination

Pre-requisites:

Basic concepts of Dependable Computing

Infrastructure Required:

Course Objectives: To provide an introduction to testing problems and to demonstrate the need for rigorous, semi-automated approaches.

Support courseware used:

Intrusion Detection and Tolerance

Taught at: Universidade de Lisboa
Description: Discuss in depth the main methods to obtain security using the dependability paradigm in the presence of malicious faults (attacks, vulnerabilities, intrusions), or in short: how to detect and tolerate intrusions.
Syllabus: Introduction to security; Introduction to fault tolerance and dependability; Intrusion Tolerance (IT) concepts and terminology; Fault Models, Classical methodologies, Error processing, Fault treatment; Intrusion Detection concepts; Intrusion Detection mechanisms; Intrusion Tolerance (IT) mechanisms and strategies; Example IT projects and systems
Language(s) of the course: Portuguese
Select Author(s): Paulo Verissimo
Select Lecturer(s): Paulo Verissimo
Submitted by: Miguel Correia
Number of credits: 6
Total hours of lectures: 28
Total hours of labs: 0
Total hours of personal study: 0
Student Interaction Type:
Individual Class Participation
Individual Presentation
Individual Homework
Assessment methods:
Pre-requisites:
Basic Knowledge of Data Structures
Basic concepts of Dependable Computing
Basic knowledge of computer networking
Basic understanding of Operating Systems concepts
Data Structures
Fundamentals of Operating Systems
Knowledge of distributed systems
Programming languages
Basic knowledge of computer architectures
Infrastructure Required:
Course Objectives: Discuss in depth the main methods to obtain security using the dependability paradigm in the presence of malicious faults (attacks, vulnerabilities, intrusions), or in short: how to detect and tolerate intrusions.
Support courseware used: Intrusion-Tolerant Architectures: Concepts and Design
Freely Available: Yes

Management of Computing Infrastructure

Taught at: Budapest University of Technology and Economics

Description:

The lectures of this course present the following topics:

- Network management (network resources, name resolution, bandwidth allocation, performance measurement and planning, traffic management).
- Security of computer networks (firewalling, protection against viruses).
- Security of information systems (formal models and standards).
- User management (authentication and authorization, access control, directory systems). Capacity planning (modelling, requirements and capacities, benchmarking).
- Configuration management.
- System operation (monitoring and system surveillance, disaster management, desktop management, integrated system management).
- Storage management (storage devices, virtual storage, data backup and recovery).
- The ITIL methodology (overview, concepts and modules).

Language(s) of the course: Hungarian

Select Author(s): Gábor Huszerl

Select Lecturer(s): Gábor Huszerl

Submitted by: Istvan Majzik

Number of credits: 5

Total hours of lectures: 60

Total hours of labs: 0

Total hours of personal study: 60

Student Interaction Type:

Individual Class Participation

Lectures

Assessment methods:

Oral Examination

Pre-requisites:

Basic knowledge of computer networking

Basic understanding of Operating Systems concepts

Infrastructure Required: None

Course Objectives: The students will be familiar with the typical management activities in a well-operated enterprise information infrastructure. They will be able to understand the principles of the main aspects of infrastructure management and gain insights into the basic planning, dimensioning, operation and maintenance techniques and standards. They will have an understanding of the importance and means of systematic management of information systems.

Support courseware used: Course slides (in Hungarian)

Freely Available: Yes

Copyright or Restrictions: Yes

Description of Cost and Copyright: Budapest University of Technology and Economics, Dept. of Measurement and Information Systems

Operational Network Security

Taught at: Institut Eurecom
Description:
- Network vulnerabilities:
Protocol attack in IP, TCP, Web, e-mail, DNS (syn flooding, teardrop, cgi-bin, smurf, etc.), application specific exposures (mail packages, web servers, viruses)
- Intrusion Processes:
Intranet attack scenarios (sniffers, spoofing), Internet attack scenarios (by-passing filters, sniffer set-up, flooding, distributed denial of service attack scripts, script kiddies)
- Intrusion Detection Basics:
Principles (behaviour analysis, pattern matching, normalization), techniques (statistical methods, rule-based systems, neural networks, genetic algorithms)
Vulnerability Testing, Ethical Hacking
Language(s) of the course: English
Select Author(s): Marc Dacier
Select Lecturer(s): Marc Dacier
Submitted by: Guillaume Urvoy-Keller
Number of credits: 4
Total hours of lectures: 42
Total hours of labs: 9
Total hours of personal study: 42
Student Interaction Type:
Group Project
Assessment methods:
Written Examination
Pre-requisites:
Infrastructure Required:
Course Objectives: This course provides both a broad survey of intrusion patterns threatening
global network operation and an investigation of countermeasures to thwart these intrusions. We analyse attacks with a systematic approach, identifying typical attack classes such as traffic subversion, masquerading, and denial of service. We present an in-depth study of intrusion detection techniques including main concepts and industrial solutions. This course
does not cover cryptographic mechanisms.
A substantial amount of personal work is expected from the students who have to study and implement, by groups, different types of attacks in a dedicated network.
Support courseware used: Computer Security: Art and Science

Privacy-Enhancing Technologies

Taught at: LAAS-CNRS, Centre de formation à la sécurité des systèmes d'information (CFSSI)
Description: Subjects covered include privacy definitions and regulations; privacy principles; management of virtual identities; anonymous communications; anonymous access to services; privacy-preserving authorization; personal data management.
Language(s) of the course: French
Select Author(s): Yves Deswarte
Select Lecturer(s): Yves Deswarte
Submitted by: Yves Deswarte
Number of credits: 0
Total hours of lectures: 6
Total hours of labs: 0
Total hours of personal study: 0
Student Interaction Type:
Lectures
Assessment methods:
Attendance
Pre-requisites
Basic knowledge of computer networking
Basic understanding of Operating Systems concepts
Infrastructure Required:
Course Objectives: To provide computer system engineers/researchers with basics on privacy problems and solutions.
Support courseware used:

Project and Dissertation (MSc Computer Security and Resilience)

Taught at: The University of Newcastle upon Tyne

Description: The individual project is a substantial piece of independent work involving the technical and research skills developed in the taught part of the degree. You will have the opportunity to contribute directly to research or development activities, develop your own specialist expertise in the project topic, and further improve your planning and communication skills. You will work closely with a member of staff from one of the School's research groups, and you may also be working with an industrial partner (possibly in an industry laboratory). The project culminates in a dissertation and workshop presentation with other students, staff and industry partners.

Outline Syllabus:

1. Project Definition and Planning: Bounding and clarifying a problem for research. Planning background research, identifying relevant information sources. Decomposing a problem and forming a project outline in terms of goals and criteria for success. Identifying resources and tooling required. Students will either select a project from a list offered by potential supervisors or propose and refine a project proposal with an academic supervisor. Projects will involve learning about some unfamiliar aspect of secure and resilient systems. In every project there will be a research component and a strong design, programming and/or analytic element.
2. Supervision Arrangements: Each project has a lead supervisor and second supervisor, both staff from the School. Additional supervision support may be provided by an industrial partner. The student and lead supervisor will meet regularly throughout the period of the project.
3. Research: Background research will be undertaken in the selected topic using the skills developed in earlier modules with access to library and online resources. The supervisor will advise on quality of sources and standards in the topic area.
4. Development and Analytic Skills: The core of the project will involve carrying out the project plan largely independently, but with guidance form the supervisors.
5. Report Writing: An Introduction to higher writing skills for a research-led project. An interim report will be produced 10 weeks into the project, documenting progress to date and future plans.
6. Technical Presentation: A presentation will be given to students and staff at the end of the project.
7. Dissertation: A dissertation will be prepared, describing the technical background, the work undertaken, the analysis of results and directions for further work.

Guidance on the style and content of an academic dissertation will be provided by means of guest lectures and through the supervisor.

Language(s) of the course: English

Select Author(s): John Fitzgerald

Select Lecturer(s):

Submitted by: John Fitzgerald

Number of credits: 0

Total hours of lectures: 0

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Individual Project

Assessment methods:

http://resist.ecs.soton.ac.uk/courseware/5a0a43a7

Pre-requisites:

Infrastructure Required:

Course Objectives:

Aims:

- To deepen the knowledge and skills acquired in the first half of the programme through practice.
- To develop an awareness of the range and limitations of technologies available for the analysis and development of secure or resilient systems.
- To enhance research skills and awareness of the professional literature.
- To develop an awareness of open problems in the subject.

Intended knowledge outcomes:

- Detailed knowledge of analysis, design and development techniques appropriate to realistic problems in system dependability.
- Improved knowledge of the professional literature and information resources.

Intended skills outcomes:

- Problem solving skills related to the provision of predictably dependable systems.
- Written and oral communication skills appropriate to technical communication and presentation to non-specialists.
- Personal skills for lifelong learning; ability to exploit professional information resources.

Support courseware used:

Requirements engineering: natural language requirements elicitation, specification and quality evaluation

Taught at: The University of Pisa

Description:

Skills in Requirement Engineering (as for instance the realisation of complete, unambiguous requirement documents, or the requirement change management) are considered determinant for the success of industrial projects for software-intensive systems. That determined an increasing awareness of the importance of the specific role of the requirement engineer.

Software engineering courses are today widely spread in almost all IT-specific curricula, because the academia in the last two decades understood the demand for skilled software professionals coming from the market. Nevertheless, software engineering courses focus more on coding, architecture and testing than requirements engineering.

Skills in requirements engineering are pressing for almost every possible scenario an engineer will operate in, because requirements are handled (written or used) during the whole development process.

MODULE I: Software Quality, principles and theory

The purpose of this introductory module is to underline the importance of a culture of quality in software engineering and to understand the consequences of lack of quality at all the phases of the software development with particular emphasis on the requirements phase. Metrics, reviews and process assessments, as means to evaluate the quality, are also treated.

- Quality of product
- Quality of process
- Software Quality Assurance and Project Management
- International standards
- Software Quality Metrics

MODULE II. Requirements Engineering

In this module the basic concepts of the requirements engineering discipline are considered. The state-of-the-art techniques for representing and managing requirements are described.

- Requirements elicitation (techniques and methods)
- Requirements documentation
- Formal methods
- Graphical notations
- Natural language-based notations
- Requirements management (techniques and tools)

MODULE III: Requirements Elicitation Workshop

A meeting between customer and supplier for eliciting customer requirements is simulated in this module. The workshop is composed of a presentation of the customer needs (the teacher plays the role of the customer) and an interactive part where the students are encouraged to interview the customer to elicit additional requirements/details, ask for clarifications and solve possible conflicts. A requirements document produced by the students is the expected outcome of this module.
-Case study presentation
- Elicitation of requirements
- Requirements document production
MODULE IV: Quality analysis of natural language requirements
After an overview of the existing tools and methods for the quality analysis of natural language requirements, the students run the quality analysis and calculate metrics on the requirements they produced in module 3 using an automatic tool.
MODULE V: Conclusions
This final module aims at sharing and discussing the results of the work made by the students in the laboratory session (module 4). The scope and validity of the analysis performed (what specific properties of the software are addressed and what not) are discussed too.
Language(s) of the course: English, Italian
Select Author(s): Giuseppe Lami
Select Lecturer(s): Giuseppe Lami
Submitted by: Giuseppe Lami
Number of credits: 4
Total hours of lectures: 15
Total hours of labs: 0
Total hours of personal study: 16
Student Interaction Type:
Individual Class Participation
Assessment methods:
Written Examination
Pre-requisites:
Software Engineering Fundamentals
Infrastructure Required: None
Course Objectives: The principal objectives of this series of lectures are (i) to make the students aware of the consequences of low quality requirements for a software project; and (ii) to provide them the basic skills to specify requirements as well to analyse and manage requirements quality.
These objectives are achieved by learning software quality concepts as well as requirements elicitation and specification techniques. The simulation of a

requirements elicitation meeting and the experience of the realisation and quality analysis of a real requirements document will consolidate the concepts learned.
Support courseware used: Engineering and Managing Software Requirements

Research Skills for Computer Security and Resilience (MSc Computer Security & Resilience)

Taught at: The University of Newcastle upon Tyne

Description: Any professional in the field of high integrity systems needs to know how to find and interpret up-to-date information, ranging from news of new risks or threats, which changes daily, to research results proposing and evaluating resilience technology. This practical module develops your awareness of the information resources available, your ability to use those resources, and to evaluate the quality of the information that you find. It is an essential preparation for embarking on group and individual projects in computer security and resilience. The module develops skills through practice in a series of seminars based around 'hot' topics.

Outline Syllabus:

1. The structure of the information space in dependability and security of computer-based systems:
    a. Informal Short-term news sources
    b. Rapid dissemination: information on threats and strategies
    c. Research life cycle: Initial Technical results: technical reports, conference proceedings, journals of record, texts.
    d. The role of peer review in system development and in research.
2. Information Literacy for Computer Security & Dependability:
    a. Identifying information need
    b. Locating and accessing information
    c. Comparing and evaluating sources
3. Critical evaluation of published material
4. Written presentation of findings:
    a. Identifying the readership
    b. Standards in the discipline regarding use and citation of sources
    c. Writing styles

Themes: an analysis of current directions in secure and resilient systems. Examples of themes to be considered include:

- International perspectives on regulation, compliance and research activity
- Flexible system structure: the move to dynamic coalitions
- Contracts, monitoring and dependability-explicit computing
- Perceptions of Risk

Examples of topics for papers and seminars include:

- Electronic voting
- Outsourcing
- Use of off-the-shelf systems, software of unknown provenance
- Intrusion detection

- Information security and corporate responsibility

Language(s) of the course: English

Select Author(s): John Fitzgerald

Select Lecturer(s):

Submitted by: John Fitzgerald

Number of credits: 3

Total hours of lectures: 10

Total hours of labs: 7

Total hours of personal study: 35

Student Interaction Type:

Individual Presentation

Individual Homework

Lectures

Assessment methods:

Individual Coursework

Pre-requisites:

Basic concepts of Dependable Computing

Infrastructure Required:

Course Objectives:

Aims:

- To give experience of reading and interpreting the professional and research literature.
- To encourage the development of an international and multidisciplinary view on dependable computer-based systems.
- To provide an insight into current and future developments in high integrity systems.

Intended knowledge outcomes:

- Understanding of the sources of information and standards of information quality that apply in Computing Science generally, and in Security and Dependability in particular.
- Appreciation of current challenges in research and topics of concern in industry and professional practice.
- In-depth knowledge about a particular technology or application area.

Intended skills outcomes:

- Improved appreciation of the research literature in high integrity systems.
- Critical evaluation of technical literature.
- Written technical communication skills

Support courseware used:

Secure Communications

Taught at: Institut Eurecom

Description: Policy, Access control models, security services, function placement in multi-layered communication systems

- Cryptography: Security evaluation, entropy, unicity distance, one-way functions, symmetric and asymmetric encryption algorithms
- Data encryption techniques, block chaining
- Data integrity: Hash functions, message digest algorithms, message authentication mechanisms
- Non-repudiation: Digital signature algorithms, non-repudiation protocols, fair exchange
- Authentication: Passwords, tokens, smartcards, authentication protocols
- Key management: Key management architecture, pseudo-random number generators, key generation, key distribution protocols, public-key certification
- Network access control: Firewall components, packet filters, proxies, circuit gateways, firewall configurations.

Language(s) of the course: English

Select Author(s): Refik Molva

Select Lecturer(s): Refik Molva

Submitted by: Guillaume Urvoy-Keller

Number of credits: 4

Total hours of lectures: 42

Total hours of labs: 12

Total hours of personal study: 42

Student Interaction Type:

Group Laboratory Practice

Assessment methods:

Written Examination

Pre-requisites:

Infrastructure Required:

Course Objectives: This course provides a broad introduction to communication security. The course covers basic information security techniques such as cryptography and access control, communication security mechanisms based on cryptography, and network security techniques based on filtering.

Support courseware used:

Building Internet Firewalls

Firewalls and Internet Security

Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition Cryptography (Discrete Mathematics and Its Applications)

Network Security: Private Communication in a Public World, Second Edition

Cryptography and Network Security (4th Edition)
Handbook of Applied Cryptography

Security

Taught at: Universidade de Lisboa
Description: This course gives an introduction to the area of security in networks and distributed systems. Students are exposed to fundamental paradigms of security--- such as cryptography, or authentication and access control--- and models of secure communication and programming. The latter are illustrated with example applications and systems.
Topics: Fundamental security concepts; Security paradigms; Models of secure distributed computing; Systems and environments for secure communication and programming.
Practical topics: Java security model; Building a sandbox; Security API in Java (SSL, JAAS); Mastering the iptables command; Mastering the snort command
Language(s) of the course: Portuguese
Select Author(s): Paulo Verissimo
Select Lecturer(s): Paulo Verissimo
Submitted by: Miguel Correia
Number of credits: 6
Total hours of lectures: 42
Total hours of labs: 0
Total hours of personal study: 0
Student Interaction Type:
Group Project
Individual Class Participation
Assessment methods
Oral Examination
Written Examination
Pre-requisites:
Basic Knowledge of Data Structures
Basic knowledge of computer networking
Basic understanding of Operating Systems concepts
Data Structures
Fundamentals of Operating Systems
Knowledge of distributed systems
Infrastructure Required:
Debugging Tools
Java programming environment
Linux
Networked PCs

Course Objectives: This course gives an introduction to the area of security in networks and distributed systems. Students are exposed to fundamental paradigms of security--- such as cryptography, or authentication and access control--- and models of secure communication and programming. The latter are illustrated with example applications and systems.

Support courseware used

Distributed Systems for System Architects (Advances in Distributed Computing and Middleware, Volume 1) (Advances in Distributed Computing and Middleware)

Security Applications in Networking and Distributed Systems

Taught at: Institut Eurecom

Description

- Advanced security Mechanisms: Public Key Infrastructures, distributed access control (capabilities, access control lists, privilege attribute certificates), Simple Public Key Infrastructure, distributed access control architectures
- Network Security Solutions: Internet security architecture (IPsec, SSL/TLS, IKE, ISAKMP,
  DNS), security protocols in mobile networks (GSM, DECT, UMTS, Mobile IP), multicast security,
- Distributed Application Security: www security, HTTP, SSL, e-mail security, PGP, S/MIME, certified mail, electronic payment systems, Secure Electronic Transactions, micro-payment, digital cash.

Language(s) of the course English

Select Author(s): Refik Molva

Select Lecturer(s): Refik Molva

Submitted by: Guillaume Urvoy-Keller

Number of credits: 2

Total hours of lectures: 21

Total hours of labs: 6

Total hours of personal study: 21

Student Interaction Type:

Group Laboratory Practice

Assessment methods:

Written Examination

Pre-requisites:

Infrastructure Required:

Course Objectives: This course presents the main applications of secure communication mechanisms in computer networks and distributed applications. The course covers complex security mechanisms (including public-key infrastructures), global security solutions for
specific areas (such as mobile networks) and application-specific security solutions (such as payment systems).

Support courseware used:

Cryptography: Theory and Practice, Second Edition

IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks

802.11 Wireless Networks: The Definitive Guide, Second Edition

Security Technologies

Taught at: Universidade de Lisboa

Description: This course presents a set of advanced topics and technologies in the area of security in distributed systems.

Syllabus:

- Cryptographic algorithms: study of encryption, hashing and digital signature algorithms (AES, DES, MD5, HMAC);
- Authentication: protocols for distributed authentication (Kerberos, Radius), including public key infrastructure (X.509);
- Secure communication: study of several protocols (IPsec, SSL/TLS, S-HTTP, Bluetooth), including secure email (S-MIME, PGP);
- Electronic payment systems: mechanisms for account transfer and digital coins (SET, Ecash, Millicent);
- Secure architectures: intrusion detection systems and firewalls, and the study of various types of malicious programs (virus, worms);
- Practical topics: Complements the topics discussed in the theoretical classes, and introduces other algorithms and protocols.

Language(s) of the course: Portuguese

Select Author(s): Nuno Neves

Select Lecturer(s): Nuno Neves

Submitted by: Miguel Correia

Number of credits: 6

Total hours of lectures: 45

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Group Project

Individual Class Participation

Assessment methods:

Oral Examination

Written Examination

Pre-requisites:

Basic Knowledge of Data Structures

Basic knowledge of computer networking

Basic understanding of Operating Systems concepts

Data Structures

Fundamentals of Operating Systems

Programming languages

Basic knowledge of computer architectures

Infrastructure Required:

Java programming environment

Linux

Networked PCs

Course Objectives: This course presents a set of advanced topics and technologies in the area of security in distributed systems.

Support courseware used:

Network Security: Private Communication in a Public World, Second Edition

Cryptography and Network Security (4th Edition)

Security and Fault-tolerance in Distributed Systems

Taught at: ETH Zurich

Description: This course presents methods for building dependable and secure distributed systems. The emphasis is on fault-tolerant and distributed cryptographic protocols. Topics include group communication, failure detectors, reliable broadcast protocols, distributed cryptography, threshold cryptosystems, Byzantine agreement, quorums, replication and secure networked storage systems. Applications to cluster computing, Internet services, and storage-area networks are presented.

Content

- Introduction
- Dependability Concepts
- Quorums
- Consensus and Broadcast
- View-synchronous Group Communication
- Distributed Cryptography
- Byzantine Agreement
- Service Replication
- Data Storage

Language(s) of the course English

Select Author(s): Christian Cachin

Select Lecturer(s): Christian Cachin

Submitted by: Christian Cachin

Number of credits: 5

Total hours of lectures: 30

Total hours of labs: 0

Total hours of personal study: 30

Student Interaction Type:

Lectures

Individual Homework

Individual Project

Individual Class Participation

Assessment methods: Oral Examination

Pre-requisites:

Fundamentals of Operating Systems

Data Structures

Basic understanding of Operating Systems concepts

Basic knowledge of modern cryptography is helpful; graduate level

Knowledge of distributed systems

Infrastructure Required:

Course Objectives:

Support courseware used: Course web page

Freely Available: Yes

Copyright or Restrictions: No

Security of Hardware

Taught at: Institut Eurecom

Description:

- Timing attack
- Simple power attack
- Differential power attack
- Electromagnetic attacks
- Fault injection
- Destructive attacks

Language(s) of the course: English

Select Author(s): Renaud Pacalet

Select Lecturer(s): Renaud Pacalet

Submitted by: Guillaume Urvoy-Keller

Number of credits: 2

Total hours of lectures2: 1

Total hours of labs: 0

Total hours of personal study: 21

Student Interaction Type:

Assessment methods:

Written Examination

Pre-requisites:

Infrastructure Required:

Course Objectives:

Embedded applications with strong security requirements use sophisticated cryptographic algorithms and protocols. Those algorithms and protocols are usually considered resistant against cryptanalysis.

Inside the complete system they are implemented either in software or hardware form. Unfortunately at least for the designers of such systems any computation is eventually performed by a piece of hardware (microprocessor or hardware dedicated accelerator) and every hardware device leaks symptoms of its activity (power consumption, electromagnetic emanations, computation time, etc.) An attacker can use such side channels to retrieve embedded secrets. She can also inject and exploit faults by modifying the power supply, the clock frequency or even by modifying the structure of the device. This course offers a survey of the known hardware attacks. For each of them the conditions of success are explained and some counter measures are proposed.

Support courseware used:

Software Development Technologies and Tools

Taught at: The University of Newcastle upon Tyne

Description:

Outline Syllabus

I. The Development Set:

- Recap of development activities and development data
- Review of model types: data, functionality, control and structure.
- Recap of formal modelling concepts in VDM-SL
- Basic Design Concepts: contracts; levels of abstraction.

II. The Search for Correctness:

- Correct by Construction?
- Correctness by Verification: principles and limitations of code verification
- Correctness by Construction: principles of refinement, data reification; operation decomposition
- Annotation and Markup approaches, e.g. JML Verification Technology:
- The spectrum: Proof, Model Checking, Inspection, Static Analysis and Testing
- Testing Fundamentals: cases, case selection, adequacy, specification-based test
- Test Strategies: control flow; data flow; transaction testing
- Metrics: size, coverage, defect classification.
- Test Engineering: regression, defect tracking, reliability growth.

III. Evidence and Arguments:

- The need for evidence and role of certification
- Argumentation: structuring and assembling
- Tracing the Development Set

IV. New Development Methodologies:

- Open source development
- Agile techniques
- Model-driven development

Language(s) of the course: English

Select Author(s): John Fitzgerald, Steve Riddle

Select Lecturer(s):

Submitted by: John Fitzgerald

Number of credits: 10

Total hours of lectures: 0

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Individual Class Participation

Individual Homework

Lectures

Assessment methods

Individual Coursework

Written Examination

Pre-requisites:

Infrastructure Required:

Course Objectives:

Aims:

To deepen students' knowledge of the underpinnings of software development so that they can:

- understand the basic principles of tools and techniques independent of current fashions
- integrate models expressed in the various UML notations introduced in earlier modules to gain a system-level view
- account for design decisions in rigorous terms
- plan and conduct appropriate validation and verification activities on software systems

Intended Knowledge Outcomes

- An understanding of the challenges in tracing and maintaining the design set
- Awareness of the design techniques for supporting large-scale or complex software development
- An understanding of the kinds of support tools available for industrial software development, including their limitations

Intended Skills Outcomes

- Ability to make informed choices of tools for larger-scale development tasks
- Ability to interpret claims made about design techniques and tools

Support courseware used:

Refinement Calculus: A Systematic Introduction (Texts in Computer Science)

Fundamentals of Software Engineering

Black-Box Testing: Techniques for Functional Testing of Software and Systems

Software Measurement

Taught at: City University, London
Description: The development and deployment of software-based systems must be conducted as an engineering discipline if the familiar problems of budget overruns, late delivery and inadequate quality are to be overcome. All branches of engineering employ measurement. In software engineering, "metrics" are sometimes used, but these are rarely defined according to the principles of measurement theory. Sound measurement needs to be practised in software engineering as it is in other branches. This module provides a comprehensive **Description** of the roles that should be played by measurement in software engineering. Emphasis is on a rigorous, but practical, approach based on sound measurement theory. It teaches how to use measurement to plan and control a software development project, to assess the dependability of the delivered system, and to manage the risks, which might materialise during development and use of the system.
Language(s) of the course: English
Select Author(s): Bev Littlewood
Select Lecturer(s): Bev Littlewood
Submitted by: Andrey Povyakalo
Number of credits: 15
Total hours of lectures: 20
Total hours of labs: 10
Total hours of personal study: 75
Student Interaction Type:
Lectures
Assessment methods:
Individual Coursework
Written Examination
Pre-requisites:
Probability Theory
Software Engineering Fundamentals
Infrastructure Required: None
Course Objectives
On successful completion of this module, a student will be expected to be able to:

- explain the purposes for which measurement is needed in software engineering, the underlying principles of the theory of measurement and its importance.
- describe the nature of the risks that affect software and how measurement is used in quantitative risk management

- identify national, European and international standards that require measurement to be applied to the software development process or to the delivered system, and state when and how they should be used
- explain why measurement is a necessary part of a mature approach to the management of software development projects
- draft a development plan including estimates of effort, cost and schedule for a software development project using software measurement to monitor and control the software development process according to the plan and to analyse and manage the risks associated with the project using a systematic approach
- draft a quality plan to define meaningful measures and quantitative targets for dependability attributes and other quality characteristics, including a programme of data
  collection to satisfy the defined quality objectives, using software measurement to assess the achieved levels of the attributes and characteristics during a realistic software product trial and during operation using a similar systematic approach.
- analyse software engineering data using statistical analysis techniques including software reliability growth modelling

Support courseware used:

Software Reliability analysis and Evaluation

Taught at: ENSICA

Description:

- Needs for software reliability analysis,
- Methods and tools for software reliability analysis and evaluation,
- Data collection and validation, data analysis,
- Descriptive statistics, software dependability evaluation,
- Dependability benchmarking,
- Software reliability and development process improvement (maturity models),
- Case studies

Language(s) of the course: English

Select Author(s): Karama Kanoun

Select Lecturer(s): Karama Kanoun

Submitted by: Karama Kanoun

Number of credits: 0

Total hours of lectures: 8

Total hours of labs: 2

Total hours of personal study: 0

Student Interaction Type:

Assessment methods:

Pre-requisites:

Infrastructure Required:

Course Objectives: Introduce software reliability analysis methods

Support courseware used:

Software Reliability: Basic Concepts and Assessment methods

Taught at: City University, London

Description: This half-day tutorial introduces the need for probabilistic measures of software dependability, and the basic principles, capabilities and limitations of the methods for assessing and predicting them. The topics covered include: the basic concepts of reliability; principles of statistically realistic testing; estimation of stable reliability; "reliability growth models" and ways for checking their trustworthiness; limits to reliability levels that can be effectively demonstrated before deployment of the software. The tutorial is designed for an audience with a software engineering background, with only knowledge of the basic concepts of probability. It is meant to give an understanding of the meaning and value of claims about software reliability, and of the practical methods available for predicting it.

Language(s) of the course: English

Select Author(s): Bev Littlewood, Lorenzo Strigini

Select Lecturer(s): Bev Littlewood, Lorenzo Strigini

Submitted by: Andrey Povyakalo

Number of credits: 0

Total hours of lectures: 4

Total hours of labs: 0

Total hours of personal study: 0

Student Interaction Type:

Lectures

Assessment methods:

Attendance

Pre-requisites:

Probability Theory

Software Engineering Fundamentals

Basic knowledge of computer architectures

Infrastructure Required: None

Course Objectives: The course objective is to give an understanding of the meaning and value of claims about software reliability, and of the practical methods available for predicting it.

Support courseware used:

Software Security

Taught at: Universidade de Lisboa
Description: The objective of the course is to give the students the mental tools necessary to understand the problem of the security of the computer, in spite of the security of the communication or distributed systems. The problems in the area are presented and solutions discussed.
Syllabus:
Security and software, the problem:
Buffer overflows, Race conditions, Trust and input;
General-purpose operating systems:
Design principles, Protection of general purpose OSs, Access control, User authentication, Root omnipotence, Rootkits, TCG;
Designing trusted operating systems:
Trusted OSs, Security policies, Security models, Designing trusted OSs, Assurance, Examples;
Applications security: Managing software risk, Selecting Technologies, Open source vs. Closed source, Auditing software, Cryptographic software, Trust management and input validation, Client side security
Language(s) of the course: Portuguese
Select Author(s): Miguel Correia
Select Lecturer(s): Miguel Correia
Submitted by: Miguel Correia
Number of credits: 6
Total hours of lectures: 26
Total hours of labs: 0
Total hours of personal study: 0
Student Interaction Type:
Individual Homework
Assessment methods:
Pre-requisites:
Basic Knowledge of Data Structures
Basic concepts of Dependable Computing
Basic understanding of Operating Systems concepts
Data Structures
Fundamentals of Operating Systems
Basic knowledge of computer architectures
Infrastructure Required:
Course Objectives: The objective of the course is to give the students the mental tools necessary to understand the problem of the security of the computer, in spite of

the security of the communication or distributed systems. The problems in the area are presented and solutions discussed.

Support courseware used:

Writing Secure Code, Second Edition

Building Secure Software: How to Avoid Security Problems the Right Way

Software Testing

Taught at: Technische Universitat Darmstadt
Description: Understanding and knowledge of a particular field of research in the area of software testing.
Language(s) of the course: English
Select Author(s): Neeraj Suri
Select Lecturer(s): Neeraj Suri
Submitted by: Brahim Ayari
Number of credits: 3
Total hours of lectures: 28
Total hours of labs: 0
Total hours of personal study: 28
Student Interaction Type:
Individual Presentation
Assessment methods:
Attendance
Oral Examination
Pre-requisites:
Infrastructure Required:
Course Objectives: Deepen understanding of a particular field of research. Practice reading and writing scientific text in English. Improved presentation skills.
Support courseware used:
http://resist.ecs.soton.ac.uk/courseware/resources/d41d8cd9

Freely Available: Yes

Copyright or Restrictions: Yes

Software Verification and Validation

Taught at: Budapest University of Technology and Economics
Description The lectures introduce the informal, semi-formal and formal techniques of verification during software development. The verification activities related to specification, architecture design, software module design, implementation, software and hardware integration are discussed. Emphasis is put on the basic formal verification techniques like model checking, equivalence checking and theorem proving. An introduction is provided to complexity reduction methods like symbolic verification and partial ordering reduction. New challenges of software testing like model based testing and OO testing are discussed. Typical tools supporting verification activities are also presented.
Language(s) of the course Hungarian
Select Author(s): Istvan Majzik
Select Lecturer(s): Istvan Majzik
Submitted by: Istvan Majzik
Number of credits: 5
Total hours of lectures: 60
Total hours of labs: 0
Total hours of personal study: 60
Student Interaction Type:
Individual Presentation
Lectures
Assessment methods:
Oral Examination
Pre-requisites:
Data Structures
Programming languages
Infrastructure Required: None
Course Objectives: The students will be familiar with the typical verification and validation activities in a well-defined software development process. They will obtain an understanding of the strengths and weaknesses of different techniques, including formal verification techniques, particularly in the face of challenges such as the development of dependable and safety-critical systems. They will be able to estimate how the verification and validation activities affect software quality and development cost.
Support courseware used: Lecture notes (in Hungarian)
Freely Available: Yes
Copyright or Restrictions: Yes

Description of Cost and Copyright: Budapest University of Technology and Economics, Dept. of Measurement and Information Systems

System Security (MSc Computer Security & Resilience)

Taught at: The University of Newcastle upon Tyne

Description: This course aims to introduce, largely with real-life case studies, how security technology has failed in the real world, and what lessons security engineers can learn from these failures to build robust secure systems. We examine not only security failures due to failure of technical mechanisms, but also those due to failure of usability and incentives.

Outline of syllabus (To inform module choice of current students):

1. Review of foundations: Computer security, Shannon theory, and modern cryptography
2. Security engineering methodology: threat model, security policy and protection mechanisms
3. Real-world security failure due to lack or failure of technical mechanisms
    a. Smashing the stack, worms, spywares, etc.
    b. Attacks on security protocols: replay, oracle, interleave, algebraic, cryptanalytic
    c. Misuse of cryptography
    d. Side channel attacks
    e. Attacks on physical security measures

4. Security failure due to usability
    a. ATM machine case: money or card first?
    b. “Why Johnny can’t encrypt”: user interface in security systems
    c. The Cambridge passwords experiment
    d. Differences between engineering expectation and user utilisation: what engineers expect to work and what users actually make to work are two different things
    e. Usable security: designing secure systems that people can use

5. Security failure due to failure of motivation
    a. Distributed denial of service, tragedy of the commons, and motivation failure
    b. Incentive-compatible security system design

Language(s) of the course: English

Select Author(s): Jeff Yan

Select Lecturer(s): Jeff Yan

Submitted by: John Fitzgerald

Number of credits: 5

Total hours of lectures: 24

Total hours of labs: 12

Total hours of personal study: 64

Student Interaction Type:
Individual Class Participation
Individual Laboratory Practice
Individual Project
Individual Homework
Lectures
Assessment methods:
Individual Coursework
Written Examination
Pre-requisites:
Basic Knowledge of Data Structures
Basic concepts of Dependable Computing
Basic knowledge of computer networking
Knowledge of distributed systems
Programming languages
Infrastructure Required:
Course Objectives:
Aims:

- To instil an appreciation of the need for security in distributed environments
- To impart an understanding of security requirements and goals as well as threats and risks
- To introduce the elements of system security: cryptographic algorithms, protocols and primitives and system security concepts and mechanisms.

Intended knowledge outcomes: Knowledge & understanding of:

- System vulnerabilities, and common attacks on security systems
- Importance of interface usability in robust secure systems
- Role of incentives in, and the ways in which incentives can be built into, secure systems
- Security engineering methods: threat model,ß security policy and protection mechanisms
- Trade-offs that needed to beß considered with any sensible security scheme

Intended skills outcomes: The ability to:

- Work out a threat model for a given scenario,
- Formulate a security policy,
- Design specific protection mechanisms to implement a security policy.

Support courseware used:
Cryptography: Theory and Practice (Discrete Mathematics and Its Applications)
Computer Security

Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition
Security Engineering: A Guide to Building Dependable Distributed Systems

System Validation (MSc Computer Security & Resilience)

Taught at: The University of Newcastle upon Tyne
Description: The inherent difficulty of validation of complex distributed computer systems, including Internet and Grid systems, is widely recognised as a major stumbling block in the development and implementation of modern networked applications. It is generally acknowledged that, ultimately, the only way of coping with this complexity problem is to use formal methods supported by computer aided validation tools. This module covers basic formal techniques and computer aided validation topics relevant to the design and validation of network software systems. In particular, automata and process algebra based frameworks are introduced and typical validation techniques discussed. An industrial strength protocol specification language and validation tool is used in the coursework part of the module.
Outline Syllabus:

- The role of formal techniques: supporting specification, computer aided verification and testing, in a protocol engineering system.
- Protocol specification languages.
- Frameworks for the verification of behavioural properties: automata based models and process algebra models.
- Verification techniques based on the state space exploration: coping with a combinatorial state space explosion.
- State-based verification techniques: coping with a combinatorial state space explosion.
- Other verification techniques: term rewriting and bisimulation.
- Computer aided verification of communication protocols.

Language(s) of the course: English
Select Author(s): Maciej Koutny
Select Lecturer(s):
Submitted by: John Fitzgerald
Number of credits: 5
Total hours of lectures: 24
Total hours of labs: 12
Total hours of personal study: 64
Student Interaction Type:
Individual Laboratory Practice
Lectures
Assessment methods:
Individual Coursework
Written Examination
Pre-requisites:

Infrastructure Required:

Course Objectives:

Aim:

To cover formal techniques and computer aided verification topics relevant to those involved in design and validation of network software systems.

Intended knowledge outcomes:

- Students will acquire the knowledge for understanding current and emerging validation technologies; and knowledge to choose appropriate cost-effective techniques for validation of distributed systems.

Intended skills outcomes:

- To learn to design and validate high-level prototypes of communication software.
- To become familiar with techniques and computer-aided model checking tool for distributed systems.
- To understand the ways in which formal methods can be used to support rigorous specification and validation of a design.

Support courseware used:

System and Network Security

Taught at: The University of Newcastle upon Tyne
Description:
Language(s) of the course: English
Select Author(s): Jeff Yan, Peter Ryan
Select Lecturer(s): Jeff Yan, Peter Ryan
Submitted by:
Number of credits: 10
Total hours of lectures: 24
Total hours of labs: 12
Total hours of personal study: 64
Student Interaction Type: Group Project
Assessment methods:
Individual Coursework
Written Examination
Pre-requisites:
Basic knowledge of computer networking
Basic understanding of Operating Systems concepts
Knowledge of distributed systems
Infrastructure Required: None
Course Objectives: To create awareness of the need for security in computer and communications systems, and to introduce some of the technical mechanisms by which security can be achieved.
Support courseware used: course introduction

The Challenge of Dependable Systems (MSc Computer Security & Resilience)

Taught at: The University of Newcastle upon Tyne

Description: This module will develop your understanding of the basic concepts that underpin the dependability of a computer-based system. What does it mean to say that a system is dependable? How can this be quantified? Does it cover the safety of users and society? What can be done about physical breakdown, design inadequacy, accidental misuse and deliberate attacks? We will examine hazard-based approaches to help eliminate or reduce risks, and defensive measures to protect against faults and ameliorate their consequences.

Scenarios based on actual system failures will be examined to both illustrate and motivate. The role of a "dependability case" as a way to provide assurance and, in some domains, gain regulatory certification will be considered. Finally, we will discuss how these approaches might be applied to the highly complex ubiquitous systems of the future.

Outline of Syllabus:

1.Components, systems, interfaces, environments

- trusted vs. trustworthy
- success vs. failure
- safe and secure and reliable

2.Flaws and weaknesses

- physical deterioration and design inadequacy
- accidental misuse and malicious attack

3.Concepts and terminology

- Humpty Dumpty, and all the King’s horses
- faults, errors, failures
- tolerance, resilience, reconfiguration

4.Hazards and accidents

- safety analysis and integrity levels
- safety culture, management and life-cycle
- risk management and ALARP

5.Dependability cases

- professional and legal aspects: the safety case and regulation
- arguments, evidence and goal structured notation (GSN)
- fault models and hypotheses

6.Developing critical software

- structure, firewalls, wrappers
- fault tolerance and self-healing
- interactive consistency

- design-fault tolerance
- testing in all its forms

Language(s) of the course: English
Select Author(s): Tom Anderson
Select Lecturer(s):
Submitted by: John Fitzgerald
Number of credits: 5
Total hours of lectures: 20
Total hours of labs: 10
Total hours of personal study: 70
Student Interaction Type:
Individual Class Participation
Individual Project
Individual Homework
Lectures
Assessment methods
Individual Coursework
Written Examination
Pre-requisites:
Infrastructure Required:
Course Objectives:
Aims:
To introduce the concepts and principles of dependable systems, including: the notions of security, safety and reliability, the flaws that undermine dependability, the analyses that expose weaknesses, the techniques that can impart resilience, and the arguments that engender trust in a system.
Intended knowledge outcomes:
- Knowledge of the broad range of dependability requirements
- Understanding of the various sources from which a loss of dependability can spring
- Knowledge of techniques by which dependability can be achieved in architecture, in design, and in operation
- Understanding of how dependability can be evaluated, assessed and advocated

Intended skills outcomes:
The ability to:
- Propose system structures and organisation for a dependable system solution
- Select and apply techniques
- for developing a dependable system
- to maintain dependable operation
- Present arguments in support of system dependability achievement

Support courseware used:

Software Fault Tolerance

Dependable Computing Systems: Paradigms, Performance Issues, and Applications (Wiley Series on Parallel and Distributed Computing)

Fault-Tolerant Computing: Theory and Techniques (Volume II)

Architecting Dependable Systems (Lecture Notes in Computer Science)

Concurrency in Dependable Computing

Safeware: System Safety and Computers, SPHIGS Software

Fault-Tolerant Computer System Design

Trustworthy Operating Systems

Taught at: Technische Universitat Darmstadt

Description: Important concepts and terminology Basics of resource management, Mutual exclusion, Process synchronization, Deadlock/Livelock.

Process

- and processor management
- Concepts: Process vs. Thread
- Scheduling

Memory management

- Memory management: data structures and strategies
- Virtual memory: concepts, problems and solutions

Distributed coordination

- Distributed operating systems
- Distributed resource sharing

Error recovery in operating systems

Testing for verification and validation

Operating system stability and security

Language(s) of the course: English

Select Author(s): Neeraj Suri

Select Lecturer(s): Neeraj Suri

Submitted by: Brahim Ayari

Number of credits: 7

Total hours of lectures: 28

Total hours of labs: 42

Total hours of personal study: 28

Student Interaction Type:

Individual Laboratory Practice

Lectures

Assessment methods:

Practical Examination

Written Examination

Pre-requisites: Basic Knowledge of Data Structures

Infrastructure Required: Linux

Course Objectives: Understanding of fundamental concepts of classical operating systems Design, structure and functionality of classical operating systems Understanding of the operating system's role as a resource manager Understanding the issues of distributed resource management Understanding of the issues involved in trusted operating systems (dependability and security)

Support courseware used:

http://resist.ecs.soton.ac.uk/courseware/resources/d41d8cd9

Freely Available: Yes

Copyright or Restrictions: Yes

Appendix B – Links to Universities and Organizations inside and outside ReSIST that deal with curricula at post-graduate level where resilience-related courses can be found.

In this appendix a non-comprehensive list is presented to links to Universities and Organizations inside and outside ReSIST that deal with curricula at post-graduate level where resilience-related courses can be found:

1) EWICS: http://www.ewics.org/docs/curricula-subgroup
2) Newcastle: http://www.ncl.ac.uk/postgraduate/taught/subjects/computing/courses/458, http://www.cs.ncl.ac.uk/pg/csr/, http://www.cs.ncl.ac.uk/degrees/pg/csr/2006CSR.html
3) Universitat Saarlandes: http://depend.cs.uni-sb.de/index.php?id=169
4) Aachen University: http://lufgi4.informatik.rwth-aachen.de/general/page/about.html
5) EPFL: http://dslab.epfl.ch/courses/pods/winter06-07/index.html (with downloadable slides)
6) Dresden Technische Universitat: http://wwwse.inf.tu-dresden.de/index.php?language=English&site=courses&file=index
7) Eurecom: http://www.eurecom.fr/teaching/engineering/curriculum.en.htm
8) Chalmers: http://www.chalmers.se/en/sections/education/masterprogrammes/programme-descriptions/secure-dependable http://www.cs.chalmers.se/Cs/Grundutb/Kurser/logcs/index.html
9) Darmstadt: http://www.deeds.informatik.tu-darmstadt.de/teaching/index.html
10) BUTE: http://www.inf.mit.bme.hu/FTSRG/Education/index.html
11) U. P. Madrid: http://lsd.ls.fi.upm.es/lsd/distribuidos.htm
12) Lancaster: http://www.comp.lancs.ac.uk/postgraduates/distsyseng.html
13) Vrie: http://www.cs.vu.nl/masters/compsys/program.html
14) Antwerp: http://www.pats.ua.ac.be/courses
15) Mannheim: http://pi1.informatik.uni-mannheim.de/index.php?pagecontent=site/Teaching.menu/Current%20Courses.menu/Principles%20of%20Dependable%20Systems.page
16) IBM Zurich: http://www.zurich.ibm.com/~cca/sft06/
17) LAAS: http://www.laas.fr/TSF/courses
18) U. Roma: http://www.dis.uniroma1.it/%7Ealberto/didattica/critto.html http://www.dis.uniroma1.it/%7Ebaldoni/SD.html
19) Cornell U.: http://www.cs.cornell.edu/ken/book/
20) FME-Formal Methods Europe: http://www.di.uminho.pt/FME-SoE/index.html
21) Indiana U. on FM: http://www.cs.indiana.edu/formal-methods-education/xxiec/statements/mjcg-slides.ps
22) U. Lugano: http://www.mdds.unisi.ch/
23) U. Konstanz: http://www.inf.uni-konstanz.de/disy/teaching/ss06/kryptographie/
24) U. P. Madrid: http://polaris.dit.upm.es/~jpuente/strl/guia.html
25) U. York: http://www.cs.york.ac.uk/MSc/Modules/rts.html
26) U. College London: http://www.cs.ucl.ac.uk/staff/J.Bowen/GS03/
27) U. Copenhagen: http://www.diku.dk/undervisning/2004f/202/
28) CMU: http://www.cs.cmu.edu/~emc/15-820A/
29) U. Twente: http://fmt.cs.utwente.nl/courses/systemvalidation/
30) MIT: http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www
31) George Mason U.: http://ise.gmu.edu/~pammann/637-sched.html
32) Usability Net: http://www.usabilitynet.org/usability/university.htm

also:
OpenSeminar: http://openseminar.org/se/modules/44/index/screen.do http://openseminar.org/se/modules/45/index/screen.do http://openseminar.org/se/courses/41/modules/206/index/screen.do http://www.intute.ac.uk/sciences/cgi-bin/browse.pl?limit=75&id=25926&type=%25&sort=record.title

Appendix C – Joint ReSIST/EWICS TC7 Workshop on Teaching Resilient Computing, Erlangen, May 2, 2007

On May 2, 2007, it was organized a Joint ReSIST/EWICS TC 7 Workshop to provide inputs to the work on WP3 on the development of the Curriculum on Resilient Computing. The Workshop was co-organised by Francesca Saglietti, Friedrich-Alexander-University Erlangen-Nuremberg, Germany, Luca Simoncini, University of Pisa, Italy (ReSIST) and Udo Voges, Institute for Applied Computer Science, Forschungszentrum Karlsruhe, Germany (EWICS TC7).
15 persons coming from different European Universities and Organizations discussed together about the guidelines for teaching Resilient Computing. The discussion during the entire day provided very useful comments and suggestions that have been incorporated into this Deliverable. The contributions presented in the Workshop (written texts and slide presentations) are available from the ReSIST web site at: http://www.resist-noe.org/ under the Events section.

The following table is the list of participants to the Joint EWICS-ReSIST Workshop on "Teaching Resilient Computing" held in Erlangen, Germany on May 2, 2007.

Name	Affiliation
Wolfgang Ahrendt	Chalmers University of Technology, Sweden
Robin Bloomfield	City University London, UK
Alessandro Fantechi	University of Florence, Italy
Vincenzo De Florio	University of Antwerp, Belgium
Maritta Heisel	University of Duisburg-Essen, Germany
Karama Kanoun	LAAS-CNRS, France
Jean-Claude Laprie	LAAS-CNRS, France
Norbert Oster	University of Erlangen, Germany
Francesca Saglietti	University of Erlangen, Germany
Erwin Schoitsch	Austrian Research Center, Austria
Luca Simoncini	University of Pisa, Italy
Neil Speirs	University of Newcastle upon Tyne, UK
Mark Alexander Sujan	University of Warwick, UK
Udo Voges	Forschungszentrum Karlsruhe, Germany
Zdzislaw Zurakowski	Private Contractor, Poland

.

Final Program

Joint Workshop on "Teaching Resilient Computing"

Erlangen, Germany - May 2, 2007

09.00 – 09.10 **Francesca Saglietti** University Erlangen-Nuremberg, Germany **Luca Simoncini** University of Pisa, Italy **Udo Voges** Forschungszentrum Karlsruhe, Germany **Opening, Welcome, Organisational remarks**		
09.10 – 09.30	**Udo Voges**	EWICS TC7 and its Subgroup Education and Training
09.30 - 09.50	**Luca Simoncini**	ReSIST NoE and its WP 3 on Training and Dissemination
09.50 – 10.55 **Session 1** (Chair: **Udo Voges**)		
09.50 – 10.15	**Erwin Schoitsch** Austrian Research Center – ARC, Austria	Overview on Training Needs, Challenges and Achievements towards "Dependable Embedded Systems Master Courses and Professional Industrial Training"
10.15 – 10.40	**Zdzislaw Zurakowski** Private Contractor, Poland	Education and Training Issues in Electric Power Industry
10.40 – 10.55	**Discussion**	
10.55 – 11.25 **Coffee break**		

11.25 – 12.30 **Session 2** (Chair: **Francesca Saglietti**)		
11.25 – 11.50	**Maritta Heisel** University of Duisburg-Essen, Germany	Software Development in the Fields of Embedded Systems, Safety, and Security
11.50 – 12.15	**Alessandro Fantechi** University of Florence, Italy	Model based Development and Verification of Software for Resilient Computing
12.15 – 12.30	**Discussion**	
12.30 – 13.45 **Lunch**		
13.45 – 14.50 **Session 3** (Chair: **Luca Simoncini**)		
13.45 – 14.10	**Francesca Saglietti** **Norbert Oster** University of Erlangen-Nuremberg, Germany	Teaching Software Reliability Engineering
14.10 – 14.35	**Jean-Claude Laprie** LAAS-CNRS, Toulouse, France	Teaching Dependability Fundamentals
14.35 – 14.50	**Discussion**	
14.50 – 15.15 **Tea break**		
15.15 – 16.20 **Session 4** (Chair: **Udo Voges**)		
15.15 – 15.40	**Vincenzo De Florio** University of Antwerp, Belgium	Teaching Resilient Computing in Antwerp: Report, Considerations, and Vision
15.40 – 16.05	**Wolfgang Ahrendt** Chalmers University of Technology, Sweden	Experiences with a Dedicated Master's Programme on Secure and Dependable Computer Systems
16.05 – 16.20	**Discussion**	
16.20 – 17.00 (Chairs: **Luca Simoncini, Udo Voges**) **General Discussion**		
17.00 **End of Workshop**