Software Engineering with Objects and Components

Massimo Felici

JCMB-1402	0131 650 5899
1BP-G04	0131 650 4408

mfelici@inf.ed.ac.uk

Course Organization

- SEOC course webpage

  http://www.inf.ed.ac.uk/teaching/courses/seoc/

- Mailing List

  seoc-students@inf.ed.ac.uk

- Newsgroup

  eduni.inf.course.seoc1

- SEOC CVS repositories

Course Organization

- **Course Textbook**

  UML, Second Edition, by Simon Bennet, John Skelton and Ken Lunn, Schaum's Outline Series, McGraw-Hill, 2005

- **Course Resources**

  Lecture Notes and References

- **Software**

  ArgoUML, Eclipse and Java

Course Organization

- **Tutorials** begin in **week 3**
  - Frequency: once a week
  - Maximum 12 people per tutorial group
- **Coursework**
  - in small teams (approx 3-4 people)
  - two deliverables equally weighted
  - **Deadlines**
    - 1st deliverable: **Monday, 29th October**
    - 2nd deliverable: **Friday, 30th November**
- **Assessment**
  - 25% coursework; 75% degree examination

What is Software Engineering?

Software Engineering *is an* ***engineering discipline*** *that is concerned with* ***all aspects of software production*** *from the early stages of system specification to maintaining the system after it has gone into use.*

Some Software Engineering Aspects

- Software Processes
- Software Process Models
- Software Engineering Methods
- Costs
- Software Attributes
- Tools
- Professional and Ethical Responsibilities

Why Software Fails?

- Complex causes (interactions) trigger software failures
- Software fails in context
- Some issues related to software engineering
    - Misunderstood requirements
    - Design issues
    - Mistakes in specification, design or implementation
    - Operational issues
- Faults, Errors and Failures

Faults, Errors and Failures

- Some **definitions**:
  - **Fault** – The adjudged or hypothesized cause of a an error is called a fault. A fault is **active** when it causes an error, otherwise it is **dormant**.
  - **Error** – The deviation from a **correct** service state is called an error. An error is the part of the total state of the system that may lead to its subsequent service failure.
  - **Failure** – A failure is an **event** that occurs when the delivered service deviates from correct service.
- **Warnings: different understandings of faults, errors and failures.**

Some “Famous” Software Failures

- Patriot Missile failure
  - Inaccurate calculation of the time since boot due to computer arithmetic errors.
  - Coding errors may effect overall software system behaviour.
- The Ariane 5 Launcher failure
  - the complete loss of guidance and altitude information 37 seconds after start of the main engine ignition sequence.
  - The loss of information was due to specification and design errors in the software of the inertial reference system.
  - The software that failed was reused from the Ariane 4 launch vehicle. The computation that resulted in overflow was not used by Ariane 5.
- The London Ambulance fiasco
- Therac 25 and other medical device failures
  - (Software) Reliability is different than (System) Safety

An Example: The Patriot Missile Failure

Accident Scenario: On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Dharan, Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people.

The Patriot Missile continued…

- **Fault** - Inaccurate calculation of the time since boot due to computer arithmetic errors.
- **Error** - The small chopping error, when multiplied by the large number giving the time in tenths of a second, lead to a significant error of 0.34 seconds.
- **Failure** – A Scud travels at about 1,676 meters per second, and so travels more than 500 meters in this time. This was far enough that the incoming Scud was outside the range gate that the Patriot tracked.

The Patriot Missile ...conclusions

- Identifying coding errors is very hard
  - seemingly insignificant errors result in major changes in behaviour
- Original fix suggested a change in procedures
  - reboot every 30 hours – impractical in operation
- Patriot is atypical
  - coding bugs rarely cause accidents alone
- Maintenance failure
  - failure of coding standards and traceability

Supporting Software Engineering Practices

- UML provides a range of **graphical notations** that capture various aspects of the engineering process
- Provides a common notation for various different facets of systems
- Provides the basis for a range of consistency checks, validation and verification procedures
- Provides a common set of languages and notations that are the basis for creating tools

Some UML diagrams

- Use Case Diagrams
- Class Diagrams
- Interaction Diagrams
  - Sequence and Communication Diagrams
- Activity Diagrams
- State Machines

Readings

- **UML course textbook**
  - Chapter 1 on the Introduction to the Case Studies.
  - Chapter 2 on the Background to UML
- B. Meyer. Software Engineering in the Academy. IEEE Computer, May 2001, pp. 28-35.
- R.N. Charette. Why Software Fails. IEEE Spectrum, pp. 42-49, September 2005.
- B. Nuseibeh. Ariane 5: Who Dunnit? IEEE Software, pp. 15-16, May/June 1997.
- J.-M. Jézéquel, B. Meyer. Design by Contract: The Lessons of Ariane. IEEE Computer, pp. 129-130, January 1997.
- M. Grottke, K.S. Trivedi. Fighting Bugs: Remove, Retry, Replicate, and Rejuvenate. IEEE Computer, pp. 107-109, February 2007.
- Rational Unified Process: Best Practice for Software Development Teams: Rational Software White Paper, TP026, Rev 11/01.

Suggested Readings

- I. Sommerville. Software Engineering, Eighth Edition, Addison-Wesley 2007.
  - Chapter 1 for a general account of Software Engineering
  - Chapter 3 on Critical Systems
  - Chapter 4 on Software Processes
- SWEBOK - Guide to the Software Engineering Body of Knowledge. 2004 Version, IEEE.
- A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing 1(1):11-33, January-March 2004.
- N.G. Leveson, C.S. Turner. An investigation of the Therac-25 accidents. IEEE Computer 26(7): 18-41, Jul 1993.
- D.R. Wallace, D.R. Kuhn. Lessons from 342 Medical Device Failures. In Proceedings of HASE 1999, pp. 123-131.
- G. Cernosek, E. Naiburg. The Value of Modeling. Rational Software, Copyright IBM Corporation 2004.

Summary

- SEOC organization
- An introduction to Software Engineering
- Why Software Fails
- Faults, Errors and Failures
- Examples of Software Failures
- An Outline of some UML diagrams
- Readings and Suggested Readings