

# Software Assurance

## ITI8610-Tarkvara töökindlus

Goals and content

2015 Autumn

# Why is this course needed?

- **EDS Fails Child Support (2004)**

Software giant EDS introduced in 2004 a complex IT system to the U.K.'s Child Support Agency. Same time, the Department for Work and Pensions restructured the agency. These actions were completely incompatible.

*Result - The system overpaid 1.9 million people, underpaid 700,000, had \$7 billion in uncollected child support payments, a backlog of 239,000 cases,*

*In total cost the UK taxpayers over \$1 billion to date.*



# LA Airport Flights Grounded (2007)

A single faulty piece of embedded software, on a network card, sends out faulty data on the U.S. Customs and Border Protection network, bringing the entire system to a halt. Nobody is able to leave or enter the U.S. from the LA Airport for over eight hours.

*Result - Over 17,000 planes grounded for the duration of the outage.*



# The Ariane 5 Launcher Failure

- A European rocket designed to launch commercial payloads, approximately 37 seconds after a successful lift-off, lost control.
- It started to break up due to uncontrollable stress.
- Ground controllers initiated self-destruct and the rocket and payload was destroyed.



# The Ariane 5 Launcher Failure

- Software failure occurred when an attempt to convert a 64-bit floating point number to a signed 16-bit integer caused the number to overflow. The lack of an associated exception handler lead to a software shutdown.
- Aggregate cost: \$640 million
- Loss of life: 160



# and many more...

- read

[https://en.wikipedia.org/wiki/List\\_of\\_software\\_bugs](https://en.wikipedia.org/wiki/List_of_software_bugs)

# Lessons learned

- A 2002 study commissioned by the National Institute of Standards and Technology found that software bugs cost the US economy **\$59.5 billion every year**
- The study estimated that more than a third of that amount, **\$22.2 billion**, could be eliminated by
  - *improved testing*,
  - and even more by **applying systematic analysis and development techniques.**

# What is software assurance?

Application of technologies and processes to achieve a required level of confidence that *software systems* and *services*

- function in the intended manner,
- are free from accidental or intentional vulnerabilities,
- provide security capabilities appropriate to the threat environment, and
- recover from intrusions and failures.



# Who needs it?

- Mostly professional software security and assurance practitioners
- But to some extent all who deal with IT and SE issues
- SW assurance is already part of many Master of software engineering curricula over world.
- Main reference curriculum issued by Software Engineering Institute, Carnegie Mellon Univ. U.S.

# Course differs from traditional SE and CS programs

- Special areas of emphasis:
  - Software and *services*,
  - their development and *acquisition*
  - *Security* and correct functionality: defective software isn't dependable or secure
  - *Software analytics*: the ability to analyze software to ensure that it has both the right security properties and the right functionality
  - *System operations*: monitor and assess to ensure that systems continue to have the right security properties in their operational environment
  - *Auditable evidence*: the ability to produce rigorous evidence of assurance processes and outcomes

# Expected outcomes<sup>1</sup> (1)

Graduates will have the following *abilities*:

- **In Assurance Process and Management**

- Assurance across life cycles: to incorporate assurance technologies and methods into life-cycle processes and development models for new or evolutionary system development, and for system or service acquisition
- Risk management: to perform risk analysis, trade-off assessment, and prioritization of security measures.
- Assurance assessment: to analyze and validate the effectiveness of assurance operations and create auditable evidence of security measures.
- Assurance management: to make a business case for software assurance, lead assurance efforts, understand standards, comply with regulations, plan for business continuity, and keep current in security technologies.

---

<sup>1</sup> - Mead et al., 2010a

# Expected outcomes (2)

- **Assurance Product and Technology**
  - System security assurance: to incorporate effective security technologies and methods into new and existing systems.
  - System functionality assurance: to verify new and existing system functionality for conformance to requirements and absence of malicious content.
  - System operational assurance: to monitor and assess system operational security and respond to new threats

# Context of the course



# Course organization

- Lectures Mon 14:00-15:30
- Practical training Fri. 10:00-11:30
- 3 modules:
  - 1. module (weeks 2-4, 10 and 16), Maili Markvardt
    - Assurance Across Life Cycles
    - Assurance Assessment
    - Assurance management
  - 2. module (weeks 5-9), Aleksandr Lenin
    - Risk Management
    - System Security Assurance
  - 3. module (weeks 11-15), Jüri Vain, Jishu Guin
    - System Functionality Assurance

# Requirements to pass

- *Total mark is arithmetic mean of marks collected from modules I – III*
- *Each module is evaluated separately on scale 0 – 100 points*
- *To pass a module at least 50 % points are required from max possible*
- *The points come from tests, lab assignments, home work, oral report, etc.*
- *Evaluation of results depends on the specifics of the assignment of module*

# Course materials and web

- Course web page
  - (Estonian) <https://cs.ttu.ee/kursused/>
  - (English) <https://cs.ttu.ee/courses/>
- Course Moodle
- [http://resources.sei.cmu.edu/asset\\_files/UsersGuide/2011\\_012\\_001\\_51607.pdf](http://resources.sei.cmu.edu/asset_files/UsersGuide/2011_012_001_51607.pdf)



# Module 1: Assurance processes & risk management



Assurance processes and requirements

SW Risk management

Tool considerations

# Learning outcomes for module I

- After completing module I, student is able to
  - Analyse and objectively describe SW system's context in terms of assurance
  - Document non-functional requirements in controlled manner as a basis for assurance
  - Conduct and document risk analysis depending on SW system's context
  - Implement tool support for SW assurance in an organisation



# Topics timeline & mandatory assignments

- Week 2: Requirements engineering considerations in assurance
  - Exercise on non-functional requirements
- Week 3: Risk management: Concept and preparations
  - Risk analysis preparations exercise for SW system
- Week 4: Risk management: Risk analysis
  - Risk analysis exercise for SW system
- Week 10: Tool support for analysis
  - Practical exercise in example of quality assurance management tool
  - Week 16: TBA



# Conditions

- All assignments are mandatory
  - Solving the assignment itself
  - Reviewing another group's solution
- My assignments are usually done in groups
- Deadline for each practice assignment is usually following week's Friday
  - Assignment & review – plan your timing with care
  - Don't be late if you have not agreed another deadline in advance!
- Submitted via e-learning environment



# Module II: Security Assurance

*IT – business enabler or a threat?*



# Module II: Security Assurance



# Module II: Security Assurance

The course is about:

- Designing reliable secure and trustworthy systems
- Critical/weak spots in software. Typical attack vectors
- Vulnerability and threat identification
- Security patterns, best practices, security testing
- Security vs usability
- Security fallacies

The course is **not** about:

- Designing security systems
- Cryptography
- Reliability, fault tolerance and safety

# Module II: Security Assurance

Practice hands-on assignments:

1. Threat identification, control and mitigation
2. Vulnerability identification
3. Secure software design

The structure of the course is subject to possible changes due to the audience background.

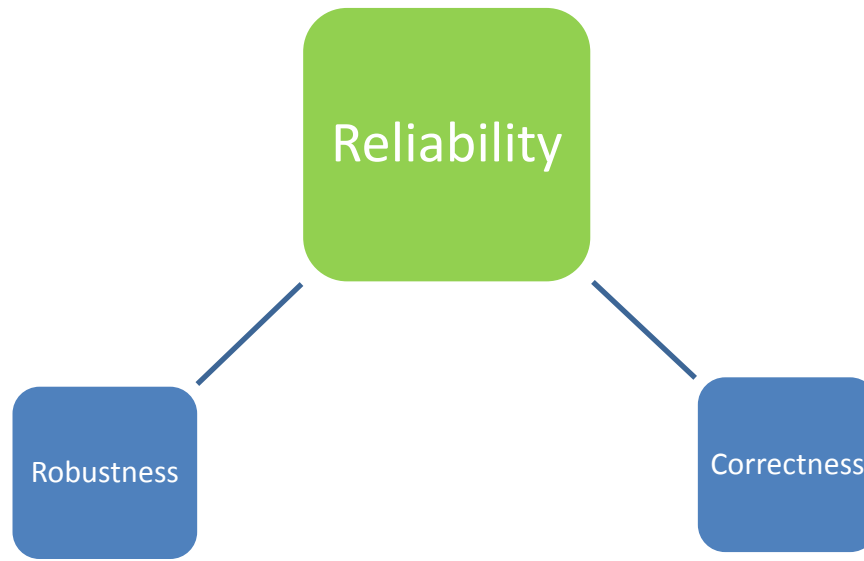
Alternatives are negotiable.



# Module III: Assured Software Analytics

## □ Focus area

- System functionality assurance



# Module III: Assured Software Analytics

## □ Method

- Design by contracts
  - Correctness - Adherence to contracts
  - Robustness - Handle violation of contracts



# Module III: Assured Software Analytics

## □ Practise Environment

- Contracts for Java (Cofoja) - A **contract programming** framework and **test tool** for Java, which uses annotation processing and bytecode instrumentation to provide run-time checking.

```
class RestrictedInteger {
    int x;

    @Ensures("x == y")
    @ThrowEnsures({ "IllegalArgumentException", "x == old (x)" })
    void setX(int y) throws IllegalArgumentException {
        ...
    }
}
```

# Module III: Assured Software Analytics

## □ Pre-requisites

- Basic knowledge of Java programming language.
- Familiarity with basic data structures (List, Map, etc.)



A word cloud of Java keywords and classes. The words are arranged in a roughly rectangular shape, with some words oriented vertically and others horizontally. The words include: try, private, HashMap, Exception, catch, class, main, else, ArrayList, public, static, and return. The colors of the words vary, including shades of brown, green, and grey.

# Module III: Assured Software Analytics

- After completing module III, student is able to
  - specify functional and non-functional requirements as contracts and use contracts in software processes and development increments
  - analyse and verify contracts
  - easily adapt to new contract based development tools.

# Module III: Assured Software Analytics

## ☐ Assignments

- The module has two assignment.
  - Short class assignment - 40% grade for Module III
  - Take home assignment - 60% grade for Module III



If bridges were built like software...



MEIL POLE AEGA EGA RAHA, ET KORRALIKULT TEHA  
AGA MEIL ON AEGA JA RAHA, ET ÜMBER TEHA

# Contacts

- Jüri Vain

[juri.vain@ttu.ee](mailto:juri.vain@ttu.ee)

- Maili Markvardt

[maili.markvardt@ttu.ee](mailto:maili.markvardt@ttu.ee)

- Aleksandr Lenin

[aleksandr.lenin@cyber.ee](mailto:aleksandr.lenin@cyber.ee)

- Jishu Guin

[jishu.guin@ttu.ee](mailto:jishu.guin@ttu.ee)