Prof. Dr. A. Podelski,
Dr. B. Westphal

Sommersemester 2017

Softwaretechnik/Software Engineering

http://swt.informatik.uni-freiburg.de/teaching/SS2017/swtvl

Exercise Sheet 0

Early submission: Wednesday, 2017-04-26, **18:00** Regular submission: Thursday, 2017-04-27, **12:00**

Exercise 1 – Are Software Engineers Managers? (5 + 5 Bonus)

Give 3 examples of management activities (as opposed to activities which produce product artefacts, such as programming, formalising requirements, conducting tests, etc.) that can be expected of people holding a B.Sc./M.Sc. degree in computer science and working in the software branch. That is, activities that require knowledge or competences typically acquired as part of a formal computer science degree.

For each example, discuss in what sense it is a management activity, and why it is specific to people with professional knowledge of computer science, i.e. whether it, in general, cannot be expected of, e.g., business administrators, psychologists, etc.

Note: Points will be awarded for your discussion of the example, just naming examples does not entitle you to any points. (5)

Originality Challenge* (5 Bonus)

Provide a fourth example and discuss it as indicated above. Make sure to clearly mark which example you would like to propose for the originality challenge.

Exercise 2 – Does Software Reliability Matter? (5 + 5 Bonus)

Consider the cases A–C presented in the appendix. Research a different, recent (that is, in the last 3 years) incident where a software problem caused a considerable damage and report it following the style of those cases.
Provide a general description of the case, followed by a more detailed description of the software-related issue and its consequences. Quantify the damages caused and argue why the case is relevant. Discuss in how far the incident (following official reports, or in your opinion) is related to issues with requirements, design, quality assurance, management, or usage under specified conditions. (5)

Originality Challenge* (5 Bonus)

You will be awarded bonus points for your answer following the rules of originality challenges.

*** Originality Challenge:** For this type of exercise, the points attained depend on how original the answer you provide is. Every different answer is given a total number of points. These points will be divided among the teams that provide the same answer (at your tutor's discretion). Concretely, the number of points you will obtain for a certain answer is $\lceil p/n \rceil$, where p is the number of points for the originality challenge exercise and n is the number of teams that provided the same answer.

Survey

1. Expectations (4 Bonus)

What do you expect from the course? More concretely: what knowledge, abilities or competences do you expect to obtain during the semester? Please elaborate on your answer, make sure it is clear what kind of achievements you expect and why you consider them important or relevant for your future career.

2. Previous Experience (5 Bonus)

Please rate the level of experience of the team members (no need to specify the names) in the following activities of software engineering according to the following scale:

- 0: I have no experience in that activity whatsoever. I have not taken any related subjects during my studies.
- 1: I have only performed the activity in the context of a lecture or programming course.
- 10: I have performed the activity in a project with a large user base (100+ users), a large work volume (36+ person-months) or a specific commercial purpose. I have been responsible for the planning and execution of the following activities in a software development project within defined resource and time constraints.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Project Management (cf. Exercise 1)											
Member 1											
Member 2											
Member 3											
Requirements Engineering (capturing and managing requirements from users or clients)											
Member 1											
Member 2											
Member 3											
Programming (writing code, fixing bugs)											
Member 1											
Member 2											
Member 3											
Design Modelling (creating an architecture or behavior model of a solution)											
Member 1											
Member 2											
Member 3											
Software Quality Assurance (e.g., testing, code review, formal verification)											
Member 1											
Member 2											
Member 3											

3. Regarding the Softwarepraktikum... (1 Bonus)

	Member 1	Member 2	Member 3
I will be participating in it this semester.	☐	☐	☐
I have already taken part.	☐	☐	☐
I will participate in it inthe following semesters.	☐	☐	☐
It is not part of my study plan.	☐	☐	☐

4. Language

What language would you prefer for your tutorial?

- ☐ German.
- ☐ I prefer German, but English is okay.
- ☐ I prefer English, but German is okay.
- ☐ English.

Note that, due to limited available places, there is no guarantee that your wish can be fulfilled.

Appendix

Consider the following cases where software failures caused important damage.

A. Therac-25 (1985-87)

The radiation therapy machine Therac-25 erroneously delivered extremely high radiation doses in at least six incidents between 1985 and 1987, causing serious injuries to the patients being treated.
The dangerous radiation overdoses occurred when the machine activated a high energy electron beam without appropriately having rotated four hardware components onto the path of the beam that would condition it to make it safe for patient treatment, thus allowing the electron beam to directly hit the patient and deliver a potentially lethal radiation dose, 100 times larger than the intended dose.
The error was caused by a race condition between the data entry routine, which communicated with a keyboard and a terminal screen to configure the machine, and the routine that monitors radiation treatment. If the operator changed the configuration using the terminal in less than 8 seconds while the machine was rotating a magnet plate into place, the changes entered would go undetected and a flag indicating that the configuration is complete was set, allowing the machine to continue the therapy sequence with the wrong parameters. After the initial incidents, the software was modified to fix the specific race condition; other problems of concurrent programming due to the poor design of the software allowed similar incidents to still occur.
The machine caused serious injuries to six patients, three of which later died from complications of the radiaton burns. The damage caused by the software error has both moral and legal implications: human life was lost as a result of poor software development practices. Furthermore, the company who developed the machine was faced with several lawsuits and subsequent economic losses derived from settling the suits in and out of court.
Source: `http://www.cs.umd.edu/class/spring2003/cmsc838p/Misc/therac.pdf`

B. Ariane-5 Rocket (1996)

On June 4th, 1996, the first flight of an Ariane-5 rocket from the European Space Agency (ESA) was scheduled to launch a group of satellites (the *Cluster* spacecraft) into orbit to study the earth's magnetosphere. Thirty-seven seconds after liftoff, the rocket veered off its programmed flight path, broke up and exploded.
The incident was caused by the adoption of a software module for the inertial control system from the rocket's previous generation. The newer rocket was subject to a greater horizontal acceleration, which caused an overflow when converting a 64-bit floating point value into a 16-bit signed integer value, thus triggering a hardware exception, which in turned caused the inertial reference system to enter a failure state and stop providing valid attitude information.
The approximate damage caused by this error amounts to approximately 1 billion US$, contributed by the taxpayers of the states participating in ESA, plus the delay and cost increase of the scientific mission on board.
Source: `http://esamultimedia.esa.int/docs/esa-x-1819eng.pdf`

C. Toll Collect (2003-06)

The German highway toll collection system for heavy traffic "Toll Collect" was scheduled to enter service in August of 2003. The consortium in charge of developing the system failed to meet the deadline. The system only entered its initial reduced load operation in January, 2005, delayed almost a year and a half. The system operates in full functionality since January, 2006.
The project planning, that involved around twelve different development partners, at twelve different locations was faulty. No standards for the interaction between the software modules were set, ranging from the access to databases to the graphical user interfaces. This situation made integration tests very difficult, because the developed software modules were, in some cases, incompatible.
The damage caused by the faulty project extends from the developing firms, who incurred contract penalties in the hundreds of millions of euros, to the taxpayers and the state, who provided the resources for the development, and missed an undetermined amount of revenue in toll fees that may be well into the thousands of millions of euros.
Source: `http://www.sueddeutsche.de/wirtschaft/chronik-der-autobahn-maut-pleiten-pech-und-pannen-1.508682`