Software Engineering

Dr. Michael Eichberg

Wintersemester 2018-2019

Contents

1 What is Software?

There are differing definitions of the word *software*, but in general it is more than just the code. It is understood to also include the documentation and configuration.

1.1 Types of Software

One can differentiate between two general groups of software.

Generic products This includes anything that you would call *shrink-wrapped software*. Examples are Microsoft Office, Open Office, Photoshop, etc.

Customised products These are products that are customised for a particular purpose. Examples of this would be TUCaN (*TU Campus Network*), an air traffic control system, etc.

1.2 Properties of Software

Software has some unique properties compared to hardware. For one thing, software does not physically exist, so it's basically free to reproduce. It also doesn't wear out or need to be replaced. It does, however, need to be *maintained*, meaning that it needs to be updated to cope with changing environments, otherwise it will become obsolete. As such, software is also incredibly hard to measure. How could one define (and quantify?) the quality of software? Are certain properties (amount of lines of code, amount of comments) correlated to the quality? How can progress be measured?

2 What is Software Engineering?

The term comes from the sixties and is often attributed to F.L. Bauer. It describes the discipline of systematically developing software following engineering principles. This is opposed to an artistic way of software development.

2.1 Areas of Software Design

Software Requirements The requirements define what the systems is expected to do.

Software Design How the system is designed.

Software Testing The systematic identification (and elimination) of errors.

Software Maintenance

Software Configuration Management The management of different versions and configuration of a software.

Software Engineering Process Definition and improvement of software development processes.

Software Engineering Tools and Methods

Software Quality

2.2 Why do Software Projects Fail?

Software development can fail for a large number of reasons. Often, the requirements and system dependencies are not well-defined, leading to more effort begin put in. Since changing the requirements during the development is much, much easier for software than for hardware, sometimes software has to accommodate for issues with the hardware. Lack of tools, methods, education and planning also often contribute to failed projects.

> **Note**
> One curious reason for the failure of software projects is *overbudgeting*. When a project needs to be finished quickly, one common idea is to add more people to the team. But in effect, while this can increase the productivity, it also greatly increases the communication overhead. Frederick Brooks discusses this in detail in his famous book *The Mythical Man-Month*.

2.3 Takeaway

Engineering software is hard; this lecture teaches you why and (to some extent) how to tackle common problems. Software engineering is about designing software and not about building software.