Software Quality
Software Quality

• **Software Quality Factors**
  B. Meyer; Object-oriented software construction; Prentice Hall, 1997
We distinguish between internal and external software quality factors.

- The internal quality factors can only be perceived by computer professionals.
- The external quality factors are ultimately the relevant ones, as they are perceived by the user. However, the external quality factors depend on the internal quality factors.
We distinguish between **internal** and **external** software quality factors.

- Internal quality factors
  - modular
  - readable

This lecture series’ main subject
An example of missing internal quality.

```csharp
/// <summary>
/// Turns true into false and false into true
/// <param name="_booInpt">True of false</param>
/// <returns>False or true</returns>
private bool trueandorfals(bool _booInpt)
{
    // I'm quite sure though there is a very
    // clever C# standard command doing this,
    // I just can't find it right now ...
    if (_booInpt == true)
        return false;
    return true;
}
```
/**
 * Checks to see if Australia is typed into the other country box
 */
function checkContactCountry(inputBox)
{
    |(N|n)(E|e)(W|w) (Z|z)(E|e)(A|a)(L|l)(A|a)(N|n)(D|d)$/);

    if(validator.test(inputBox.value))
    {
        alert("Your Residential Address must be outside Australia. "
            + "Enter your residential address outside this country,"
            + "or visit redacted-travel.com.au to make a booking if "
            + "you live in Australia.");
        inputBox.focus();
        inputBox.select();
    }
}
[...] Have you ever noticed that when someone checks in some complex and, oftentimes, horrific piece of code, the check-in is greeted with an almost deafening silence? [...] 

The explanation for why this occurs was first given by C. Northcote Parkinson [...] He stated that if you were building something complex, then few people would argue with you because few people could understand what you were doing. If you were building something simple [...] which most anyone could build, then everyone would have an opinion.

George V. Neville-Neil

Painting the Bike Shed - A sure-fire technique for ending pointless coding debates; ACM Queue, ACM 2009 1542-7730/09/0600
If you want to study code with missing quality...

But, reading other people’s code - in particular if the code is good - is one of the best ways to learn to program.
If you want to study code...
Software quality in commercial software.

Part of the source code for Comanche, build 055. It is part of the source code for the Command Module's (CM) Apollo Guidance Computer (AGC), Apollo 11.

```plaintext
VXSC  PDVL
-KVSSCALE  # KVSSCALE = .81491944
UNITW  # FULL UNIT VECTOR

VXV  VXSC
UNITR  # VREL = V - WE*R
KWE

VAD  STADR
STORE  -VREL  # SAVE FOR ENTRY GUIDANCE.  REF COORDS

UNIT  LXA,1
36D  # ABVAL( -VREL) TO X1
STORE  UXA/2  # -UVREL  REF COORDS

VXV  VCOMP
UNITR  # .5 UNIT  REF COORDS

UNIT  SSP  # THE FOLLOWING IS TO PROVIDE A STABLE
S1  # UN FOR THE END OF THE TERMINAL PHASE.

DEC  .019405  # 1000/ 2 VS
TIX,1  VLOAD  # IF V-VQUIT POS, BRANCH.
CM/POSE2  # SAME UYA IN OLDUYA
SPVQUIT  # OTHERWISE GO TO
```

We distinguish between internal and external software quality factors.

- Correctness
- Robustness
- Extendibility
- Reusability
- Compatibility
- Efficiency
- Portability
- Ease of use
- Functionality
- ...
**Correctness** is the ability of software products to perform their tasks as defined by their specification.

- To achieve correctness a precise requirements definition is needed.
- Correctness is usually only conditional - we guarantee the correctness of our program on the assumption that the lower layers - upon which our product is built - are correct (E.g. we assume that a processor calculates correctly, that the compiler compiles our program correctly, ...).
Robustness is the ability of software systems to react appropriately to abnormal conditions.

- Robustness characterizes what happens outside of the specification
- Robustness complements correctness
**Extendibility** characterizes the ease of adapting software products to changes of specification.

- Important principles to achieve extendibility:
  - Design simplicity
    A simple architecture is easier to adapt.
  - Decentralization
    Autonomous modules (modules which have minimal coupling to other modules → Software Engineering Design & Construction) are easier to change.

Change is pervasive in software development.
• **Reusability** is the ability of software elements to serve for the construction of many different applications

• **Compatibility** is the ease of combining software elements with others

• **Portability** characterizes the ease of transferring software products to various hardware and software environments
**Efficiency** is the ability of a software system to place as few demands as possible on hardware resources.

- Resources are the processor time, the space occupied in internal and external memories, the bandwidth used in communication devices, ....

- Always try to use “good” algorithms over “bad” ones, because a computer that is twice as fast as a previous model can handle problem sizes near $2^N$ if the algorithm’s complexity is $O(n)$. Do ask yourself: **If the complexity is $O(2^n)$ a computer that is twice as fast can handle problems of size?**

Do not worry how fast it is unless it is also right! Efficiency nearly always have to be balanced with other goals.
**Functionality** characterizes the extent of possibilities provided by a system.

- Avoid featurism; remain consistent with existing features if you add new ones
Ease of Use is the ease with which people of various backgrounds and qualifications can learn to use software products and apply them to solve problems.
Software Quality

- Good Software
  Ian Sommerville; Software Engineering - Eighth Edition; Addison Wesley, 2007
• **Maintainability**
  Software should be written in such a way that it may evolve to meet changing needs of customers.

• **Efficiency**
  Software should not waste system resources; it includes: responsiveness, processing time, memory utilisation, etc.

• **Usability**
  Software must be usable by the intended users.

• **Dependability** (dt. **Verlässlichkeit**)
  Dependable software does not cause physical or economic damage in the event of system failure. Further properties: Repairability, Survivability, Error Tolerance...
Some Aspects of Dependable Systems

Availability = dt. Verfügbarkeit
Reliability = dt. Zuverlässigkeit
Safety = dt. Betriebssicherheit
Security = dt. Systemsicherheit

Dependability

- Availability
  - The ability of the system to deliver services when required
- Reliability
  - The ability of the system to deliver services as specified
- Safety
  - The ability of the system to operate without catastrophic failure
- Security
  - The ability to protect itself against accidental or deliberate intrusion

This includes:

- correctness
- precision
- timeliness
Software Quality

• ... or the lack thereof.
Software failures can be disastrous.

- **Therac-25**
  People died due to an overdose of radiation (1985)

- **Ariane 5**
  A system from Ariane 4 was reused but the specification was ignored (1996)

- **Mars Climate Orbiter**
  There was some confusion about the units (i.e. metric system or english system) that are used (1999).

- ...
Software failures can be disastrous.

- *hessische Schulsoftware LUSD* “just” unusable (2007)

  - ...
Lack of software quality.

- CampusNet error message shown to the end user (2010)

**Magic uniPaaS Partitioning Message**

Error: "The Requester could not connect to the Enterprise Server" (-109)

<table>
<thead>
<tr>
<th>Enterprise Server</th>
<th>cmapp1.cn.pvww.tu-darmstadt.de/3300</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>CampusNet</td>
</tr>
<tr>
<td>Program</td>
<td>ACTION</td>
</tr>
</tbody>
</table>
Error creating task - java.io.IOException: StoreElement release error -
de.espirit.firstspirit.server.storemanagement.ReleaseFailedException: page 'index_17'
(ID=722542) of pageref 'tatjana_korbmacher' (ID=722547) is never released
Error creating task - java.io.IOException: StoreElement release error - de.espirit.firstspirit.server.storemanagement.ReleaseFailedException: page 'index_17' (ID=722542) of pageref 'tatjana_korbmacher' (ID=722547) is never released

java.io.IOException: StoreElement release error - de.espirit.firstspirit.server.storemanagement.ReleaseFailedException: page 'index_3' (ID=722534) of pageref 'roman_knoell' (ID=722541) is never released
at de.espirit.firstspirit.server.taskmanagement.TaskImpl.doTransition(TaskImpl.java:988)
at de.espirit.firstspirit.server.taskmanagement.TaskImpl.doTransition(TaskImpl.java:811)
at de.espirit.firstspirit.client.action.WorkflowAction.startWorkflow(WorkflowAction.java:446)
at de.espirit.firstspirit.client.action.WorkflowAction.startWorkflow(WorkflowAction.java:74)
at de.espirit.firstspirit.client.action.WorkflowAction.startWorkflow(WorkflowAction.java:80)
at de.espirit.firstspirit.client.gui.workflow.WorkflowPopupSWFStartAction.actionPerformed(WorkflowPopup.java:413)
at javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2351)
at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:387)
at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:242)
at javax.swing.AbstractButton.doClick(AbstractButton.java:389)
at javax.swing.plaf.basic.BasicMenuPopupUI.doClick(BasicMenuPopupUI.java:1220)
at apple.laf.CUIMenuItem.doClick(CUIMenuItem.java:119)
at
Missing software quality in commercial software.


...
Missing software quality in commercial software.

- Lufthansa Buchungssystem “Totalausfall” (2009)

...
Missing software quality in commercial software.
Missing software quality in commercial software.
Does Distributed Development Affect Software Quality?
An Empirical Case Study of Windows Vista

Christian Bird¹, Nachiappan Nagappan², Premkumar Devanbu¹, Harald Gall³, Brendan Murphy²

¹University of California, Davis, USA
²Microsoft Research
³University of Zurich, Switzerland

{cabird,ptdevanbu}@ucdavis.edu {nachin,bmurphy}@microsoft.com gall@ifi.uzh.ch
Browser Security: Lessons from Google Chrome

Google Chrome developers focused on three key problems to shield the browser from attacks.

Charles Reis, Google; Adam Barth, UC Berkeley; Carlos Pizano, Google

The Web has become one of the primary ways people interact with their computers, connecting people with a diverse landscape of content, services, and applications. Users can find new and interesting content on the Web easily, but this presents a security challenge: malicious Web-site operators can attack users through their Web browsers. Browsers face the challenge of keeping their users safe while providing a rich platform for Web applications.

Browsers are an appealing target for attackers because they have a large and complex trusted computing base with a wide network-visible interface. Historically, every browser at some point has contained a bug that let a malicious Web-site operator circumvent the browser’s security policy and compromise the user’s computer. Even after these vulnerabilities are patched, many users continue to run older, vulnerable versions. When these users visit malicious Web sites, they run the risk of having their computers compromised.

Generally speaking, the danger posed to users comes from three factors, and browser vendors can help keep their users safe by addressing each of these factors:

• **The severity of vulnerabilities.** By sandboxing their rendering engine, browsers can reduce the severity of vulnerabilities. Sandboxes limit the damage that can be caused by an attacker who exploits a vulnerability in the rendering engine.

• **The window of vulnerability.** Browsers can reduce this window by improving the user experience...
Software Quality

• Summary
Recall the “fifteen principles of Software Engineering”.

Take responsibility!

There are no excuses. If you develop a system, it is your responsibility to do it right. Take that responsibility. Do it right, or don’t do it at all.
The goal of this lecture is to enable you to systematically carry out small(er) software projects that produce quality software.

Software quality is not just about the (internal) quality of the source code. Software quality means different things to different stake holders. To produce quality software a holistic view on a software project is required.
The goal of this lecture is to enable you to systematically carry out small(er) commercial or open-source projects.