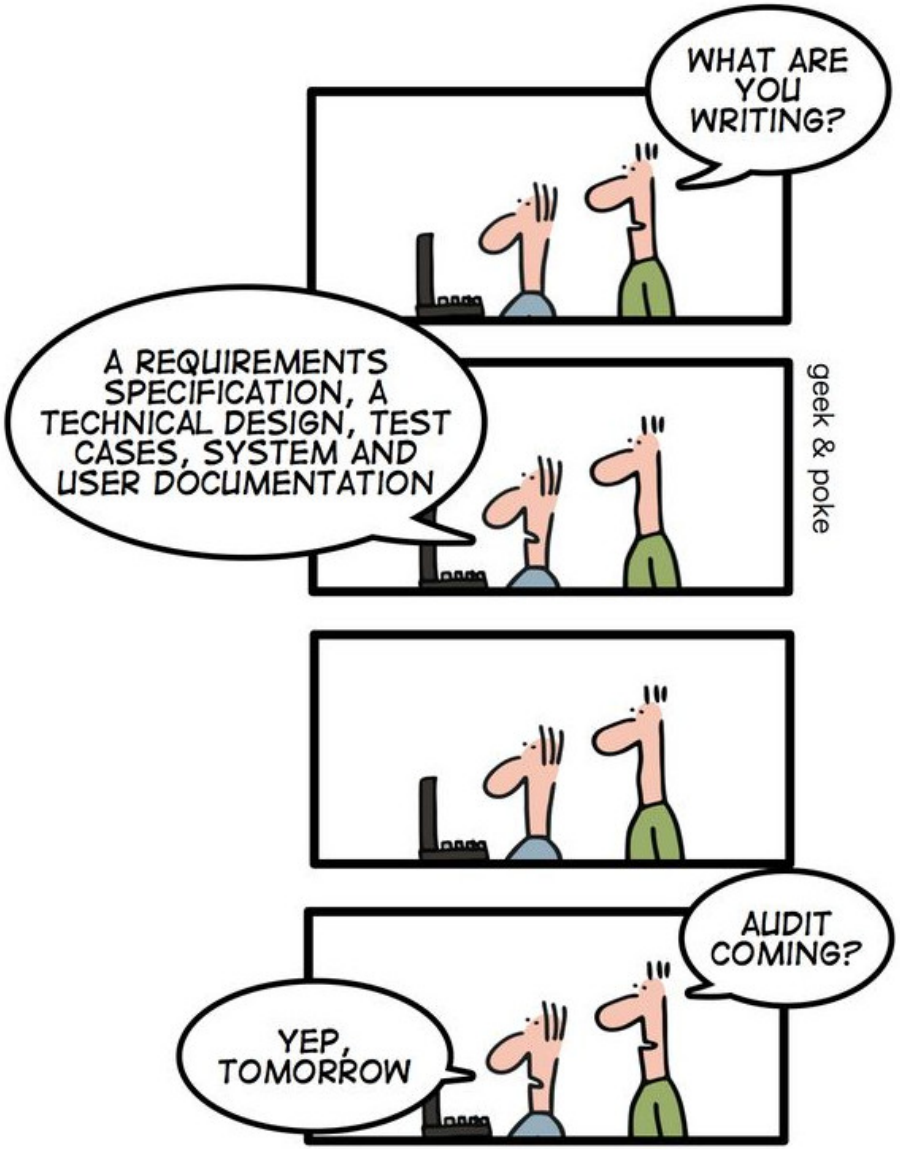
Software Quality and Sustainability Guidelines

CLARIAH

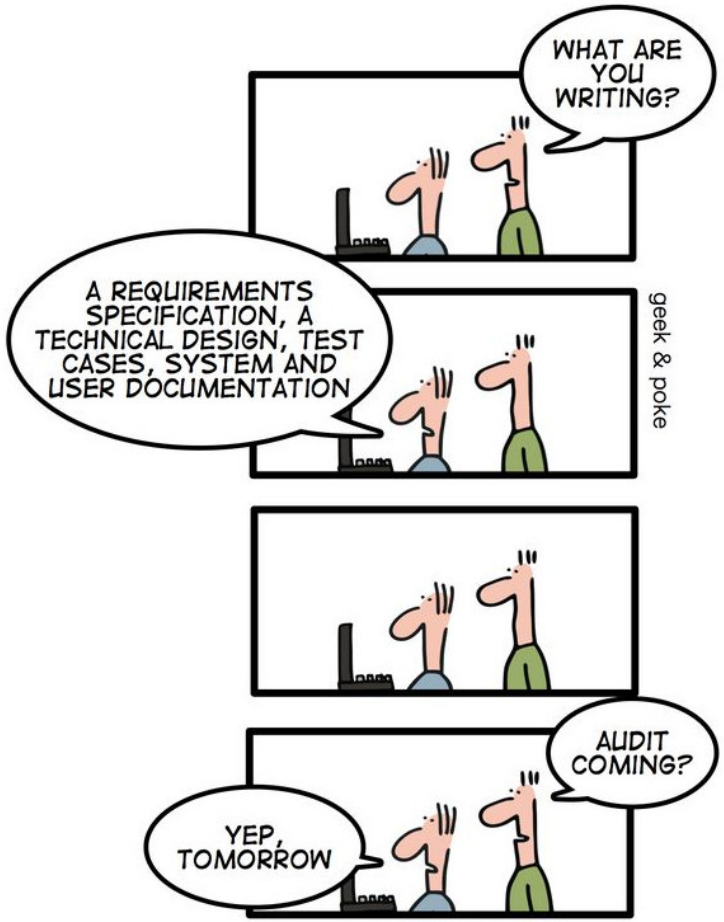
Maarten van Gompel (RU), Reinier de Valk (DANS), Jauco Noordzij (Huygens), Andrea Scharnhorst (DANS)

October 7th, 2016

Introduction



Introduction



Software quality and sustainability should be an **essential component** of the development process rather than an afterthought.

Introduction

Software plays an essential role

- Development of advanced sofware is a core activity within CLARIAH.
- The CLARIAH infrastructure consists of numerous interconnected software components.
- The success of the whole depends on the success of its parts

Questions addressed

- How do we ensure software is of good quality?
- ... How to measure this?
- ... What quality standards do we aim for?
- How do we ensure software is sustainable towards the future?

Introduction

Context

- DANS & eScience-NL report (Doorn et al., 2016):
  - Research software is a **fundamental component** of modern research.
  - Software in academia **lags behind** commercial software.
  - Pressure is on publication of new results: software is fragile and not sustainable or usable beyond project lifetime.
- Research Software Sustainability: Report on Knowledge Exchange workshop
  - Software is not data: software **must adapt** to the constant changes in its environment or will decay.
- Software Sustainability Institute (UK): `http://software.ac.uk`

Goals

1. **Improve** software quality & sustainability
2. **Emphasise** the importance of good software development in academia
3. **Provide practical guidelines** that allow developers and adopters to assess software on their own.

Form

- A **set of specific assessment criteria** grouped into various **dimensions**
- A web-based survey anybody can use to assess software: `http://softwarequality.clariah.nl`
- A list of minimal requirements for developers that offers a different perspective on the criteria.

Usage of the guidelines

How to use the guidelines?

For developers..

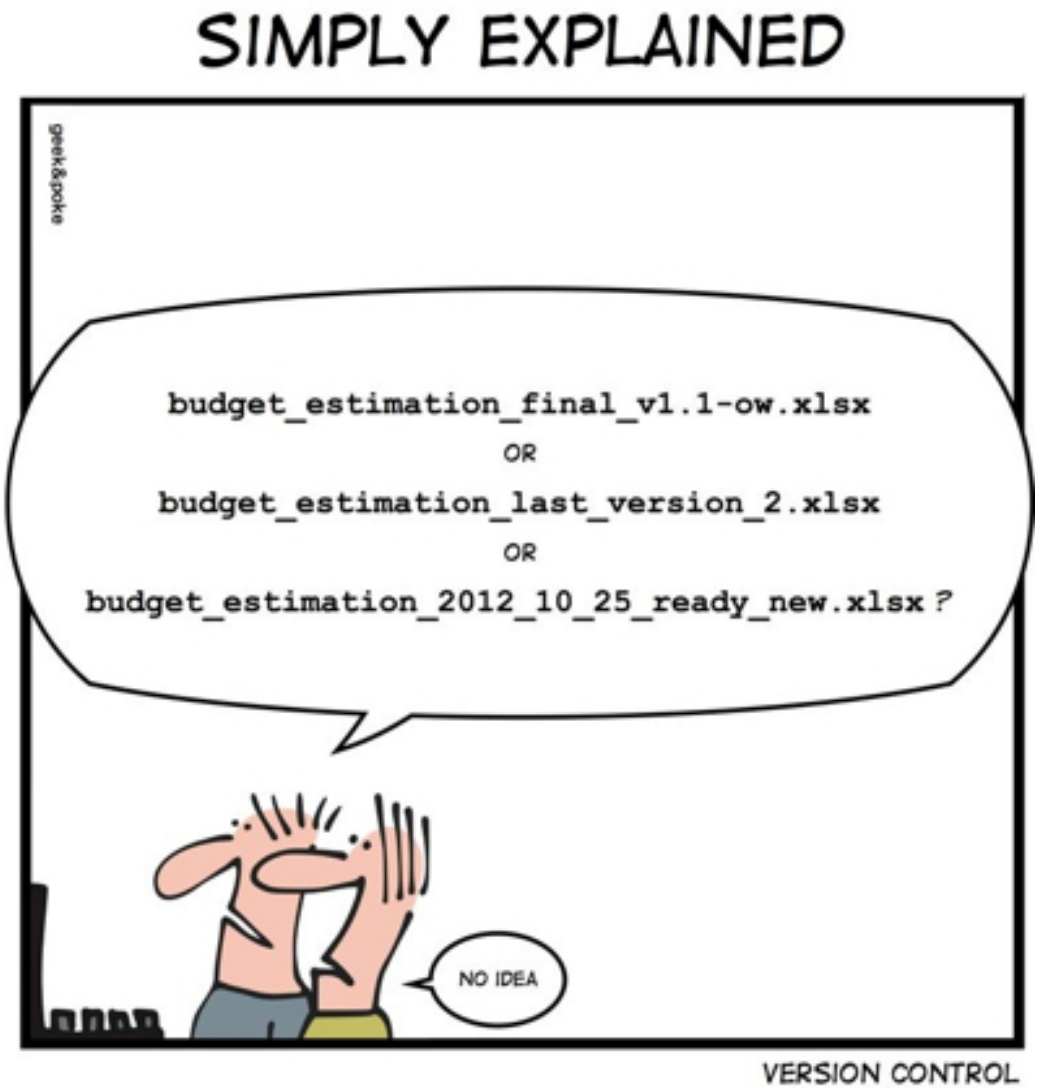
- self-assessment using the web-survey (`http://softwarequality.clariah.nl`)
- add the survey results to your VC repository so others can see it
- provide us with feedback through our issue tracker! (`https://github.com/CLARIAH/software-quality-guidelines/issues`)

Future

- Revision of guidelines based on feedback after usage
- Incorporation of CLARIAH-specific interoperability criteria
- A software seal of approval?

Highlights

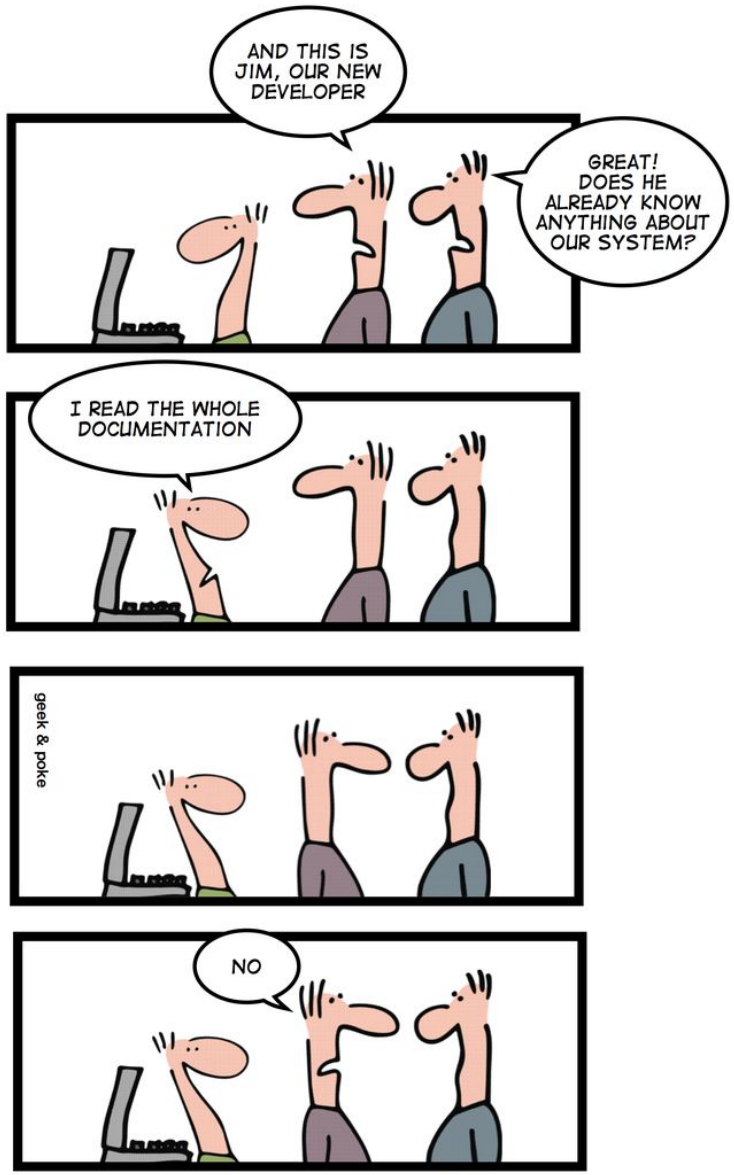
Accessibility: Version Control



Accessibility: Version Control

- Always use **version control** for any software project
- Use **public** version control (e.g. Github) whenever possible
  - increases project visibility/findability
  - encourages contribution and interaction.
  - provides infrastructure with issue trackers (bug reports), release management facilities
  - transparency: allows users to assess the project's activity
  - `https://github.com/CLARIAH`
- Periodically publish releases of your software, using a sane version numbering scheme, accompanied with release notes or a changelog.

Documentation



Documentation

- Have a good and comprehensive **README** (plain-text) in your VC repository.
  - **Tip:** Use Markdown or ReStructureText for mark-up
- Document how to **build and install** the software
- Address **all the intended audiences** at the **appropriate levels**.
- Documentation should **live alongside the code** in the VC repository
- API documentation should be **automatically generated** from documentation comments in the source code *(Doxygen, sphinx, javadoc)*
- Documentation should be up to date with the latest version.
- **Tip:** Have documentation auto-regenerate on every commit to your VC repo, using services like https://readthedocs.io

Buildability & Installability

INSTALL.SH

```
#!/bin/bash

pip install "$1" &
easy_install "$1" &
brew install "$1" &
npm install "$1" &
yum install "$1" & dnf install "$1" &
docker run "$1" &
pkg install "$1" &
apt-get install "$1" &
sudo apt-get install "$1" &
steamcmd +app_update "$1" validate &
git clone https://github.com/"$1"/"$1" &
cd "$1";./configure;make;make install &
curl "$1" | bash &
```

Buildability & Installability

- There should be an automated build and install procedure for the software *and all dependencies* on the target platform(s). Preferably through **one command**.
- Use established build tools (GNU Autotools, CMake, Maven, Ant)
- Package and upload to the programming language's infrastructure to facilitate installation:
  - Python - Python Package Index (http://pypi.org)
  - Java - Maven (http://maven.org)
  - Ruby - Gems (http://rubygems.org)
  - Perl - CPAN (http://cpan.net)
  - Javascript - npm
- or ...

Buildability & Installability

- Package and upload to the platform's repository to facilitate installation:
  - Build packages for Linux distributions (DEB packages for Debian/Ubuntu packages, Arch User Repository (AUR), RPM for Fedora/CentOS/RHeL, etc)
  - Consider Homebrew or macports for Mac, or the Mac App store.
  - Apple's App store (iOS) Google's Play (Android) store for mobile apps.
- Last resort only: statically linked download archive from a project website
- Additionally, consider container or VM solutions (e.g, Docker, Vagrant, Ansible) for complex setups.

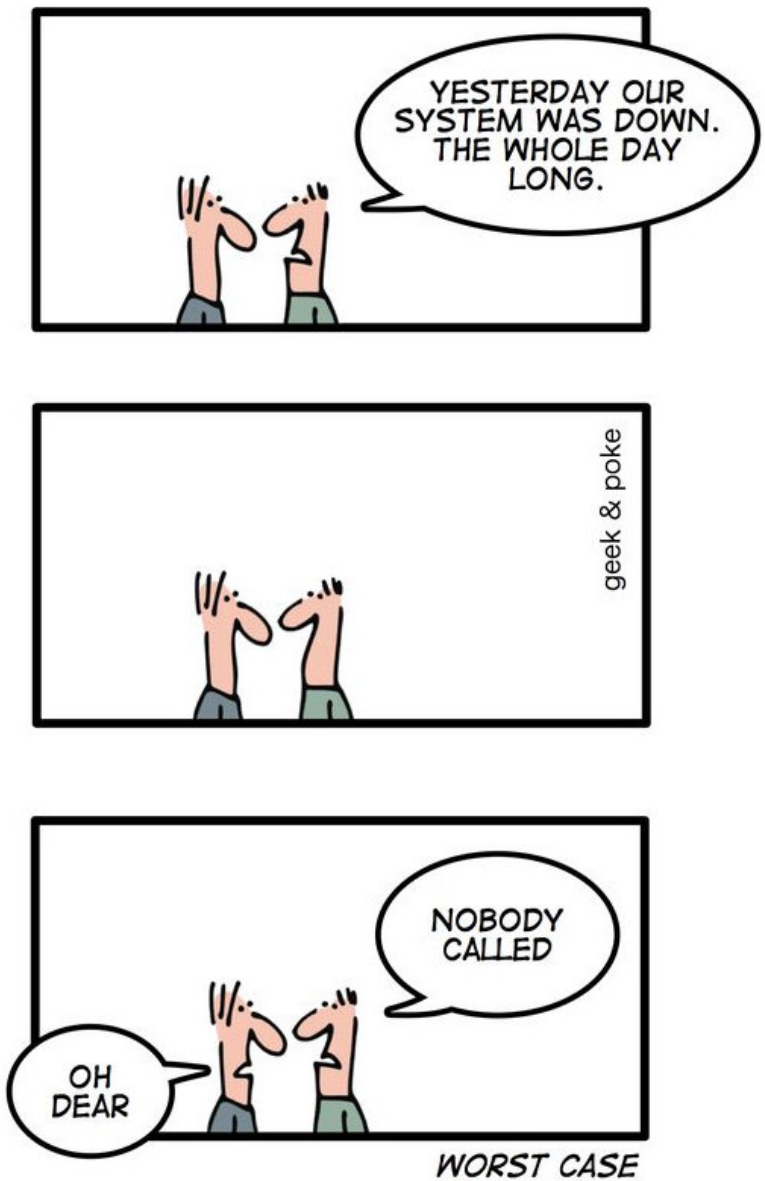
Testability

Testability

Testability

- If you didn't test it, it most likely doesn't work.
- Write automated tests (unit & integration tests) for your software.
- Ensure automated tests cover all enough of your software.
- Use continuous integration platforms such as Travis-CI (https://travis-ci.org), Jenkins (http://jenkins.io) or others.
  - These automatically run tests each time you commit code to your VC repo.

Community, Changeability & Supportability



Community, Changeability & Supportability

Community

- Is the software used? Do you have usage statistics?

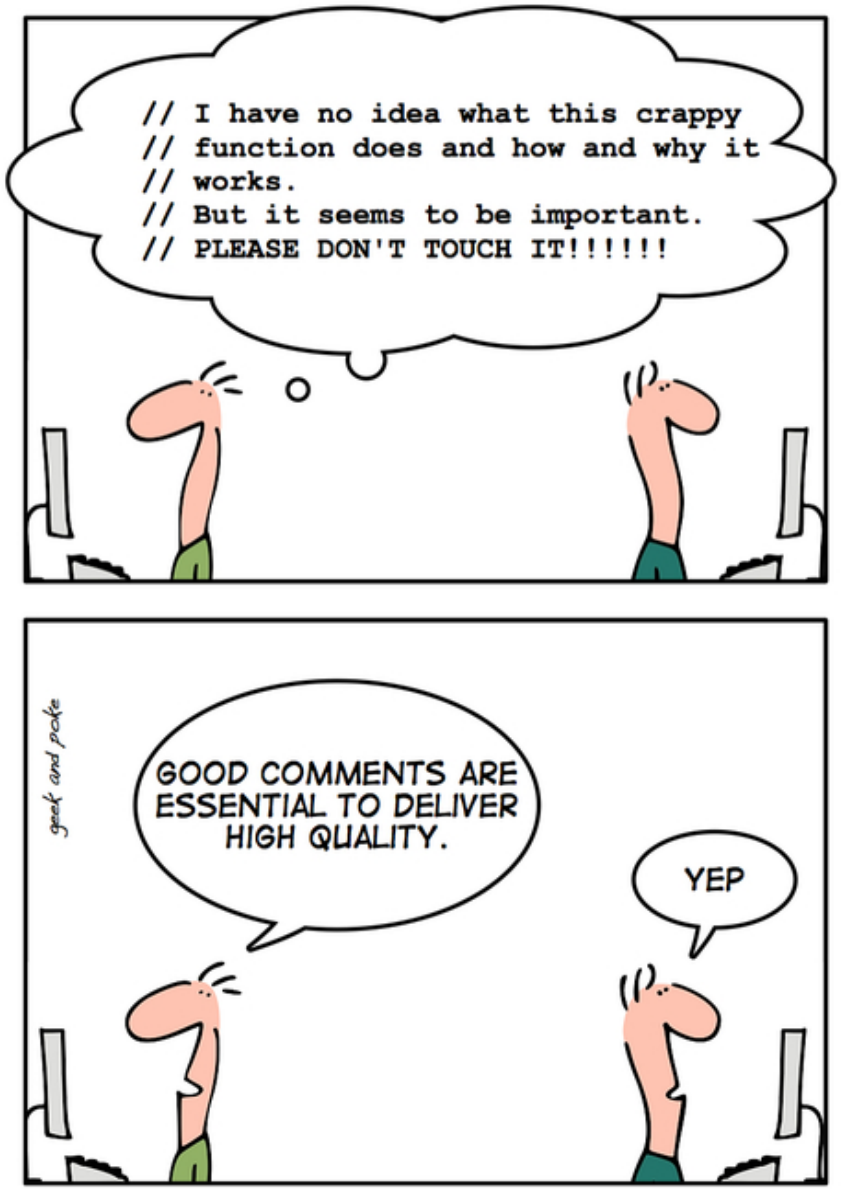
Changeability

- Be open to outside contributors
- Code changes and authorships should be publicly visible through versioning control, with sane commit messages
- Is the software sufficiently backwards compatible?

Supportability

- Be in touch with the users your software.
- Have a **public issue tracker** for tracking bugs and feature requests. Alternatively, have a mailing list (publically archived).
  - Platforms like github provide an issue tracker automatically.

Analysability & Reusability



Analysability & Reusability

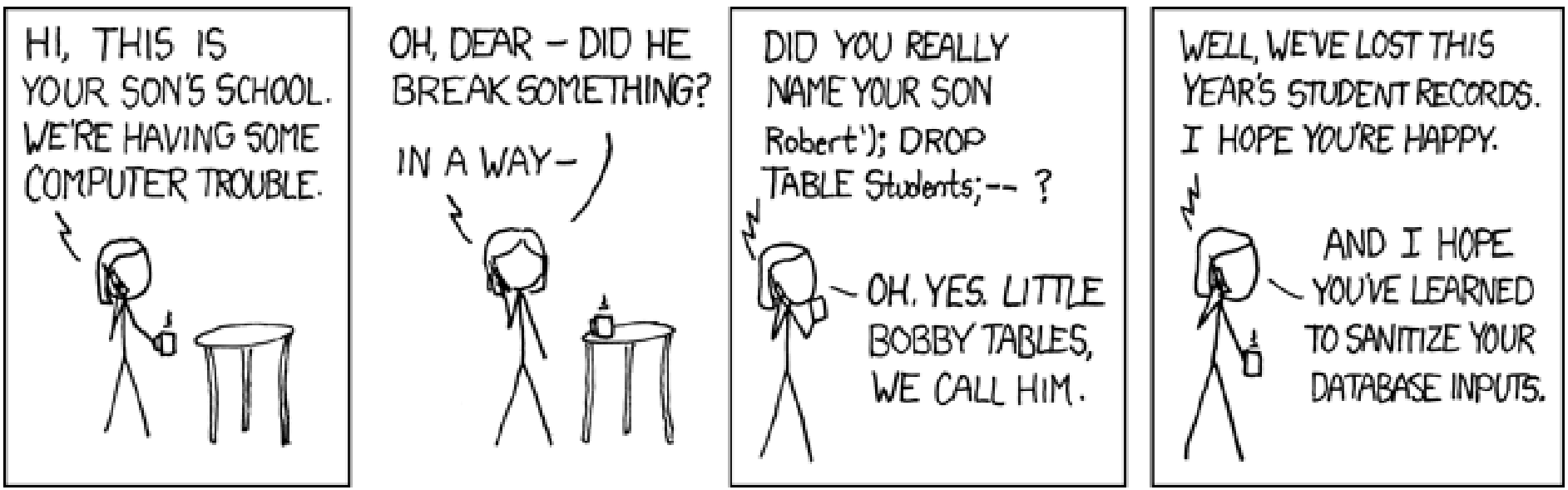
Analysability

- Is the source code commented adequately?
- Is the source code cleanly laid out? (e.g. proper indentation)
- Are sensible named used?
- Follow established conventions for the programming language

Reusability

- Does the software offer the **appropriate interfaces** for the respective audiiences? (command-line, web, GUI, API, web-api..)
- Is the software set up in a **modular** fashion?
  - on **code level**: functions, classes
  - abstraction of core logic from interfaces (front-end/back-end separation)
  - architectural: modules for different functionality

Security & Privacy



Security & Privacy

Security

- Validate user input
- Be aware of possible attack vectors:
  - DB injection in e.g. SQL queries
  - shell injection (when invoking system calls)
  - cross-site request forgery
  - denial of service
- Do not rely on outdated platforms or libraries
- Never run anything as root

Privacy

- Never store unhashed passwords
- Require authentication to access sensitive user data

Questions?

Comics by: Geek & Poke and XKCD

Doorn, P., Aerts, P., and Luscher, S. (2016). Research software at the heart of discovery. Technical report, DANS, NLeSC.

Extra: Dimensions (1/2)

Usability Dimensions

- **Understandability** - Is the software easily understood?
- **Documentation** – Comprehensive well-structured documentation?
- **Buildability** – Straightforward to build from source on a supported system?
- **Installability** – Straightforward to install and deploy on a supported system?
- **Learnability** – Easy/intuitive to learn how to use its functions?

Sustainability & Manageability Dimensions

- **Identity** – Project/software identity is clear and unique?
- **Copyright** – Licencing and ownership Adoption of appropriate open-source license
- ...

Extra: Dimensions (2/2)

Sustainability & Manageability Dimensions

- **Community** – Evidence of current/future community
- **Accessibility** – Good facilities to obtain versions of the software?
- **Testability** – Easy to verify if the software functions correctly?
- **Portability** – Usable on multiple platforms?
- **Supportability** – Evidence of current/future developer support?
- **Analysability** – Easy to understand at the source-code level?
- **Changeability** – Easy to modify and contribute changes?
- **Performance** – Resource demands/consumption
- **Interoperability**
- **Security & Privacy**