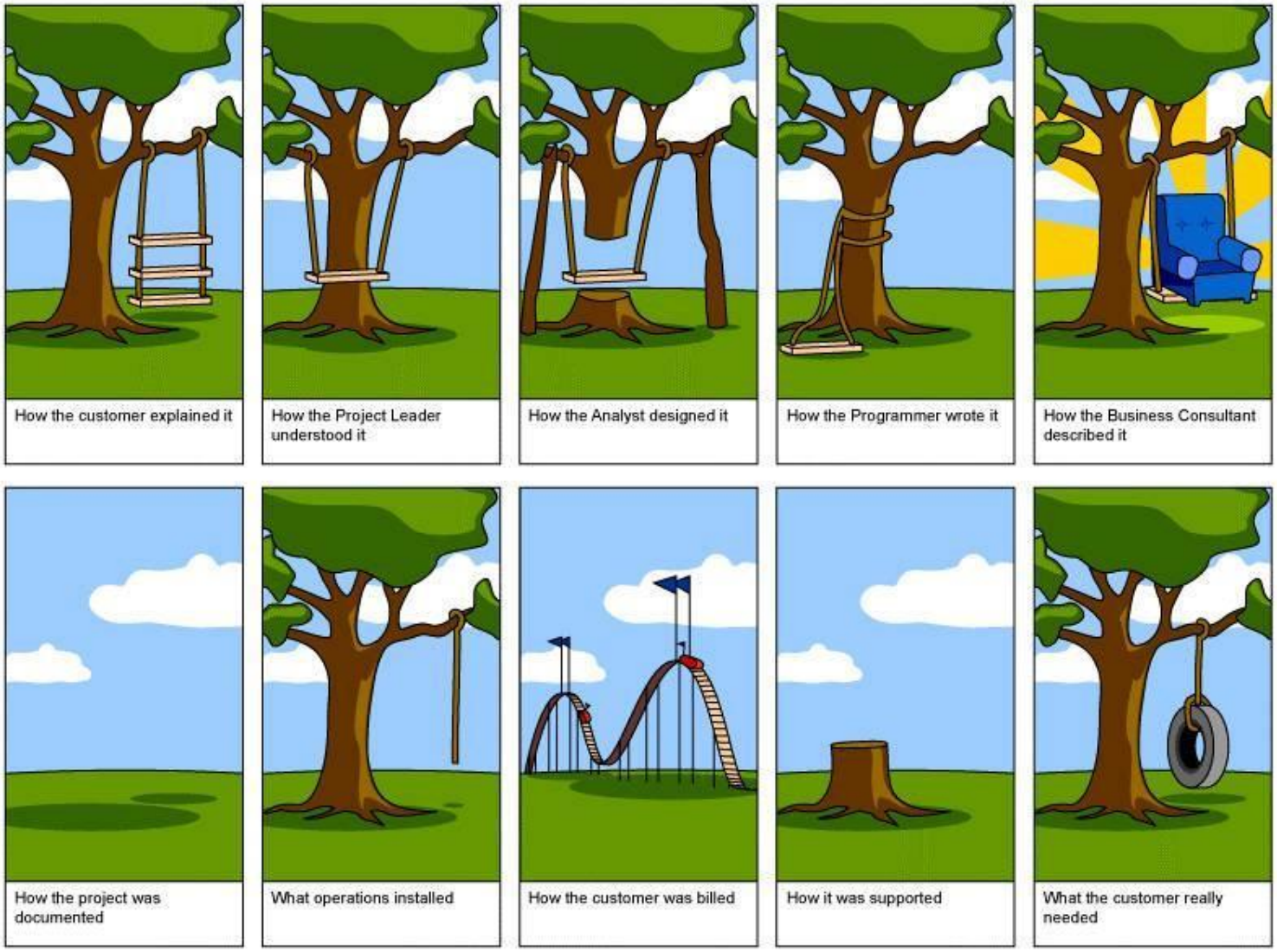
Introduction to Programming and Computing for Scientists

Oxana Smirnova

Lund University

Lecture 2: Essentials of code development in a collaborative environment

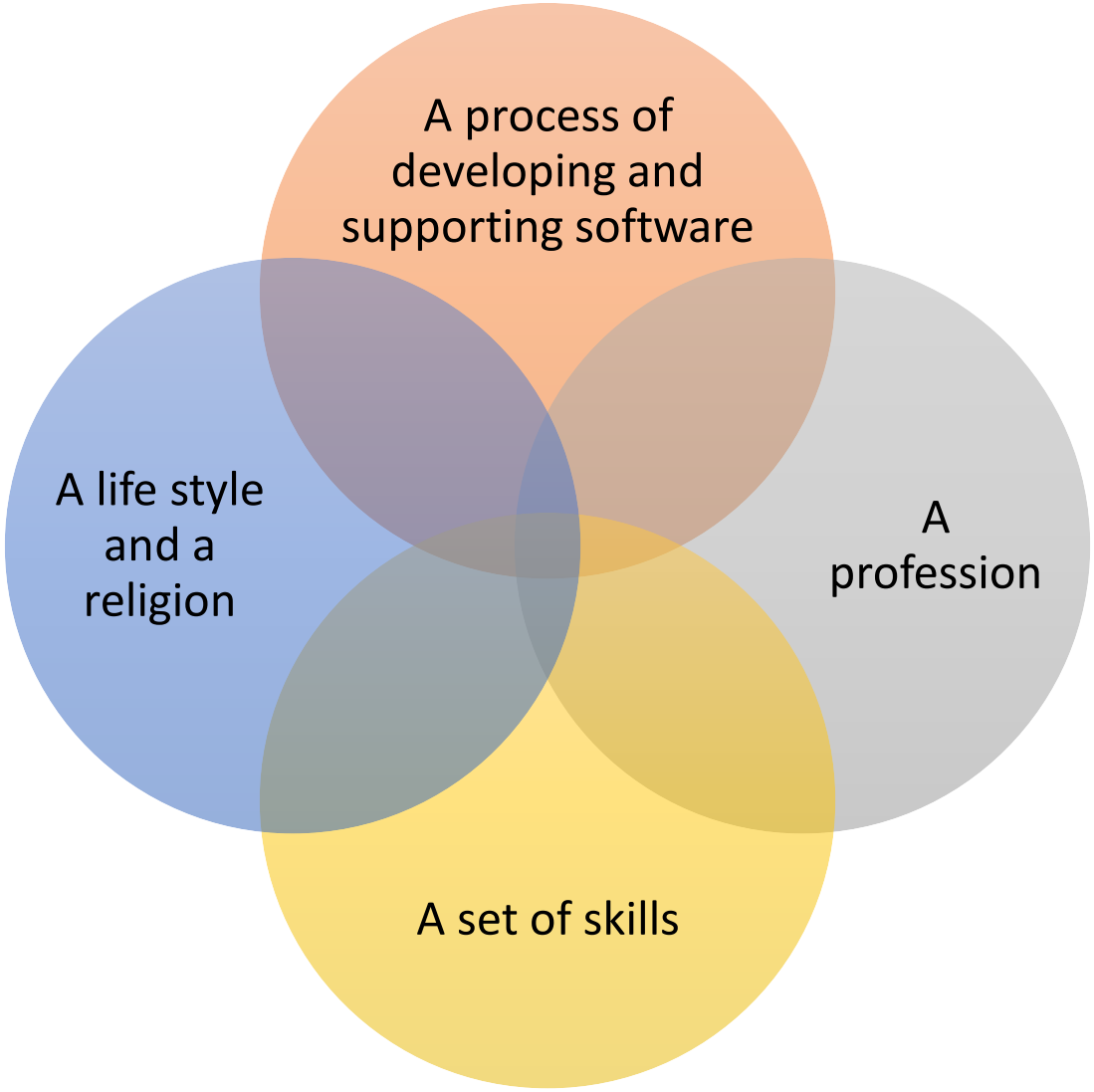
Software development is not simply programming



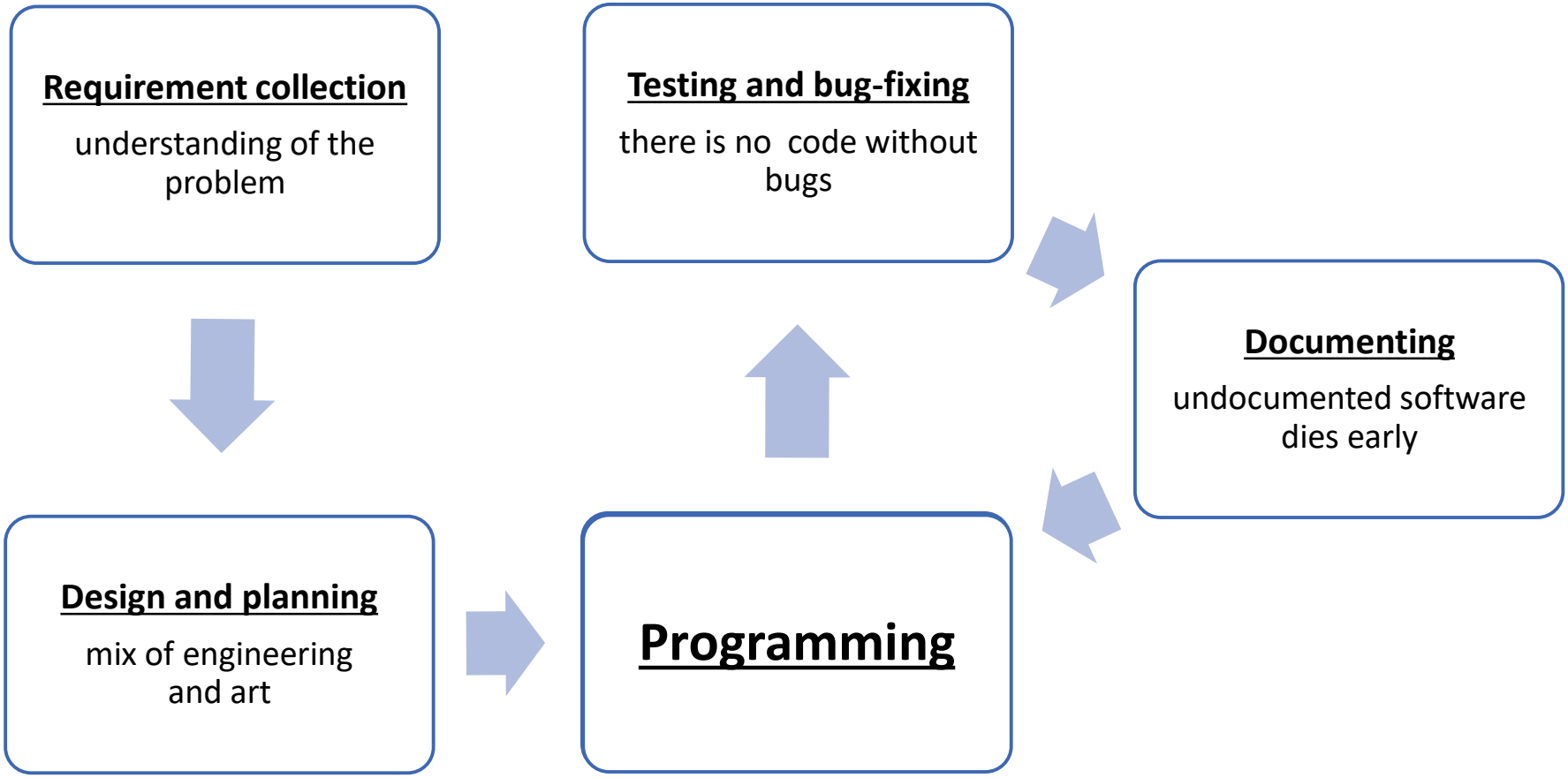
Author unknown

- Most software/IT projects fail, even with excellent programmers

Software development is many things

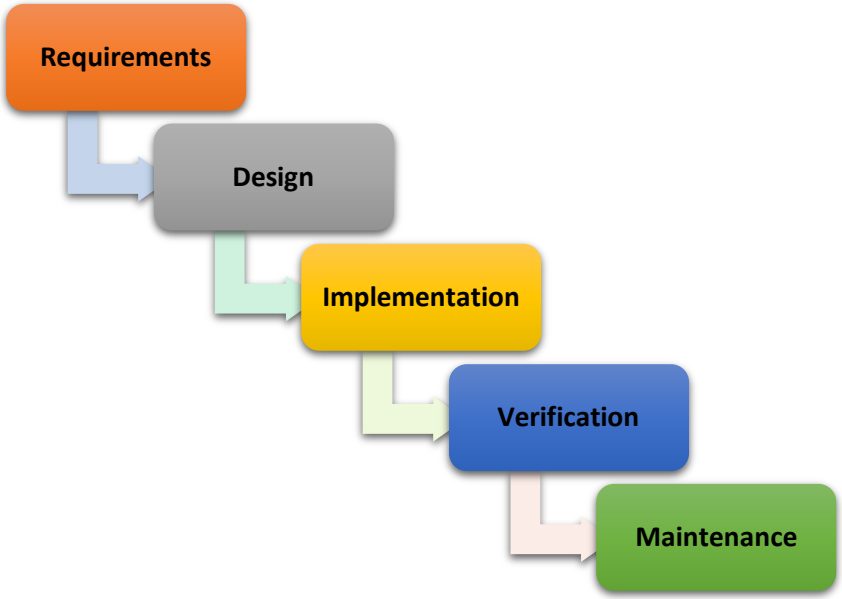


Software development as a process: a simplified picture

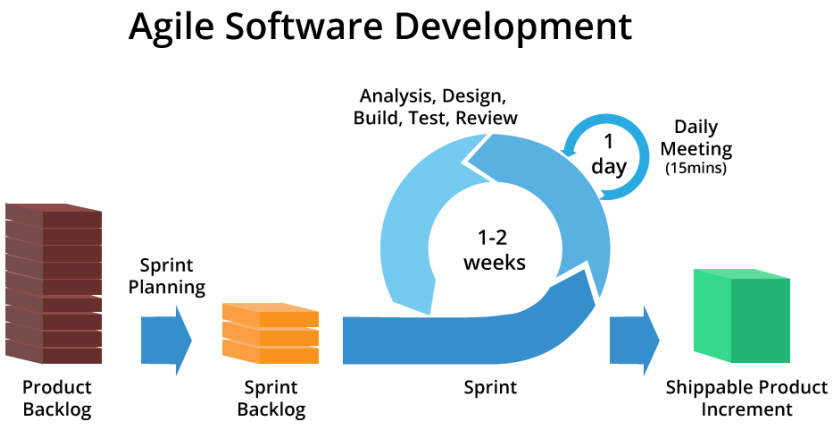


Different software development methodologies

- **Waterfall** model: a straightforward sequential approach



- **Agile development**: too many bugs to do long-term planning

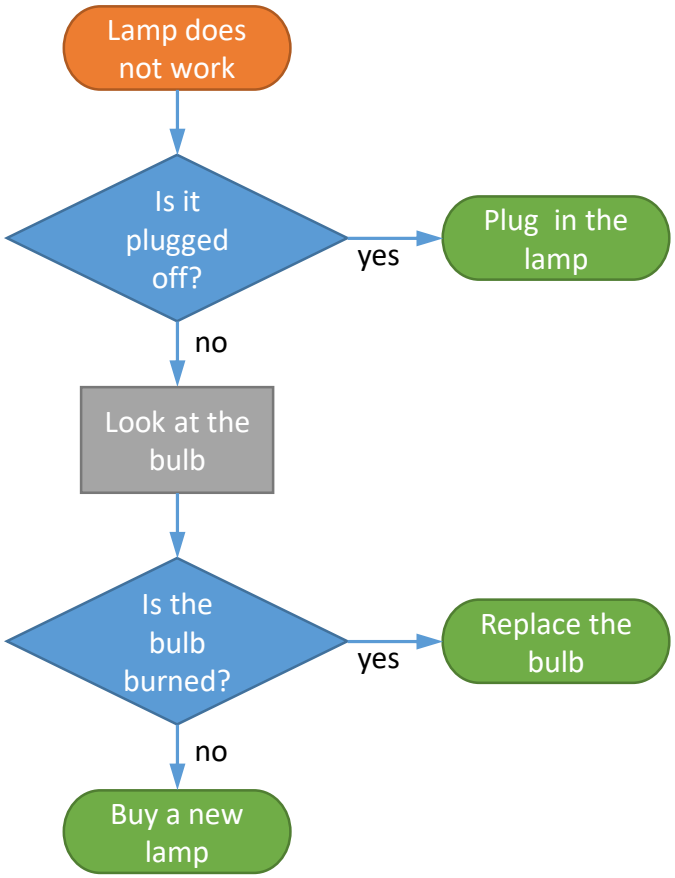


- There are also **rapid prototyping**, **incremental development**, various combinations of methodologies, and even **cowboy coding** (every student does it)

Most programs implement an **algorithm**

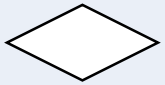
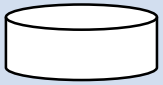
- **Algorithm** is a well-defined sequence of actions to be performed
  - Starts from **initial state**
    - May need initial input
  - Proceeds through a sequence of **instructions** in a strict order
    - May include conditional statements
  - Terminates with a **final state**
- Algorithms can be expressed through:
  - *Human language* (ambiguous)
  - *Pseudocode* (no standard)
  - *Flowcharts*
  - *Other charts, tables, programming languages*

- **Flowchart** is a graphical representation of an algorithm
  - Warning: complex flowcharts may lead to "*spaghetti code*" with many redirections



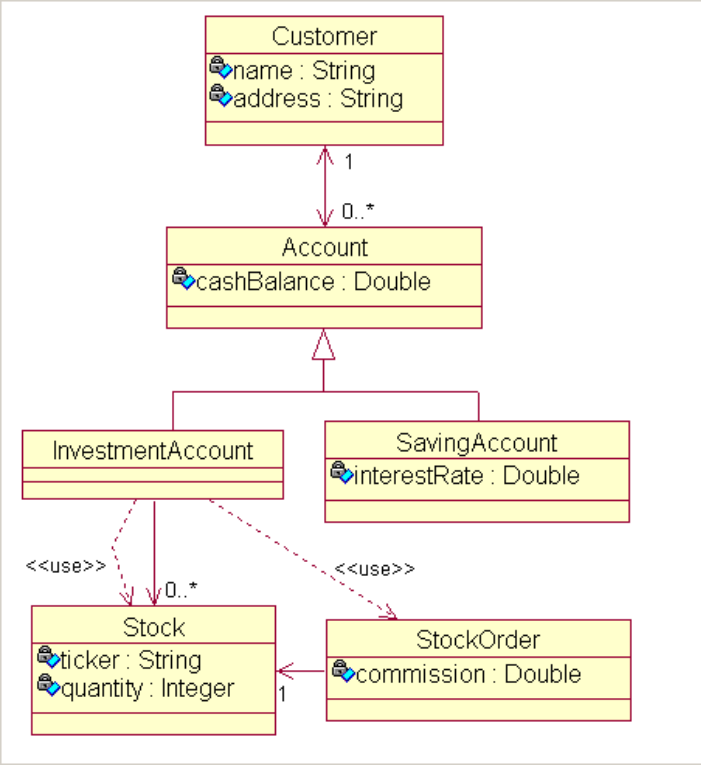
Flowchart symbols overview

- Can be found in any presentation-making software
- Often used to describe not only algorithms, but also workflows

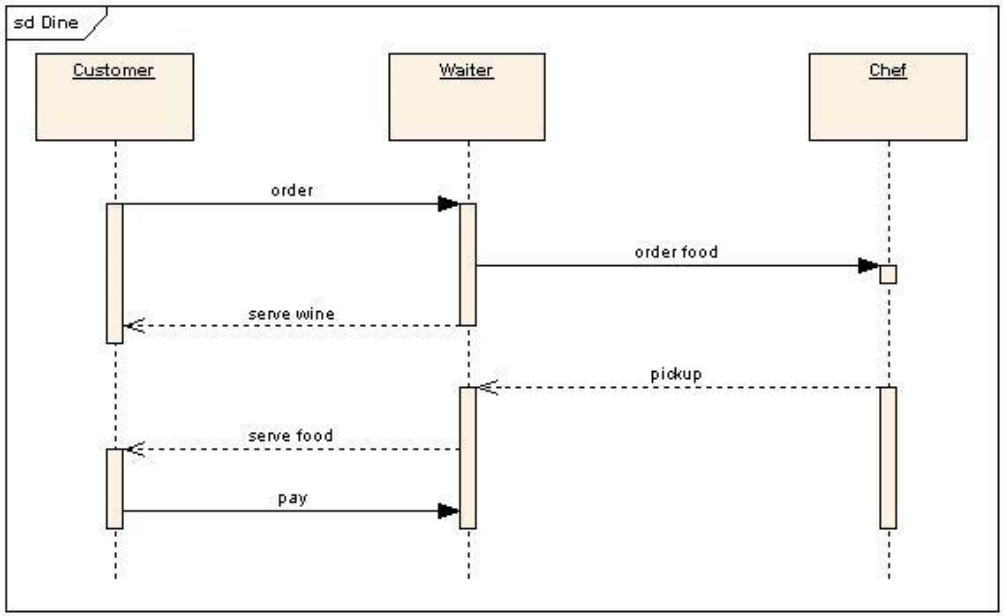
Start and end	
Flow arrows	
Process	
Conditional	
Data	
Document	
Disk	
Tape	

Unified Modelling Language (**UML**)

- A standard way to visualize complex processes or systems
  - You may never need to use it, unless you'll become a professional developer
- Designed for object-oriented methods
- Uses diagrams to describe systems:
  - **Structure** diagrams show objects and their relations
  - **Behaviour** diagrams show activities and state changes
- There are many different "styles" of diagrams, but each has well-defined "language"



Graph by IBM



Graph from Wikimedia

A bit of legalism

- Even if you are not a professional programmer, the code that you write is an **Intellectual Property**
  - Much like scientific publications, music, photos etc
  - Computer programs and even their design are protected by copyright
    - Different laws exist in different countries
- In Swedish universities, it is **your** Intellectual Property
  - In other countries and companies, your employer may own the code – check the contract
  - You should remember to mention code written by you in your CV
- What does ownership give you:
  - Right to authorise copying (including copying for usage)
  - Right to authorise modifications
  - Right to authorise distribution

Software licenses

- **Software license** is a legal instrument that defines rules of software usage, modification and distribution
- Different licenses allow different freedoms
  - Proprietary (end-user license agreements, EULA): least permissive
  - Open Source: some limitations, several combinations exist
  - Public domain: basically, no license, everything is permitted
- Scientific software has no common approach regarding licenses
  - Large pieces of code are in a "grey zone", having no explicit license and used without clear rules
- We like **Open Source** licenses because they allow free code usage, modification and sharing
  - A number of different Open Source licenses exist (see next slide)
  - Open Source software can still be sold (if anyone wants to pay)
  - Software developed using public funding (as in universities) should normally have an Open Source license
- Note: documents and data also have licenses! We like **Open Access** ones.

Some Open Source licenses

- GPL, Apache and BSD are the most commonly used ones
  - Some, like GPL, are “contagious”
  - It is up to the code owner to decide what license to use

Comparison of the Open Source Licences

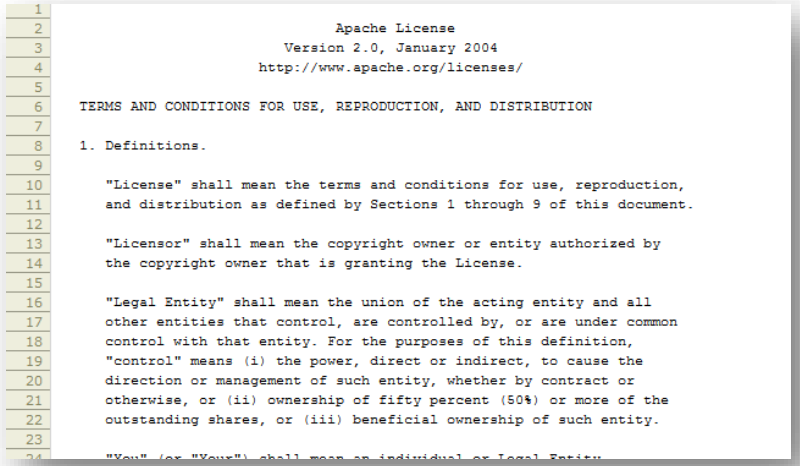
The bullets mark if the the licence explicitly states the item in question. Implicit items are not marked by this chart

	Must distribute license with binray or source	Cannot use contributors name to endorse	There has to be a notification for changed files	Any change must distributed in source form	Lets you provide warrenty if you want to, normally no	Lets you explicitly charge for providing warrenty or gurantee or transfer of code	All derivative work must be under the same license	Must show License when Run from command line	Non derivative works can have different license	May exclude countries where there is a contradiction with patent in that ocuntry	Must describe any deviation due to regulation
Apache License 2.0	●	●	●	●	●						
Common Development and Distribution License	●		●				●				
GNU General Public License (GPL)	●		●	●		●	●	●	●	●	
GNU Library General Public License (LGPL)	●		●	●		●			●	●	
Microsoft Public License (Ms-PL)	●	●									
Microsoft Reciprocal License (Ms-RL)	●	●							●		
Mozilla Public License 1.1 (MPL)	●		●	●							●
New BSD License	●	●									
The MIT License	●										

Graph from stackoverflow.com , original spelling

Practical use of licenses

- Licenses protect (or not) both the developers and the software
  - If everybody is allowed to change the code, the original author can not guarantee its quality or features
  - If nobody is allowed to change it, the author will be held responsible for all wrongdoings
  - In practice, a good balance is needed: changes should be allowed under certain conditions
- License is implemented as a piece of text, distributed together with the software
  - Some add it to every file
  - If a software package has many files, license can be a separate file



Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

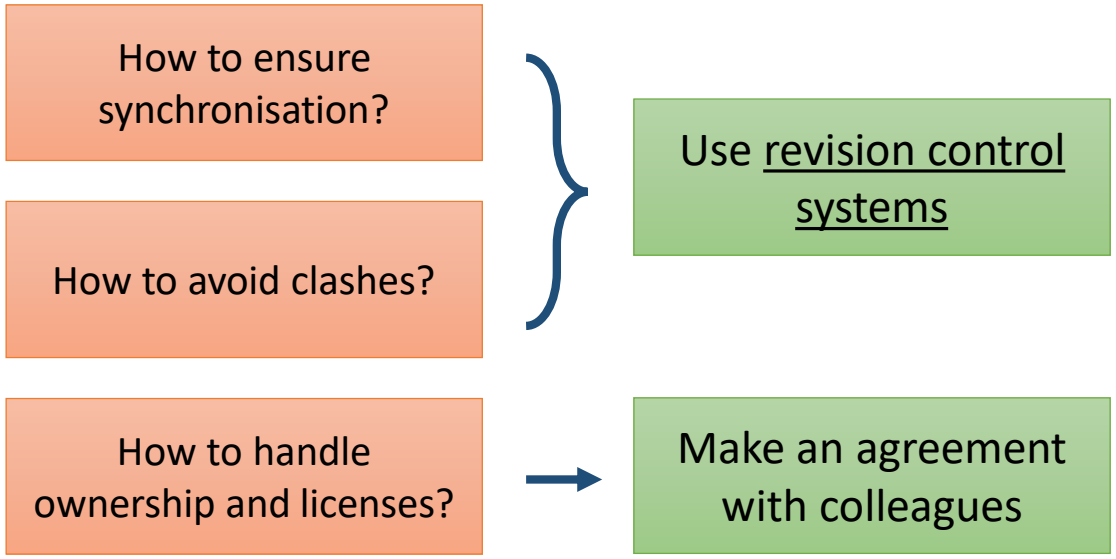
"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

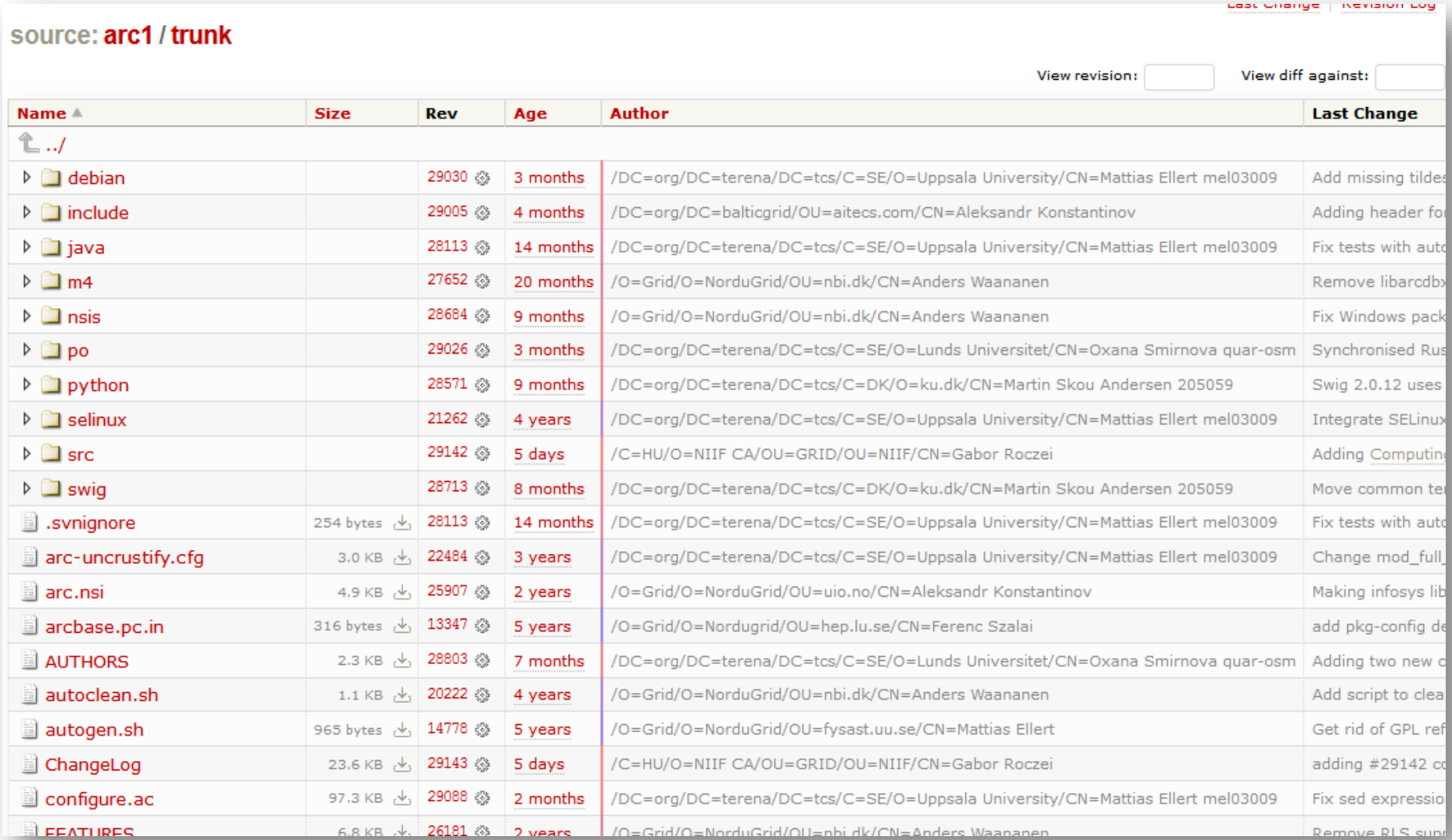
"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

When you're not the lone developer

- When your code grows big, it is always good to split it into separate files
  - A program is one algorithm (rarely a few)
  - Software is a collection of various algorithms that work together
- Large softwares consist of many files and are usually developed by several people



Example of a complex software with many authors



source: arc1 / trunk

View revision: View diff against:

Name	Size	Rev	Age	Author	Last Change
../					
debian		29030	3 months	/DC=org/DC=terena/DC=tcs/C=SE/O=Uppsala University/CN=Mattias Ellert mel03009	Add missing tildes
include		29005	4 months	/DC=org/DC=balticgrid/OU=aitecs.com/CN=Aleksandr Konstantinov	Adding header fo
java		28113	14 months	/DC=org/DC=terena/DC=tcs/C=SE/O=Uppsala University/CN=Mattias Ellert mel03009	Fix tests with auto
m4		27652	20 months	/O=Grid/O=NorduGrid/OU=nbi.dk/CN=Anders Waananen	Remove libarcdbx
nsis		28684	9 months	/O=Grid/O=NorduGrid/OU=nbi.dk/CN=Anders Waananen	Fix Windows pack
po		29026	3 months	/DC=org/DC=terena/DC=tcs/C=SE/O=Lunds Universitet/CN=Oxana Smirnova quar-osm	Synchronised Rus
python		28571	9 months	/DC=org/DC=terena/DC=tcs/C=DK/O=ku.dk/CN=Martin Skou Andersen 205059	Swig 2.0.12 uses
selinux		21262	4 years	/DC=org/DC=terena/DC=tcs/C=SE/O=Uppsala University/CN=Mattias Ellert mel03009	Integrate SELinux
src		29142	5 days	/C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=Gabor Roczei	Adding Computin
swig		28713	8 months	/DC=org/DC=terena/DC=tcs/C=DK/O=ku.dk/CN=Martin Skou Andersen 205059	Move common te
.svnignore	254 bytes	28113	14 months	/DC=org/DC=terena/DC=tcs/C=SE/O=Uppsala University/CN=Mattias Ellert mel03009	Fix tests with auto
arc-uncrustify.cfg	3.0 KB	22484	3 years	/DC=org/DC=terena/DC=tcs/C=SE/O=Uppsala University/CN=Mattias Ellert mel03009	Change mod_full_
arc.nsi	4.9 KB	25907	2 years	/O=Grid/O=NorduGrid/OU=uio.no/CN=Aleksandr Konstantinov	Making infosys lib
arcbase.pc.in	316 bytes	13347	5 years	/O=Grid/O=Nordugrid/OU=hep.lu.se/CN=Ferenc Szalai	add pkg-config de
AUTHORS	2.3 KB	28803	7 months	/DC=org/DC=terena/DC=tcs/C=SE/O=Lunds Universitet/CN=Oxana Smirnova quar-osm	Adding two new c
autoclean.sh	1.1 KB	20222	4 years	/O=Grid/O=NorduGrid/OU=nbi.dk/CN=Anders Waananen	Add script to clea
autogen.sh	965 bytes	14778	5 years	/O=Grid/O=NorduGrid/OU=fysast.uu.se/CN=Mattias Ellert	Get rid of GPL ref
ChangeLog	23.6 KB	29143	5 days	/C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=Gabor Roczei	adding #29142 co
configure.ac	97.3 KB	29088	2 months	/DC=org/DC=terena/DC=tcs/C=SE/O=Uppsala University/CN=Mattias Ellert mel03009	Fix sed expressio
FEATURES	6.8 KB	26181	2 years	/O=Grid/O=NorduGrid/OU=nbi.dk/CN=Anders Waananen	Remove RLS supp

Revision control systems

- **Revision** in our context stands for an updated piece of code (or several pieces)
- Since several developers may update different pieces of code simultaneously, a system is needed to keep everything synchronised and to avoid clashes
- Sometimes bad updates need to be reverted, too – previous revisions need to be kept
- When a software is ready to be used, it has to be tagged, for reference
  - A tag is basically a snapshot of all the code, labelled by a number or a special string
  - Tags are a good reference point for testing
  - When tested and proven to work as expected, a tag is released as a new software **version**
- Therefore, the main functionalities of such common development systems are:
  - Reference software repository (“master copy”)
  - Accepting changes (commits) from different developers
  - Revision history (“backup” of files)
  - Versioning

Few words on software versions

- Software changes very often, so it is important to know what exact code was used
  - Primarily, to make results **reproducible**
  - But also to simplify maintenance, debugging, user support etc
- Most code developed by students has no versions – very bad practice!
- Some examples of versions:
  - Operating systems: Windows **11**, iOS **15.6.1**, Ubuntu **22.10 "Kinetic Kudu"**, Android **9.0 "Pie"**, Fedora **20 "Heisenbug"**
  - Software: ROOT **v6.26/06**, gcc **12.2**, Photoshop **CS5.5**, Office **2019**, Linux kernel **5.19.5**
- In the Linux world, most common versioning scheme is **MM.mm.bb**
  - **MM** – major version with massive changes; usually backwards incompatible with MM-1
  - **mm** – minor version with some new functionality; versions MM.mm and MM.mm-1 are usually compatible
  - **bb** – bugfix version, always compatible with MM.mm.bb-1

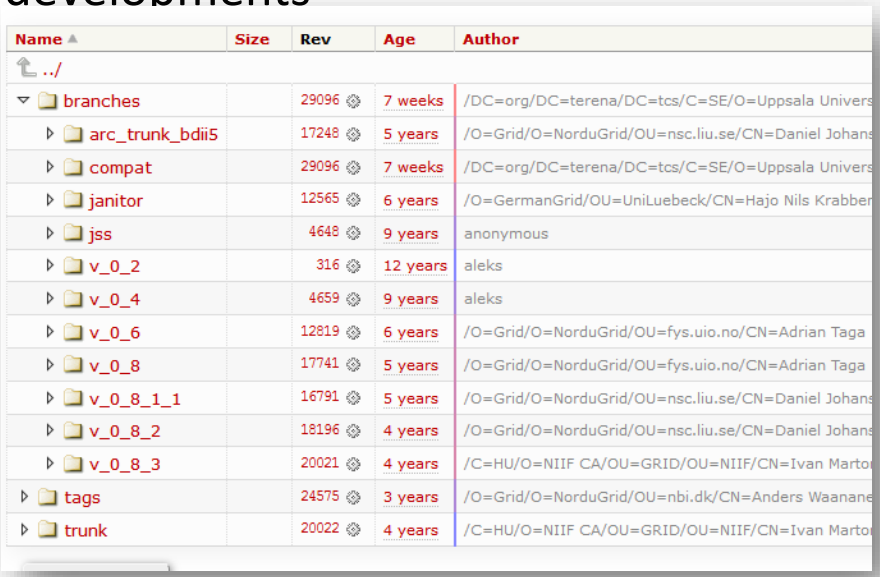
General principles of work with revision control systems

- There is a code repository, from which software releases are made
  - Repository can be centralised or distributed
- Each developer makes – **clone** or **checkout** – an own **working copy** of the repository
  - Many systems allow to clone/checkout only a part of the entire repository
- After doing local code changes, the developer synchronises – **commit** or **push** – the change to the repository
  - In most systems, only the differences are communicated to the repository
  - It is a good practice to commit often, avoid mega-commits
- If the system notices that the code has changed meanwhile, it will try to merge the changes, if they were committed to different parts of the code
  - Beware! The changes may still turn out to be incompatible, no system is clever enough to figure it
  - If automatic merging is impossible, you will have to do it manually
- Changes can be **reverted** to any previous revision if it turns out they caused troubles
- Release manager can decide which changes should be accepted for the software release

Traditional approach: central code repository

- A straightforward approach is to have one central repository
- A **trunk** would contain the main reference code, and **branches** would contain specialised/private developments

Old ARC software repository



Name	Size	Rev	Age	Author
../				
branches		29096	7 weeks	/DC=org/DC=terena/DC=tcs/C=SE/O=Uppsala Univers
arc_trunk_bdii5		17248	5 years	/O=Grid/O=NorduGrid/OU=nsc.liu.se/CN=Daniel Johans
compat		29096	7 weeks	/DC=org/DC=terena/DC=tcs/C=SE/O=Uppsala Univers
janitor		12565	6 years	/O=GermanGrid/OU=UniLuebeck/CN=Hajo Nils Krabber
jss		4648	9 years	anonymous
v_0_2		316	12 years	aleks
v_0_4		4659	9 years	aleks
v_0_6		12819	6 years	/O=Grid/O=NorduGrid/OU=fys.uio.no/CN=Adrian Taga
v_0_8		17741	5 years	/O=Grid/O=NorduGrid/OU=fys.uio.no/CN=Adrian Taga
v_0_8_1_1		16791	5 years	/O=Grid/O=NorduGrid/OU=nsc.liu.se/CN=Daniel Johans
v_0_8_2		18196	4 years	/O=Grid/O=NorduGrid/OU=nsc.liu.se/CN=Daniel Johans
v_0_8_3		20021	4 years	/C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=Ivan Marto
tags		24575	3 years	/O=Grid/O=NorduGrid/OU=nbi.dk/CN=Anders Waanane
trunk		20022	4 years	/C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=Ivan Marto

Plus

- Easy to control and manage
- Easy to prevent conflicts
- Each developer makes a checkout of only the code they need

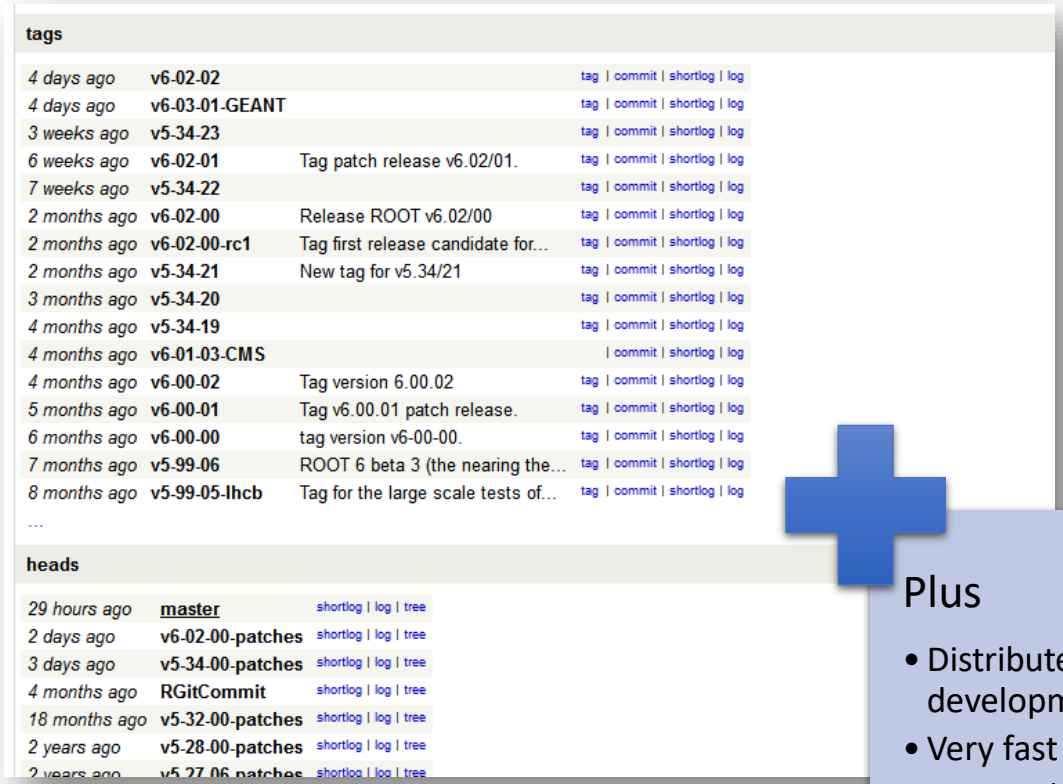
Minus

- No general agreement how to deal with branches
- Single point of failure if the server goes down (or is slow)

Modern approach: distributed repository

- Every developer has a local copy of the entire repository
  - Can commit off-line and synchronise later
    - Allows for frequent commits, hence better revision control

ROOT software repository



tags

4 days ago	**v6-02-02**		tag | commit | shortlog | log
4 days ago	**v6-03-01-GEANT**		tag | commit | shortlog | log
3 weeks ago	**v5-34-23**		tag | commit | shortlog | log
6 weeks ago	**v6-02-01**	Tag patch release v6.02/01.	tag | commit | shortlog | log
7 weeks ago	**v5-34-22**		tag | commit | shortlog | log
2 months ago	**v6-02-00**	Release ROOT v6.02/00	tag | commit | shortlog | log
2 months ago	**v6-02-00-rc1**	Tag first release candidate for...	tag | commit | shortlog | log
2 months ago	**v5-34-21**	New tag for v5.34/21	tag | commit | shortlog | log
3 months ago	**v5-34-20**		tag | commit | shortlog | log
4 months ago	**v5-34-19**		tag | commit | shortlog | log
4 months ago	**v6-01-03-CMS**		| commit | shortlog | log
4 months ago	**v6-00-02**	Tag version 6.00.02	tag | commit | shortlog | log
5 months ago	**v6-00-01**	Tag v6.00.01 patch release.	tag | commit | shortlog | log
6 months ago	**v6-00-00**	tag version v6-00-00.	tag | commit | shortlog | log
7 months ago	**v5-99-06**	ROOT 6 beta 3 (the nearing the...	tag | commit | shortlog | log
8 months ago	**v5-99-05-lhcb**	Tag for the large scale tests of...	tag | commit | shortlog | log

...

heads

29 hours ago	**master**	shortlog | log | tree
2 days ago	**v6-02-00-patches**	shortlog | log | tree
3 days ago	**v5-34-00-patches**	shortlog | log | tree
4 months ago	**RGitCommit**	shortlog | log | tree
18 months ago	**v5-32-00-patches**	shortlog | log | tree
2 years ago	**v5-28-00-patches**	shortlog | log | tree
2 years ago	**v5-27-06-patches**	shortlog | log | tree

Plus

- Distributed development
- Very fast
- Easy and quick to branch and merge code

Minus

- Have to keep entire repository locally
- Can not lock files
- Non-trivial access control
- Tangled history

Most popular revision control systems

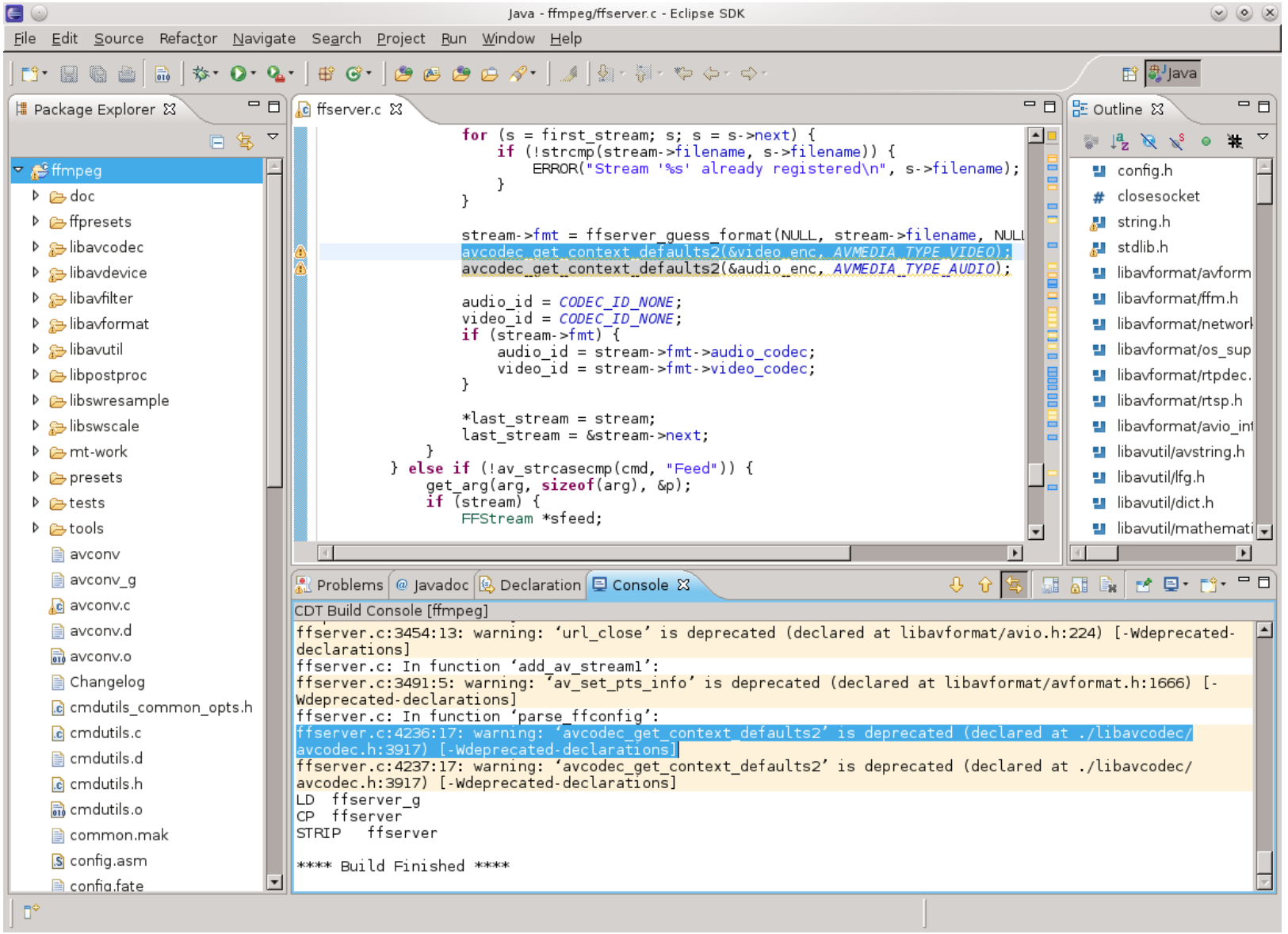
- **Git**: a distributed system
  - Developed by Linus Torvalds in 2005
  - Open Source (GPL/LGPL)
  - Each local copy is a complete repository, with all the revision history and such
  - Light-weight easy to merge branches
  - No own access control, but many add-ons
  - Basis for popular platforms: Github, Gitlab
  - Very well documented: http://git-scm.com/book

- **Subversion (SVN)**: a centralised system
  - Release 1.0.0 in 2004
  - Open Source (Apache)
  - Branches retain no knowledge of the trunk
  - Strict history and revision control
  - Allows authorisation per directory
  - Well documented, e.g.: http://svnbook.red-bean.com/

Integrated Development Environment (IDE)

- For non-professional developers, a good editor and command-line build tools is enough to write and **build** software
  - Warning: avoid **building** in your Git/SVN working copy! This may create many files that you don't want to commit!
- For professionals, special IDEs exist, that include:
  - Context-aware software editor
  - Build automation tools
    - Some include compilers and interpreters
  - Debugging tools
  - Some integrate with revision control systems
- Very many IDEs exist for C++
  - There are no good IDEs for Linux (they are not really needed there)
  - **Code::Blocks** is a good IDE, available at LUNARC
    - Some even use **Emacs** editor as an IDE
  - **Eclipse** is one of the most powerful and complicated
  - On Windows, **Microsoft Visual Studio** is the best

Eclipse screenshot



There is no code without bugs

- With an Open Source code, everybody can find a bug… or many…
- So we need a system where bugs can be reported and followed-up: a **bug tracking system**
- Such system is essentially a database where every authorised person can register a discovered defect
- Typical information to be entered:
  - Summary of the problem and ways to reproduce it
  - Software version that has the problem
  - Operating system version where the problem occurs
  - Severity of the problem
- Bugs have life cycle: from being new, to assigned, to fixed
  - Different systems have different such states
  - States are changed by administrators in charge of bug tracking
    - E-mail notifications are sent to all the involved parties (reporters, developers etc) on each state change
- When you find a bug, please **always report it**!

Bug tracking systems

- Many bug tracking systems exist
  - Some are stand-alone
  - Some are integrated with revision control systems
    - Some are even distributed
  - Some are integrated with IDEs
  - Some are a part of larger issue-tracking systems

Some commonly used bug trackers:

- Bugzilla (standalone), http://www.bugzilla.org/
- JIRA (project management tool), https://www.atlassian.com/software/jira
- Savannah (development service), http://savannah.gnu.org/
- Trac (integrated with Wiki), http://trac.edgewall.org/
- Redmine (project management), http://www.redmine.org/

Screenshot of the NorduGrid's Bugzilla

Bugzilla – Bug List

Home | New | Browse | Search | Search [?] | Reports | Help | New Account | Log In | Forgot Password

Sun Nov 30 2014 22:47:43 CET

Any program that runs right is obsolete

Hide Search Description

Resolution: --- **Component:** ARClib **Product:** NorduGrid ARC

13 bugs found.

ID ▲	Product	Comp	Assignee	Status	Resolution	Summary	Changed
804	NorduGri	ARClib	skou	ASSI	---	Insufficient client error messages when job information is not found	2013-10-08
868	NorduGri	ARClib	skou	ASSI	---	Incorrect error message from arcsub when queue information is unavailable	2012-08-16
1951	NorduGri	ARClib	waananen	ASSI	---	Outdated CRLs render all client tools useless	2014-04-22
2787	NorduGri	ARClib	skou	ASSI	---	feature request for updating status of ExecutionTargets	2013-03-27
2874	NorduGri	ARClib	akonstantinov	ASSI	---	arcsub cannot parse a simple JSDL file on Windows	2013-10-21
2890	NorduGri	ARClib	skou	ASSI	---	should not treat "executable" as a local input file	2012-08-08
2977	NorduGri	ARClib	skou	ASSI	---	arcmigrate does not seem to be working as described in manual	2013-10-08
3099	NorduGri	ARClib	skou	ASSI	---	bulk operations including queries for EMI-ES	2013-10-23
3190	NorduGri	ARClib	akonstantinov	ASSI	---	Memory leak during multiple job submission with files to BES interface	2013-10-08
3244	NorduGri	ARClib	skou	ASSI	---	Inconsistent ComputingService Name between gridftp and	2014-06-04

Software development hosting platforms

- If you start a new software project and don't want to set up an own code repository, Wiki, bug tracker etc, several free Open Source hosting services exist

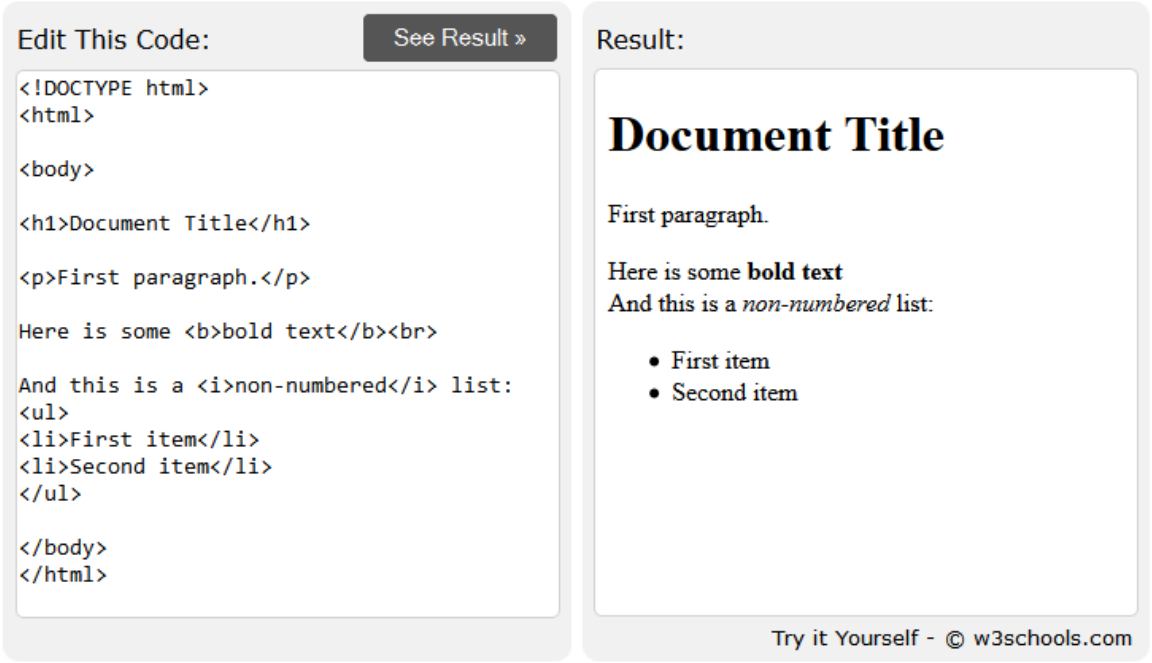
Sourceforge, http://sourceforge.net/: historically, the first platform of the kind (launched in 1999), interfaces to SVN, Git and other revision control systems

GitHub, https://github.com/: the most popular and largest IT-project hosting service (started in 2008), based on Git (obviously); free for Open Source projects

- Some other hosting services: BitBucket, several Google platforms, GitLab, Launchpad, Savannah, Assembla etc

Simplest languages: markup languages

- Markup languages add special tags to plain text
  - These tags will be processed and interpreted by software
  - Tags must be distinguishable from normal text
- An example of a markup language at work you see every day in Web pages
  - Did you ever try to click “show source code” on a Web page?
  - If yes, you probably noted `<!DOCTYPE html>` in the very beginning
  - **HTML** stands for HyperText Markup Language
    - Was developed at CERN, inspired by an earlier SGML (Standard Generalized Markup Language)



Usage of markup languages for text processing; LaTeX

- Once your results are ready, it is time to publish!
  - Or to write a project report
- Softwares that make your papers looking good are called **word processors**
  - All good word processors cost money (like Microsoft Office)
  - All free word processors are desperately bad (like LibreOffice)
- What do word processors do under the hood?
  - They make use of different markup languages to add special tags to your text and figures, and convert them to a visually pleasant layout (hopefully)
- **LaTeX** is a markup language for word processing, with which you add the tags yourself, and LaTeX system converts it to a publishable material

Plus	Minus
• It is free • It supports most complex mathematics • It is extensible • It is accepted by all publishers	• You don't see the result "live" • Tables and figures are very difficult to pin into place • No way to track changes (unless you use a revision control system)

So what is LaTeX?

- Actually, the language itself is called **TeX**
  - TeX was released in 1978, designed by Donald Knuth in Stanford
  - The goal was to create a complete typesetting system that would produce identical results on any computer
    - Hence the markup language: plain text can be transferred everywhere
  - Stable since 1989, when support for non-English languages was added to TeX 3.0
    - Software version is currently 3.14159265 (guess the next version….)
    - Public domain software
- Some basic TeX rules:
  - TeX tags (commands) start with a backslash `\` and use curly brackets `{ }` to group command input
  - Simple mathematics is included in `$$`
    - `$\sqrt{2}$` results in $\sqrt{2}$
  - Paragraphs are separated by blank lines
  - Comments start with `%`

From TeX to LaTeX

- Plain TeX uses elementary instructions and is rather difficult to learn and use for complex documents
- Leslie Lamport developed LaTeX in 1984 using TeX, to provide a higher-level language
  - Added pre-defined commands for sections, cross-references, bibliography etc
  - Easy to use with non-Latin scripts
  - Current version: LaTeX2ε (since 1994)
  - Several online platforms exist

```
\documentclass[a4paper]{article}

\begin{document}

\section*{Document Title}

First paragraph.

Here is some \textbf{bold} text\\
And this is a \textit{non-numbered} list:
\begin{itemize}
\item First item
\item Second item
\end{itemize}

\end{document}
```

Document Title

First paragraph.
Here is some **bold** text
And this is a *non-numbered* list:

- First item
- Second item

Exercise: go to https://v2.overleaf.com and try to reproduce this document

More LaTeX features

- Can do almost all imaginable formatting, section numbering, headers and footers, lists etc
  - Note: (La)TeX uses own fonts, not system ones
    - This ensures identical results everywhere

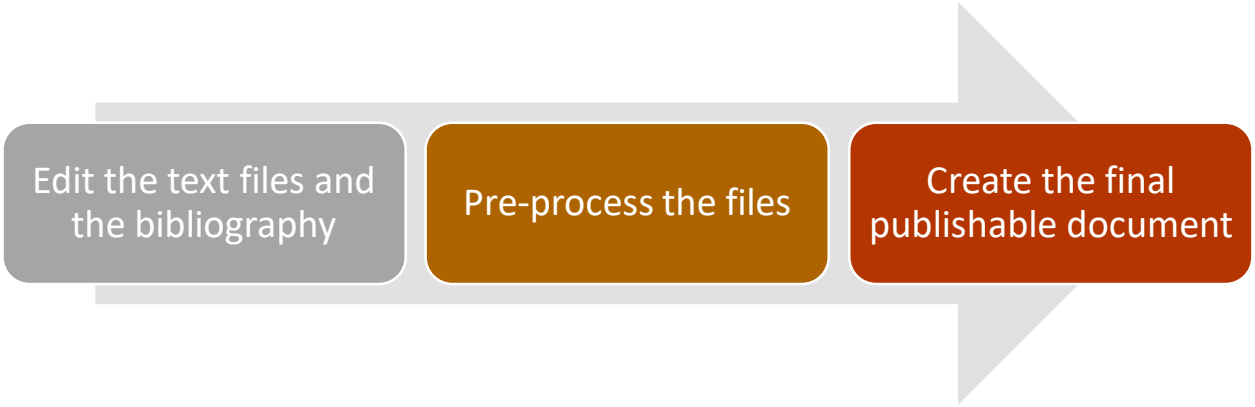
Is very good with equations	`\begin{equation}`
Can include figures	`\begin{figure}` `\includegraphics{cat.jpg}`
Can create tables	`\begin{table}` `\begin{tabular}`
Handles cross references	`\label{sec:intro}` `\ref{sec:intro}`

Handles bibliography	`\begin{thebibliography}`
	`\bibitem{mybook}` …
	`\cite{mybook}`

Can include other files	`\input{section2.tex}` `\include{appendix.tex}`
Can auto-generate table of contents etc	`\tableofcontents`
Can even do nice slides	`\documentclass{beamer}`

Steps to create a LaTeX document

- Writing a LaTeX document is similar to real software development:



- You can use Linux command line, any of Windows IDEs (*TeXnicCenter* is good), or one of the many online LaTeX systems (Overleaf is perhaps the most popular)

A more advanced document: part 1

```
\documentclass[a4paper,12pt,twoside]{article}

\usepackage[utf8]{inputenc}
\usepackage[pdftex]{graphicx}
\usepackage[pdftex]{hyperref}

\author{Oxana Smirnova}
\title{LaTeX exercise paper}
\date{\today}

\begin{document}

 \maketitle

 \begin{center}
  \texttt{oxana.smirnova@hep.lu.se}
 \end{center}

\section{Introduction}
\label{sec:intro}

This is introduction. Summary will be given in Section \ref{sec:sum}.

\section{About Linux}
\label{sec:linux}

\begin{figure}[h]
  \begin{center}
    \includegraphics[width=2cm]{penguin.png}
    \caption{Penguin symbolises Linux}
    \label{fig:penguin}
  \end{center}
\end{figure}
```

Use additional features:
utf8: non-latin fonts
graphicx: pictures
hyperref: web links

Create title here

Empty line starts a new paragraph

Use labels for cross-referencing from text

Use Upload button to upload a picture to Overleaf

Use labels to cross-reference figures as well

A more advanced document: part 2

```
Figure \ref{fig:penguin} shows a \textit{penguin}. For more details, check
the Linux Web page~\cite{linux}.

\subsection{Linux flavours}
\label{sec:flavours}

Table~\ref{tab:flavours} lists some Linux flavours~\footnote{Only one is shown
for simplicity}.

\begin{table}[h]
  \begin{center}
    \begin{tabular}{c||c|c|c} % | is for border, c is for centering
      \textbf{Distribution}&RedHat&Debian&SuSE\\ \hline \hline
      Fedora 23            & X    &      &    \\ \hline
    \end{tabular}
    \caption{Different flavours of Linux}
    \label{tab:flavours}
  \end{center}
\end{table}

\section{About mathematics}
\label{sec:math}

In-line math in \LaTeX \ is enclosed in \$ symbols. Backslash \textbackslash \
is used to denote special symbols.

Subscripts and superscripts are always math: $A_x$, $A_{xy}$,
$e^x$ and $e^{x^2}$. Using underscore \_ outside math without \textbackslash \
causes big\_troubles.

All special symbols are also math: $\alpha$, $\beta$, $\gamma$, $\delta$, $\sin
x$, $\hbar$, $\lambda$, $\ldots$ More information can be
found in Ref.~\cite{latex}.
```

Cited papers are defined in the end of the document

Tables are difficult: & divides cells and \\ divides rows

Remember {} to group characters

A more advanced document: part 3

```
Equation~\ref{eq:chi2} shows $\chi^2$.

\begin{equation}
\label{eq:chi2}
 \chi^2 = \sum\limits_i \left(\frac{F_i-D_i}{\sigma_i}\right)^2
\end{equation}

\section{Summary}
\label{sec:sum}

We learned the following:
\begin{itemize}
 \item Linux is good
 \item \LaTeX \ is good for:
    \begin{enumerate}
     \item Structuring documents
     \item Writing mathematial equations
    \end{enumerate}
\end{itemize}

We can also write unformatted text using \texttt{verbatim} environment, but
sometimes we have to specify this in the preamble:
\begin{verbatim}
 \usepackage{verbatim}
\end{verbatim}

\begin{thebibliography}{99}
 \bibitem{linux} Linux web site: \url{www.linux.com}
 \bibitem{latex} Leslie Lamport, \textsl{LaTeX: A Document Preparation
System}, second edition, Addison-Wesley (1994).
\end{thebibliography}

\end{document}
```

Auto-numbered equations is why we love LaTeX
Use labels to cross-reference equations, too

99 indicates maximum number of papers in the bibliography list

Papers are referenced not by labels but by bibitem tags

Summary

- Software development is a profession and requires professional tools
- Open Source code drives the technological and scientific progress
- “Language” can mean many things: a programming language, a visual modelling language, a markup language…
  - …and actually many other languages