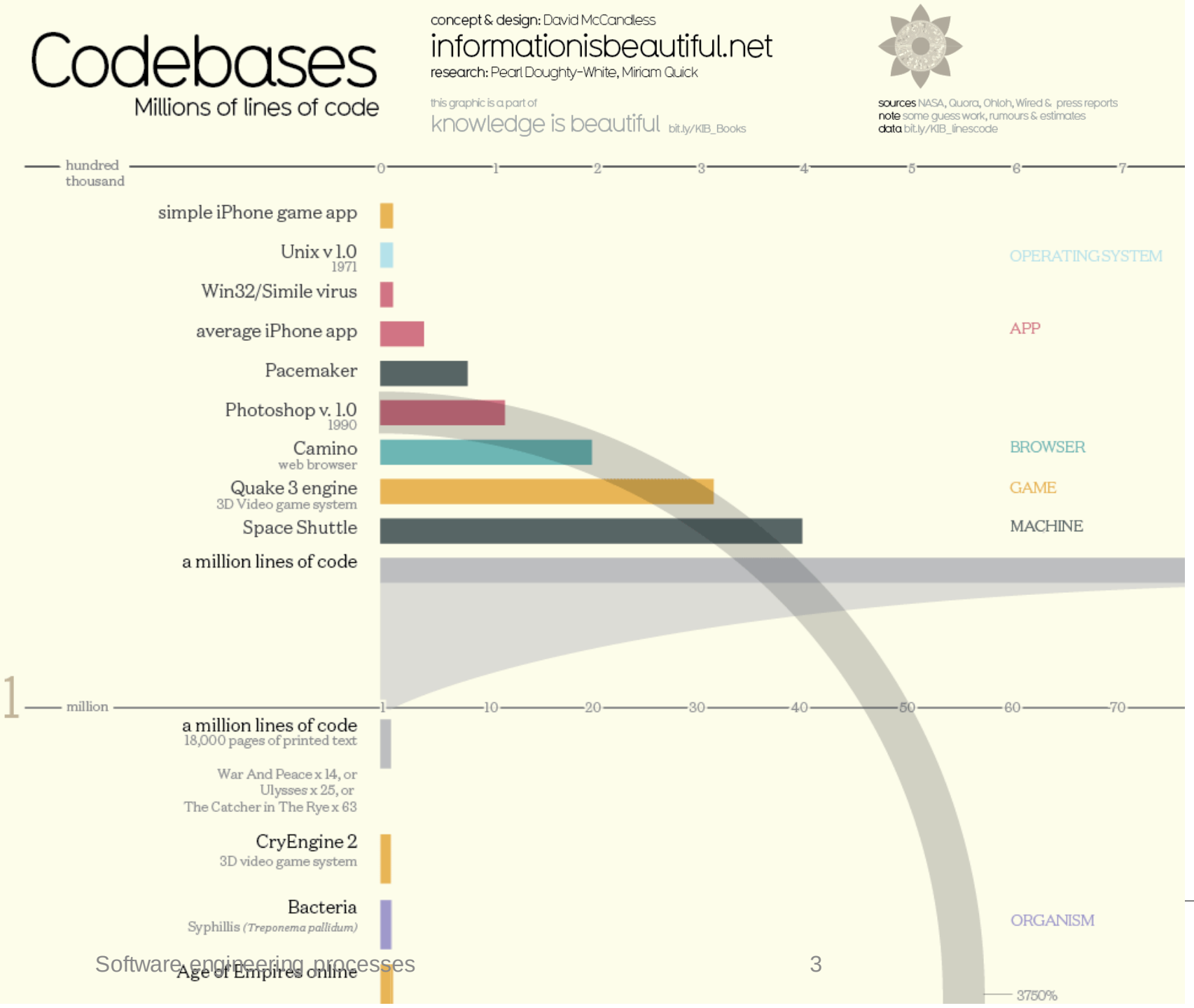
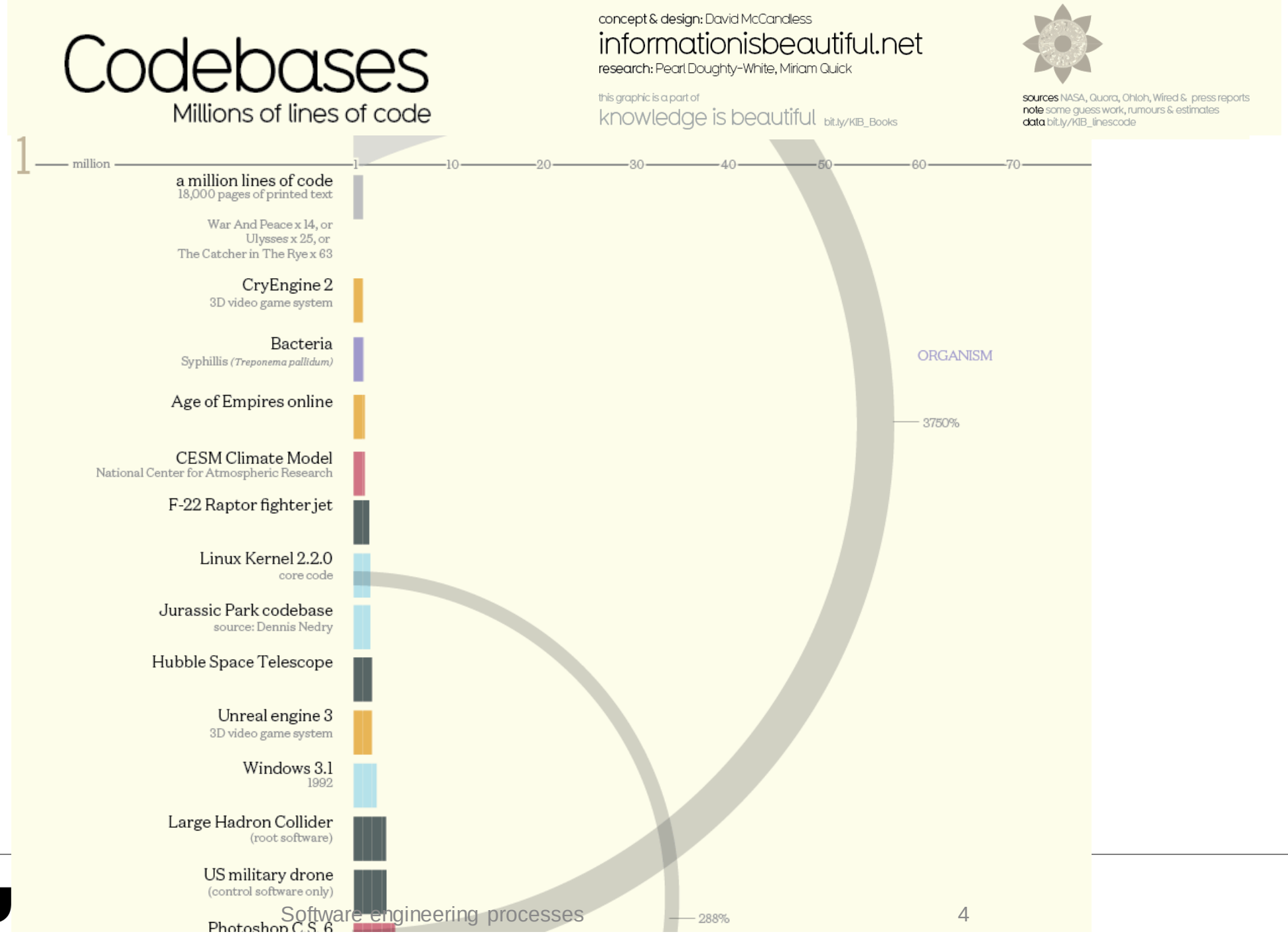
Programming Project with Open Source Code (TDDE52/726A89)

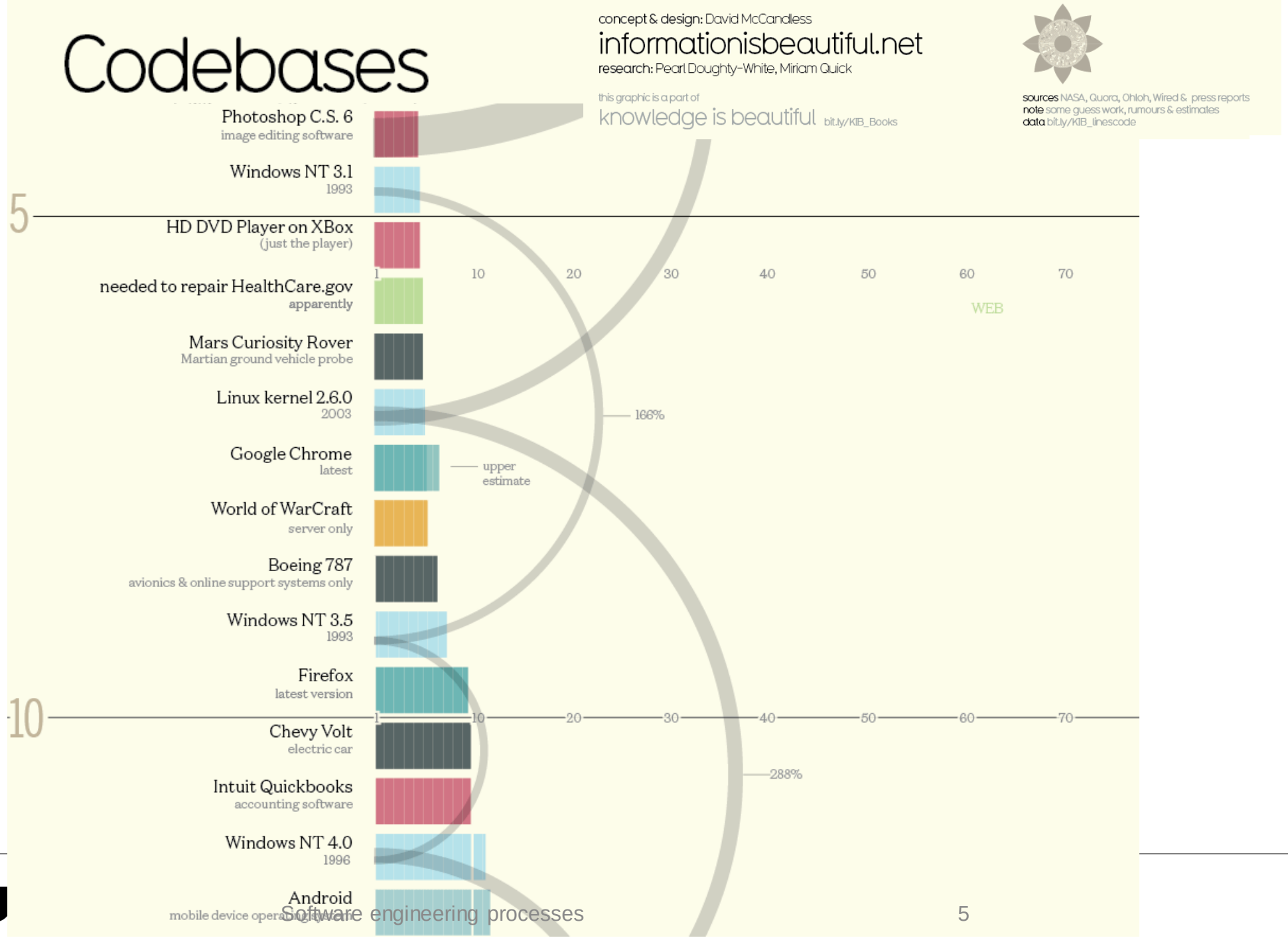
Dániel Varró, Anders Fröberg, Máté Földiák

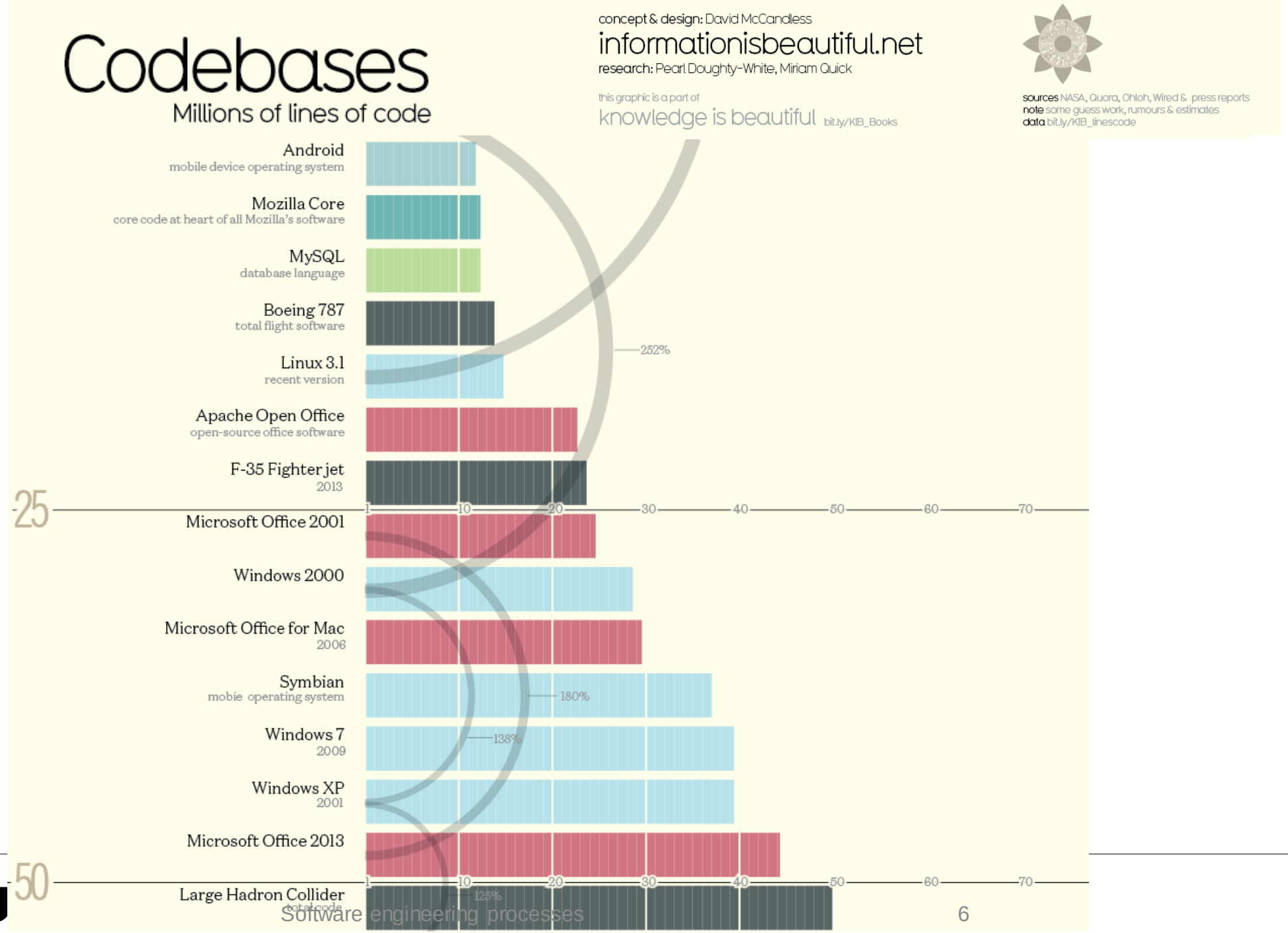
LINKÖPINGS UNIVERSITET

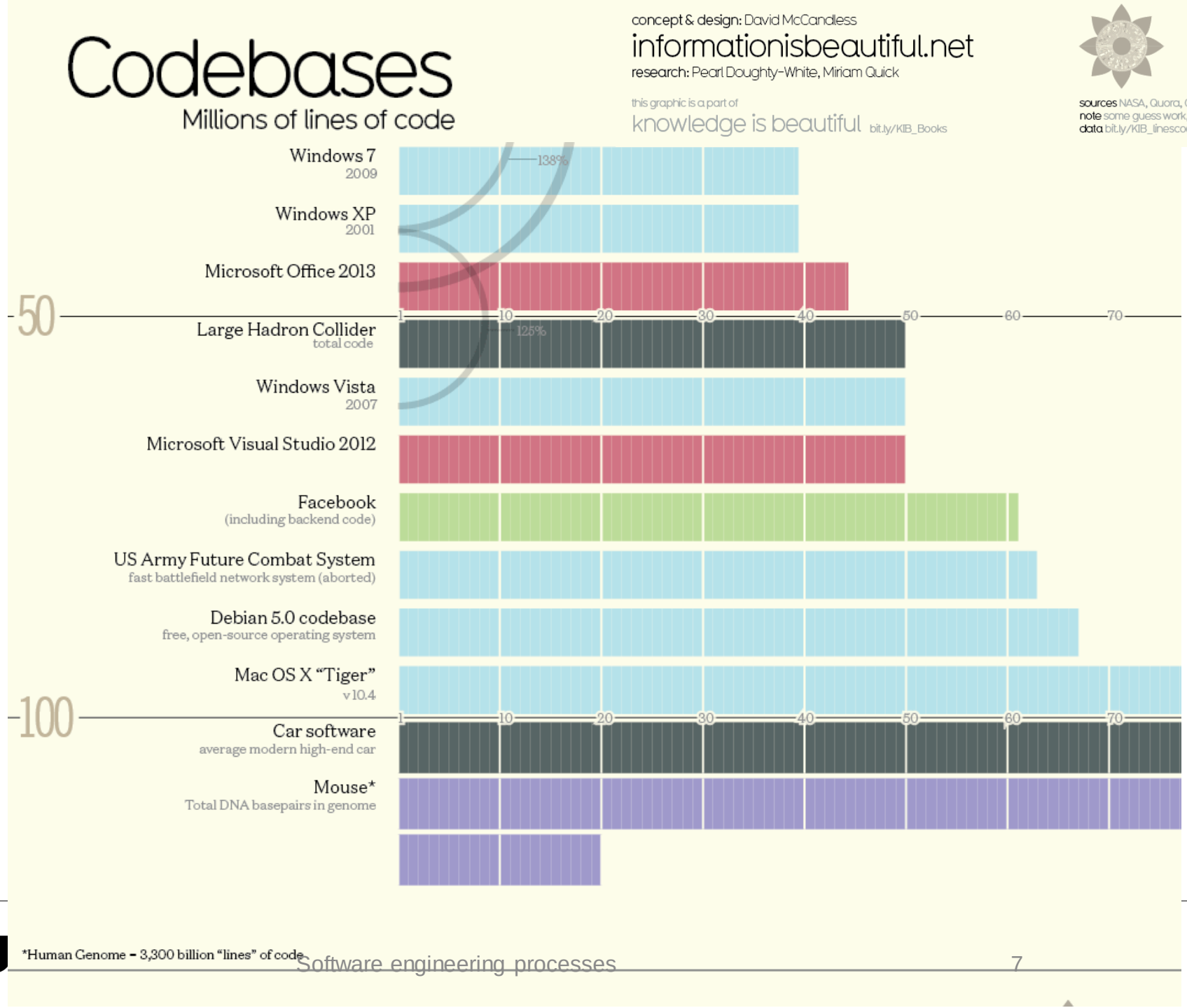
- A note on scale...


Codebases
Millions of lines of code
concept & design: David McCandless
informationisbeautiful.net
research: Pearl Doughty-White, Miriam Quick
this graphic is a part of
knowledge is beautiful bit.ly/KIB_Books
sources NASA, Quora, Ohloh, Wired & press reports
note some guess work, rumours & estimates
data bit.ly/KIB_linescode
hundred thousand
0
1
2
3
4
5
6
7
simple iPhone game app
Unix v 1.0
1971
Win32/Simile virus
average iPhone app
Pacemaker
Photoshop v. 1.0
1990
Camino
web browser
Quake 3 engine
3D Video game system
Space Shuttle
a million lines of code
OPERATING SYSTEM
APP
BROWSER
GAME
MACHINE
1
million
1
10
20
30
40
50
60
70
a million lines of code
18,000 pages of printed text
War And Peace x 14, or
Ulysses x 25, or
The Catcher in The Rye x 63
CryEngine 2
3D video game system
Bacteria
Syphillis (Treponema pallidum)
ORGANISM
Age of Empires online
3750%


Codebases
Millions of lines of code
concept & design: David McCandless
informationisbeautiful.net
research: Pearl Doughty-White, Miriam Quick
this graphic is a part of
knowledge is beautiful bit.ly/KIB_Books
sources NASA, Quora, Ohloh, Wired & press reports
note some guess work, rumours & estimates
data bit.ly/KIB_linescode
1 million
1
10
20
30
40
50
60
70
a million lines of code
18,000 pages of printed text
War And Peace x 14, or
Ulysses x 25, or
The Catcher in The Rye x 63
CryEngine 2
3D video game system
Bacteria
Syphillis (Treponema pallidum)
Age of Empires online
CESM Climate Model
National Center for Atmospheric Research
F-22 Raptor fighter jet
Linux Kernel 2.2.0
core code
Jurassic Park codebase
source: Dennis Nedry
Hubble Space Telescope
Unreal engine 3
3D video game system
Windows 3.1
1992
Large Hadron Collider
(root software)
US military drone
(control software only)
Photoshop C S 6
ORGANISM
3750%
288%


Codebases
concept & design: David McCandless
informationisbeautiful.net
research: Pearl Doughty-White, Miriam Quick
this graphic is a part of
knowledge is beautiful bit.ly/KIB_Books
sources NASA, Quora, Ohloh, Wired & press reports
note some guess work, rumours & estimates
data bit.ly/KIB_linescode
Photoshop C.S. 6
image editing software
Windows NT 3.1
1993
5
HD DVD Player on XBox
(just the player)
needed to repair HealthCare.gov
apparently
Mars Curiosity Rover
Martian ground vehicle probe
Linux kernel 2.6.0
2003
Google Chrome
latest
upper estimate
World of WarCraft
server only
Boeing 787
avionics & online support systems only
Windows NT 3.5
1993
Firefox
latest version
10
Chevy Volt
electric car
Intuit Quickbooks
accounting software
Windows NT 4.0
1996
Android
mobile device operating system
1
10
20
30
40
50
60
70
WEB
166%
288%


Codebases
Millions of lines of code
concept & design: David McCandless
informationisbeautiful.net
research: Pearl Doughty-White, Miriam Quick
this graphic is a part of
knowledge is beautiful bit.ly/KIB_Books
sources NASA, Quora, Ohloh, Wired & press reports
note some guess work, rumours & estimates
data bit.ly/KIB_linescode
Android
mobile device operating system
Mozilla Core
core code at heart of all Mozilla's software
MySQL
database language
Boeing 787
total flight software
Linux 3.1
recent version
252%
Apache Open Office
open-source office software
F-35 Fighter jet
2013
25
1
10
20
30
40
50
60
70
Microsoft Office 2001
Windows 2000
Microsoft Office for Mac
2006
Symbian
mobie operating system
180%
Windows 7
2009
138%
Windows XP
2001
Microsoft Office 2013
50
Large Hadron Collider
total code
125%


Codebases
Millions of lines of code
concept & design: David McCandless
informationisbeautiful.net
research: Pearl Doughty-White, Miriam Quick
this graphic is a part of
knowledge is beautiful bit.ly/KIB_Books
sources NASA, Quora, Ohloh, Wired & press reports
note some guess work, rumours & estimates
data bit.ly/KIB_linescode
Windows 7
2009
138%
Windows XP
2001
Microsoft Office 2013
50
1
10
20
30
40
50
60
70
Large Hadron Collider
total code
125%
Windows Vista
2007
Microsoft Visual Studio 2012
Facebook
(including backend code)
US Army Future Combat System
fast battlefield network system (aborted)
Debian 5.0 codebase
free, open-source operating system
Mac OS X "Tiger"
v 10.4
100
1
10
20
30
40
50
60
70
Car software
average modern high-end car
Mouse*
Total DNA basepairs in genome
*Human Genome = 3,300 billion "lines" of code

Characteristics of ultra-large-scale systems

- Everything is decentralized (data, control,...)
- Requirements: conflicting, diverse, unkown,...
- Continuous evolution
- Heterogeneous, inconsistent and changing elements.
- Eroded people-system boundary
- Failure is the norm

Adapted from Northrop et al. 2006, CMU

- How can we study this at university?

Agenda

- Course goals, requirements
- Weekly plan
- Tools and meetings
- Forming teams

Course goals

1. Use existing conventions and follow established processes to **contribute through software to a distributed, large-scale development project**.

2. Present changes and updates so external parties may approve submissions.

3. Create a time plan and monitor progress through a common development project

4. Use appropriate tools for contemporary, large-scale software development

5. **Independently acquire new knowledge and skills** in order to contribute to a large-scale software project.

Informal goals

- Portfolio of contributions
- Hands-on experience
- Professional and technical skills
- Creative, hopefully fun and self-steered
- Demonstrating how great LiU students are to the world!

What happens in practice during the course

- You choose a project to work on.
- You make plans for how to contribute to the project.
- We meet every week to discuss how it has been going, try to support each other and make plans for next week.
- Between meetings you work with the project and use Teams to communicate with the rest of the team.

Projects

- OSS projects (Github, Gitlab, SourceForge, …) of sufficient size:
  - 30+ developers,
  - 1000+ commits,
  - actively maintained during the last three months by at least 15 developers
- Project type/difficulty will determine grade possible

Points table

Level	Technical difficulty	Process difficulty	Development process	Individual contributions
Easy	6	3	4	8
Medium	12	6	8	16
Expert	18	9	12	24

Grades

- 21-34 points: grade 3
- 35-48 points: grade 4
- 49-63 points: grade 5
  - Example: Lowest technical and process difficulty and highest points on development process and individual contributions: 6+3+12+24 = 45, grade 4
  - Example: Highest technical and process difficulty and lowest points on development process and individual contributions: 18+9+4+8 = 37, grade 4

Technical difficulty	Build and test environment	Domain knowledge requirements	Size
Easy	No configuration needed	No expertise apart from course-level knowledge required	50-400 KLOC, 0-3 external static libraries required.
Medium	File settings and manual build steps	A few specific new algorithms or tools to learn	400 KLOC-1MLOC, several components built and tested against both internal and external components
Hard	Custom configuration languages, build tools	Several weeks or months of study estimated for initial contributions	More than 1 MLOC, several components shipped in multiple versions with dependencies on each other as well as external components.

Process difficulty	Issue tracker	Documentation	Communications
Easy	Contains issues tagged as suitable for beginners	Contains guides and tutorials for new developers	Contains specific communication channels for new developers
Medium	Contains issues marked with difficulty	Contains introduction chapters and suggested readings in documentation	Contains open communication channels
Hard	No metadata on difficulty in issue tracker	No clear introduction to new developers	Restricted communication channels

Development process	Group contributions	Planning and reflection	Engineering practice
Easy	Shares code, tutorials and development tips to other members of the team. Actively participates in project meetings.	Creates a plan for each sprint with sufficient details to guide the work, amounting to a number of hours that correspond the amount of credits given by the course.	Uses the required set of tools and suggested practices for code contributions in the host project
Medium	Shares documentation and material, and also gives valuable feedback on others' contributions	Uses the plan actively and updates the plan to take into account new information and events	Takes care to follow established good practices in continuous development and testing
Hard	Actively helps team members improve, and provides tutorials or other help to team members	Uses external sources from the host project to validate the feasibility of the plan and to update accordingly	Contributes with general quality improvements for the process of developing, building and testing software in the project

Contributions	Quantity of code contributions	Complexity of code contributions	External communications
Easy	At least one contribution that has been accepted by the project	Non-trivial code contribution	Has been able to communicate successfully with at least one developer in the external project on a suggested contribution
Medium	Several contributions that has been accepted by the project	Code contributions that solve different problems or a moderately challenging problem	Has participated in several discussions with project members before and after own contributions
Hard	Several types of contributions (e.g. new features, bug fixes or documentation) that has been accepted by the project	Code contributions that solve a challenging problem	Has successfully proposed new features to the project

Next steps

- Select a project that you would like to contribute to. You will also form X number of teams, possibly with the same project that you work on.
- Write individual project plans, with general ambitions and detailed plans for the first sprint, and additional information before each coming sprint
  - Deadline: **September 11 23:59**
  - Discussion on preliminary/draft plans on first meeting (Sept 6)
- 45 minutes/week for discussing progress within each group

Course evaluation and changes

Schedule

Week	Session
35	This lecture
36	First round of project selections
37	Planning report review
38-41	Weekly meetings
42	Mid-semester review
45-50	Weekly meetings
51	Final presentation (Dec 18)

Experiences from last years

- **Zulip** - Good and welcoming, open communication, would choose again.
- **OpenRCT** - Good general documentation on the process, but not about the actual project, good communication (discord), Good to play the game (or use the sw), can recommend.
- **NIFI** – Good but requires (significant) domain knowledge
- **Strapi** - Difficult communicaiton, hard to get in.
- **Oppia** - Welcoming community, good documentation.
- **Bootstrap -** Core-team hard to communicate with
- **Bitwarden -** Have to get approval, good to communicate with.

Other projects from the past

- Material-UI
- Oppia
- Godot
- Atom
- Electron

Discussion of project proposals

Next steps

- Select an OSS project and start drafting a plan
- You are free to choose projects as you like, but you will be graded on your ability to help each other out in your student teams as well as your contributions to the project
- Good if people with same/similar project in same group
- Preliminary grouping in next meeting