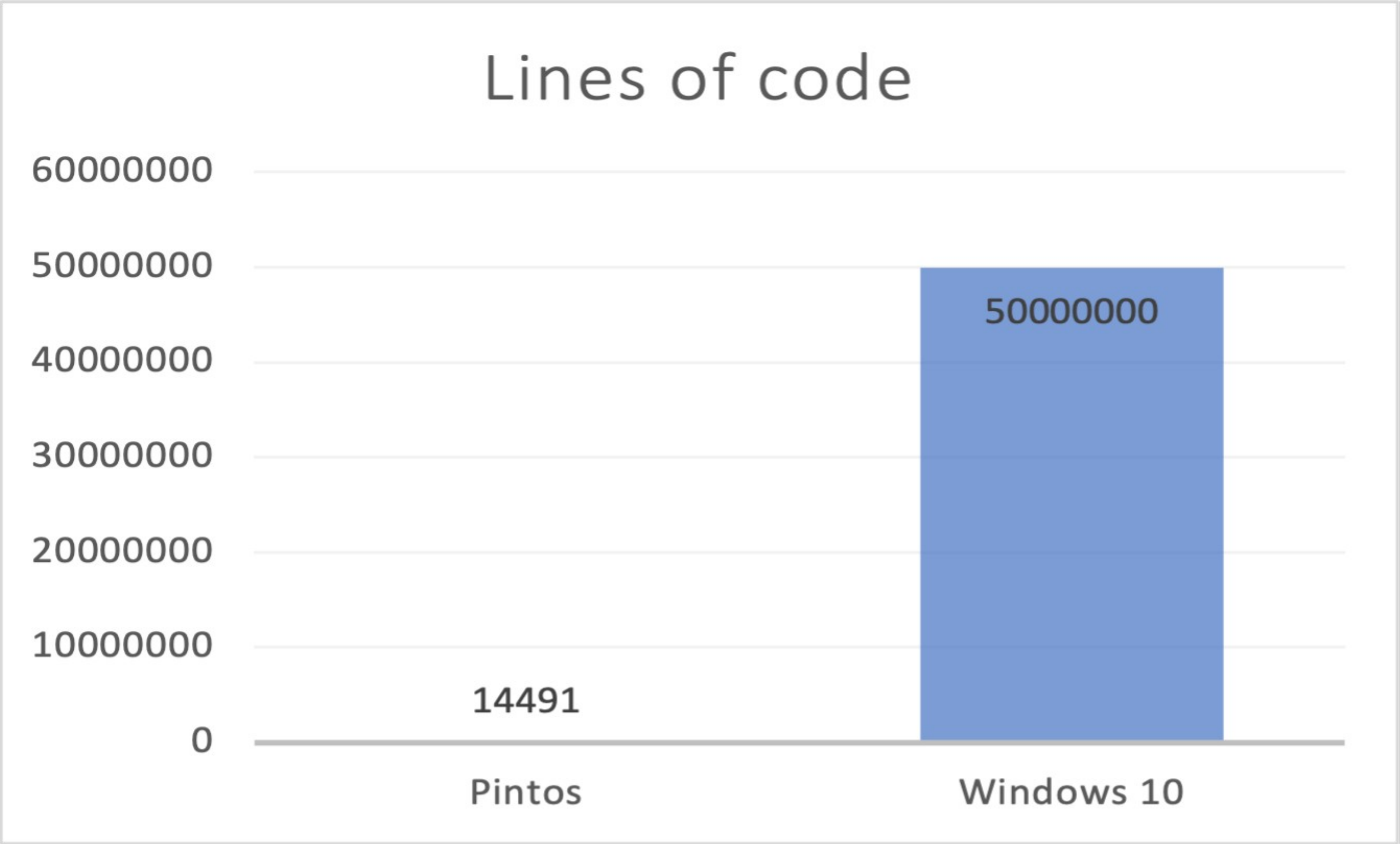
Programming Project with Open Source Code (TDDE52/726A89)

Mikael Asplund & Anders Fröberg

LINKÖPINGS UNIVERSITET

A note on scale...


Lines of code
60000000
50000000
40000000
30000000
20000000
10000000
0
14491
50000000
Pintos
Windows 10

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!

Projects

- OSS projects (Github, 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	7	3	4	8
Medium	14	6	8	16
Expert	21	9	12	24

Grades

- 22-36 points: grade 3
- 37-51 points: grade 4
- 52-66 points: grade 5
  - Example: Lowest technical and process difficulty and highest points on development process and individual contributions: 7+3+12+24 = 46, grade 4
  - Example: Highest technical and process difficulty and lowest points on development process and individual contributions: 21+9+4+8 = 46, grade 4

Technical difficulty	Build environment	Domain knowledge requirements	Size	Activity
Easy	no configuration needed	no expertise apart from course-level knowledge required	50-400 KLOC, 0-3 external static libraries required.	Developed by at least 15 developers in the last three months. At least 30 developers in total. 1000+commits.
Medium	file settings and 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	Developed by at least 20 developers in the last three months. At least 50 developers in total. 3000+commits.
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.	See Easy

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	Internal process
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	External code contributions	External communications
Easy	has submitted at least one non-trivial code contribution that has been accepted by the host project	has been able to communicate successfully with at least one developer in the external project on a suggested contribution
Medium	has committed several non-trivial code contributions accepted by the project	has participated in several discussions with project members before and after own contributions
Hard	has committed several types of significant contributions (e.g. new features, bug fixes or documentation) to the project, that have been generally approved or appreciated	has successfully proposed new features to the project

Course outline

- Select a project that you would like to contribute to. You will also form 2-2-4 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 13 23:59**
  - Discussion on preliminary/draft plans on first meeting (Sept 8)
- 45 minutes/week for discussing progress within each group

Course evaluation and changes

Schedule

week	Activity	Seminar
36	Contribution plan submission, forming teams	
37		Contribution plan review
38		Mid-sprint 1
39		Sprint 1 review
40		Mid-sprint 2
41		Sprint 2 review
…		…
51		Public presentation of contributions

Example projects

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

Discussion of project proposals

Forming teams

- Select an OSS project and write 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