issel
Intelligent Systems & Software Engineering Labgroup

ARISTOTLE UNIVERSITY OF THESSALONIKI

SOFTENG GROUP
Software Engineering Research

Upon assessing the **software quality** of open source **multimedia tools**

Keynote @ MMM 2019, Thessaloniki

Andreas L. Symeonidis and Michail Papamichail
Softeng group
Electrical and Computer Engineering
Aristotle University of Thessaloniki, Greece
asymeon@eng.auth.gr, mpapamic@issel.ee.auth.gr

Outline

- The multimedia software landscape – suites and tools
- Key questions to answer when adopting/adapting/developing a multimedia tool
- The Software engineering viewpoint when answering your key questions
- On making data-driven decisions
- Possible extensions to the presented approach

The multimedia world as we stand

Or else, what is the trend now in multimedia suites/tools development

Types of multimedia software in terms of scope

- Software Suites/Standalone platforms (hereon suites)
  - Software that exhibits complete functionality, usually oriented towards domain end-users
  - Typical cases: Photoshop, Blender, Audacity etc
- Software projects that act as tools or add-ins (hereon tools)
  - Software that is developed to serve a specific purpose
    - Answer a specific research question
    - Provide specific functionality for an application
  - Typical cases: Android-based libraries/applications, online multimedia editors

Proprietary vs Open Source:
a proprietary business going open…

- Traditionally, multimedia software suites require resources and computational power, thus developing such software requires optimization.
- For many years, this was a land for the "few and the skilled"
- However, during the last 5 years Open source alternatives have been developed and are gaining traction
  - Inkscape (OSS alternative for Adobe Illustrator functionality)
  - GIMP (OSS alternative for Adobe Photoshop functionality)
  - Avidemux (OSS alternative for Adobe Premiere functionality)
  - OpenCV (OSS alternative for Matlab multimedia functionality)

Popular open source multimedia suites…

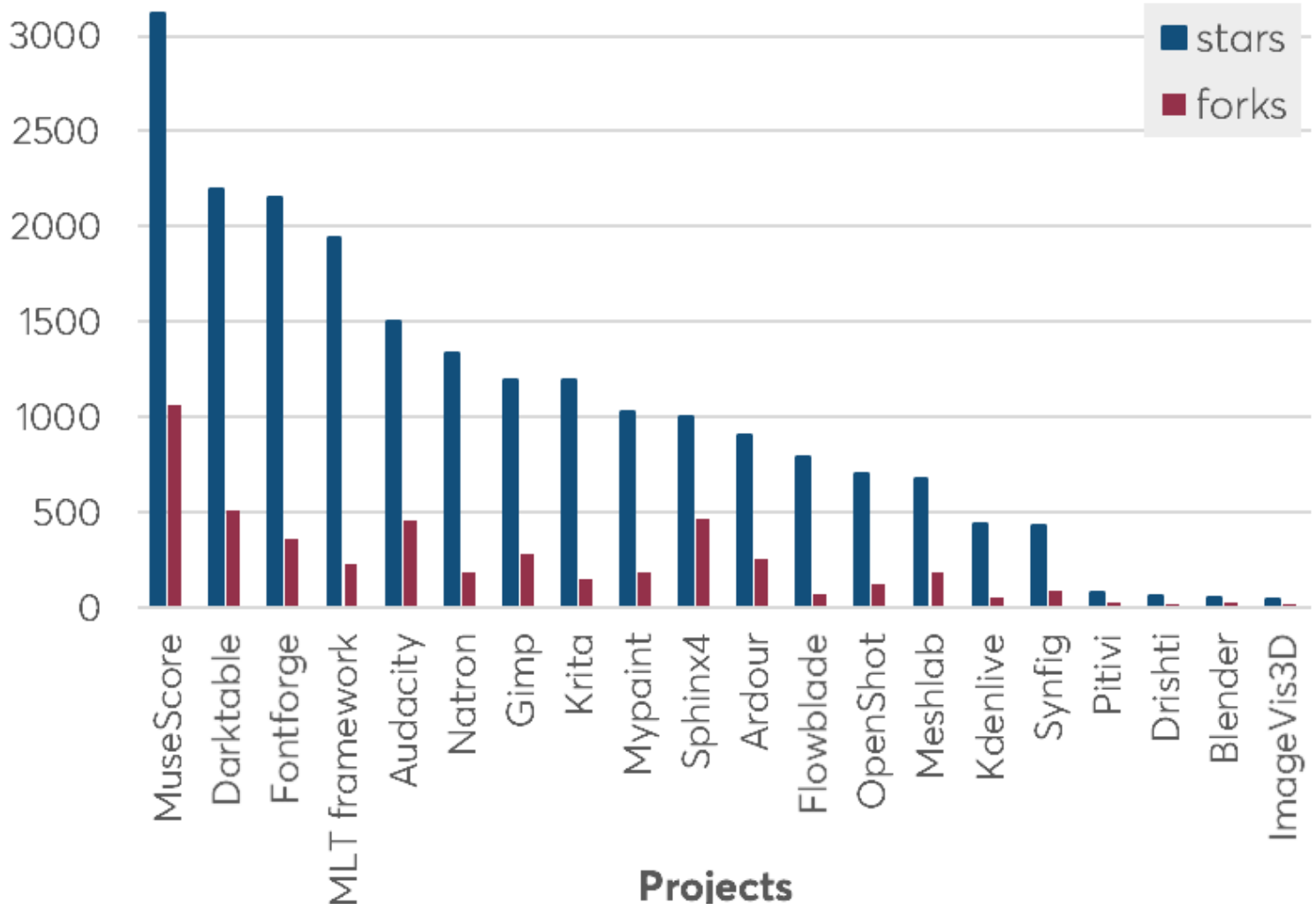
TOOL NAME	TYPE
GIMP	PD/Im
MYPAINT	PD/Im
BLENDER	3D
KRITA	PD/Im
DARKTABLE	Im
SYNFIG STUDIO	Vi
AUDACITY	So
ARDOUR	So
SPHINX	So
MUSESCORE	So

- Painting & Drawing (PD)
- 3D Modeling (3D)
- Image editing (Im)
- Video editing (Vi)
- Sound editing (So)

TOOL NAME	TYPE
FLOWBLADE	Im/Vi
KDENLIVE	Im/Vi
PITIVI	Im/Vi
SHORTCUT	Im/Vi
OPENSHOT-QT	Im/Vi
NATRON	Im/Vi
FONTFORGE	Im/Vi
MESHLAB	Im/Vi
IMAGEVIS3D	Im/Vi
DRISHTI	Im/Vi

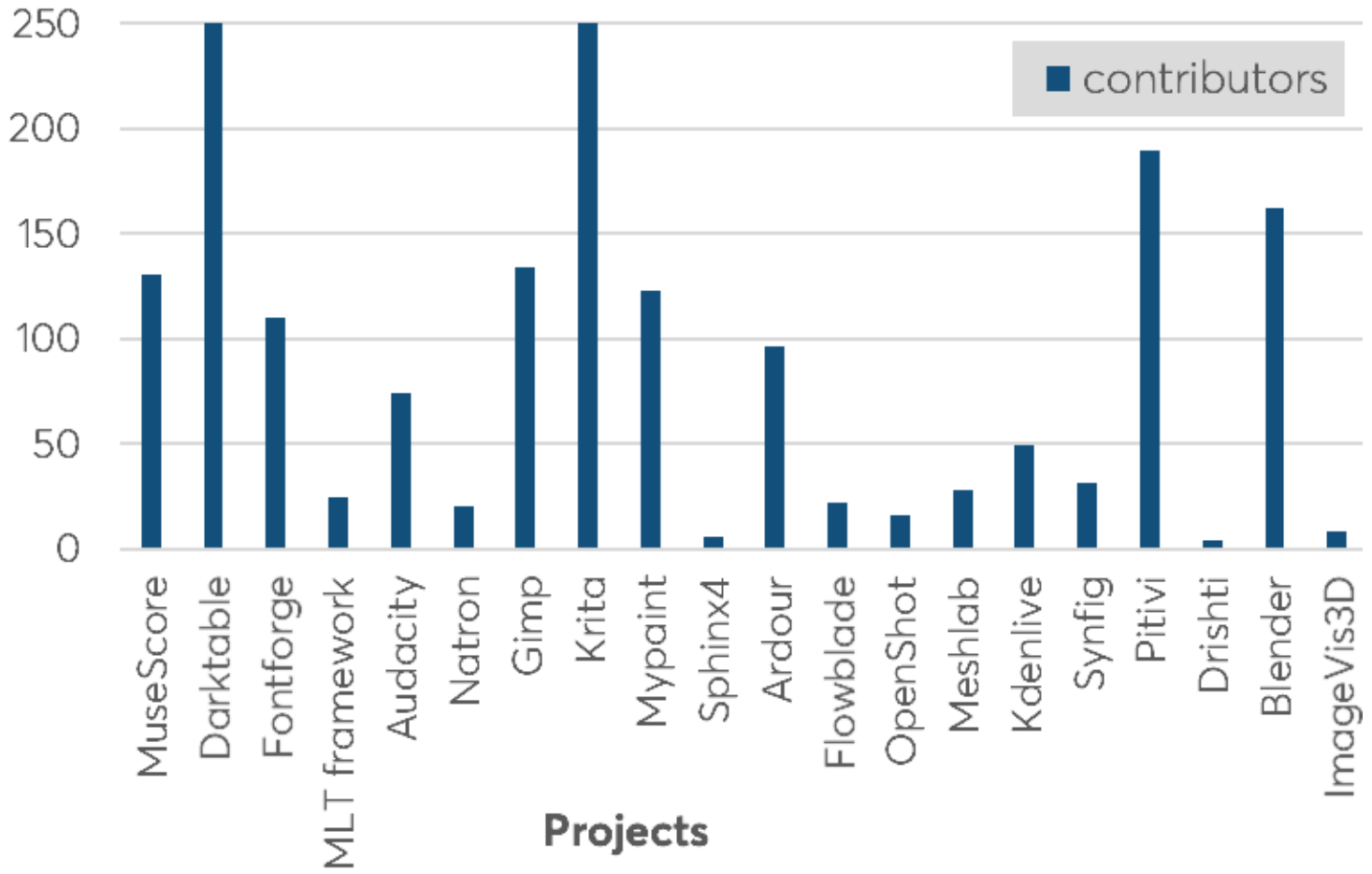
OS Multimedia suites: popularity and uptake

MIN Stars	46
MAX Stars	3117
AVG Stars	1044
MIN Forks	18
MAX Forks	1064
AVG Forks	238



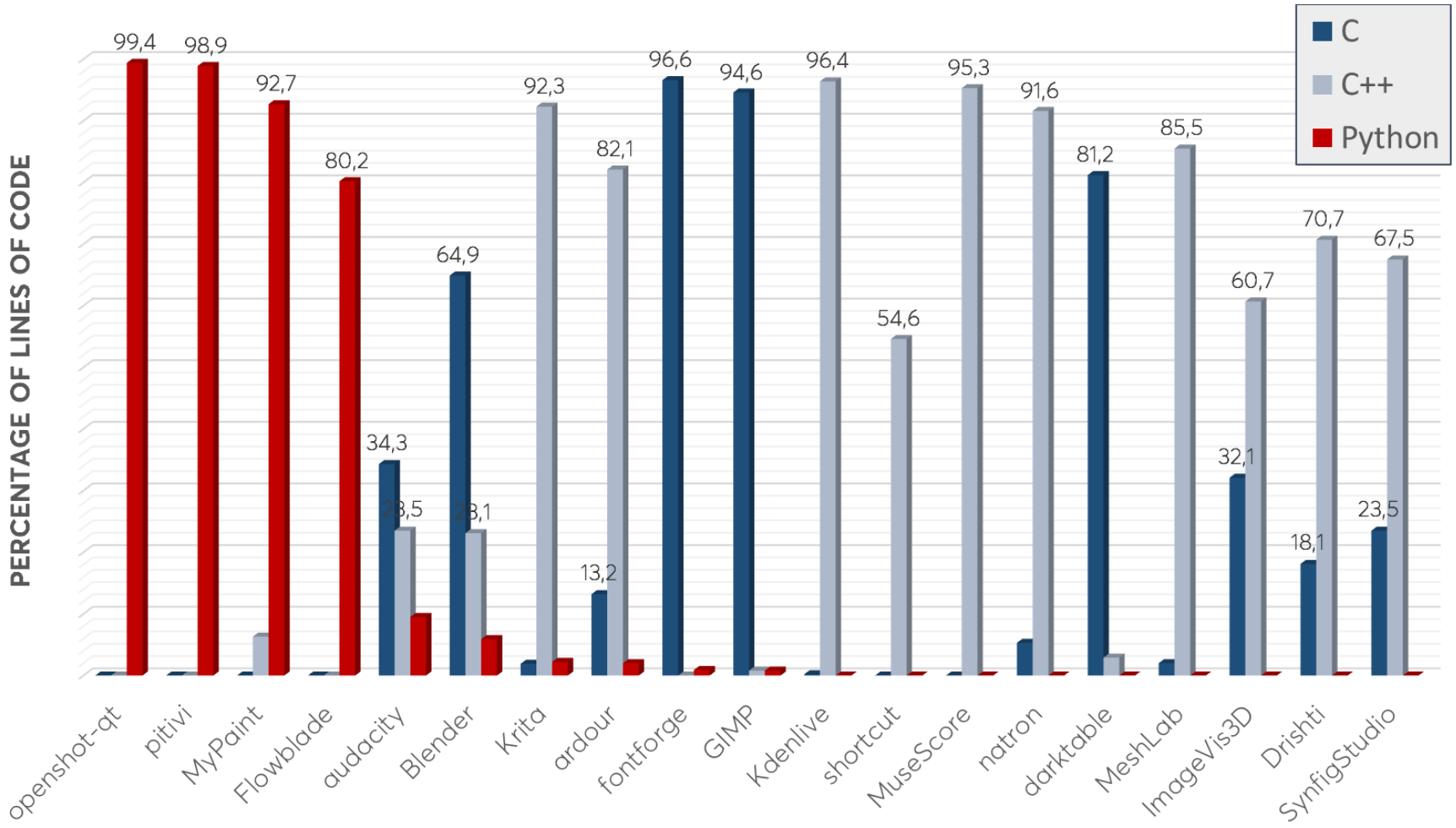
OS Multimedia suites: support & developer community

MIN Contributors	4
MAX Contributors	250
AVG Contributors	86
MIN Releases	3
MAX Releases	247
AVG Releases	59



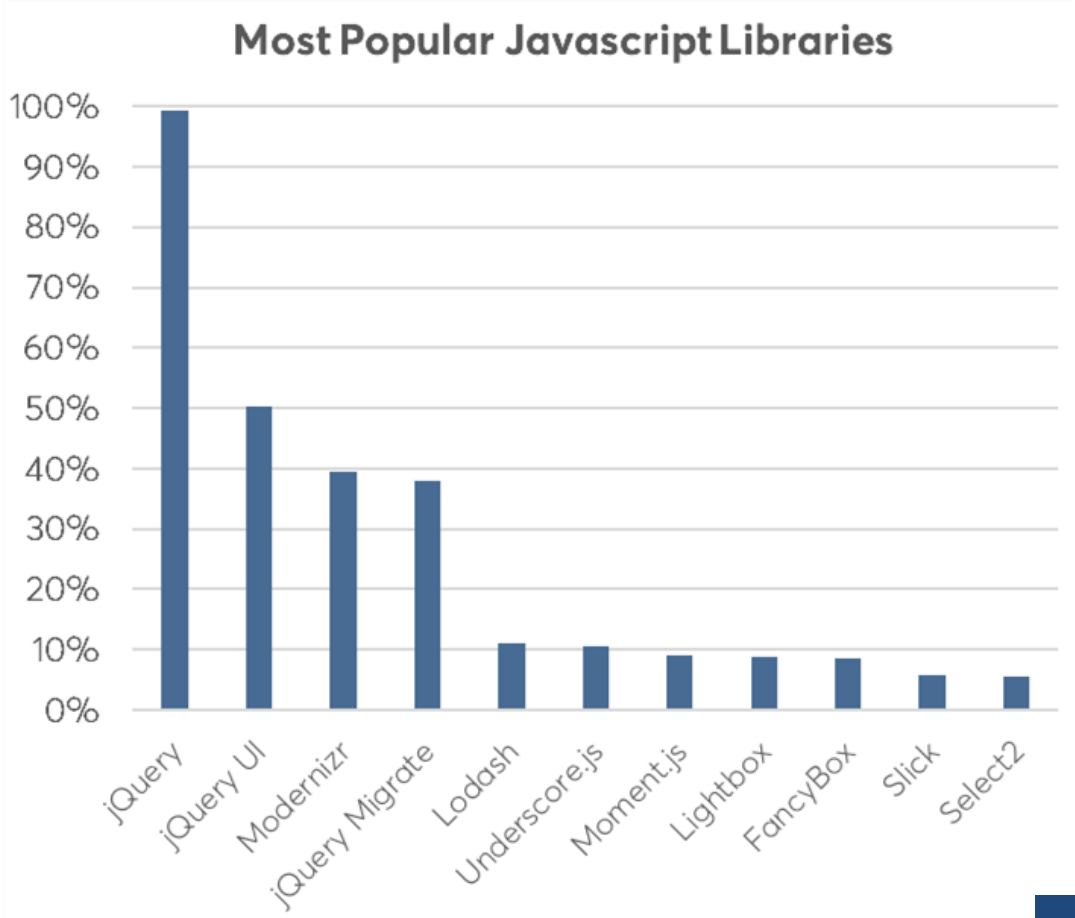
Popular OS Multimedia suites: coding languages

- C/C++ are the leading programming languages for multimedia suites
- Multimedia suites written in Python are an upcoming (all inclusive) trend



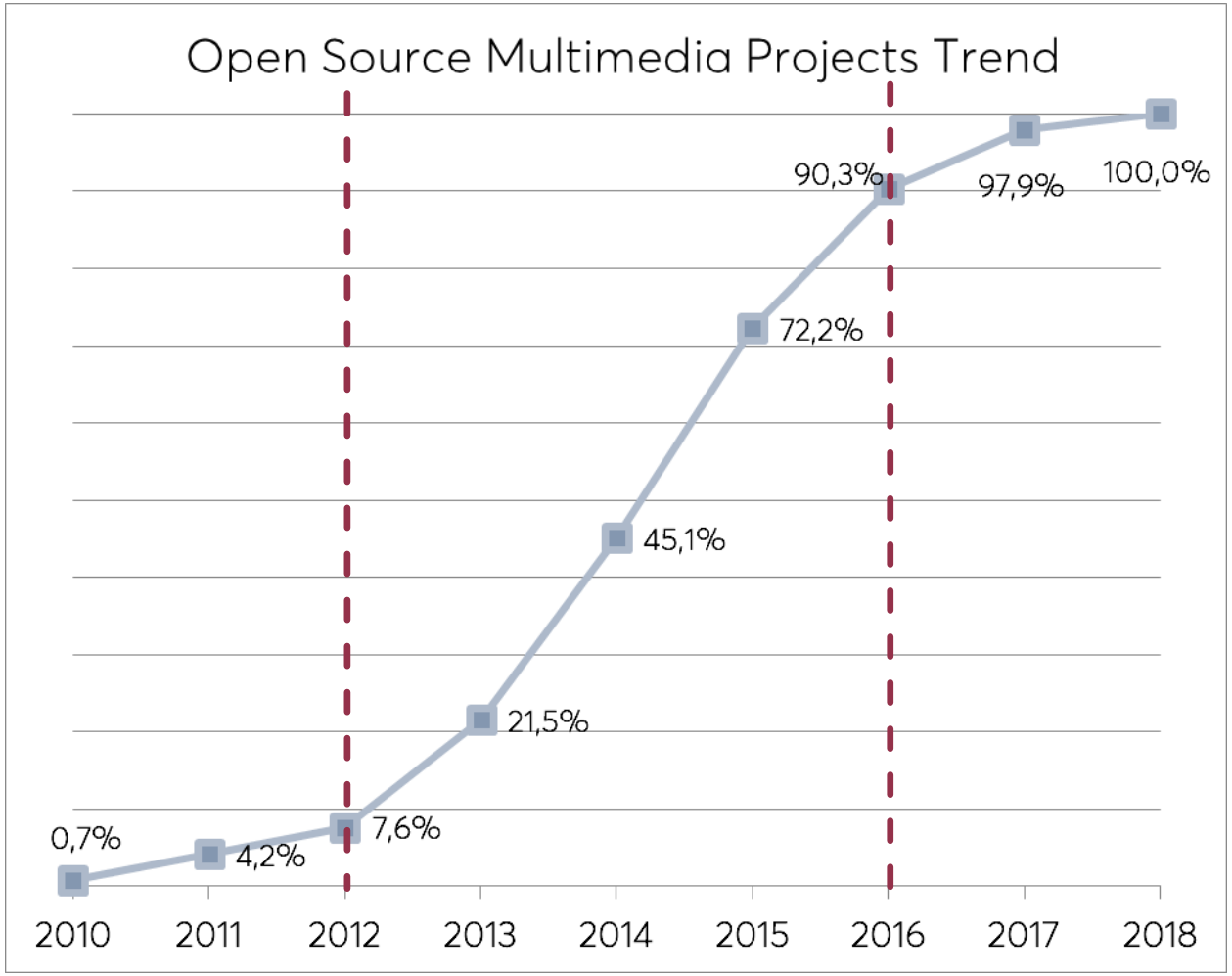
Popular open source multimedia tools

- All multimedia tools contained in the 1000 most starred GitHub projects
- 114 multimedia-related projects (11,4%)
- Top categories:
  - Animation
  - Video
  - Image
  - Graph/Network analysis
- Dominant languages:
  - Java for mobile development
  - Javascript for web development



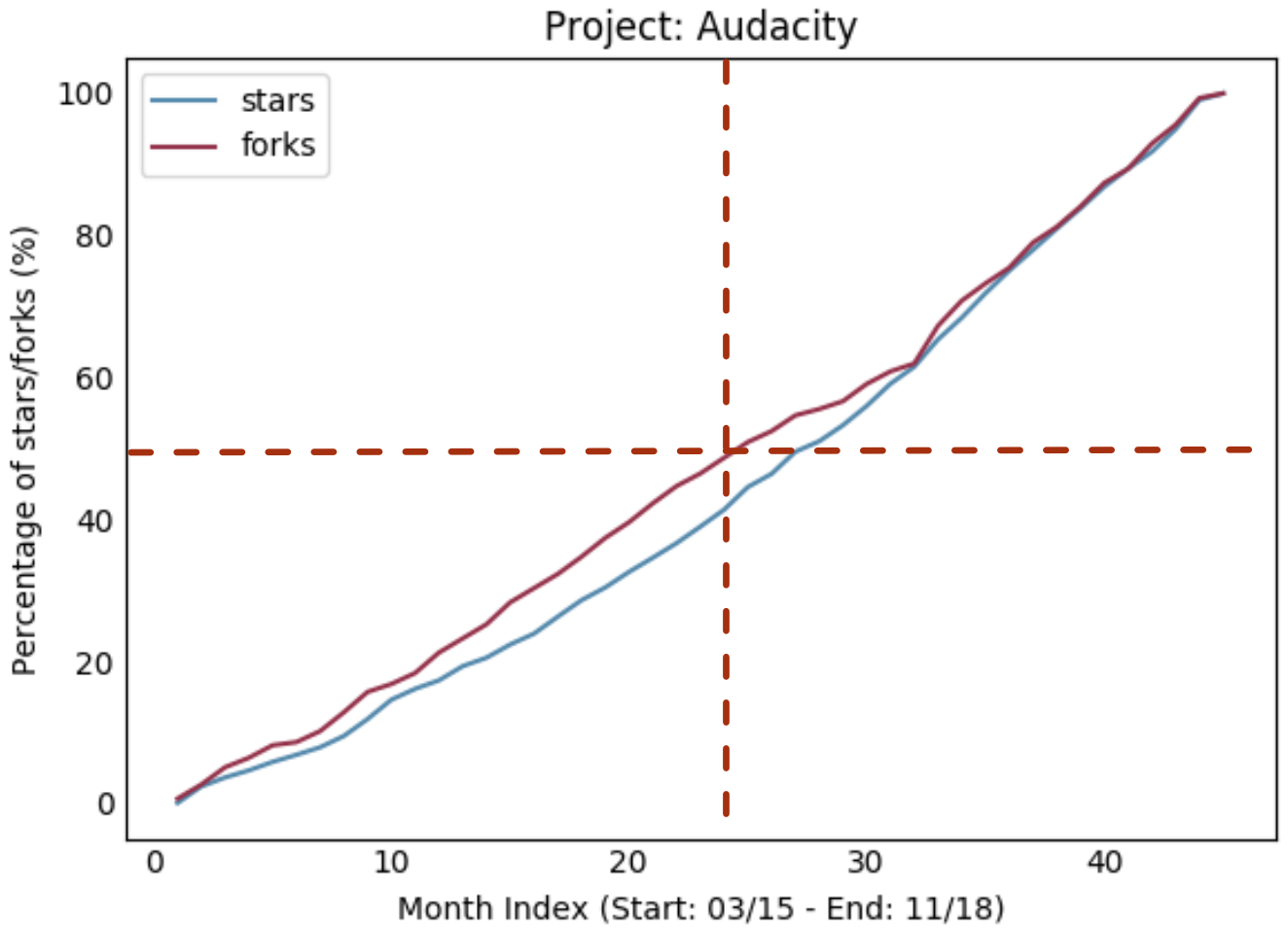
The growing popularity of OS Multimedia tools

- OS multimedia tools started gaining popularity around 2010
- From that point on, the number of popular multimedia tools is increasing rapidly.
- Multimedia add-ins evolution shows a 2-year adoption (becoming popular) rate



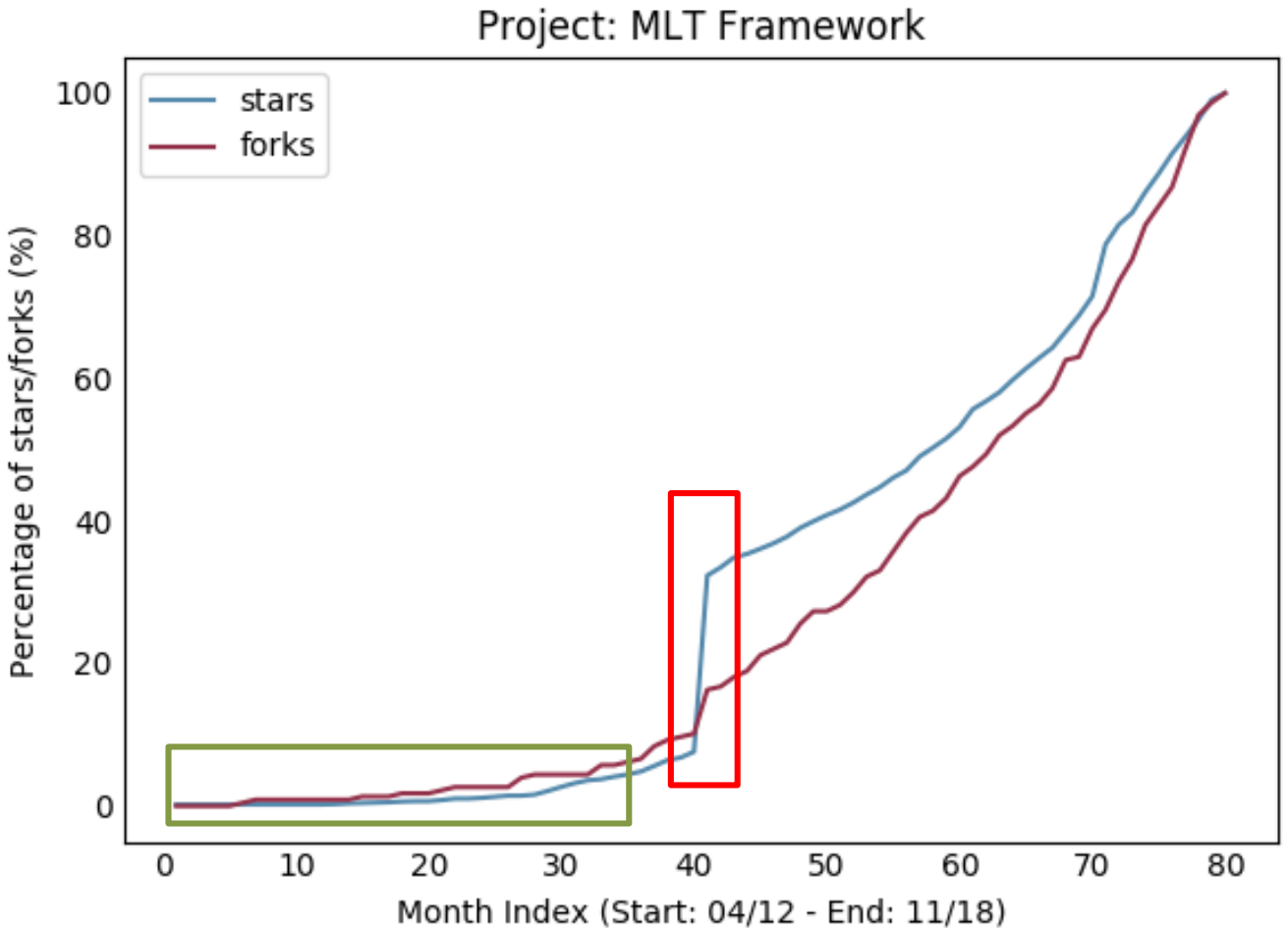
How popularity usually grows: a typical case…

- Tool: Audacity
- Period: 03/15 – 11/18
- Linear/analogous evolution of stars and forks
- Almost 24 months to gain 50% popularity and uptake from the developer community



How popularity usually grows: a market-driven case…

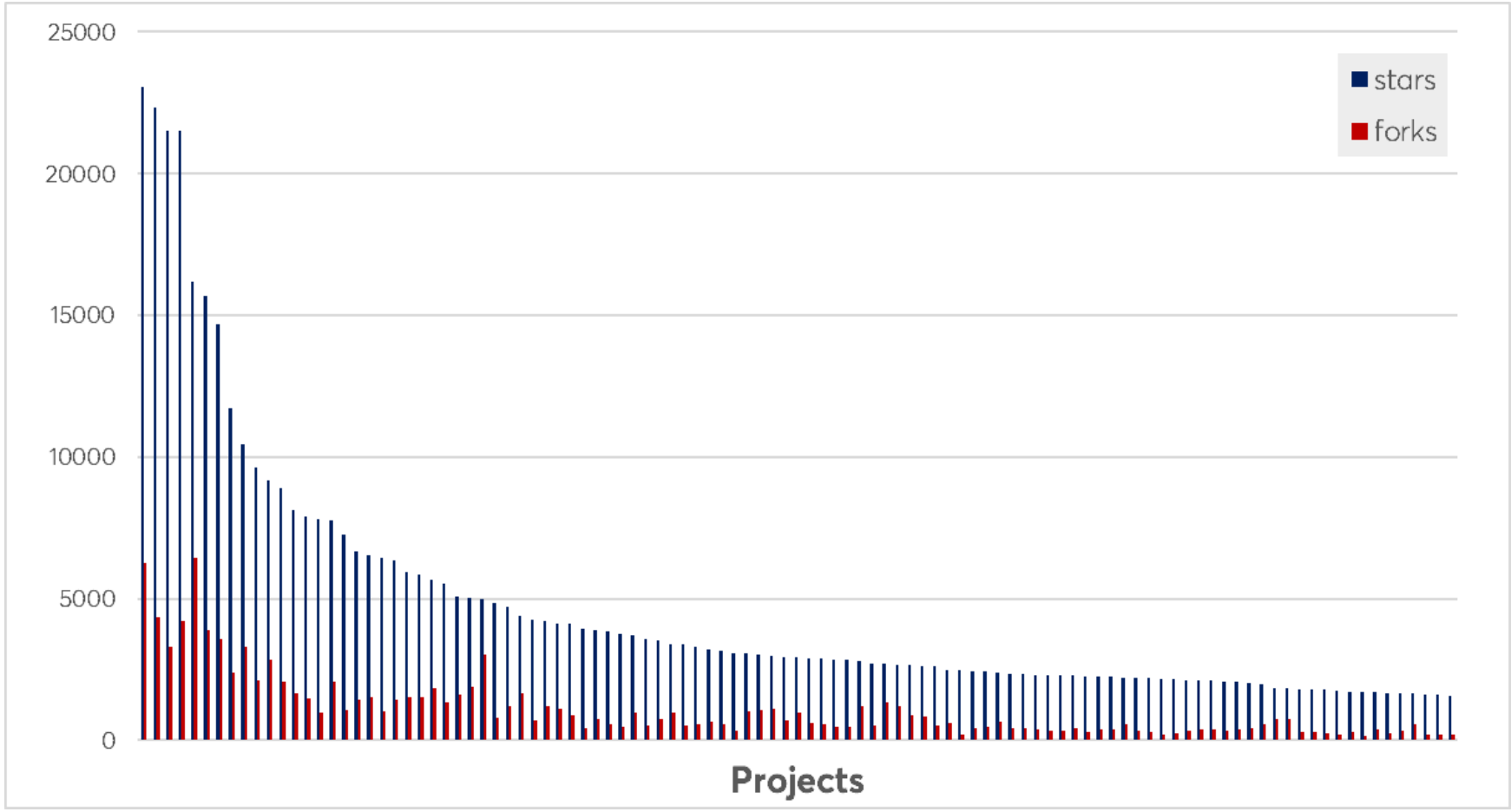
- Tool: MLT framework
- Period: 04/12 – 11/18
- Gained 8x more stars during 4 months (months 40-43) than in the time period until month 40
- Reason: major marketing and branding campaign launched, supported by big players (Facebook)



OS Multimedia tools: community uptake

MIN Stars	1583
MAX Stars	23070
AVG Stars	4695
MIN Forks	151
MAX Forks	6453
AVG Forks	1091

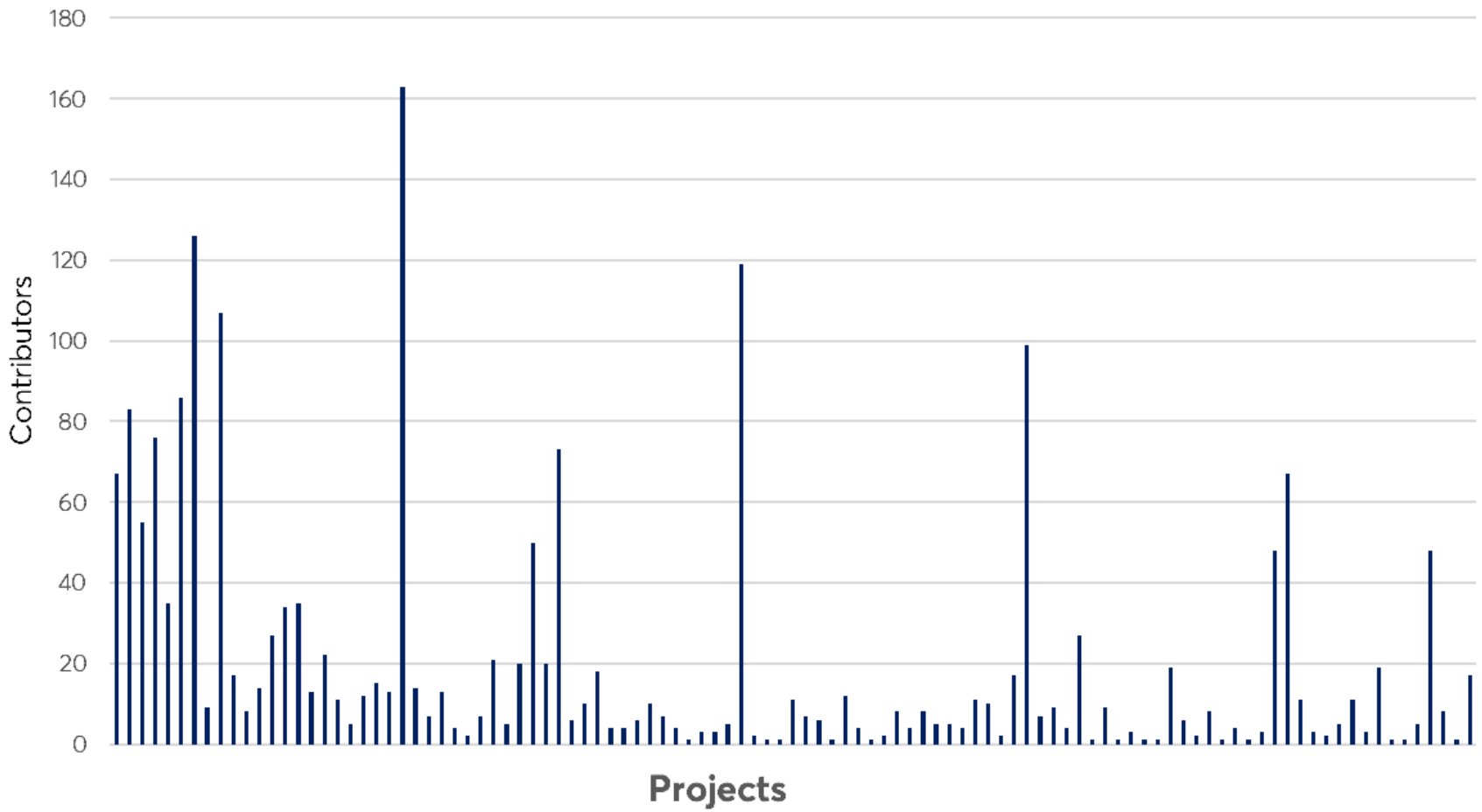
10 times more uptake wrt multimedia suites!



OS Multimedia tools: developer support

MIN Contributors	1
MAX Contributors	163
AVG Contributors	19

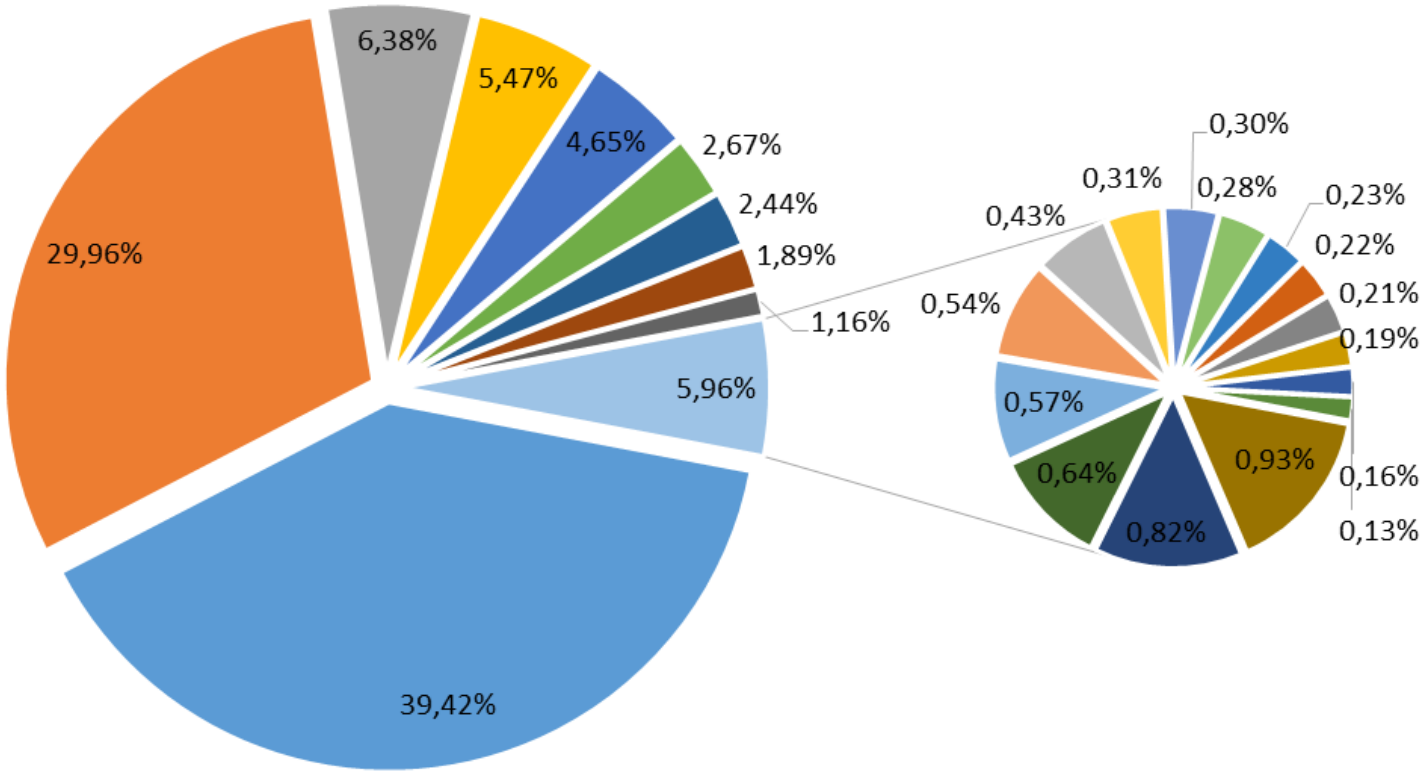
4 times less contributors on average!



OS Multimedia tools: support & developer community

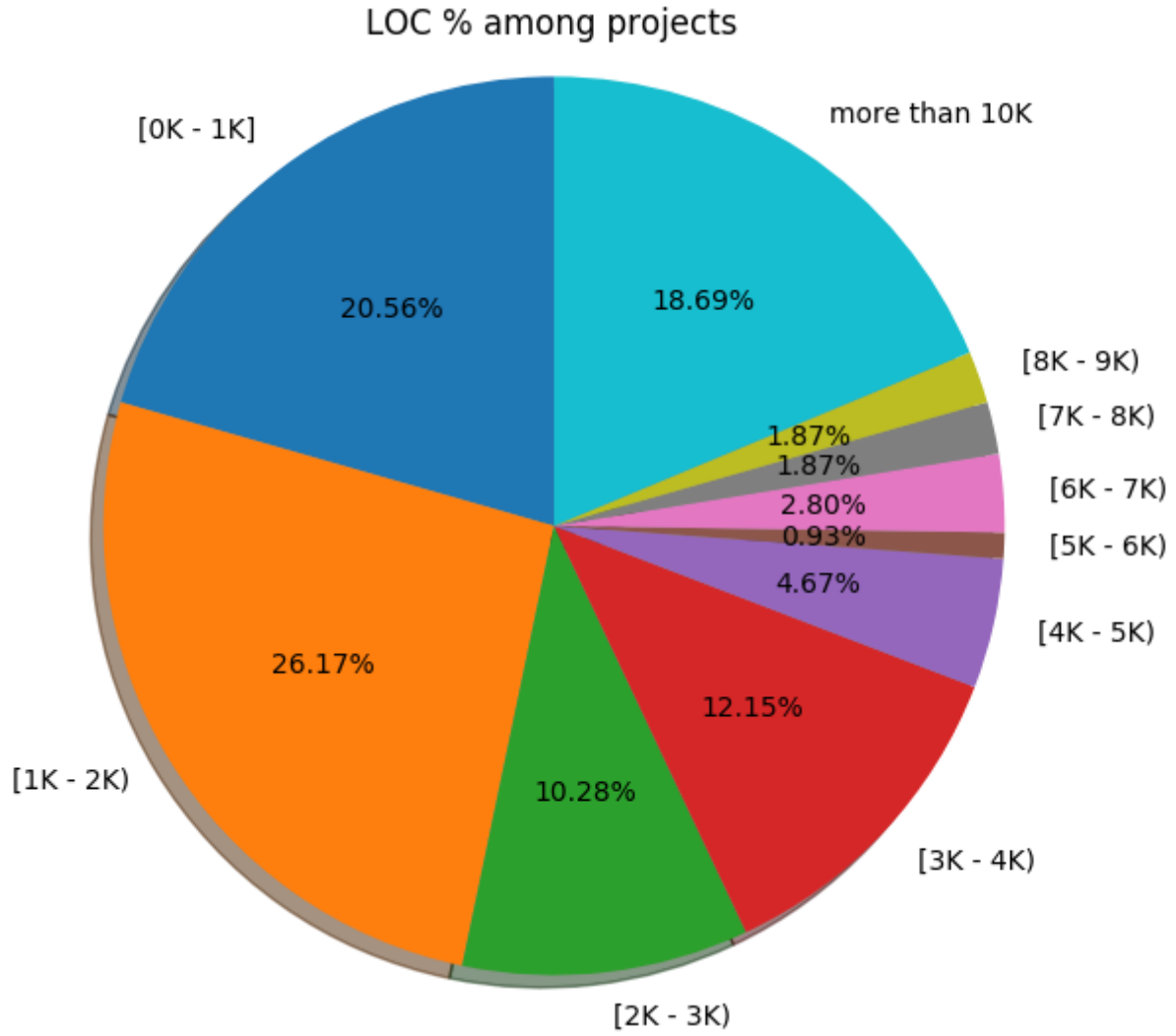
- Tool: Ardour
- 96 contributors in total
- 2 lead contributors with 69,38% of contributions
- 9,4% (9 contributors with more than 1% commits)
- In the end, a committed team is essential to ensure that the project runs properly

Project Ardour: Contribution analysis



Popular OS Multimedia tools: project size

- Size doesn't really matter
  - >45% of projects contain less than 2K LOC
  - ~20% are big projects (>10K LOC)



Upon developing OS multimedia tools: why is this information important?

Practically, when you want to develop a multimedia tool (add-on, application, research prototype), you should probably be asking the following questions:

Q1 Which library/api should I embed into my software tool in order to solve the problem efficiently and effectively?

Q2 A paper I found provides a link to the prototype developed. Would it be easy to test it/adapt it, or is it going to take ages to understand what the tool does?

Q3 I have a great idea for an add-on to the xxx tool. Is it easy to build?

This is where software analytics are applied!

The merits of applying software analytics

Data deluge at its best!

The Software engineering domain: an oasis for the data analyst

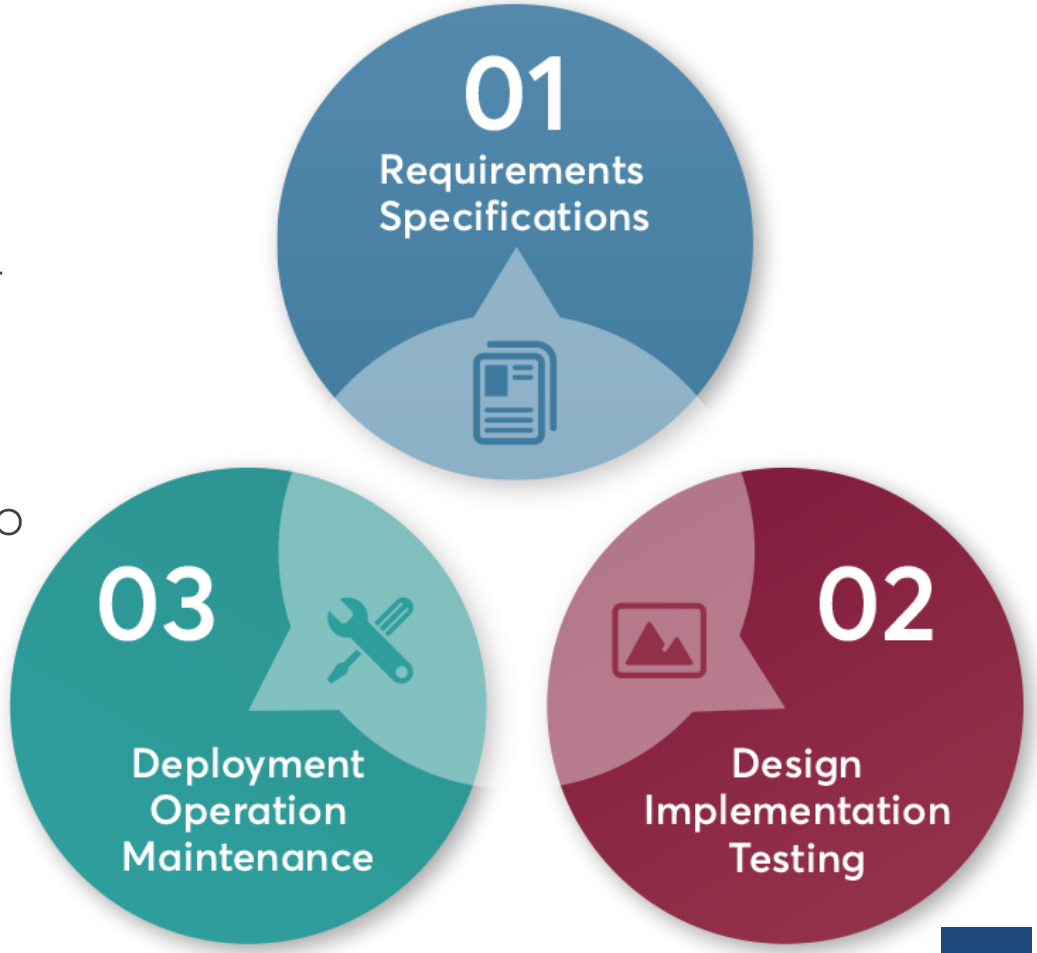
- In contrast to other domains, software engineering provides enough data openly available for any problem you want to solve
- The software engineering domain in numbers (late 2018 snapshot):

	Github	Bitbucket	SourceForge	Stack Overflow
# users	31M	6M	3.7M	40M
# projects	96M[1]	1M	340K	14M

	# Files	Size	#Languages	#Pull requests
Github (active)	1.2Bn	> 20TB	314	77.6M (2016)

The modern software engineering lifecycle: taking a step back

- Each one the software lifecycle phases produces/uses data
  - Software requirements (functional and non-functional)
  - Software models (UML, Database, MDE)
  - Source code (test cases, parameter files also included)
  - Documentation (user documentation, developer documentation)
  - Logs (development and operations)



Software engineering phases and data

L1 — Requirements and Specifications

- Semi-structured text
- Unstructured text
- UML models
- XMI files
- UML Images

L2 — Design and Implementation

- Source code
- Unstructured Text
- Source code Structure and Semantics
- APIs
- Call graphs

L3 — Operations and sustainability

- Source code metrics
- Repository meta-data
- Developer Logs
- Operation logs
- Software usage data
- Social data

The three L's and what they represent

L1 Looking at software as a black-box, from a user perspective

- Is this the right tool for the job?
- Am I missing functionality (critical or desired)?

L2 Watching under-the-hood of software tools

- Has the software been designed properly?
- Is it functionally suitable?
- Are the software components reusable and/or extensible?

L3 Seeing how the software performs

- Is the software easy to use?
- Is it supported and well-maintained?

The ECE Softeng group – vision

- A team of researchers focused on solving state-of-the-art problems in Software engineering and especially in:

- SE lifecycle analysis and auditing
- Design and development of tools for supporting the modern SE lifecycle
- Requirements and specifications Elicitation
- Automation and modeling of SE processes
- Software quality analysis
- Service-oriented software engineering

The ECE Softeng group – the team…

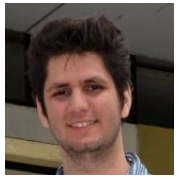
Dr. Andreas Symeonidis
Team lead

Dr. Kyriakos Chatzidimitriou
Team Architect

Dr. Themistoklis Diamantopoulos
Software Reusability expert

PhD Cand. Christophoros Zolotas
Software automation

PhD Cand. Michail Papamichail
Software lifecycle analysis



PhD Cand. Manios Krasanakis
Source code graph analysis

PhD Cand. Thomas Karanikiotis
Deep learning on software

Our way of enhancing the modern SE lifecycle:
Representative Use cases

L1 Enhancing the Requirements and specification elicitation process

- Mining for Functional requirements
- Automating the process of annotating requirements
- Mining for User scenarios

L2 Enhancing Software design and writing better/faster code

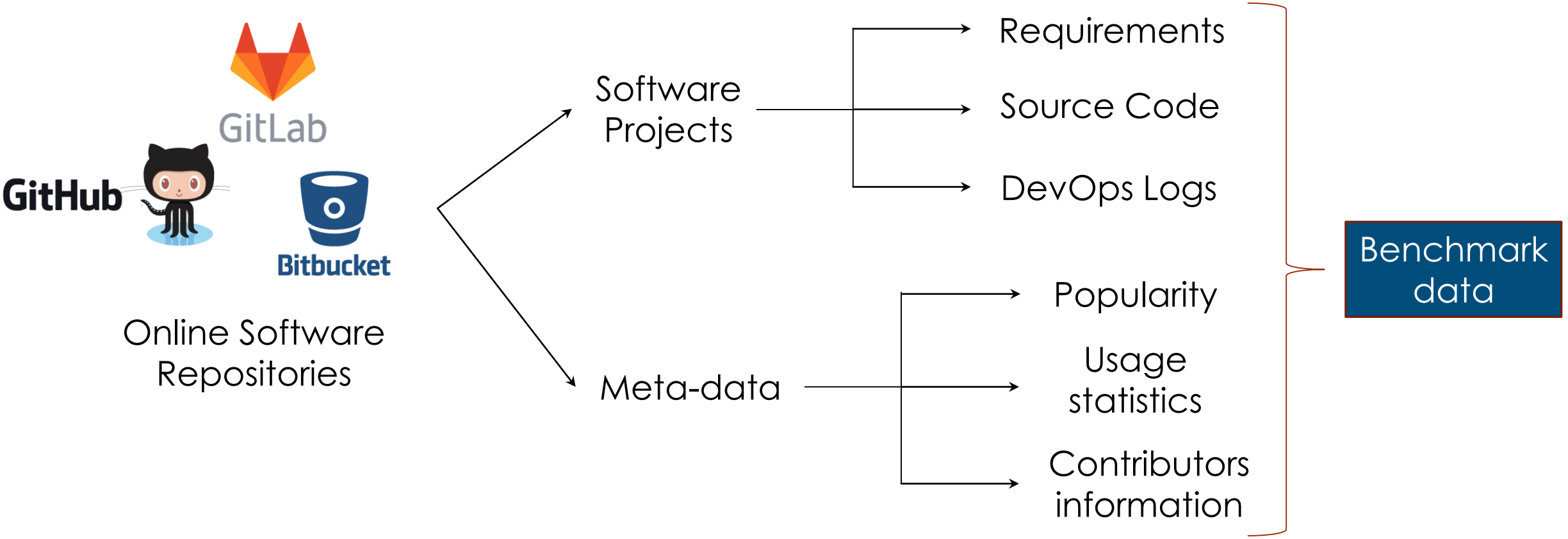
- Recommending reusable software components
- Test-driven reuse
- API usage mining
- Improving Question-Answering in Stack Overflow
- Summarizing source code semantics
- Improving source code writing through collective intelligence

Our way of enhancing the modern SE lifecycle:
Representative Use cases (cont.)

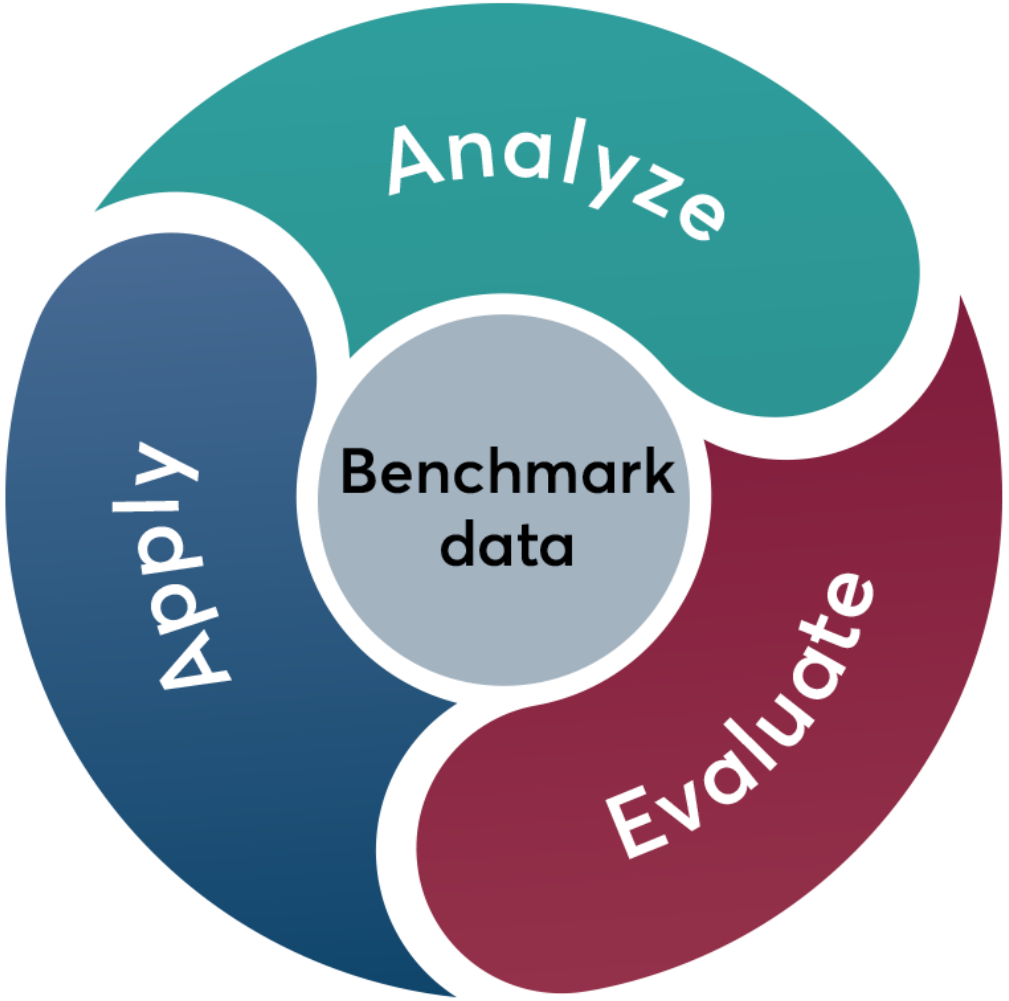
L3 Software quality monitoring and operation analytics

- Developing frameworks for quality assessment
- Localizing Software Bugs
- Predicting maintainability breaches
- Mining for popular UI design elements
- Assessing software based on user-perceived quality
- Mining for user behavior patterns

A Data-driven methodology for performing Software analytics



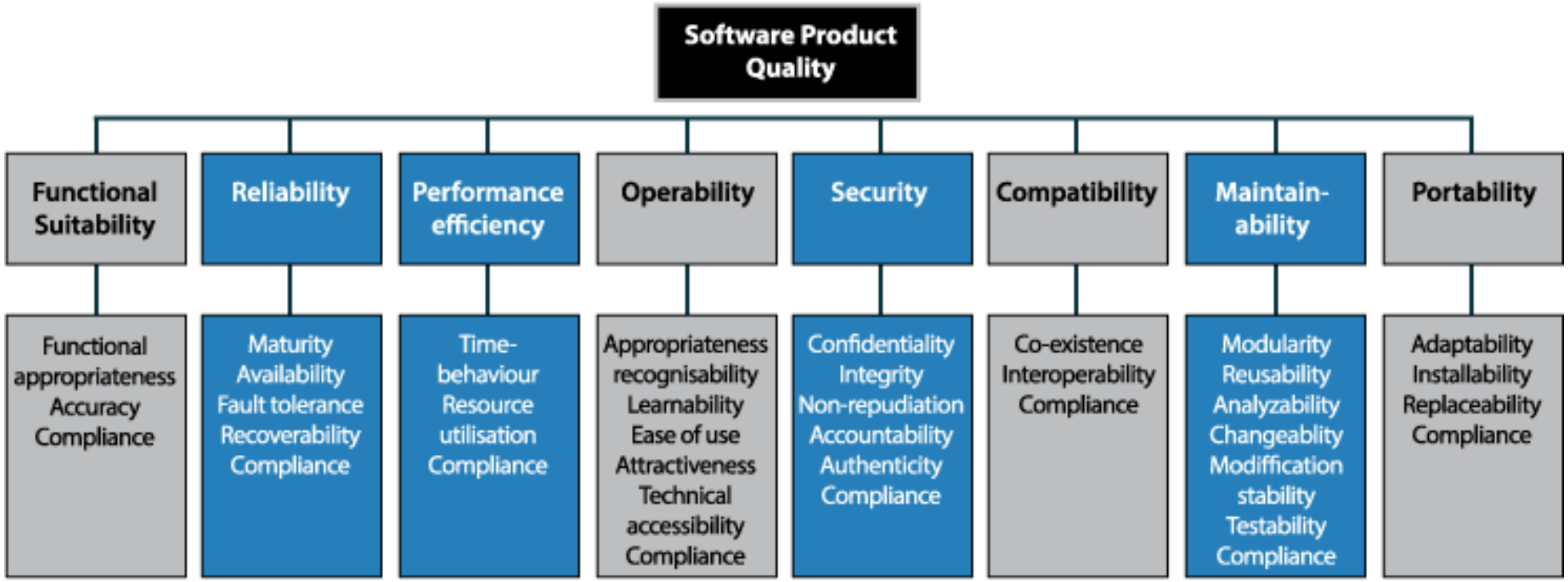
A Data-driven methodology for performing Software analytics (cont.)



- Natural Language Processing
- Static Analysis
- Dynamic Analysis
- Features Selection
- Features Extraction
- Machine Learning

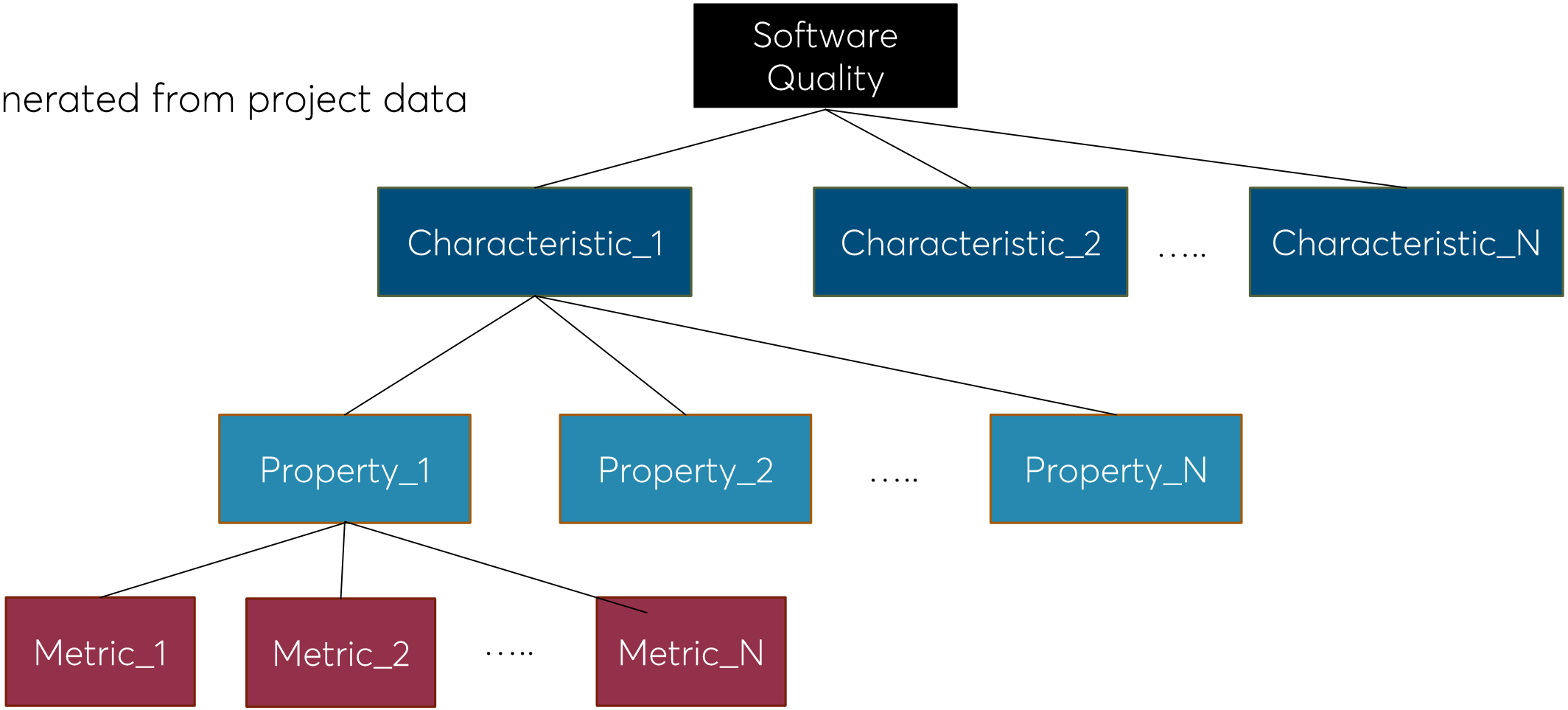
Software analytics related to software quality characteristics

- The methodology follows the ISO/IEC 25010:2011 software quality standard



Software quality characteristics decomposed to software project metrics…

…generated from project data



Now, back to the Multimedia world

Answering questions one-by-one…

Q1 Which library/api should I embed into my software tool in order to solve the problem efficiently and effectively?

Q2 A paper I found provides a link to the prototype developed. Would it be easy to test it/adapt it, or is it going to take ages to understand what the tool does?

Q3 I have a great idea for an add-on to the xxx tool. Is it easy to build?

These questions are related to various L1-L3 parameters

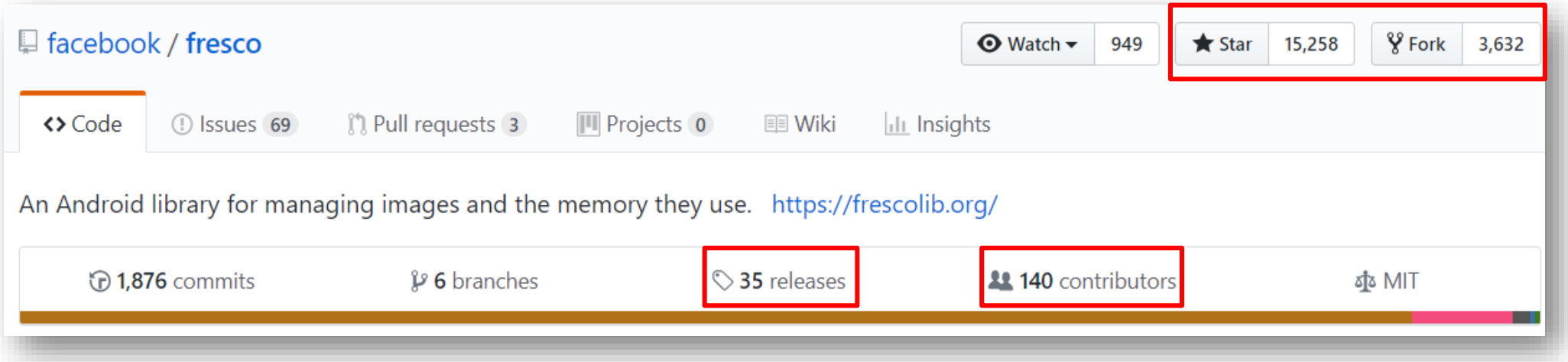
- Related to operation-wise parameters:
  - Developer community
  - User community
  - Developer support
- Related to development-wise parameters:
  - Component reusability
  - Software maintainability
  - Software performance and reliability

Easy to get operation-wise information

Q1

Q3

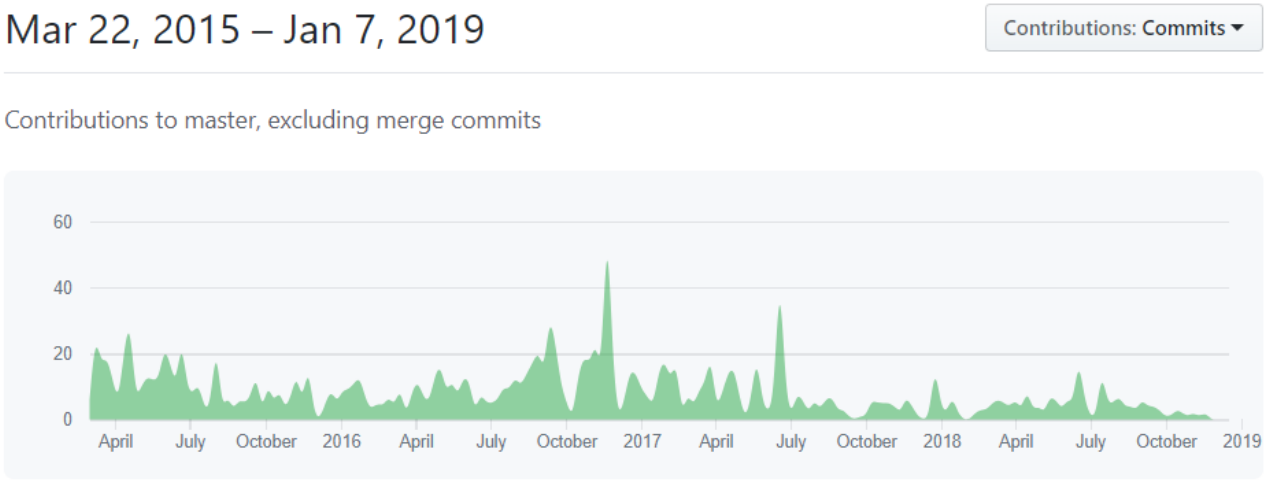
- Just by browsing on Github:



Q1

Q3

Easy to get operation-wise information (cont.)



Conclusions easy to reach:

- Large Community (stars and forks)
- High Reuse Rate
- Many Contributors
- Active Development and Maintenance

But:

- Are 15,258 Stars and 3,632 Forks enough?
- Are these 140 contributors active?

Q1

Q3

More informed operation-wise decisions based on benchmark data

- ✓ This project lies in the top 2% regarding its reuse rate and popularity
- ✓ This project has 2x faster release rate against similar projects
- ✓ This project has 8 main contributors that have committed the 85% of its source code
- ✓ The average issue close rate is 2.3 days (top 5%)

This is a good project to use operation-wise

Q1
Q2
Q3

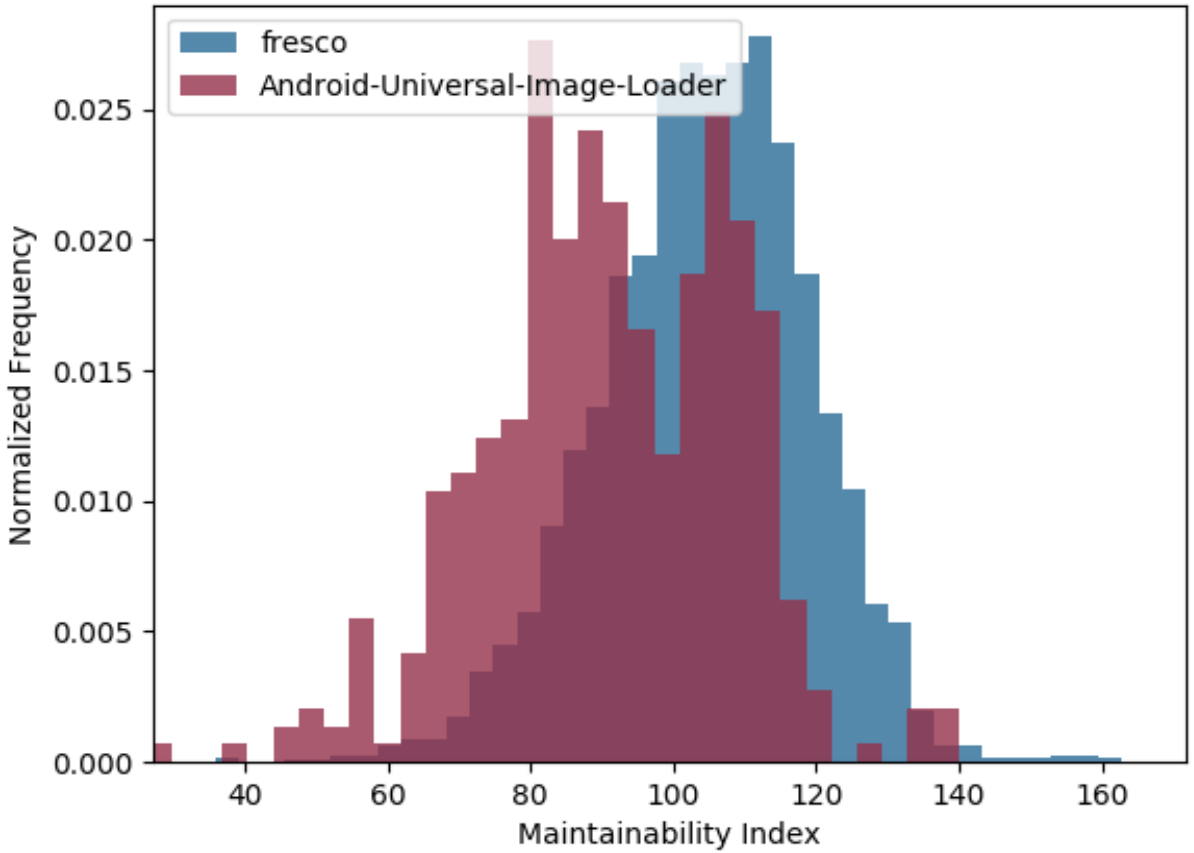
Development-wise decisions based on benchmark data

- Typical questions that can be answered through software analytics
  - Is the library/tool well documented?
  - Does it have critical dependencies to third party projects?
  - Is this a well-maintained project?
  - Can I easily reuse the whole (or part) of the tool?
  - I found two libraries that exhibit the same functionality. Which one should I adopt?
- Consider the following two projects providing the same functionality (loading an image in a mobile application):
  - Fresco (supported by Facebook)
  - Android-Universal-Image-Loader (supported by Sergey Tarasevich)

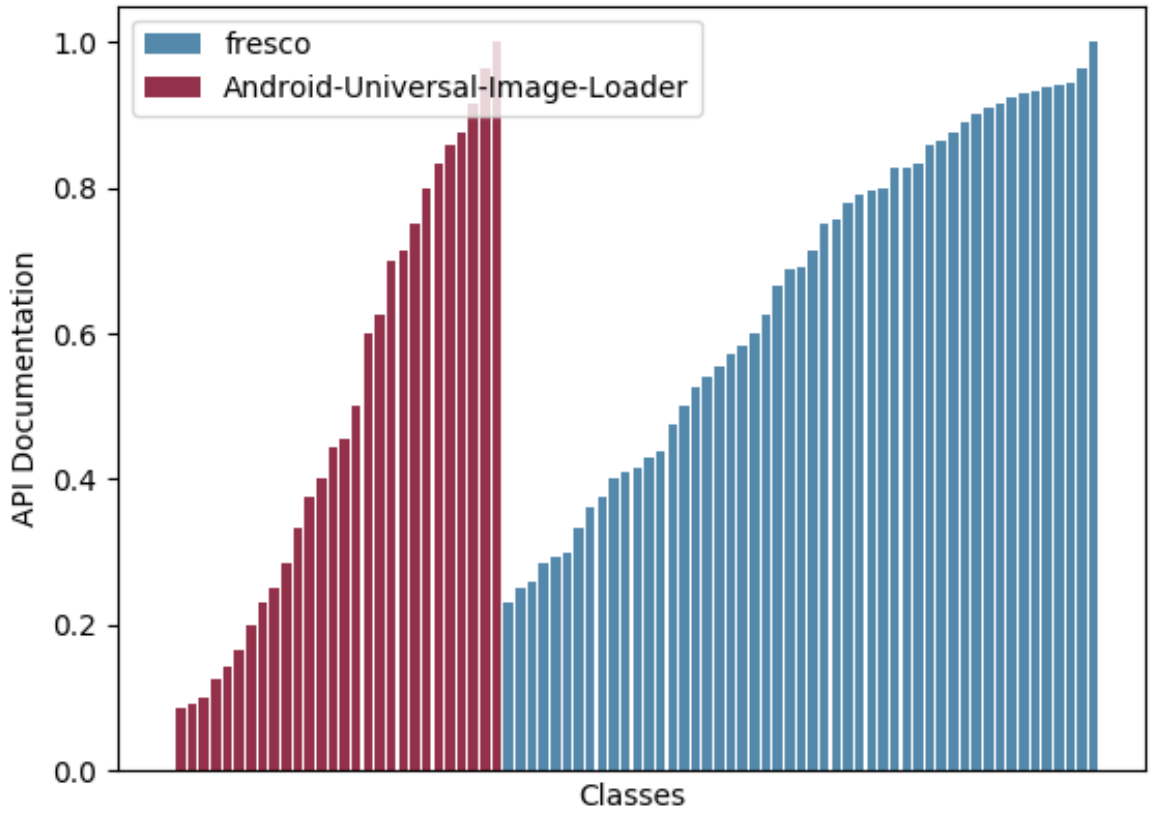
Q1
Q2
Q3

Development-wise decisions on maintainability and readability

On maintainability



On readability



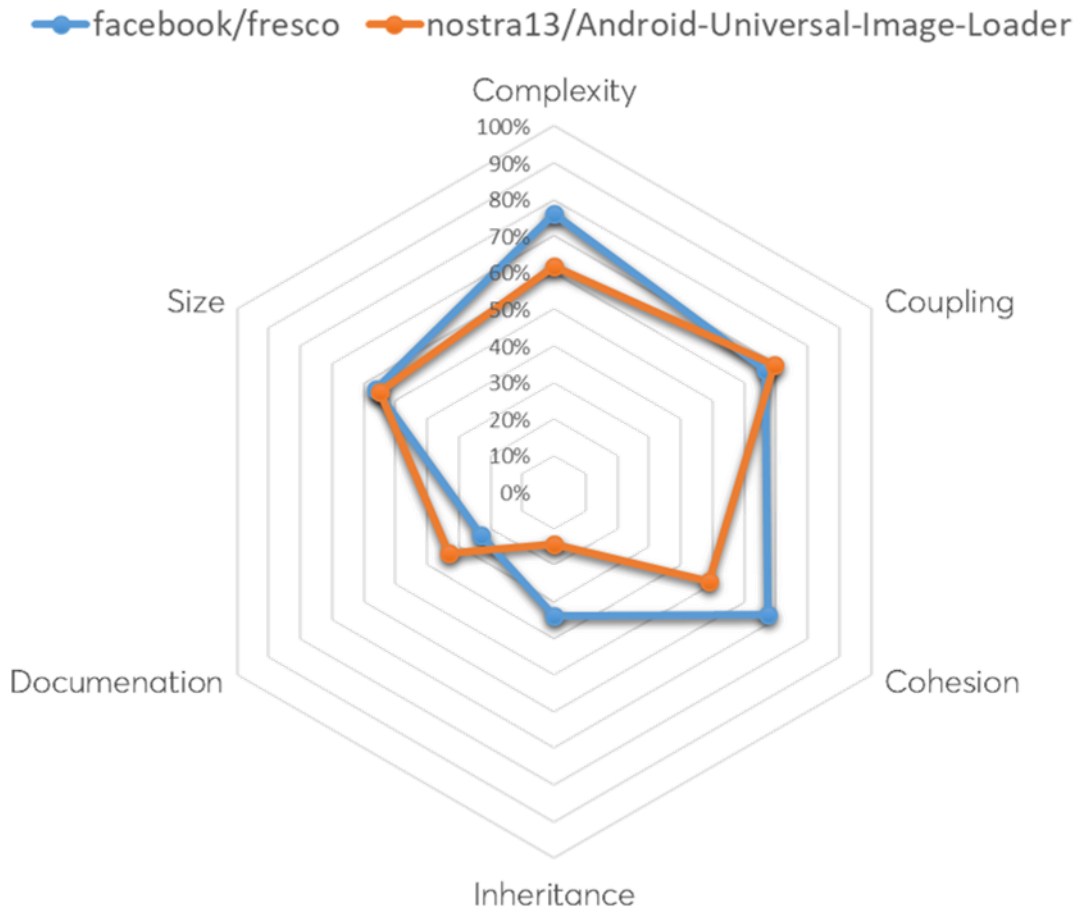
Development-wise decisions at code quality level

Q1

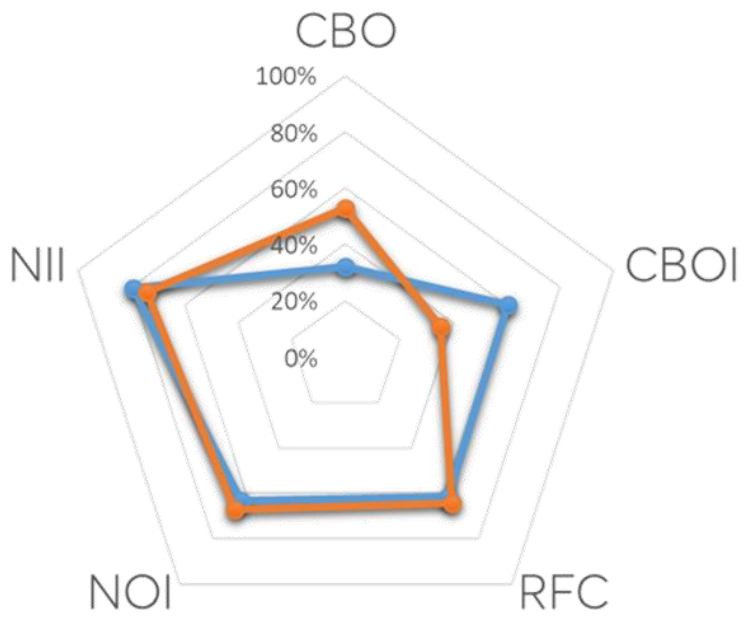
Q2

Q3

Going deeper…



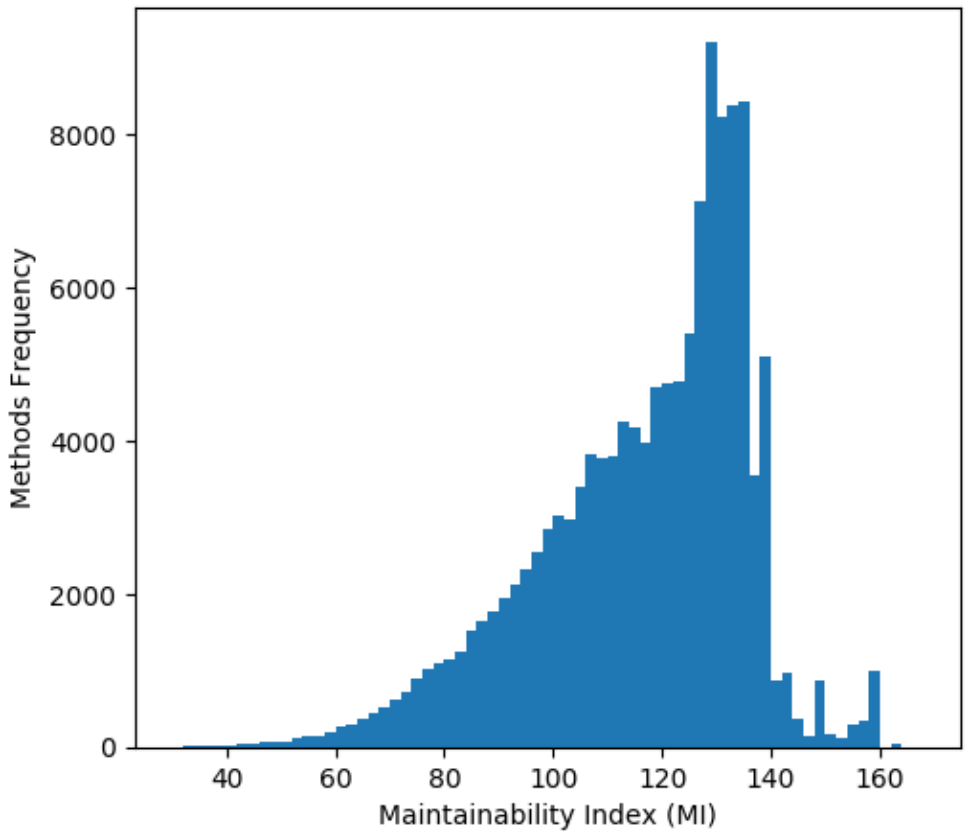
…and deeper…



Development-wise decisions on reusability

Q1

Q3



Overview of Maintainability Index at Method level

57.5	66.4	68.9	69.6	72.8	78.7	79.3	81.0	81.7
82.8	83.9	86.8	86.8	87.0	87.1	88.3	88.4	89.5
93.5	93.9	96.6	97.8	98.2	98.2	98.2	98.2	98.2
98.2	98.2	100.8	100.9	101.1	106.0	106.0	106.2	110.0
113.0	114.3	116.1	116.1	118.7	118.7	120.0	120.0	120.0
120.0	120.0	120.0	120.0	120.0	120.0	120.0	120.0	120.3
122.3	122.4	122.5	124.0	124.7	125.2	125.5	126.6	126.6
126.8	126.8	127.3	128.7	128.7	128.9	129.4	129.4	131.5
131.5	131.5	132.2	133.8	133.8	134.3	134.3	134.5	145.1

Bonus question

Q4 I have a great idea for a new multimedia tool. How can I make it popular?

Facts about developing software tools – NOT prototypes

Q4

- However good the planning, almost 20% of software projects fail, while another 50% is challenged (deviations in timing and maneffort needed)
- Software maintenance effort (bug fixing and evolution) is always underestimated. In practice, it corresponds to ~40-80% of the expected total man-effort.

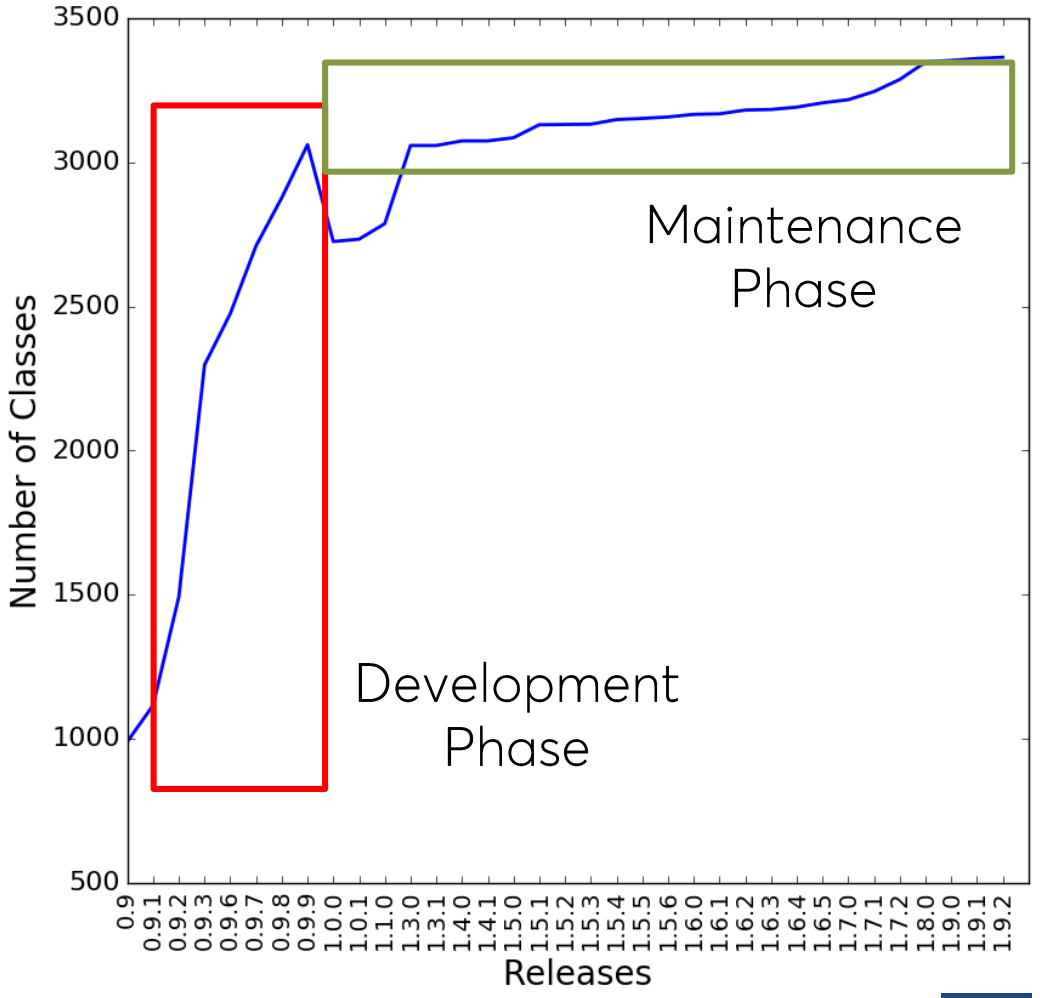
Software Lifecycle analysis is essential!

Lifecycle Analysis: the Libgdx Project example

Q4

Libgdx information sheet:

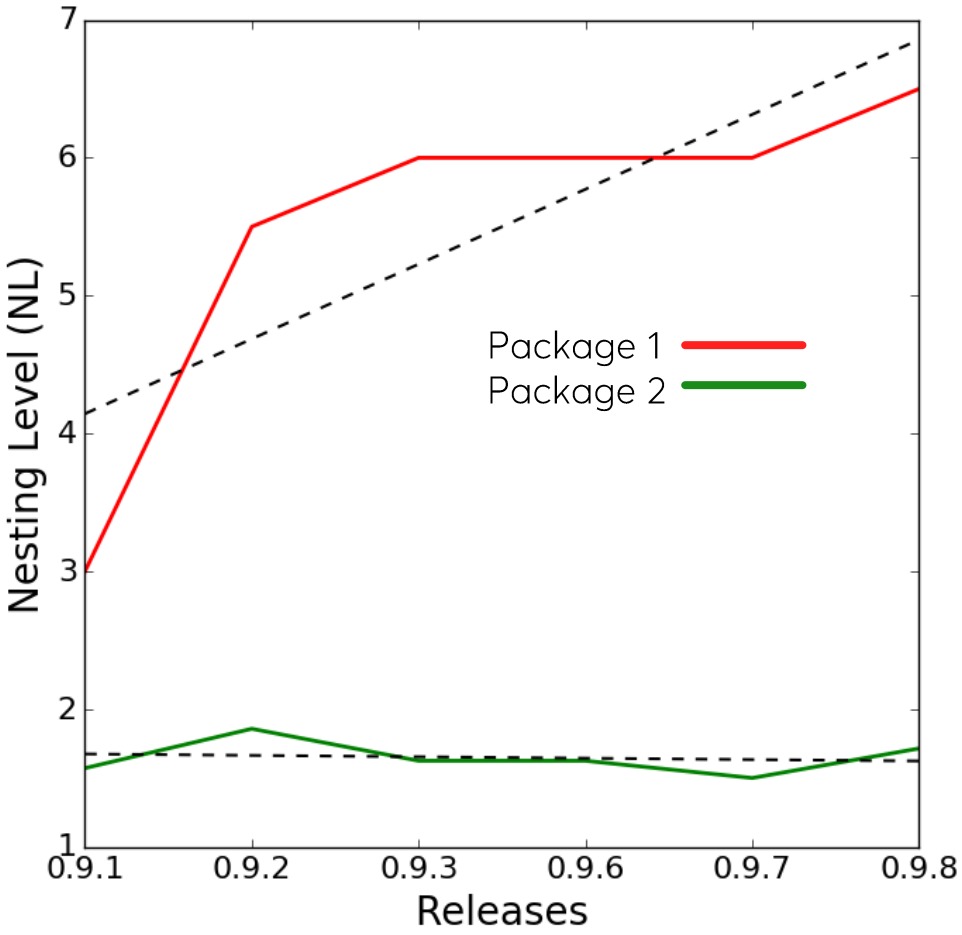
- Scope:
  - Game development application framework
- Project Duration:
  - Feb 2011 – Feb 2016
- Development stats
  - 35 Releases (0.9.0 – 1.9.2)
  - 11.1M LoC
  - 5.6K packages
  - 101K Classes
  - 1.2M Methods



Lifecycle analysis: how does code evolve

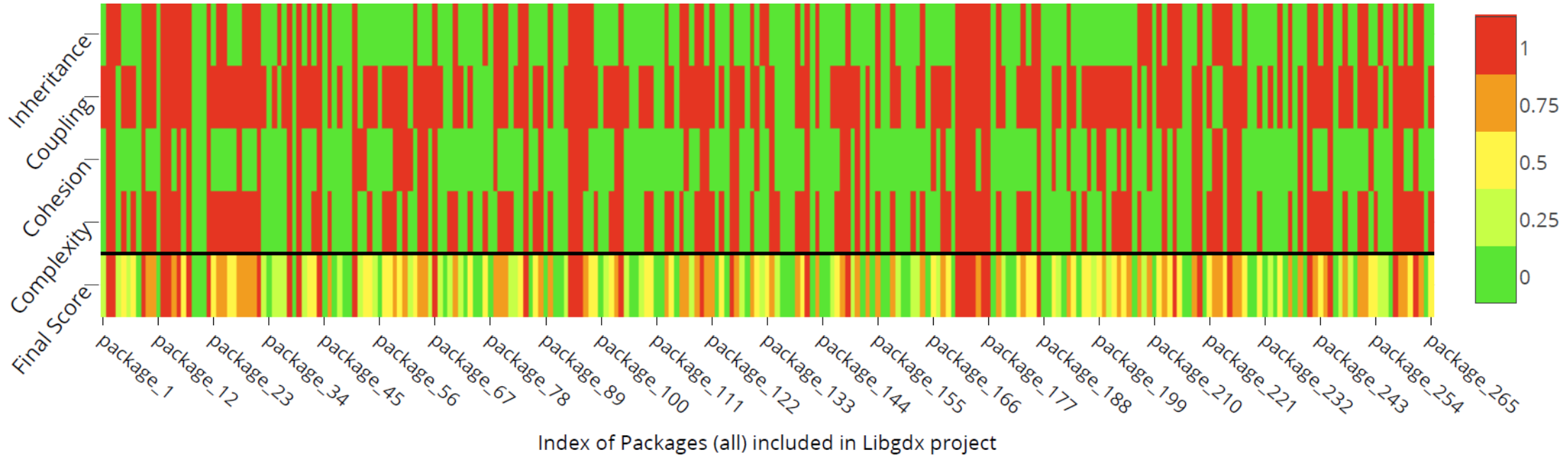
Q4

- Monitoring metrics as the project evolves
  - Evolution of NL values for 2 packages
  - The slope of the trends can be used as a modeling feature
  - Package 1 is dropped at release 0.9.8



Lifecycle analysis: how to monitor maintainability

Q4



- Gain a clear insight on which software parts require attention early enough

Instead of an epilogue

How do we see the future of OS multimedia tools development

Can OS multimedia software antagonize commercial solutions?

- Currently there are multiple Open Source alternatives for practically all commercial products

OS pros

- Free to use, modify and adapt
- Large and active community especially for the leading OS multimedia software tools
- Closer to the research/prototype/experimentation culture

OS cons

- User/developer support is not granted and credibility risks may occur
- May need to integrate various tools/libraries in order to generate the envisaged functionality

Quality assessment is needed in order to avoid deadlocks and frustration

REST API for the OS Multimedia tools dataset

- OS multimedia tools service
  http://83.212.104.23:5000/api/v1/

- Get list of all analyzed multimedia software tools
  http://83.212.104.23:5000/api/v1/projectsList

xxx: {Class, Method, Package}
yyy: {metric_name}

- Get analysis information for a specific project
  http://83.212.104.23:5000/api/v1/xxx?where={"project":"my_project"}

- Get analysis information for a specific Class/Method/Package
  http://83.212.104.23:5000/api/v1/xxx/{xxx_id}

- Perform specific query based on metrics
  http://83.212.104.23:5000/api/v1/xxx?where={"yyy":{"$gt":7}}

Analysis supported by:

CYCLOPT

Thank you – Questions

Andreas L. Symeonidis and Michail Papamichail
Softeng Group, Electrical and Computer Engineering
Aristotle University of Thessaloniki, Greece
asymeon@eng.auth.gr, mpapamic@issel.ee.auth.gr
http://users.auth.gr/symeonid, https://issel.ee.auth.gr/staff/papamichail/
http://softeng.issel.ee.auth.gr