Design of a System to Support Open Source Software License Compliance Checking in Large Enterprises

Navina Lang, 25th May 2020, Bachelor Thesis Kick-off-Presentation

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Outline

1. Motivation
2. Research Questions
3. Research Approach
4. Current state
5. Timeline

Motivation

FicTech violates GPL license

Case:

- FicTech embedded component-x which is under a GPL license
- component-x is essential for their product
- Vendor of component-x sues FicTech for 10,000 Euro

Copyleft:

- “Copyleft licenses are conditional licenses. One of the conditions you must satisfy before distributing copylefted software is that any changes you make to that software be likewise released under the copylefted license. A copyleft license ensures that all modified versions of your project remain free in the same way. Such licenses are said to keep code “forever free”.” [9]

Consequences FicTech:

- must remove component out of product
- Expenses
  - 10,000 Euro
  - 10,000 Euro court costs

Motivation

Challenges:

- An application architect knows about only 10% of the associated software components because of transitive dependencies between components
- Components are all under a specific license which are bound by different conditions (e.g. burdens must be fullfilled)
- Architects are not educated on legal issues
- Lawyers have no know-how about architecture and OSS technologies

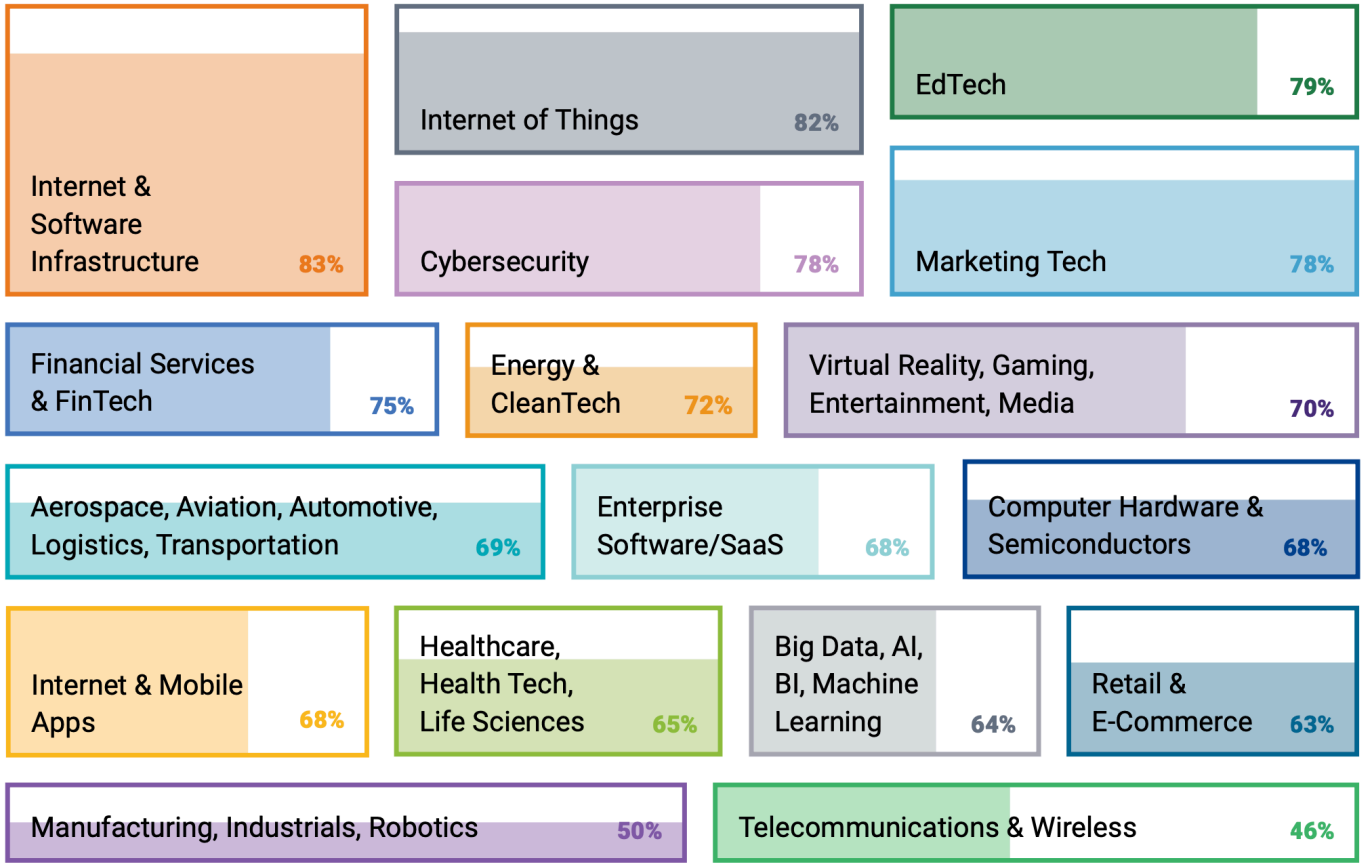
OSS: Open-source software

Motivation

Investigation of 1,250 commercial codebases in 17 industries by Synopsys

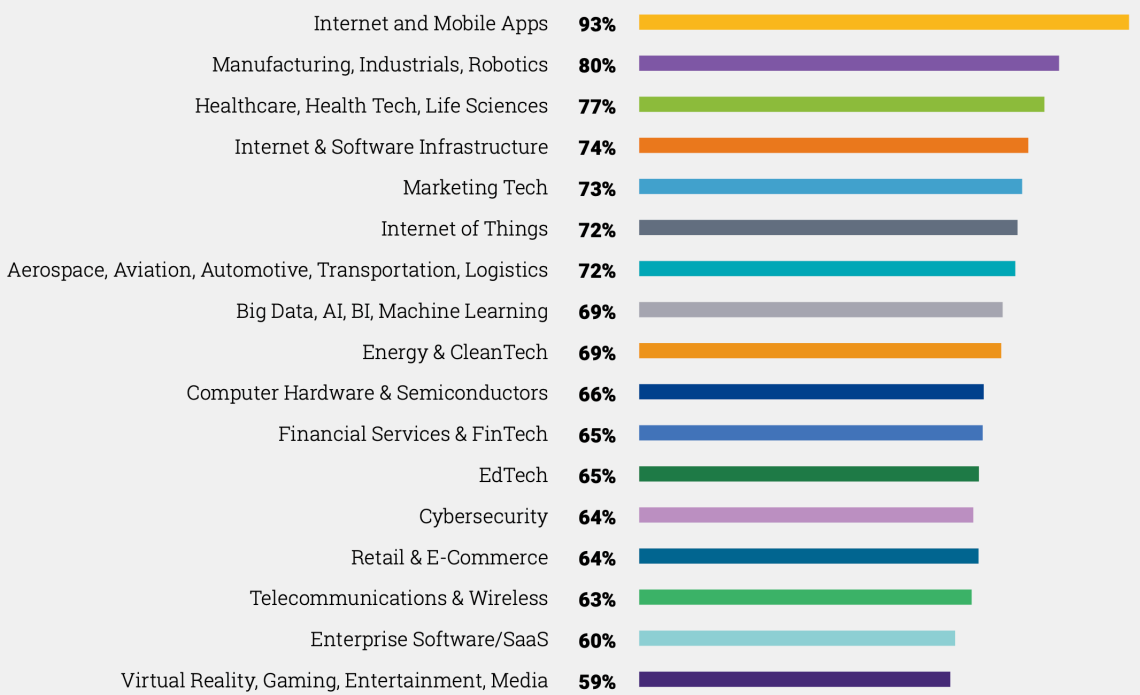
Industries represented in the 2020 OSSRA report

Percentage reflects amount of open source in codebases by industry



source: Synopsys. "OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT"
https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2020-ossra-report.pdf

License conflicts by industry (percentage of codebases)



Outline

1. Motivation
2. Research Questions
3. Research Approach
4. Current state
5. Timeline

Research Questions

What are the challenges of using OSS in terms of legal compliance?

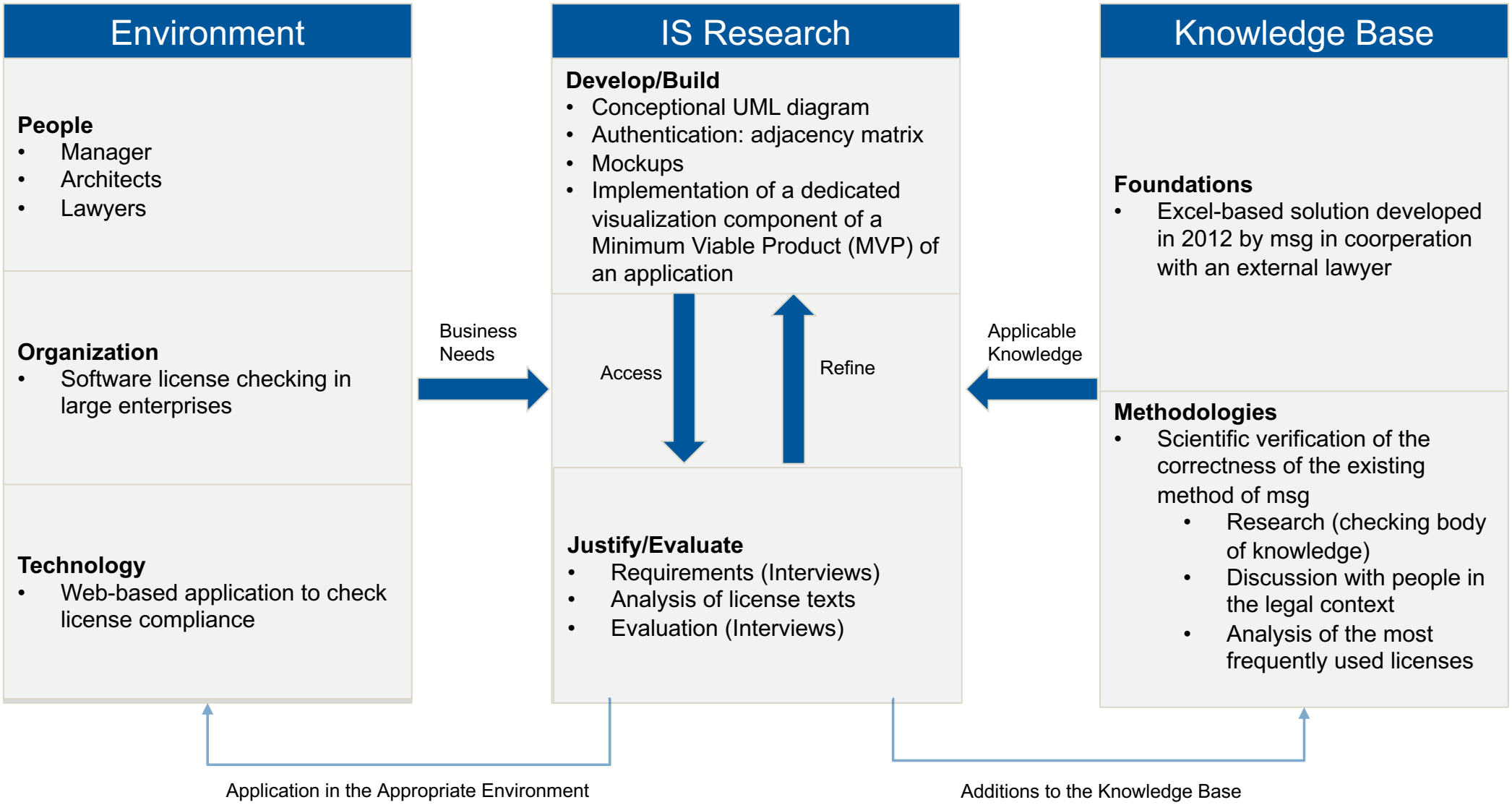
How can we improve the OSS license compliance legal process?

What are the benefits and limitations of the proposed solution?

Outline

1. Motivation
2. Research Questions
3. Research Approach
4. Current state
5. Timeline

Design Science Research



Outline

1. Motivation
2. Research Questions
3. Research Approach
4. Current state
5. Timeline

Current state: Interviews

Interviews

- 4 Architects (Gillardon, GBP, XT)
- 1 OSS Expert
- 1 Product Owner
- 1 Product Manager
- 2 Legal Experts (msg Legal Department)

Personas

- Architect-Individual: No knowledge of OSS
- Architect-Product: Good knowledge of OSS
- OSS Expert
- Legal Expert
- Product Owner

RQ1: What are the challenges of using OSS in terms of legal compliance?

- No defined process [1] [6]
- No know-how on legal issues [1] [3]
- Communication between legal experts and architects [1] [3]
- Checking transitive dependencies [1] [3]
- License can change within a new upgrade [1]
- Long and complicated license texts [1]
- License check is too complicated [1] [2] [4]

Basis: Excel-based solution

- Capture components used in product
- Specify the usage of each component
  - Conditions depend on usage

DEFCON 1
DEFCON 2
DEFCON 3
DEFCON 4
DEFCON 5

Description:		Use Types													
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.		format: source	format: compiled	dependency: optional	dependency: mandatory	delivery: internal	delivery: distributed	usage: local-call	usage: remote-call	communication: process	communication: system	bundling: standalone	bundling: embedded	artifact: pristine	artifact: modified	Update
Component Id	Component Version	FMT		DEP		DLV		USG		COM		BND		ART		DEFCON
																4

Basis: Excel-based solution

MIT License

Copyright <YEAR> <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

source: https://opensource.org/licenses/MIT

License: MIT

	Use Types													
	format: source	format: compiled	dependency: optional	dependency: mandatory	delivery: internal	delivery: distributed	usage: local-call	usage: remote-call	communication: process	communication: system	bundling: standalone	bundling: embedded	artifact: pristine	artifact: modified
Condition Id	FMT		DEP		DLV		USG		COM		BND		ART	
NO-LIABILITY	X	X	X	X	X	X	X	X	X	X	X	X	X	X
KEEP-COPYRIGHT	X	X	X	X	X	X	X	X	X	X	X	X	X	X
PROVIDE-LICENSE	X	X	X	X	X	X	X	X	X	X	X	X	X	X
PROVIDE-SOURCE														
ADV-CLAUSE														
RENAME														
NO-RELICENSE														
CTX-NON-MIL														
CTX-NON-COM														
COPYLEFT-STRONG														
COPYLEFT-WEAK														
NON-OSS-DEF														
OTHER														

RQ2: How can we improve the OSS license compliance legal process?

First idea for improvement

Example: Provide-license

- Format: must either be source or compiled
  - that means: if you use the component internally, you still must provide the license text
  - can be simplified, so the user input will be reduced

License: GPL-3

	Use Types													
	format: source	format: compiled	dependency: optional	dependency: mandatory	delivery: internal	delivery: distributed	usage: local-call	usage: remote-call	communication: process	communication: system	bundling: standalone	bundling: embedded	artifact: pristine	artifact: modified
Condition Id	FMT		DEP		DLV		USG		COM		BND		ART	
NO-LIABILITY	X	X	X	X	X	X	X	X	X	X	X	X	X	X
KEEP-COPYRIGHT	X	X	X	X	X	X	X	X	X	X	X	X	X	X
PROVIDE-LICENSE	X	X	X	X		X	X		X	X	X	X	X	X
PROVIDE-SOURCE		X	X	X		X	X		X	X	X	X	X	X
ADV-CLAUSE														
RENAME														
NO-RELICENSE	X	X	X	X		X	X		X	X	X	X	X	X
CTX-NON-MIL														
CTX-NON-COM														
COPYLEFT-STRONG	X	X	X	X		X	X		X	X	X	X	X	X
COPYLEFT-WEAK														
NON-OSS-DEF														
OTHER	X	X	X	X	X	X	X	X	X	X	X	X		X

RQ2: How can we improve the OSS license compliance legal process?

Requirements

Architect-Individual | Product Owner | Legal Expert | Architect-Product | OSS Expert

- ◯ Hide irrelevant Use Types / Default Use Types

Architect-Individual | Legal Expert | Architect-Product

- ◯ Yes / No evaluation

Architect-Individual

- ◯ Push notification for security issues / updates

Architect-Individual | Legal Expert | Architect-Product

- ◯ Clear definitions for Use Types

Architect-Individual | Product Owner | OSS Expert

- ◯ API / build-process integration

Architect-Individual | Product Owner | Architect-Product

- ◯ Import / export dependencies (e.g. pom.xml)

Architect-Individual | Product Owner | Architect-Product

- ◯ Creation of customer reports

Architect-Individual | Architect-Product | OSS Expert

- ◯ Automatic evaluation of transitive dependencies

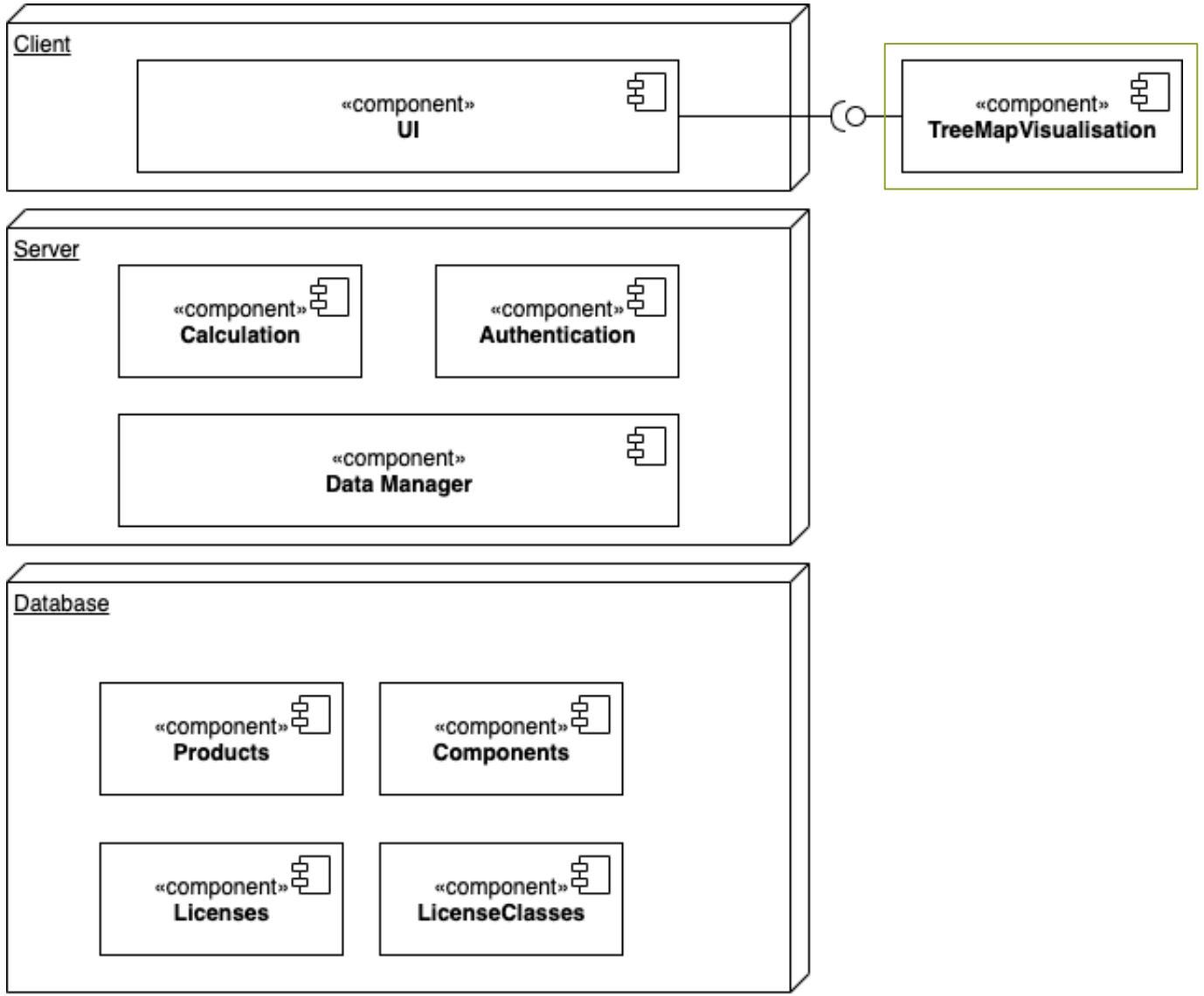
Architect-Individual | Legal Expert | Architect-Product

- ◯ License information as support for choosing a library

RQ2: How can we improve the OSS license compliance legal process?

Proposed solution

- Verify / improve method from 2012
- Web-based application to check license compliance
- Create products in order to check whether license conditions are fulfilled
- Requests for legal help

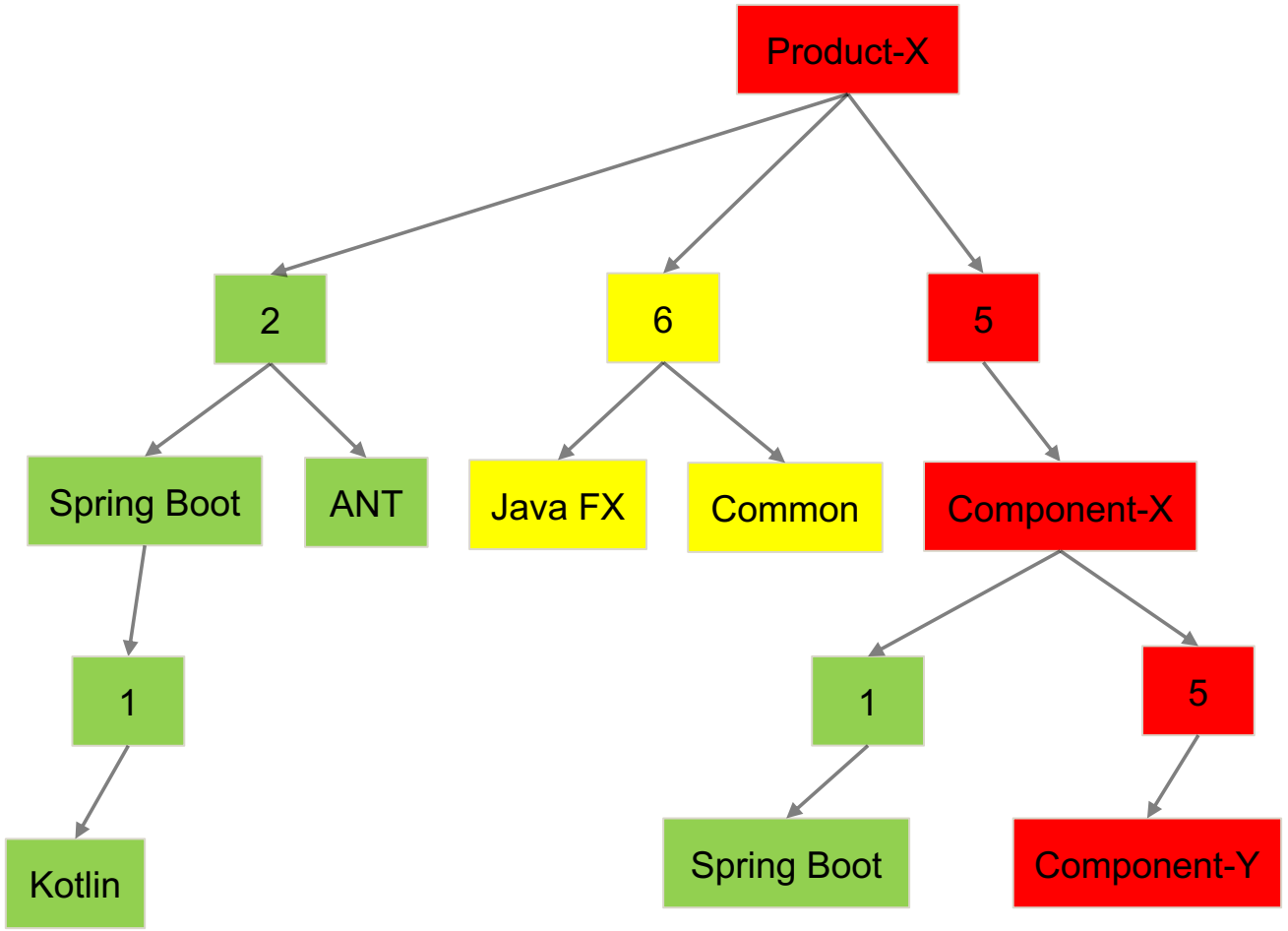


RQ2: How can we improve the OSS license compliance legal process?

Legal status illustration

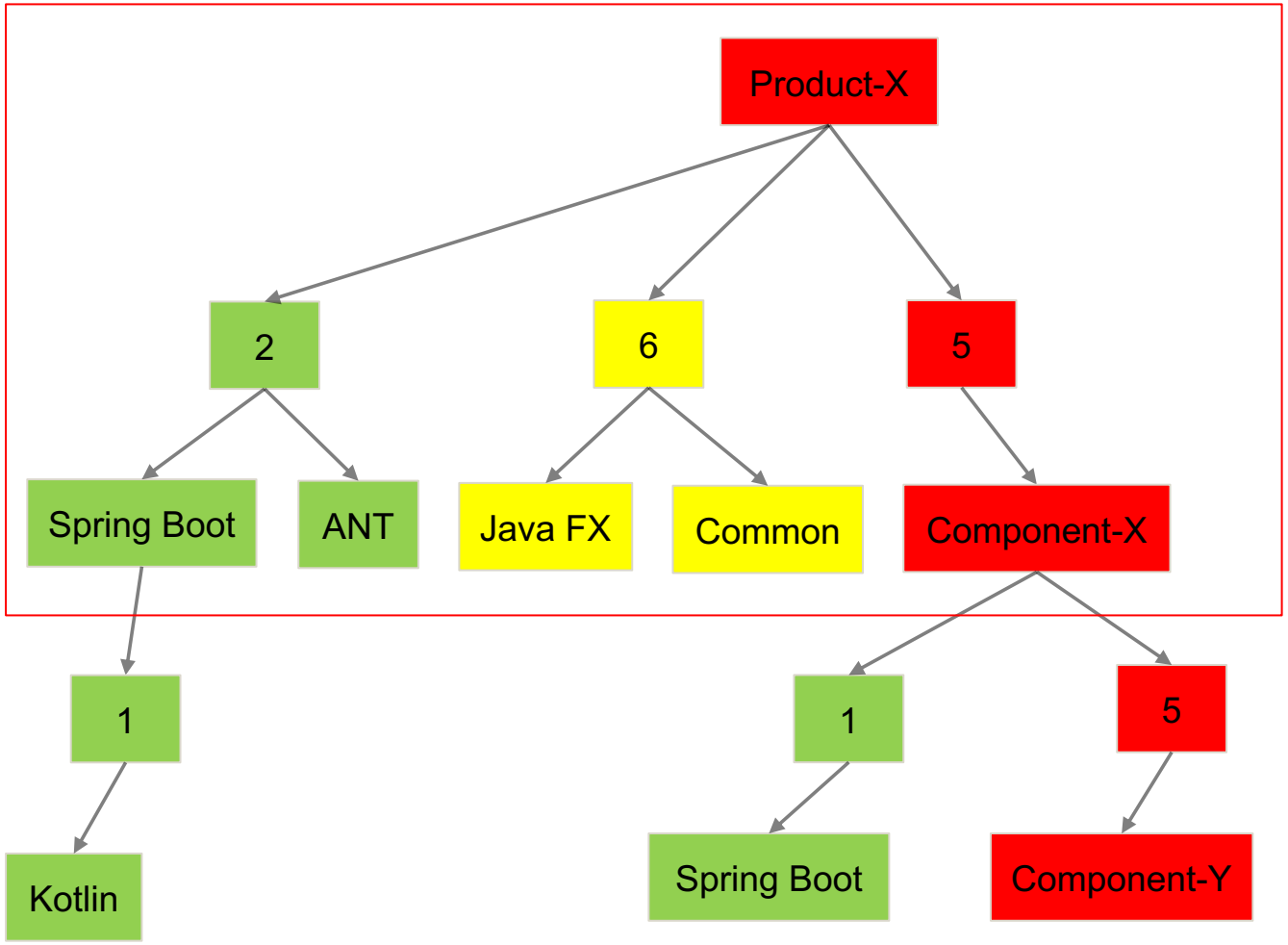
Product-X (Fictive product)

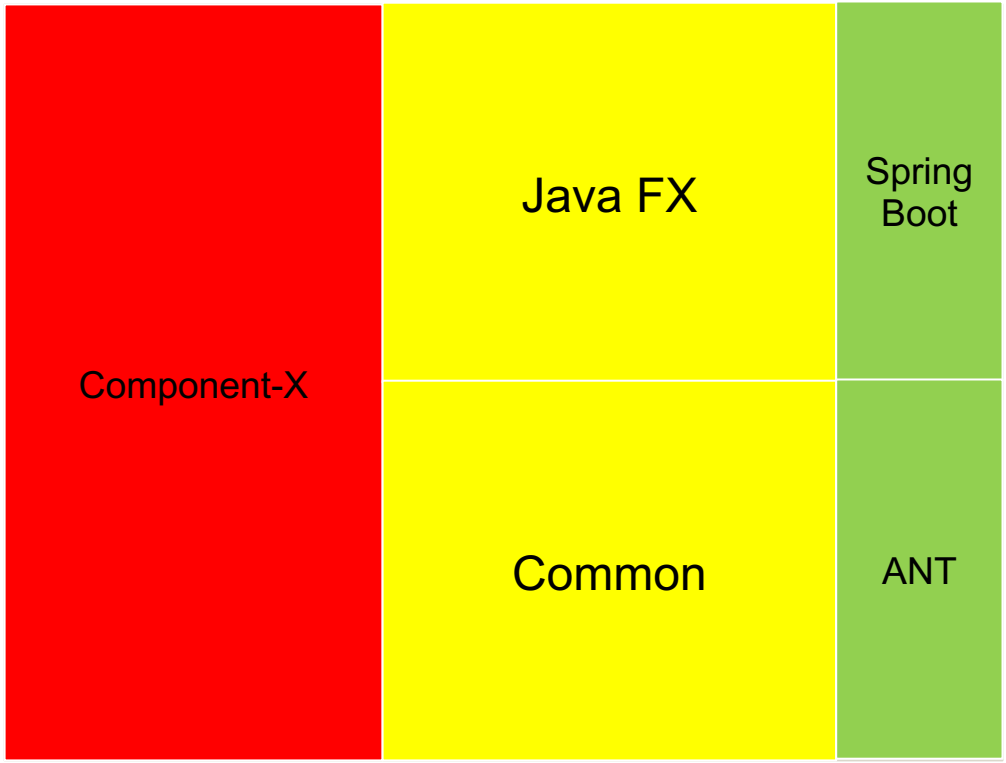
- Spring Boot
  - Kotlin
- ANT
- JavaFX
- Common
- Component-X
  - Spring Boot
  - Component-Y



RQ2: How can we improve the OSS license compliance legal process?

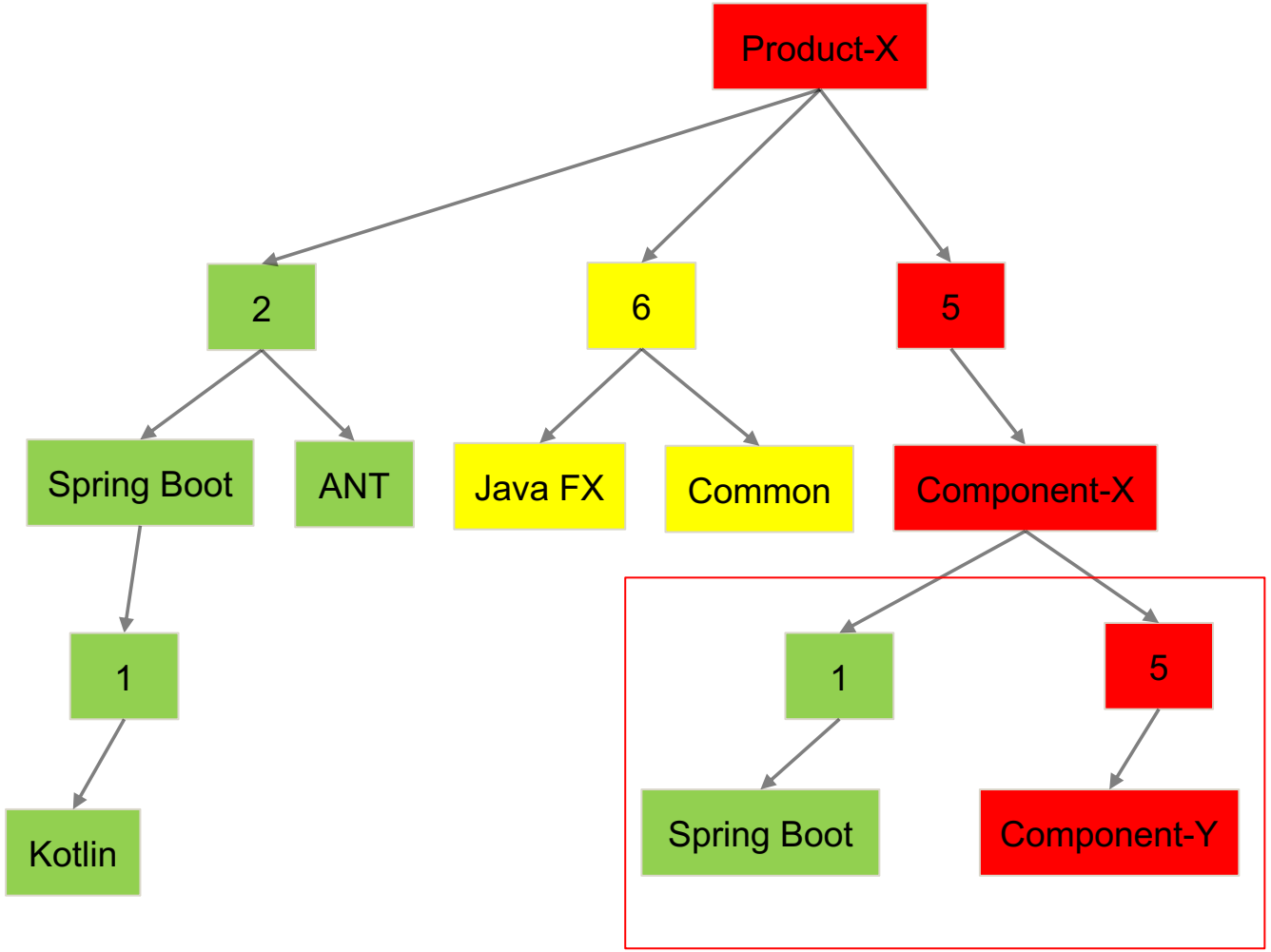
Legal status illustration

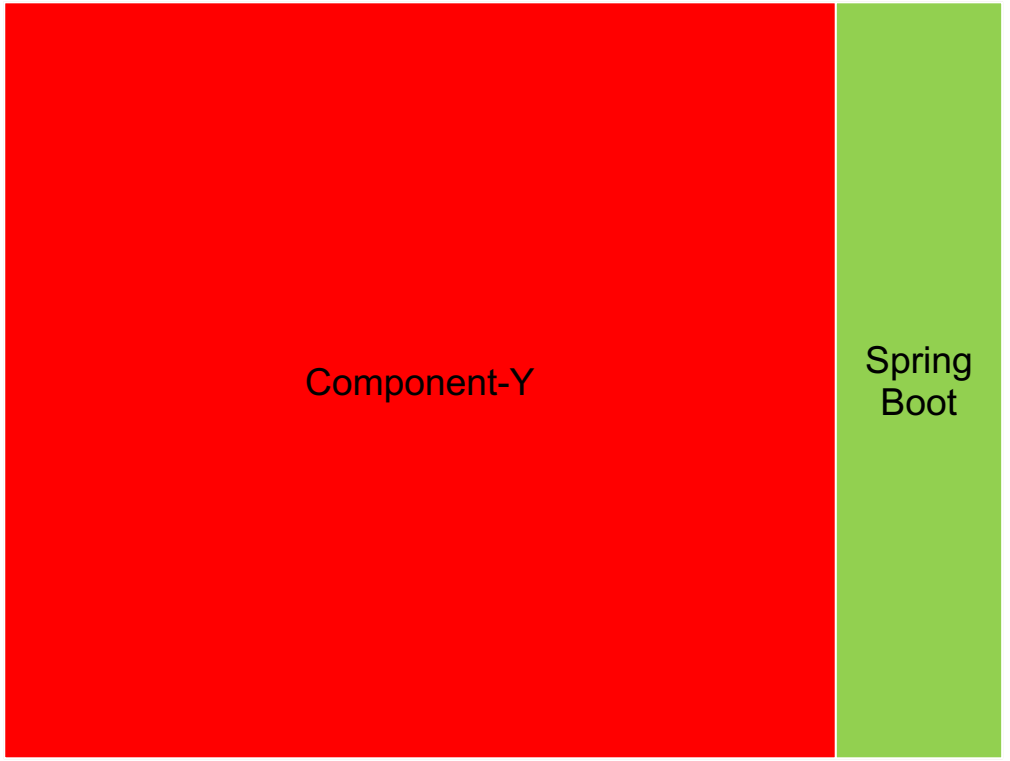




RQ2: How can we improve the OSS license compliance legal process?

Legal status illustration

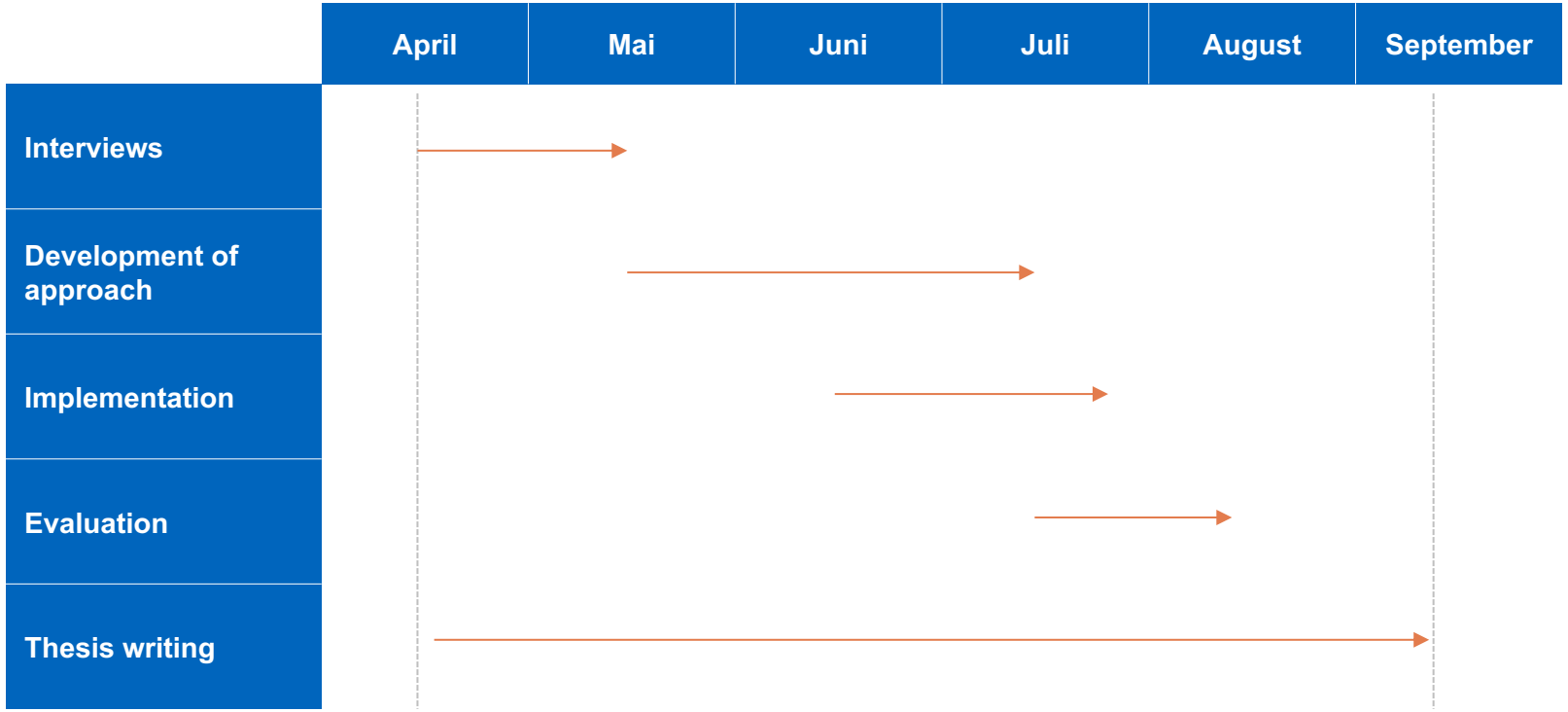




Outline

1. Motivation
2. Research Questions
3. Research Approach
4. Current state
5. Timeline

Timeline



References

[1]: Interviews with 9 interview partners

[2]: K.-J. Stol and M. A. Babar. “Challenges in Using Open Source Software in ProductDevelopment: A Review of the Literature”.

[3]: D. R. S. Engelschall. “ARCHITEKTUR VS. LIZENZRECHT: LIZENZKONFORME VER-BAUUNG VON OPEN-SOURCE-SOFTWARE”.

[4]: K.-J. Stol, M. A. Babar, P. Avgeriou, and B. Fitzgerald. “A comparative study of challengesin integrating Open Source Software and Inner Source Software”

[5]: J. Whitehurst, RedHat. “The State of Enterprise Open Source”

[6]: D. H. Schoettle. “Mitwirkung an OSS-Projekten durch Mitarbeiter”

[7]: Synopsys. “OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT”

[8]: GitHub survey https://opensourcesurvey.org/2017/

[9]: Richard Fontana, Bradley M. Kuhn, Eben Moglen, Matthew Norwood, Daniel B. Ravicher, Karen Sandler, James Vasile, Aaron Williamson “A Legal Issues Primer for Open Source and Free Software Projects”

[10]: Alan R.Henver, Salvatore T. March, Jinsoo Park, and Sudha Ram “Design Science in Information System Research”

Navina Lang

Technische Universität München
Faculty of Informatics
Chair of Software Engineering for Business Information Systems

Boltzmannstraße 3
85748 Garching bei München

navina.lang@tum.de

Definitions

Definition: Use-Types (of Components)

Use-Type Group	Use-Type	Use-Type Description	Use-Type Example/Hint/Association
FMT	format: **source**	component is provided in pristine source format	WEB-INF/jquery.js
	format: **compiled**	component is provided in compiled/converted/compressed format	com/example/foo.class
DEP	dependency: **optional**	component is loaded on demand and product would reasonably work without it	JDBC driver
	dependency: **mandatory**	component is loaded/linked dynamically/statically and product does not function without it	Hibernate ORM
DLV	delivery: **internal**	component is used internally without distribution to other legal entities (e.g. built-time components)	Gradle/Ant/Maven
	delivery: **distributed**	component is distributed to other legal entities (e.g. run-time components)	lib/example-1.2.3.jar
USG	usage: **local-call**	component (via product) is locally called for execution	C:\Applications\Example\example.jar
	usage: **remote-call**	component (via product) is remotely called for execution (SaaS)	service.example.com:1234
COM	communication: **process**	component is called from product via direct in-process mechanism (function call, dispatch table, etc)	component_function()
	communication: **system**	component is called from product via system/network service (pipe, socket, dlsym, execve, etc)	/var/run/component.socket
BND	bundling: **standalone**	component artifacts still provided fully standalone and recognizable as such	example-1.2.3.jar
	bundling: **embedded**	component artifacts embedded into product and/or not obviously recognizable	product-1.2.3.ear/example-1.2.3.jar
ART	artifact: **pristine**	component artifacts are all as-is, i.e. exactly as originally received from upstream vendor	example-1.2.3.jar!com/example/foo.class
	artifact: **modified**	component artifacts were added/replaced/removed	example-1.2.3.jar!com/example/addon.class

Definition: Conditions (of Licenses)

Condition Id	Condition Name	Condition Description	DEFCON
NO-LIABILITY	No Liability	You cannot make the author of the component liable for any damages it causes	5
KEEP-COPYRIGHT	Keep Copyright Information	The copyright information of the component's author has to be kept	5
PROVIDE-LICENSE	Provide License Text	You have to provide the full license text of the component	5
PROVIDE-SOURCE	Provide Source Code	You have to provide the full source code of the component	4
ADV-CLAUSE	Advertizement Clause	Documentation and/or application has to show hint to the component (and its author)	3
RENAME	Name Change Required	The name of the component has to be changed (in case of modifications + redistribution)	5
NO-RELICENSE	No Relicensing Allowed	The component cannot be relicensed under a different custom license	4
CTX-NON-MIL	Non-Military Use Only	Component is not allowed to be used in military or nuclear-power contexts	4
CTX-NON-COM	Non-Commercial Use Only	Component is not allowed to be used in commercial contexts	1
COPYLEFT-STRONG	Strong Copyleft Effect	License applies a strong/full copyleft effect	1
COPYLEFT-WEAK	Weak Copyleft Effect	License applies a weak/restricted copyleft effect	2
NON-OSS-DEF	Non Open Source Software Definition Compliant	License contains conditions which are not compliant to Open Source Software definition	2
OTHER	Other Conditions	License contains arbitrary other major conditions not modeled/covered by us (fallback)	3

RQ2: How can we improve the OSS license compliance legal process?

Legal status illustration