Goal-aware Analysis of Software License Risks

Fitsum Meshesha Kifetew[1], Mirko Morandini[1], Denisse Munante[1], Anna Perini[1], Alberto Siena[2], and Angelo Susi[1]

[1] Fondazione Bruno Kessler (FBK)
kifetew,morandini,munante,perini,susi@fbk.eu
[2] Delta Informatica
alberto.siena@deltainformatica.eu

Abstract. Open Source Software (OSS) components are characterised by heterogeneous licenses that give the possibility to use, modify and often redistribute the source code. Their adoption meets several adopter's needs, such as cost reduction, standards alignment, and so on. However, often OSS projects retain several different (or missing) licenses for the various components and files, which raise risks of violations and potential legal issues, if not correctly managed. This makes necessary to understand the characteristics and implications of licensing and their relation to the adopter's goals. In this paper we report the use of risk assessment techniques to make inference about license risk exposure associated with each business goal. We rely on existing knowledge, gathered from domain experts, and map it onto formal models that can be automatically analysed to provide some evidence about relevant license information and related risk. Goals are used to drive the software license selection. We illustrate the approach for the case of a research and innovation action project funded under the H2020 framework

Keywords: Open Source; Risk Analysis; Software Licenses

1 Introduction

Open Source Software (OSS) relies on the definition of licenses that on one had guarantee to users the possibility to use, modify and often redistribute the source code without paying a fee, but on the other hand include restrictions for redistribution, attribution, advertising, and so on. Many different license types have been defined over the years, and many OSS components – built on top of other components – contain collections of different licenses. The differences among licenses concerns important aspects such as the possible forms of free and commercial redistribution, compatibility with other licenses, forms of attribution, license modifiability, and so on. Requirements engineers start to be aware that licensing is an important issue to face in order to support business objectives, and that the characteristics of license texts may have a critical impact on the commercial success of the adopter.

As with other project types, research project often deliver software artefacts with the objective to promote research dissemination in a given field and to create opportunity of innovation. As such, although not pursuing any commercial objective, when releasing software, research projects need to tackle with the licensing problem. In the SUPERSEDE project, a prototype software has to be delivered. The license selection is a mandatory task, whose actual implementation faces the need to (i) have experts in the license field; (ii) agree among the various project partners; and (iii) make the best decision with respect to the project strategic objectives. There have been numerous work that deal with the simple OSS licensing analysis [4, 5] However, all of them seem to be limited to numerical measurements of OSS characteristics, while the assessment of the overall licensing relevance with respect to the adopter's goals, is not addressed.

In this paper, we present a preliminary report about the use of a combined RiskML+$i*$ modelling and analysis framework to (a) build a comprehensive model of licensing issues as well as the higher level strategic objectives, (b) feed an automated analysis session with real data collected from existing OSS repositories, and (c) derive some *evidence* about the impact of licenses on the strategic goals, in order to drive a more aware license selection.

The paper is structured as follows: Section 2 presents the RiskML+$i*$ modelling framework; Section 3 reports the application of the modelling framework to an European research project; finally, Section 4 concludes the paper.

2 RiskML

RiskML is [8] is a modelling language whose meta-model is integrated with that of $i*$ to model risky events and their relation to strategic objectives. It borrows come modelling primitives from other $i*$ extensions, such as the goal-risk framework [1], to reduce the introduction of redundant constructs. The language builds upon the main concepts of Indicator, Situation, Event and Goal. An ***Indicator*** is an abstract representation of a measure about a certain property of the OSS [2]. An indicator determines the evidence of being in a certain situation. We use the concept of ***Situation*** to model the circumstances under which a certain risk holds. A situation is *satisfied* if the state of affairs that it represents holds [2, 7]. We use the concept of ***Event*** to model a change in circumstances, with a potential negative impact on goals. The concept of ***Goal*** and, in a more broad sense, that of *Intentional* are the same as in $i*$. Event occurrence may happen with a certain *likelihood* and *severity.* If ϕ is a proposition describing an event, $lik(\phi)$ describes the *likelihood* of the event. $sev(\phi)$ describes the *severity* of the event. A *Risk* is a composed concept which expresses a lack of knowledge about some happening and what could be the (negative) consequences on goals. A set of relations link indicators, situations, events, and goals, and thus define implicitly the occurrence of a risk: *Expose*, from a situation to an event: the higher the evidence that the situation is satisfied, the more likely the event is to happen (the *evidence* value represents the degree of confidence that a fact is true); *Protect*, from a situation to an event: the higher the evidence that the situation is satisfied, the less likely the event is to happen; *Increase*, from a situation to an event: the higher the evidence that the situation is satisfied, the more severe is the event consequence; *Mitigate*, from a situation to an event: the higher the evidence that the situation is satisfied, the less severe is the event consequence. *Impact*, from an event to a goal: the more the event is likely and the more there is evidence that its consequences are severe, the more the goal is at risk to fail. *Indicate*, from an indicator to a situation: the higher the value of an indicator, the higher the evidence that the situation is satisfied.

3 Goal-aware license risk analysis

In the context of the SUPERSEDE European project, the need arose, to select a license for the software to be implemented. In order to select the proper license, two aspects had to be taken into consideration: (a) the optimal license had to be selected to achieve project-wide objectives, such as increasing the project visibility and acceptance in the industry, as well as fostering the integration with the OSS communities; and (b) an important requirement was that of choosing a safe license combination, allowing to achieve the project objectives without generating legal issues. In such context, the RiskML framework was used to help in performing the right choice.

Firstly, we have built the RiskML+*i** risk model, which is depicted in Figure 1. The model is generic and can be tailored to requirements for specific closed or open source licenses for the final product, i.e. linking the model to the goals of the adopter organisation. The model has been built using state-of-the-art literature [6] and on interviews with OSS licensing experts. The model contains 17 license indicators, such as "number of GPL licenses", "number of MIT licenses" and so on. Additionally, there are some indicators to capture the target license (the license to cover the released software product, containing the OSS components) or the linking type (i.e., whether a given OSS component is linked statically or dynamically). Some Situations are used to capture specific values of the indicators: for example, the occurrence of particular combinations of licenses. Risk indicators and situations are then aggregated into higher level risks, up to the identification of the top level risks. Overall, 12 risk types have been identified:
Internal incompatibility risk. Risk that two (or more) of the adopted components have licenses that are compatible with each other.
External incompatibility risk. Risk that the target license (licenses, in case of dual licensing) is incompatible with one or more of the component licenses.
Lack of affinity risk. Risk that arises from the need of maintaining a given corporate licensing scheme. It measures how this set of components, although being compatible, deviates from the desired scheme.
Future uncertainty risk. Risk due to the low degree of freedom in the choice of the target license.
Reduced target license set. Risk due to the low degree of freedom in the choice of the target license because of the licenses of all components.
Declining components licenses. Risk that the project includes components

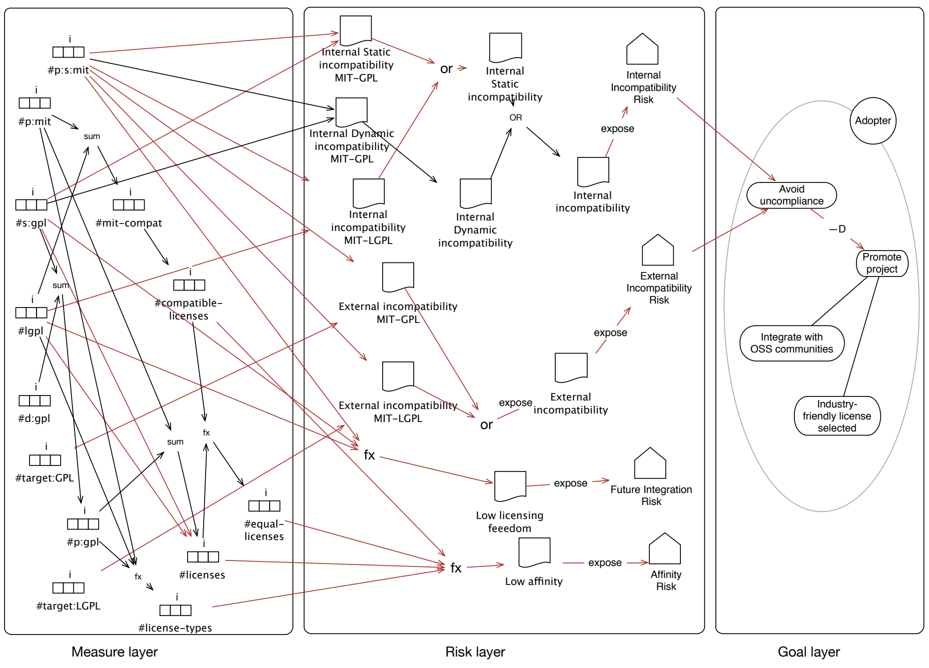
Measure layer
Risk layer
Goal layer
#p:s:mit
#p:mit
sum
#s:gpl
#mit-compat
#compatible-licenses
#lgpl
#d:gpl
#target:GPL
#p:gpl
#target:LGPL
#licenses
#license-types
#equal-licenses
fx
Internal Static incompatibility MIT-GPL
Internal Dynamic incompatibility MIT-GPL
Internal incompatibility MIT-LGPL
External incompatibility MIT-GPL
External incompatibility MIT-LGPL
or
Internal Static incompatibility
Internal Dynamic incompatibility
OR
Internal incompatibility
Internal Incompatibility Risk
expose
External Incompatibility Risk
External incompatibility
Low licensing feeedom
Future Integration Risk
Low affinity
Affinity Risk
Adopter
Avoid uncompliance
D
Promote project
Integrate with OSS communities
Industry-friendly license selected

Fig. 1. RiskML license risk graph

released under declining licenses.
Declining target license. Risk that the selected target license is declining.
Infrequent components licenses. Risk that the project includes components released under rare or unusual licenses.
Infrequent target license. Risk that the selected target license is rare or unusual.
Lack of knowledge. Risk that, because of lack of knowledge in relation to the number of components licenses, the rest of the risk analysis is not completely trustable.
Obsolete components licenses. Risk that the project includes components released under obsolete licenses — a license is obsolete when at least a greater version of the same license exists.
Obsolete target license. Risk that the selected target license is obsolete — a license is obsolete when at least a greater version of the same license exists.

In order to drive the license selection process, the goals of the project have been analysed. Starting from the two aspects listed above, some relevant goals ere identified. As an example, we have:
Industry-friendly license selected This goal describes the need to select a license that can be used by potential third parties interested in exploiting the results of the project.

Number of components	25
Number of OSS libraries	194
ASL2	67
BSD3	3
BSD4	1
CC3.0	1
CDDL	9
CPL-EPL	31
GPL2	4
LGPL2.1	3
LGPL3+	5
MIT	31
Other/unknown	18

Table 1. Excerpt of the gathered licensing information.

Integrate with OSS communities This goal is motivated by the will to operate in the OSS context ant therefore contribute (if possible) with the communities.
Avoid uncompliance This goal makes explicit the intention to select a safe licensing policy, which protects the project members from licensing issues.

Each task leader was in charge of identifying and reporting the OSS components used by the software module(s) under his responsibility. Table 1 illustrates an excerpt of the data produced for the analysis. The project consisted in 4 workpackages responsible for software delivery. Overall, 25 software modules were in development in the various workpackages. The total count of OSS components used in the various modules amounts to 194. Out of these, 176 OSS components had a known license, belonging to 10 different license types. The other 18 had a license whose nature was either unknown or not captured by the model; moreover, for 1 of them if was not possible to find the license attribution.

Once collected, the data has been used to quantify the risk exposure for the project as a whole through automated reasoning. RiskML supports automated reasoning through a label propagation inference algorithm, which is an extension of goal reasoning techniques [3]. The algorithm is fed with numerical values, which correspond to the model input nodes (the indicators in Figure 1), and propagates those values through the graph, until the root nodes are reached (i.e., the goals). During this propagation, functions (added to the model as annotations) are used to, e.g., normalise the ranges of the indicators values, or to provide a semantics to AND/OR operators. In the end, a numerical quantification of a certain truth value, expressed as a real number in the range [0..1], is assigned to root nodes.

4 Results and conclusion

We described an approach to analyse software licence risks in multi-component software, which is goal-aware. We applied it to the analysis of the licences of the tool-suite under development in the SUPERSEDE research project, with the main objective of identifying potential violations as cause of strategic failures. As a preliminary result, the performed analysis allowed to identify 5 license violations that prevented the project from achieving its goals. On the other hand, the approach allowed use to to infer the exposure of goals to license risks in a context of a real research project wanting to adopt open source software. License risk models were previously built taking into account knowledge from OSS experts about compatibility and available metrics. The risks ave been put in relation to goals after internal discussion sessions. The risk models were able to capture an important part of the expert knowledge and would thus be able to create risk awareness for non-expert analysts and managers about the impact of risks on the organisational goals. The approach uses a label propagation algorithm on the extended RiskML+*i** risk model, to give evidence on license risks and their impact on goals. Main threat to this approach stay on one side in the difficulty to capture effectively the decision making models that are used by experts in practice, and on the other hand in the general difficulty to find correlations between indicators, risk and goals.

Acknowledgement

This work is a result of the SUPERSEDE project, funded by the H2020 EU Framework Programme under agreement number 644018. We also thank the EU FP7 RISCOSS project for the research results exploited here.

References

1. Y. Asnar, P. Giorgini, and J. Mylopoulos. Goal-driven risk assessment in requirements engineering. *Requir. Eng.*, 16(2):101–116, 2011.
2. D. Barone, L. Jiang, D. Amyot, and J. Mylopoulos. Reasoning with key performance indicators. In P. Johannesson, J. Krogstie, and A. Opdahl, editors, *The Practice of Enterprise Modeling*, volume 92 of *Lecture Notes in Business Information Processing*, pages 82–96. Springer Berlin Heidelberg, 2011.
3. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Formal reasoning techniques for goal models. *J. Data Semantics*, 1:1–20, 2003.
4. I. Herraiz, D. Izquierdo-Cortazar, and F. Rivas-Hernández. Flossmetrics: Free/libre/open source software metrics. In A. Winter, R. Ferenc, and J. Knodel, editors, *CSMR*, pages 281–284. IEEE, 2009.
5. D. Izquierdo-Cortazar, G. Robles, J. M. González-Barahona, and J.-C. Deprez. Assessing floss communities: An experience report from the qualoss project. In *OSS*, page 364, 2009.
6. M. Morandini, A. Siena, and A. Susi. Risk awareness in open source component selection. In *Business Information Systems (BIS'14)*, 2014.
7. A. Siena, I. Jureta, S. Ingolfo, A. Susi, A. Perini, and J. Mylopoulos. Capturing variability of law with Nòmos 2. In *ER'12*, LNCS 7532, pages 383–396, 2012.
8. A. Siena, M. Morandini, and A. Susi. Modelling Risks in Open Source Software Components Selection. In *ER'14*, LNCS, 2014.