1.1 Systematic literature review: Risks in OSS adoption

We conducted a systematic literature review (SLR) on risks in OSS adoption, with various objectives: firstly, to describe the state of the art in the field, to understand the main research problems, the difficulties, the limitations of current approaches and open issues; secondly, to study the terminology used in the field, to structure, categorize and detail the problem domain, i.e. risks, risk measures, mitigation strategies, and the peculiar role of the OSS development and business model in OSS component adoption; and lastly, to contribute to the discovery of the initial requirements for a new risk analysis process, with a preliminary screening of the techniques used for data gathering, for giving evidence to claims and for the evaluation of the proposed approaches, as an indicator for their reliability and maturity, as a potential source for data, and as a guide for the evaluation of own approaches.

The SLR focuses on the identification of technical, legal and organisational risks and risk analysis techniques in the domain of open source software adoption, where we intend adoption mainly as component integration. To obtain a more complete view on the state of the art in risk analysis in this domain, it includes also literature, which mainly concerns off-the-shelf (OTS) components in general. It excludes literature on business-related risks, which are analysed in Work Package 3, security and safety risks. Moreover, it excludes OSS adoption by the final user (i.e. deployment), e.g. we are not considering to analyse the challenges and risks in the adoption of Linux, Firefox or LibreOffice by an organisation, nor the use of server applications such as the LAMP stack.

In the following, we detail the SLR process, summarize the results, present the obtained terminologies, and discuss the achievements and lacks of current approaches.

1.1.1 Systematic literature review: protocol

For the SLR we defined the following protocol, adopting the guidelines by [Kitchenham and Charters, 2007]. The single steps are documented and the retrieved data stored in a repository accessible by the project partners.

1. Definition of the purpose and intended goals of the review
2. Definition of the details of the literature search. The search strategy includes:
   - definition of the publication channels to be searched (primary conferences, journals, other traceable sources of knowledge);
   - definition of search terms and selection of the libraries to be searched;
   - selection criteria, which are used to determine which studies are included in, or excluded from, a systematic review. The justification that the comprehensiveness is still assured with the adopted exclusion criteria

4. Relevance and quality assessment procedures (screening for exclusion), describing what will be the selection criteria to remove papers of insufficient quality or that are out of the domain, e.g. with checklists. The assessment is made, in this order, on title, abstract, introduction and the full text.
5. The data extraction strategy that defines how the information required will be obtained systematically.
6. The quantitative or qualitative analysis of the extracted data, in a way that it could be independently reproduced, and the writing of the detailed results of the review.
1.1.2 Purpose of the SLR

The purpose of this SLR is to define the state of the art, the terminology and the techniques used, in the field of risk analysis for OSS component adoption for the development of commercial and open-source software, focusing on technical and organisational risks. It excludes business-related risks (analysed in Work Package 3), security and safety risks. Moreover, it excludes the adoption of OSS by end-users.

1.1.3 Literature search: publication channels

We defined an a-priori a set of source publication channels where we expect to find a comprehensive set of scientific publications on the argument. This set is coordinated with the sources used for the literature review on open source ecosystems and ontologies, contemporarily carried on in WP1, and with the involved project partners. During the manual screening for identifying promising publications, other relevant channels were added.

1.1.3.1 Journals and Magazines


1.1.3.2 Conferences and Workshops


1.1.3.3 Newly added sources recognized to be important in the field


1.1.4 Search terms and libraries

To find relevant articles, we searched 5 popular, complete and comprehensive meta-libraries for publications in computer science. Moreover, if we recognized that one of the sources listed above is not indexed in anyone of these libraries, we added the corresponding library. Search in meta-libraries is carried out with a conjunction of the following search strings. Most of the search engines used did not give a larger set of results by using also the plural or longer acronyms such as "FLOSS". As a result of some tests, the terms "Free software" and
"Libre software" were not considered as synonyms of OSS, since they are mainly used in the field of end-user software. We list a general form of the search string. The concrete search string used depends on the syntax available in each meta-library search facility. Sometimes, also a stepwise search was necessary.

1.1.4.1 Search terms (used in conjunction)
The keywords to find relevant papers have been the following:

- To find papers related to Open Source Software: OSS OR “Open Source”
- To find papers related to Risks: Risk OR Obstacle OR Problem
- To find papers related to Adoption: Adoption

We would like to note that we excluded synonyms of risk such as pitfall, complication, trouble, danger, or peril, because they are related more to business issues, to safety issues, or not commonly used in this context. We included the term Problem, because it gives some relevant results, even if many of the papers retrieved are out of the topic.

1.1.5 Selection/exclusion criteria

Literature collected searching the repositories previously defined is first scanned for duplicates. Then, the criteria for relevance and quality assessment (i.e. screening for exclusion) are defined.

1.1.5.1 Step 1: direct exclusion criteria

The criteria for direct exclusion of papers have been the following:

- Papers not written in English.
- Introductions, indexes, book reviews, PhD symposium papers, editorials.
- Short papers or demo papers shorter than 3 pages.
- Papers, which are not published in one of the venues (Journals/ Magazines/ Conferences/ Workshops) defined above need to undergo a manual screening for relevance by evaluating, in this order, venue, title, keywords and abstract. In this step, the list of venues could be completed. Additional workshop papers are included only if they have a particularly high relevance and quality.

While performing the review, additional exclusion criteria appeared, that are documented in each phase.

1.1.5.2 Step 2: Exclusion based on title and keyword relevance

Papers have also been excluded by means of a screening, based on the venue and topic relevance (removing papers which speak only about security risks, social risks, software use in education, electronic devices, mathematical problems, medical risks, disaster management).

1.1.5.3 Step 3: Exclusion based on reading of abstract and introduction

Lastly, papers have been excluded on the basis of their content, with the objective to find the most relevant papers for an in-depth reading, aimed at contributing to the main purpose of the SLR: to identify technical and organizational risks and risk analysis techniques in the domain of open source software adoption. On the basis of a screening by reading the abstracts and, if necessary, the main parts of the paper, we excluded papers which do not contribute to this approach; in particular papers prevalently about:
• Business-related and legal risks (analysed in WP3)
• General security and safety risks, and risks which are not related to software engineering (e.g. papers on disaster management)
• Mathematical problems, optimisation problems, etc.
• Focussing only on the development of open source
• Papers on the end-user adoption or deployment of open source programs such as Linux, Firefox or LibreOffice by end users or on Servers (e.g. Tomcat, PhP)
• "OSS" used as a different abbreviation (Operational Support System, Operating SystemS)

The selected papers will thus contain information on:
• Technical risks in OSS adoption and use.
• Risk analysis techniques in the domain of open source software (adoption).
• Limitations of current approaches, capabilities needed for new approaches.
• Guidelines for risk mitigation with COTS/OSS use.

1.1.6 Data Analysis and Extraction

Each of the papers remaining from the previous steps has been read in-depth in a collaborative way, with focus on the following aspects: the use and definition of concepts related to OSS, risks, risk measures and mitigation activities, with the aim of creating a terminology of the concepts used; the community or OSS ecosystem structure and the underlying business strategy of involved companies and individuals; the modelling approach used, if any; the documented problems encountered, the risks that were identified (in explicit or implicit way), the mitigation strategies, and their validation. The papers are then clustered by research group (to find correlated works) and by the domain of the paper (OSS, adoption, COTS, risk analysis,..., SLR). With this information, the following form is filled.

1. Main message of the paper
2. Paper type (exploratory, theoretical, experimental,...)
3. Important concepts defined
4. Risks or issues described
5. Risk measures described, data analysis techniques described
6. Decision and mitigation techniques described
7. Paper type (exploratory, theoretical, experimental)
8. Maturity (intuitive, tested on example, qualitative (on questionnaires), quantitative (on collected data), semantic (NLP on text), Integrated (ETL, various data sources in sync.)
9. Evaluation (none, informal, qualitative or quantitative experiment, formal) and data obtained
10. Subjective importance for the RISCOSS project
11. Evaluation results and reader's subjective comments
12. Identified differences between COTS and OSS adoption
13. Lessons learned (in the paper and by the reader)

1.2 Systematic Literature Review: Execution and Paper Analysis

The search is done on the following five meta-libraries, by using the web-search option available in the Open-Source application JabRef. A search for duplicates is made while adding the results for each library. Moreover, a first screening on title and keywords is
performed, to ensure the efficiency of the search terms and to remove clearly unrelated papers. Note, that the search engines index different data to be used in the searches, e.g. some include the abstracts, others not.

- **DBLP**: Search executed with several variants and conjunctions of the search terms: 125 results. 44 papers added.
- **IEEE Library**: Search executed with several variants of the search terms, including plurals: 11 searches, 236 results without duplicates. 207 added.
- **ACM DL** (search is limited to the 20 most relevant papers for every search key): 6 searches, 86 papers retrieved, 10 duplicates, 76 added.
- **Additional manual search**:
  - SpringerLink: not exportable. Manual check of the first 20 publications for relevant articles for search string 'risk AND "open source"'. 2 added.
  - ACM DL: manual search on "risk AND OSS", results ordered by recent date, first 100 results (2011-2013) analysed. 3 relevant journal papers identified and added

Total papers in the review: 332.

**Relevant paper extraction STEP 1**
Manually delete publications, which are not articles (e.g. full proceedings (18), collections (1), books (2)). Remaining papers: 311.


**Relevant paper extraction STEP 2**
Screening for venue and additionally for topic relevance, based on title and keywords. Remaining: 93 plus 13 interesting, not directly relevant papers.

**Relevant paper extraction STEP 3**
Screening based on reading of abstract and introduction: The result of this step, performed as described above, are the 47 relevant papers for the review, reported in Section 1.3.

**1.2.1 Paper Analysis**
A detailed review of the selected 47 papers reported in Section 1.3 (please note that these references are indicated as [SLR1] to [SLR47] in this section, and on purpose not merged with the rest of the references) gave a comprehensive overview on the actual state of the art and the community in the field of risk analysis and management in the domain of Open-Source-Software adoption. Furthermore, we extracted several relevant aspects in a structured manner, in order to build glossaries of terms used and to discover relevant measures and techniques. In the following, we categorise the papers, highlight the differences between OSS and COTS adoption as individuated in these papers, extract and structure the issues and risks (which can also be seen, in a positive way, as opportunities) that were evidenced or analysed in the papers, and explain the measures and mitigation activities proposed, used or validated.
1.2.1.1 Categorisation

For the purpose of a bitter overview, we divided the resulting articles into seven main areas, based on the main focus of their content. The categories were defined based on a partitioning of the articles, guided by the relevant concepts in the domain of this SLR, that are OSS, risk, and component adoption. These categories are not mutually exclusive, rather, they define different levels of coverage of the concepts of interest.

- OSS component adoption: 12 papers, thereof one paper low-level (line-wise) adoption
- OSS in general (use, participation, OSS practices): 17 papers
- Off the shelf (OTS) components and development (commercial and OSS): 10 papers
- Software risk analysis (in general): 1 paper
- Software quality assurance (relevant for risk management): 2 papers
- Software in general: 2 papers
- Systematic literature reviews (SLR): 3 (1 on OSS adoption, 2 on OSS development in general)

The reader can note that only one quarter of the papers explicitly deals with OSS component adoption. However, most of the other papers include parts treating this topic, among other, more general aspects.

An analysis of research groups collaborating on a single topic gave only one mentionable cluster of work, namely the nine papers by the first authors Jingyue Li [SLR18][SLR19][SLR20][SLR47], O. Hauge [SLR12][SLR13], R. Conradi [SLR7] and C. Ayala [SLR2][SLR3].

1.2.2 Differences between COTS and OSS adoption identified in literature

The RISCOSS project has a concrete focus on the adoption of open-source software. Processes for commercial component selection and risk evaluation are adopted, in a more or less formal way by many medium and big companies [Li, 2006] [Carvallo et al., 2007] [Ayala et al., 2011] [Comella-Dorda et al., 2002]. However, these processes are not suitable, or are suitable only partially, to the evaluation of OSS components, because they rely on direct costs, quality metrics, responsibilities and certifications such as CMMI (Capability Maturity Model Integration), which are not available or difficult to obtain without the possibility to evaluate the company behind a component [SLR29]. Also, quality metrics are difficult to obtain, because traditional project monitoring focuses on human reporting in a tightly coupled organization [SLR44]. Moreover, OSS demands for decision criteria where indirect costs, an evaluation of the community, etc. need to get particular attention. Here we give a structured view on the main differences between COTS and OSS adoption, as identified in literature, as a starting point for focussing on the relevant aspects for an OSS adoption risk analysis process.

1.2.2.1 Access to data

By the definition of the Open Source Initiative (OSI), open source software must comply, among others, with the following criteria: the source code must be available, it must be allowed to copy and freely redistribute it, to improve the program, and to release these improvements.

This definition and its consequence of possibly having a community working on the software bring about the most evident differences from commercial software (certainly, exceptions...
prove the rule), the availability of the source code and of a bug tracker [SLR15][SLR34][SLR19][SLR17], which make white box testing possible [SLR34], and access to data by and about the developer and user community, which makes various community measures possible [SLR17][SLR34].

1.2.2.2 Differences emerging from the organizational structure

Having no purchasing costs, but typically various hidden costs, implies the need for a different focus in component evaluation processes [SLR40][SLR4][SLR39][SLR12]. Another critical difference, vendor (in)dependency [SLR40], was already highlighted in the previous sections. Evaluation thus needs to be project-, not company-oriented [SLR29], and commercial contact for support [SLR19][SLR12] can bring different opportunities (e.g. competition) and challenges (e.g. missing competence), and the need for creating expertise in the adopting company [SLR12].

The community-oriented organisational structure leads to novel license, liability, responsibility, and intellectual property issues [SLR9][SLR12][SLR40], highlighted in most literature on OSS risks. Uncertainty about product future [SLR40][SLR4], unknown development schedules [SLR17] and the uncertain long-term support [SLR20][SLR27] provide further challenges.

1.2.2.3 Differences emerging from the OSS development process

Due to the development processes followed in the OSS community and the reduced influence from companies' business strategies, various qualities are attributed to OSS and emphasized as differences to COTS, such as a missing long-term roadmap [SLR34][SLR46], lower quality documentation [SLR40][SLR4], the different testing process [SLR27], and missing unambiguous specifications [SLR1] with few architectural choices or design for usability [SLR32]. Moreover, rapid code and API change [SLR1][SLR4], lacks in quality evaluation [SLR40], but also better adherence to open standards are attributed to OSS projects.

However, various works also underline the similarities of OSS and COTS. In practice, code changes are seldom, and external support can be ensured, so update difficulty, liability, licenses, and testing do not differ significantly [SLR3][SLR18].

1.2.3 “Risk” in OSS adoption – a taxonomy from literature

In the following we give a structured list of terms for the various risks which were identified, which were discussed or which emerged from surveys or other analyses, in the reviewed papers. Inspired by the various categorisations given in some of the works, we define six main types of risks (which can, however, also be seen as opportunities) in our domain of interest: risks inherent to the component selection and integration process, knowledge-related risks (e.g., depending on experience or on availability of documentation and training), risks linked to legal issues (property rights) ad business models, risks in maintenance and support, risks emerging form the community and its organisation, and risks related to code quality.

1.2.3.1 Selection and Integration Risks

- component selection risks
  - requirements satisfaction [SLR14][SLR38], requirements not negotiable [SLR16][SLR47]
  - lack of products [SLR12]
  - decide what fork should be chosen [SLR40]
identification of product quality, uncertainty about quality [SLR40]
- adoption (integration) process risks [SLR19]
  - uncontrolled adoption [SLR12], less caution in selection of OSS [SLR19]
  - no care for quality in adoption [SLR1]
  - lack of OTS-driven requirements engineering process [SLR16]
- integration risks [SLR40][SLR6]
  - integration effort ill-estimated [SLR16][SLR19][SLR47][SLR40]
  - deployment risks [SLR47] compatibility [SLR28]
  - complicated multi-OTS components arrangement [SLR16]
- final product project risks (budget, delivery time, required quality) [SLR34]
  - impact on product reliability [SLR17][SLR47][SLR40][SLR41][SLR42]
  - impact on product security [SLR1][SLR28][SLR47]
  - impact on product performance [SLR47]

1.2.3.2 Knowledge-Related Risks
- company human capital risks [SLR21]
  - availability of skilled personnel [SLR38]
  - lack of expertise [SLR12][SLR28]
  - product training availability [SLR2][SLR34]
- lacks in documentation
  - documentation availability [SLR17][SLR34]
  - Insufficient component documents [SLR16]
  - tutorial and book availability [SLR34]

1.2.3.3 Legal and Business Risks
- legal issues
  - legal, intellectual property [SLR2][SLR9][SLR22][SLR40][SLR46]
  - licensing [SLR12][SLR14][SLR22][SLR24][SLR34][SLR40]
- liability [SLR9]
  - no ownership [SLR28]
  - liability for component quality [SLR24]
  - lack of providers [SLR12]
  - project-specific, not company-specific [SLR29]
  - lack of information on provider [SLR16]
  - (no) vendor lock in [SLR28]
- monetary risks
  - cost, monetary risk [SLR25][SLR28] availability of financial resources [SLR38]
  - hidden costs [SLR12]
  - market success [SLR26]
- non-IT organisational risks (business agreements, ideology, management support) [SLR47]
- OSS component business model [SLR40]
  - commercial versions [SLR34]
  - availability of additional non-OSS functionalities, licensing costs [SLR38][SLR2]

1.2.3.4 Maintenance Risks
- versioning, update risks [SLR2][SLR28][SLR37][SLR40][SLR46]
  - update frequency [SLR34]
  - innovation (as benefit) [SLR28]
  - explicit and hidden semantic changes [SLR15]
    - API changes [SLR4]
    - quality changes [SLR15]
    - impact on self-implemented code [SLR15]
    - upgrade backwards compatibility [SLR40]
    - upgrade feasibility [SLR16][SLR47]
• lack of support [SLR9][SLR12][SLR16][SLR47][SLR40]
  o uncertainty in service and support [SLR21]
  o bug correction/fix time [SLR5][SLR15][SLR34]
  o unknown development schedules [SLR17]

1.2.3.5 Community-Related Risks

• project community activeness [SLR34]
  o presence of heroes [SLR32]
  o absence of key committers [SLR44]
  o de-motivation of the project community [SLR44]
  o fail to create community [SLR8]
  o lacks in marketing [SLR28]

• possibility of contribution [SLR34][SLR35][SLR40]
  o patch rejection, change acceptance [SLR35][SLR40]
  o cost of contribution [SLR35][SLR42]
  o not adaptable to requirements changes [SLR16][SLR47]
  o reduced control of future evolution [SLR16]

• uncertainty in future and roadmap [SLR34][SLR46]
  o standards implementation [SLR34]
  o maintenance planning unfeasible [SLR16][SLR47]
  o impossibility of ensuring future response time [SLR34]

1.2.3.6 OSS Component Code Quality Risks

• lack of information on quality [SLR24]
  o lack of a testing process [SLR27]
  o testing shortfalls [SLR34]
  o overlook NFR’s [SLR9]
  o maturity [SLR38]

• usability, user friendliness [SLR28][SLR31]

• component dependability [SLR1]

• bad code quality [SLR15][SLR24][SLR30]
  o difficult defect localisation [SLR19][SLR47]
  o bug risk [SLR30] error proneness [SLR36]
  o component dependency (code level) [SLR34]

• flexibility of use [SLR28]

1.2.4 Measures used in literature

Various measures are used in literature that explore the particular data provided by OSS projects, to quantify various qualities of the source code, data on bugs and bug fixing, on the community and its future roadmap, and on the adopting organisation. Also, subjective expert opinions can be a valuable source for measure. In the following subsections, we define five categories of measures and list them as they were identified or used in the single papers, without going into detail for single values.

1.2.4.1 Code metrics:

API usage metrics, tool, informal results analysis [SLR4], analyse dependencies, detect 3rd party signature changes, detect 3rd party code internal changes, code quality: QBench benchmarks [SLR15], module usage (coupling), fault count, complexity (cyclomatic number) [SLR17][SLR36], software quality metrics, documentation and Interaction support, integration and adaptability metrics [SLR1], Sotograph code metrics [SLR31], regression to find associations metric-fault [SLR36], lines of code [SLR37].
1.2.4.2 Bugtracker:
Bugginess measures [SLR37], correction time [SLR5], invalid method parameters fault, wrong method usage fault, faulty method implementation, environment fault [SLR6], analyse bug tracker [SLR15], bug reports [SLR17], keyword search in logs for "bug", "problem" [SLR30], repository measures (number of committers, number of commits by committer, etc.) [SLR33], bug reports [SLR37], mean time between software failures [SLR42].

1.2.4.3 Community and Roadmap:
Various developers community and users community measures, project activity and release delivery frequency [SLR8], development environments, community group activities, and ongoing efforts [SLR17], mails per month, developer interactions, active power users, postings vs. downloads [SLR44].

1.2.4.4 Expert opinions, involved companies, support
Significance (expert opinion) [SLR17], interviews to developers and managers [SLR18], support reputation [SLR19], perception [SLR28][SLR29], questionnaire-based on all aspects of adoption risk, measure level of usability-awareness: preliminary, recognized, defined, streamlined, institutionalized [SLR32], understand where the managers see the risk [SLR43].

1.2.4.5 Metrics for the adopting company, business value metrics
Organization size, magnitude of outsourced IT, prior experience in adoption of OSS and proportion of IT employees within an organization [SLR21], Real Options Theory: analyse, in a monetary fashion, the economic value, Petri nets, RiskSimulator tool [SLR25], testing process certification, TMM Testing Maturity Model [SLR27], developer experience [SLR37].

1.2.5 Risk mitigation
Few publications in this field speak about risk mitigation. Mostly, mitigation of the various risks encountered in OSS adoption is only mentioned informally, in form of general hints, such as to train the people, i.e. to develop the existing stock of human capital [SLR21], to follow general COTS adoption decision processes, to evaluate the community [SLR24], to evaluate similarity to previous projects [SLR44], to evaluate the OSS project’s roadmap and possible future directions [SLR46], or to make managers aware of risks and opportunities [SLR37]. Empirical experiments were used to identify risks and to identify effective mitigation activities. However, none of the works showed evidences for causal relationships between these risks, concrete measures and the effectiveness of the mitigation activities. Only for more concrete risks, as e.g. for lowering the risk of introducing errors when upgrading to new versions, concrete mitigation activities were proposed, such as automatically checking API compatibility [SLR4] or exploring code executability with test cases [SLR14] to ensure correct functionality.

1.2.6 Risk, events, metrics, mitigation: a conceptual map
The following figure shows an excerpt of a map of the main concepts related to risk and risk analysis in OSS adoption, putting terms in context on basis of [Kenett and Raanan, 2010] and the papers resulting from the SLR. The resulting terminology, which needs to be discussed with the RISCOSS partners, refined and completed, and can be seen as a first step towards a formal ontology for risks in OSS adoption.
1.2.7 Data retrieval and empirical evaluation for OSS risks and mitigation activities

In this section we report the forms in which data was acquired and proposals were evaluated in the identified literature.

Out of the 47 analysed papers, three were literature reviews, of which two can be defined as systematic. Twelve papers are theoretical, but evaluated only informally, by a proof-of-concept application to a single example. 13 papers report, analyse and draw conclusions from surveys, questionnaires and interviews with developers and managers in software firms, basing mostly on qualitative data such as opinions. Other ten papers based their analyses on...
quantitative data, with data from log files for retrieving the software used by servers, from bug tracking systems for activity, bug reporting and resolution data, from code (various code metrics, available test cases, interface analysis), and code repositories metadata, e.g. for changes in time and changes per developer (see Section 1.2.4).

Notice that there are a huge number of publications, which report and interpret results from qualitative and quantitative studies to identify possible risks. It stands out that only [SLR40] calculates threshold values for defining bug risks, and evaluates their performance, while no paper identifies risks based on quantitative data of project failure or created losses and revenues, or correlates project failures and losses with quantitative data such as the number of bug reports and bug fixes.

Few papers consider quantitative measures on community qualities (number of contributors, activity, presence of heroes [SLR39] etc.), while no work empirically evaluates the existence of causal relationships between the metrics applied, the risks identified and the actual faults happening. Moreover, an empirical evaluation of the effectiveness of mitigation activities by their influence on the retrieved metrics is also missing in the works, which propose specific mitigation activities. Also for very complete works, such as the ones by the group [SLR12][SLR19][SLR20][SLR47], whose surveys retrieve data for risks and mitigation activities, do not show any link between these two: typical mitigation activities adopted in software companies are very general (see Section 1.2.5) and would reduce various risks.

1.3 References: SLR Selected Papers


1.4 Discussion about limits of the current approaches in OSS adoption risk management and about research opportunities

Our focus in this overview is adoption, intended as integration of OSS components, by developers and companies, for the realisation of close and open source software products. The works we have analysed give a broad coverage of the state of the art in this area. However, we also recognised several limits, which give space to research opportunities.

Various publications analyse, on one side, the security risks of OSS and challenges and failures in OSS deployment, and on the other side, they analyse risks and propose processes for COTS adoption, while a few works have been done specifically in our topic of interest.

The majority of recent papers on risks in OSS and COTS adoption analyse the as-is situation, applying literature studies and empirical studies mainly basing on surveys and interviews, and some times (for OSS) on a quantitative analysis of source code, repositories metadata and bug trackers. For example, the complete and detailed joint work by Conradi, et al. [SLR7][SLR12][SLR18][SLR47] classifies risks and mitigation activities on the basis of literature and various surveys, but relies, like other works, mostly on subjective perceptions, performing no evaluation of the effectiveness of the mentioned mitigation activities. Experiments to show the causal relations between failures, risks, measures and mitigation activities with statistical evidence, were performed merely for security and business aspects.

Some of the analysed approaches propose quantitative measures and analyse their effectiveness (e.g. [SLR4][SLR6][SLR15] in the SLR for API metrics and code changes, [SLR14] for code executability, [SLR25] for business values). [SLR17] proposes a reliability
model combining qualitative and quantitative metrics, but does not consider OSS-specific community and repository measures. In general, a few works that propose metrics to support OSS selection take account for the organisation that adopts the component, including the company’s size and structure, its internal processes and the quality requirements for its products.

Our aim is to define a process that improves the selection, management and maintenance of OSS components. We will review existing COTS quality assessment processes, including the ones enacted in the project partner companies in practice. Giving a particular focus on the differences between OSS and closed source components identified in the SLR, we try to understand the causal relationships between risks and failures, risk metrics, and risk mitigation. To propose such measures we investigate into two directions: (i) evaluating the OSS project community ecosystem and the adopting company, by organisational modelling and analysis, and (ii) evaluating the data available in the OSS projects repositories, such as code repositories, bug trackers, and mailing lists, applying measurements based on a statistical analysis. Techniques and tools for this analysis, the prioritisation and multi-criteria decision making will be presented in the next section.