To appear in *Computer Science Education*
Vol. 00, No. 00, Month 20XX, 1–37

ARTICLE

Open Source Projects in Software Engineering Education: A Mapping Study

Debora Maria Coelho Nascimento[a*], Roberto Almeida Bittencourt[b], Christina Chavez[c]

[a] *UFS - Federal University of Sergipe, Brazil*; [b] *UEFS - State University of Feira de Santana, Brazil*; [c] *UFBA - Federal University of Bahia, Brazil*

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

Context: It is common practice in academia to have students work with "toy" projects in software engineering courses. One way to make such courses more realistic and reduce the gap between academic courses and industry needs is getting students involved in Open Source Projects (OSP) with faculty supervision. Objective: This study aims to summarize the literature on how open source projects have been used to facilitate students' learning of software engineering. Method: A systematic mapping study was undertaken by identifying, filtering and classifying primary studies using a predefined strategy. Results: 72 papers were selected and classified. The main results were: a) most studies focused on comprehensive software engineering courses, although some dealt with specific areas; b) the most prevalent approach was the traditional project method; c) studies' general goals were: learning software engineering concepts and principles by using OSP, learning open source software, or both; d) most studies tried out ideas in regular courses within the curriculum; e) in general, students had to work with predefined projects; f) there was a balance between approaches where instructors had either inside control or no control on the activities performed by students; g) when learning was assessed, software artifacts, reports and presentations were the main instruments used by teachers, while surveys were widely used for students' self-assessment; h) most studies were published in the last seven years. Conclusions: The resulting map gives an overview of the existing initiatives in this context and shows gaps where further research can be pursued.

Keywords: software engineering education; open source software; systematic mapping study

1. Introduction

Learning software engineering (SE) requires more than just acquiring content knowledge. It usually demands learning skills and attitudes that are hard to acquire inside a higher education environment. For example, taking part in a long-term project that usually lasts one or more years is an activity that does not fit well in an undergraduate curriculum split into four-month academic terms. Authentic environments and situations, with different roles to play, with novices learning from experts, with time and budget constraints, where customers dialogue with team members, and where team members practice different skills such as communica-

*Corresponding author. Email: dmcnascimento@ufs.br

tion, leadership, conflict resolution and decision making, are very hard to reproduce inside academia.

Some initiatives to foster cooperation between undergraduate programs and industry have been reported, e.g., co-op programs (Reichlmayr, 2006). However, typical schedules of co-op programs are limited by the delay between learning concepts in class in one term, and practicing them in an industrial setting in another term. In addition, concepts introduced by SE faculty are usually illustrated with "toy problems" with reduced size and complexity, raising the issue of authenticity.

To deal with those issues, one approach is gaining momentum in SE courses: having students take part in open source projects (OSP) with faculty supervision. Practical sections of SE courses are performed and assessed from student participation in such projects. OSP give room to an environment where experts and novices interact, real products are developed and evolved, where work structure is well-defined, and different roles are played by stakeholders. Since this is an authentic environment where real software is being produced, faculty can usually cover most SE knowledge areas.

However, regardless of the authentic environment provided by OSP, students must perform activities leading to significant learning. Previous studies provide evidence that active learning approaches promote student engagement, enhance academic achievement, student attitudes and student retention, offerong a natural environment to enhance interpersonal skills, and to develop problem-solving and life-long learning skills (Prince, 2004). Barg et al. (2000), for instance, reports that problem-based learning (PBL) "*without losing any of the technical skills, it makes room in the course for activities that encourage generic skills*". We argue, thus, that active learning practices are a good combination with OSP in order to learn SE.

Given that context, we decided to perform a literature review of the scholarly production that relates SE education to OSP, in order to uncover experiences where OSP facilitates learning of relevant SE concepts. We were not particularly interested in academic reports of open source software as a development tool or a lab environment. Instead, we decided to focus on studies where students actively engage with OSP, analyzing their source code, designing and implementing changes, performing tests and other quality checks, and participating in their communities. The present systematic mapping study was produced to achieve this goal and reach a comprehensive view that allows steering future research.

Systematic mapping studies are secondary studies similar to systematic reviews. Both studies search primary studies indexed in scientific databases, select revelant studies, and classify them according to objective criteria. The main difference is that while systematic reviews examine primary studies in depth, aiming to generate conclusions about a particular research question, mapping studies mainly intend to categorize them, usually providing a visual summary of results (Petersen, Feldt, Mujtaba, & Mattsson, 2008). Not only are evidence clusters identified, but also evidence gaps where additional research could be conducted are uncovered (Kitchenham & Charters, 2007). A mapping study is mainly an exploratory research strategy, which fits well to the purposes of our research.

In this study, we extend a mapping study we previously published (Nascimento et al., 2013). Here we add steps to the process, snowballing selected papers after paper screening to find additional relevant studies not identified in the electronic databases. We also added new facets to the classification scheme, to describe learning objectives, how the approach is embodied in the curriculum, how the OSP is

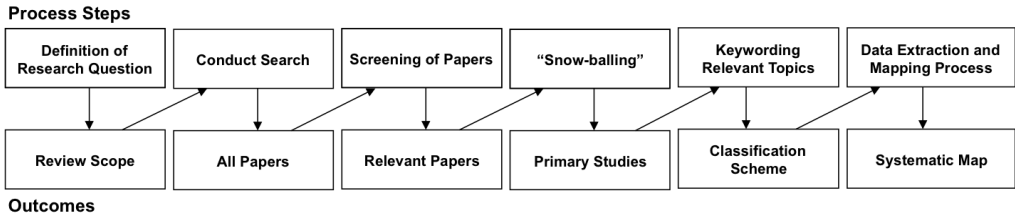

Figure 1. The systematic mapping study process, adapted from Petersen et al. (2008).

chosen, and how much control instructors have on student activities. As a result, 72 papers were selected and classified.

The main findings were: i) most studies focus on comprehensive software engineering courses (50 articles), although there are some studies on specific areas, e.g., design/architecture, development/construction and evolution; ii) there is seldom explicit information about a pedagogical theory or framework used as a basis to the studies, and the most prevalent approach is the traditional project method; iii) studies' general goals are: learning SE concepts and principles by using OSP, learning OSS, or both; iv) most studies embody the approach in regular courses within the curriculum; v) in general, students have to work with predefined projects; vi) there is a balance between approaches where the instructor has either complete or no control of the activities performed by students; vii) when learning is assessed, software artifacts, reports and presentations are the main instruments used by teachers, while surveys are widely used for student self-assessment; viii) most studies were published in the last seven years in journals and conferences related not only to computing education, but also in general venues related to computing, software engineering or OSS; ix) there are various active research communities around the world, especially in the US, Canada and Europe.

The main contribution of this study is to provide a comprehensive view of the use of OSP to learn software development and SE principles. Not only do we identify the active research communities, long-term projects and venues for publication, but we also provide an overview of objectives, contributions and methods applied in 72 studies over the latest 14 years. This view is important both to researchers interested in the identified trends and gaps, and to instructors interested in trying out their own experiences in their classes, using the repertoire of the different approaches we identified.

This paper is organized as follows. Section 2 presents an overview of the mapping study process. In Section 3, we present the results. Then, in Section 4, we discuss the main findings and threats to validity. Finally, in Section 5 we draw conclusions and suggest areas for further research.

2. The Systematic Mapping Study Process

In this work, we performed a systematic mapping study (Budgen, Turner, Brereton, & Kitchenham, 2008; Petersen et al., 2008) to analyze the extent and range of previous research that relate the use of OSP in software engineering education. Figure 1, adapted from Petersen et al. (2008), describes the process we used in this mapping study. This section describes how each step of the process was performed.

Definition of Research Questions.

The main research question of a systematic mapping study is typically broad.

Table 1. Standard Search String

"open source" **OR " free software" OR "libre software" OR**
"FLOSS" **OR "FOSS" OR "OSS" OR "OSP"**
AND
"course" **OR "curriculum" OR "education" OR "educational" OR**
"teaching" **OR "learning" OR "mentoring" OR "training" OR**
"apprentice" **OR "tutoring" OR "coaching" OR "skills" OR "competencies"**
AND
"software engineering" **OR "software requirement" OR "software modeling" OR**
"software analysis" **OR "software design" OR "software architecture" OR**
"design patterns" **OR "software verification" OR "software validation" OR**
"software evolution" **OR "software maintenance" OR "software process" OR**
"software quality" **OR "software metrics" OR "software management" OR**
"software testing" **OR "software configuration management" OR**
"computing" **OR "computer science"**

To give readers a broad view of previous research relating SE education and OSP, we asked the following question.

RQ1. How are Open Source Projects used in Software Engineering Education?

There is a bulk of literature on the use of OSP for pedagogical purposes, and we aim to uncover the experiences where such projects were used to foster learning of software engineering. We also would like to find out which software engineering knowledge areas are usually addressed by such initiatives, and we used the SWEBOK (2013) areas to classify them (e.g., software engineering in general, requirements, design, quality, maintenance, among others).

We also added two secondary research questions related to pedagogical issues, as follows.

RQ2. Which of the initiatives combine open source projects with active learning in software engineering courses?

RQ3. How is student learning assessed in the reviewed initiatives?

Conducting Search. We built a search string using the following steps, defined by Kitchenham, Mendes, and Travassos (2007): (a) derive major terms from the main question, by identifying the population and intervention; (b) identify alternative spellings and synonyms; (c) check keywords from known studies; (d) use the Boolean OR to incorporate alternative spellings and synonyms, and finally, (e) use the Boolean AND to link the major terms. To obtain a broad coverage, special attention was given to identify alternative spellings and synonyms. The resulting search string is presented in Table 1.

The search strategy included the following scientific electronic databases: Engineering Village[1], IEEE Xplore[2], ACM[3], Scopus[4], Springer[5] and Science Direct[6]. These libraries are important sources of SE and computer science studies, indexing relevant conferences and journals. Furthermore, their search engines support automatic search based on different criteria and string formats – which may help to ensure adequate coverage.

[1]http://www.engineeringvillage.com
[2]http://ieeexplore.ieee.org
[3]http://portal.acm.org
[4]http://www.info.sciverse.com/scopus
[5]http://www.springer.com
[6]http://www.elsevier.com

We used the search string to query the libraries against paper *title*, *abstract* and *keywords*. The standard search string had to be gauged for each database, to cope with different word stemming and syntax requirements.

The search was performed from 1st to 15th October, 2012. Search results identified 2204 papers, from which 1099 were duplicates (many studies were indexed by more than one digital library), leading to 1105 papers selected for screening.

Screening of papers and Snow-balling. The main goal of the screening process is to select relevant studies that properly address the research questions. This means that inclusion and exclusion criteria must be carefully chosen and applied to the data retrieved. In our screening process, four inclusion criteria (IC) and 20 exclusion criteria (EC) were used, among them:

IC 1. Studies that address the use of OSP to learn/teach Software Engineering should be included, regardless of their application in either SE programs or in other programs;

EC 1. Documents written in languages other than English should be excluded (the universality of this language supports reproducibility);

EC 2. Documents whose full text is not available should be excluded;

EC 3. Studies whose main content is not related to learning or teaching Software Engineering should be excluded.

The whole set of inclusion and exclusion criteria can be found in our mapping study website[7].

After we searched the papers and eliminated the duplicates, we applied two filters. In the first, two of the authors screened the remaining papers. Each reviewer examined title and abstract of each paper, and marked it as included or excluded, using the previous criteria. Results were subsequently compared. A third reviewer analyzed conflicts and took the final decision. In this first step, 156 papers were included. In the second filter, two reviewers reexamined the remaining studies, skimming through introduction and conclusion. A third reviewer compared their results, and again, took the final decision, resulting in 53 studies. These were the relevant papers considered for classification in our preliminary study (Nascimento et al., 2013). However, given the limitations of our search string and of each digital library, and aiming at a more comprehensive view of our area of interest, we followed a "snow-balling" procedure (Budgen et al., 2008) to gather additional references. To do so, we followed up references cited in the previously selected studies and identified 11 more relevant papers. In addition, during our analysis, we carefully re-read each selected paper, which led us to accept eight previously rejected studies in order to be more consistent, and to gather a comprehensive view of the area. Therefore, we also included papers that report cases of other computing courses (not a software engineering course), since these papers report cases of teaching software development skills that are foundations to SE practices. Finally, we selected a total of 72 studies to be considered for classification.

Keywording Relevant Topics. We developed a classification scheme based on the research questions RQ1, RQ2 and RQ3, and also on the relevant topics identified during paper reading. This step was based on the keywording process described by Petersen et al. (2008). Table 2 presents the facets defined for classification purposes.

Facets 1-5 were presented in a conference paper (Nascimento et al., 2013), whereas facets 6-9 have not been previously published. Open source software has

[7] https://sites.google.com/site/dmcnascimento/mapping

Table 2. Defined Facets

	Facet	Description
1	Software Engineering Area	The SE topic(s) addressed in the study (software engineering in general, requirements, models and methods, design and architecture, quality, testing, evolution and maintenance, development and construction, process, management and configuration management), based on the SWEBOK (2013).
2	Research Type	The research approach used in the paper; Table 3 provides a description of each category, adapted from Petersen et al. (2008).
3	Learning Approach	The pedagogical approach that was applied together with OSP in SE courses. Table 4 presents a description of each category.
4	Assessment Perspective	The perspective from which student learning is evaluated – see Table 5.
5	Assessment Type	The instrument used to assess student learning – see Table 6.
6	Approach Goal	Which goal in introducing OSP in SE courses was prevalent (either learning SE principles/concepts or learning to develop OSS).
7	Curriculum Choice	How the content was embodied in curriculum – see Table 7.
8	Control Level	How much control faculty/staff had on the project – see Table 8.
9	Project Choice	How the projects were chosen – see Table 9.

Table 3. Research Types (adapted from Petersen et al. (2008)) - Facet 2

Category	Description
Experience report	Paper describes the authors' personal experience using a particular approach, explaining what and how something has been done in practice. It usually includes lessons learned.
Case study	An empirical inquiry that investigates a phenomenon within its real-life context. Paper may deal with single or multiple cases, may include quantitative evidence, relies on multiple sources of evidence, and benefits from the prior development of theoretical propositions.
Action research	Interactive inquiry process that balances problem solving actions implemented in a collaborative context with data-driven collaborative analysis or research to understand underlying causes enabling future predictions about personal and organizational change.
Experiment/Quasi-experiment	A collection of research designs that use manipulation and controlled testing to understand causal processes. Generally, one or more variables are manipulated to determine their effect on a dependent variable.
Survey	Encompasses any measurement procedures that involve asking questions of respondents, generally by sampling them from a population. Most common types of surveys use questionnaires or interviews.
Opinion paper	These papers express the personal opinion of somebody whether a certain approach is good or bad, or how things should been done. They do not rely on related work and research methodologies.
Solution proposal	A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution are shown by an example or a good line of argumentation.
Philosophical paper	These papers sketch a new way of looking at existing things by structuring the field in form of a taxonomy or conceptual framework.
Report	Document describes a project. Can be a technical or a progress report.
Literature review	Critical and in-depth evaluation of previous research. It is a summary and synopsis of a particular area of research.

gained importance in the software industry, not only for their high quality, but also due to the use of new development processes and practices. As a consequence, some universities decided to include OSP in computing courses. Since this topic can be seen as a software engineering subarea *per se*, we included studies with this goal in our mapping. To clarify matters, the *Facet 6 - Approach Goal* was included in the classification scheme to characterize the course approach with respect to its goals. Finally, the other three facets elaborate on some aspects related to fitting OSP into students' activities.

The item "does not apply" was added to some facets anywhere the proposed

Table 4. Learning Approaches - Facet 3

Category	Description
Active learning (general)	General term that refers to several models of education that focus the responsibility of learning on learners. Usually students engage in higher-order thinking tasks such as analysis, synthesis, and evaluation.
Case-based learning	Approach where students develop skills in analytical thinking and reflective judgment by reading and discussing complex, real-life scenarios. A case is already organized and synthesized for students.
Game-based learning	Learning that involves students in some sort of competition or achievement in relationship to an educational goal. Attempts to increase student motivation by providing a playful environment.
Peer/Group/Team learning	Educational practices in which students interact with other students to attain educational goals. Such approaches enhance the value of interaction and information sharing among peers.
Project-/Problem-/Inquiry-based learning	A collection of approaches that use projects or problems to drive the learning process. Students learn about a subject through the experience of problem solving, by working in groups with the help of facilitators. Assessment is performance-based and authentic.
Studio-based learning	Approach from professional education, where students undertake a project under the supervision of a master designer. It uses a learning cycle of construction, presentation, critique and response, that is repeated until project completion.
Project method	The traditional method where students participate in a project development.
Other	Other approaches different from the previous categories.
Not specified	Authors do not state the learning approach used.
Does not apply	Paper is not related to an experience where a learning approach is needed.

Table 5. Assessment Perspective - Facet 4

Category	Description
Student perspective	Students assess their learning by either self- or peer evaluation.
Teacher perspective	Students are assessed by faculty or teaching assistants.
Product perspective	Specific criteria are defined to assess students' products.
Not specified	No assessment is mentioned in the paper.
Does not apply	Work is not related to an experience where assessment is necessary.

Table 6. Assessment Type - Facet 5

Category	Description
Exams	Students are assessed by means of written exams.
Reports	Students should write a report for assessment.
Software artifacts	Students are assessed through developed software artifacts.
Interviews	Interviews are conducted to assess learning.
Seminars	Students are assessed by their performance in seminars.
Exercises	Students are assessed by means of exercises.
Surveys	A survey is conducted to assess learning.
Reflective essay	A reflective essay is written.
Presentations	Students are assessed by presenting their performed work.
Participation	Student's participation in class, in community, interaction with group or team.
None	No assessment instrument is mentioned in the paper.
Does not apply	Work is not related to an experience where assessment is needed.

Table 7. Curriculum Choice - Facet 7

Category	Description
Extra activity	Students worked with OSP in extra activities, e.g., internships.
Capstone Project	Students worked with OSP in a capstone project.
Course	Working with OSP was a student's assignment in a regular course.
Not specified	The paper does not mention where, in the curriculum, the approach is used.
Does not apply	Related to an experience where curriculum issues are not declared.

categorization was not appropriate.

Data Extraction. The data extraction strategy was designed to gather the required information to address the objectives of this study. Each paper accepted in the screening process was fully read to collect the required data. We extracted information on title, author(s), author's affiliation details, venue and year of pub-

Table 8. Control Level - Facet 8

Category	Description
No control	Faculty/staff only monitor student's activities inside the project. Students work with community requests and the community approves students' contribution.
Inside initiative / External approval	A new feature is proposed and built inside college, but later, it is submitted to community approval.
Inside control	Faculty/staff branch the OSP code, prepare assignments, and evaluate themselves the students' contribution.
Full control	Project core development has been sustained by faculty/staff.
Not specified	No control level could be identified in the paper.
Does not apply	Work is not related to an experience where control level is needed.

Table 9. Project Choice - Facet 9

Category	Description
Predefined	Faculty/staff decide the project students work.
Choice list	Students can choose any project from the list provided by faculty/staff.
Free choice	Students should seek and decide which OSP (from their interest) they will work with.
Not specified	Paper does not mention issues related to project choice.
Does not apply	Work is not related to an experience where choosing a project is needed.

lication for each selected paper, as well as the information required to classify it according to each facet.

We classified some studies in more than one category for some facets (e.g., *Facet 1 - Software Engineering Area*, *Facet 4 - Assessment Perspective*, *Facet 5 - Assessment Type* and *Facet 7 - Curriculum Choice*).

During data extraction, we also looked at how studies related to each other, to identify the continuity of research projects.

To assist us in managing the data generated in the whole process, we used StArt (State of the Art through Systematic Review)[8], Mendeley reference manager[9], and spreadsheets.

3. Mapping

In this section, we present data extraction results. However, before presenting the outcomes, structured by facets and research questions, firstly we provide an overview of the studies, with a brief description of their objectives and contributions (Section 3.1). One of the main contributions of mapping studies is the *map*, usually a bubble plot representation of different facets or perspectives. In this study, we classified the articles for each facet and produced three maps to summarize the results. These maps are illustrated in Sections 3.2, 3.3 and 3.4. Section 3.5 presents the distribution of publications with respect to the main venues and the most active research communities, and a temporal view of the topic. Finally, Section 3.6 reports some long-term projects.

3.1. *Overview of Study Objectives and Contributions*

The objectives of studies of OSP in SE education may vary. Some intend to enhance student motivation (Auer, Juntunen, & Ojala, 2011; Costa-Soria & Pérez, 2009; Ellis, Hislop, Rodriguez, & Morelli, 2012; Nandigam, Gudivada, & Hamou-Lhadj,

[8]http://lapes.dc.ufscar.br/tools/start_tool
[9]http://www.mendeley.com

2008; Sowe, Stamelos, & Deligiannis, 2006), others aim at attracting more students (Budd, 2009; Morelli et al., 2009; Raj & Kazemian, 2006; Xing, 2010), hence, improving enrollments in computer science courses (Kussmaul, 2009; Seiter, 2009; Tucker, Morelli, & de Lanerolle, 2011). In addition, there are papers whose aim is supplying students with an experience with OSS, i.e., studying concepts, licensing, communities, roles, and development process, motivated by the recent growth in interest in this area (Beaufait, Chen, Dietrich Jr., Dietrich, & Vanhoy, 2011; Budd, 2009; Horstmann, 2009; Lundell, Persson, & Lings, 2007; Robles, Caballe, & González-Barahona, 2008). Recognizing the wide acceptance of OSS, Yamakami (2012) discusses the impacts and implications of OSS advances from the viewpoint of education, and concludes by proposing a paradigm transition. According to the author, software engineering education needs to be reengineered due the changes brought by OSS.

Some studies discuss how OSS can be embedded in computer science courses (Li, Zhang, & Li, 2009; O'Hara & Kay, 2003; Smrithi Rekha, Adinarayanan, Maherchandani, & Aswani, 2009). O'Hara and Kay (2003) not only advocate the use of OSS in lab infrastructure, due its low-cost, but also assert that students should develop OSS so that they participate in large distributed software communities and interact with large software code bases. Li et al. (2009) highlight topics such as open source code, open course resource and open discussion, and then provide some advices on curriculum setup, teaching strategies, and resource construction. Finally, Smrithi Rekha et al. (2009) explore the use of OSS to bridge the gap noticed by software industry between what ACM Computer Science curriculum proposes and the actual skills students presently acquire.

We also found articles that encourage the use of OSS on computing education by listing a set of potential benefits (Whitehurst, 2009), or even highlighting the importance of participating in OSP for improving developers' background (Spinellis, 2006). Specifically in the software engineering area, Kamthan (2007) examines fundamental SE practices from an OSS perspective, discusses ways to use OSS, and provides some guidelines to embody OSS in SE education.

Various studies describe one particular approach. Some specify a set of criteria to help students select one project to work with, others propose specific assignments or activities that students should perform, while a few even describe the assessment method used in the course. We describe some illustrating examples. Nandigam et al. (2008) teach basic software engineering principles by focusing on practical issues. For instance, students explore a code base of an OSP, reverse engineer the source code to get an abstract view of the project, assess design quality based on software metrics, perform some refactoring, and analyze the impact of some proposed changes. Kussmaul (2009), based on instructional design, proposes a five-step USABL (Use-Study-Add-Build-Leverage) model. First, students *use* the features of one chosen OSS; second, they *study* the code of the project; then, students *add* simple enhancements; later, they *build* something more complex, and, lastly, they *leverage* OSS for other intentions. Papadopoulos, Stamelos, and Meiszner (2012) propose that students assume a particular role of their preference (e.g., tester, developer, requirements engineer) and perform a set of assigned tasks, according to the chosen role. Petrenko, Poshyvanyk, Rajlich, and Buchta (2007) teach software evolution by requesting teams to implement new features in OSS using a software change process model. The authors divide the course into three phases, gradually increasing change complexity and communication needs between students. Carrington (2003) also suggests a gradual increase in difficulty, structuring the practical

work as a sequence of four assignments that require an increasing depth of knowledge and understanding of the selected open source project. Finally, Liu (2005) proposes the use of GROw (Gradually Ripen Open-source Software), an educational OSS team development process comprising: i) a sequence of milestones with established artifacts for each deliverable, ii) test adequacy and evaluation criteria, and iii) documentation and coding style guidelines. However, this study does not describe the process in enough detail. The following subsection describes more examples of approaches.

Some studies focus on the challenges of adopting OSP. Ellis, Morelli, and Hislop (2008) identify challenges such as: student inexperience, limited course duration, informal and uncontrolled development practices, sustaining a development effort, and product complexity. Horstmann (2009) discusses challenges of student inexperience mainly related to technology (version control tools, build automation tools and operating systems other than Windows). In an overview, Meiszner, Moustaka, and Stamelos (2009) discuss the issues of traditional "closed" and "semester based" structures of educational systems.

One of the main challenges of using OSS in SE education is the difficulty to identify ideal or appropriate projects to work with (Toth, 2006). This happens not only because of the large number of existing projects (in a wide range of size, complexity, domains and communities (Ellis, Purcell, & Hislop, 2012)), but also due to the students' diverse background and interest. Selected OSS should be neither too large and complex to overwhelm students, nor too small and simple to be so trivial and not favor the applicability of SE principles (Gokhale, Smith, & McCartney, 2012; Kon et al., 2011).

Several studies address how to identify or choose adequate projects by providing comments, directions, or even a list of established criteria (Buchta, Petrenko, Poshyvanyk, & Rajlich, 2006; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Gehringer, 2011; German, 2005; Hepting, Peng, Maciag, Gerhard, & Maguire, 2008; Jaccheri & Østerlie, 2007; Kamthan, 2007; Meneely, Williams, & Gehringer, 2008; Papadopoulos et al., 2012; Sowe & Stamelos, 2007; Sowe et al., 2006; Stroulia, Bauer, Craig, Reid, & Wilson, 2011; Toth, 2006). However, two studies are devoted only to the process of selecting OS projects. The study reported by Gokhale et al. (2012) is a work-in-progress that intends to develop a systematic methodology based on metrics to facilitate the selection. They train a predictive regression model and a classification model by using metrics collected from manually selected, previously used projects. Ellis, Purcell, and Hislop (2012) provide a framework that guides instructors to select OSP for student involvement. The framework sets criteria related to viability, suitability and approachability of the project, using two levels: "Mission Critical", i.e., essential features that must be present, and "Secondary Criteria" or non-mandatory, i.e., the project can fail to meet one or two features.

On the other hand, Meneely et al. (2008), although presenting a list of criteria to select projects, conclude that it is not viable to find a single project that meets all their needs. Therefore, they created a repository named ROSE (Repository for Open Software Education) to share OSP built for educational purposes. Projects should be manageable in size and scope, and adhere to defined pedagogical criteria.

Another example of environment created to facilitate the use of OSP in education is the PicoLibre platform. Cousin, Ouvradou, Pucci, and Tardieu (2002) describe this free pedagogical collaborative platform that allows students to create their own OSP and promotes reusability of previous projects. The platform hosts other

projects that provide services such as code and documentation repository, mailing lists and bug tracker.

Nachbar (1998), in 1998, had already raised the need to generate a new organization to facilitate student contributions to a freely available and useful software, providing not only available code but also continuous maintenance and technical support. Students could have relevant experiences with very large software projects, demanding not so much effort from instructors. Instructors could share their own created exercises based on available software. The author named the collection of available software that this entity should maintain as "Public Software".

Following the idea of "learning by example", Zhang and Su (2007) propose BRIDGE, a web-based collaborative education system. Their goal is to enable a connection between theory and real-world examples by using a collaborative educational resource creation system and open source code as examples. In a similar thread, Kume, Nitta, and Takemura (2006) propose a method for editing teaching materials from OSP. Through this method, teachers find real examples of software design decisions and extract them to use on their practical training scheme.

In a different direction, Shockey and Cabrera (2005) report the outcomes of the SNAP Development Center, an OSS development project within an university, resulting from the collaboration between academia, industry and government. According to the authors, the project represents a feasible combination that promotes students' expertise in topics such as process, tools and code reuse.

Finally, Megias, Tebbens, Bijlsma, and Santanach (2009) describe the Free Technology Academy (FTA), a distance learning program created by an European consortium. This project aims to provide knowledge, skills and competences on free technologies (free software and open standards) to students, including the possibility of their getting a master's degree in this subject at one of the participating universities.

3.2. *Open Source Projects in Software Engineering Education*

The main purpose of our mapping study was to find out how open source projects have been used in software engineering courses (RQ1). In addition, we were also interested in the learning approaches used (RQ2). Hence, we decided to arrange the first map as a combination of the *Facet 1 - Software Engineering Area* with the *Facet 3 - Learning Approach* and *Facet 2 - Research Type* (see Figure 2).

Figure 2 shows some trends: works that deal with *software engineering in general* combined either with the traditional *project method* (16 studies) or with a *solution proposal* (26 studies).

Let us now move to an analysis of each facet. Beginning with the *Facet 1 - Software Engineering Area*, we noticed that 69.4% of the papers address *Software Engineering in general*. Less than half of the studies (22 papers) focus on specific software engineering areas. *Design/Architecture* leads the count with eight papers, followed by *Evolution/Maintenance* and *Development/Construction*, both with six papers (Table 10). Some studies address more than one area; therefore they were categorized twice in this facet, resulting in different totals in the map (Boldyreff, Capiluppi, Knowles, & Munro, 2009; Carrington, 2003; Costa-Soria & Pérez, 2009; Qian & Fu, 2008; Williams & Shin, 2006). No papers were found that relate OSP with specific knowledge areas such as *Requirements*, *Models and Methods* or *Configuration Management*. The only instance of *Management* we found is the paper by Conlon and Hulick (2005), which discusses the feasibility of either building

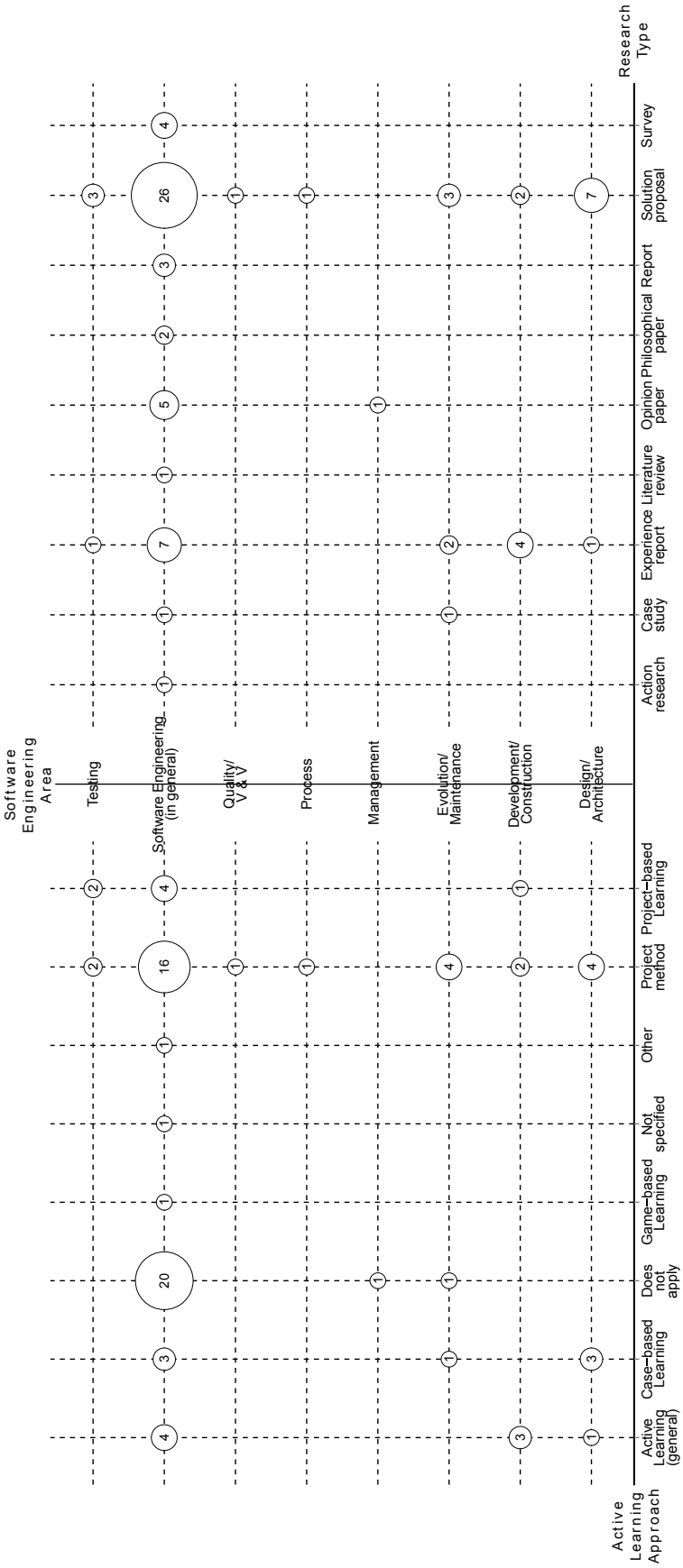

Figure 2. Learning Approach *versus* Software Engineering Area *versus* Research Type

Table 10. Studies classified by *Facet 1 - Software Engineering Area*

Area	**Studies**	#
Software Engineering (in general)	(Auer et al., 2011; Budd, 2009; Chen et al., 2008; Cousin et al., 2002; de Lanerolle, Morelli, Danner, & Krizanc, 2008; Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Ellis & Morelli, 2008; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Ellis et al., 2008; Ellis, Purcell, & Hislop, 2012; Gehringer, 2011; German, 2005; Hislop, Ellis, & Morelli, 2009; Horstmann, 2009; Jaccheri & Østerlie, 2007; Kamthan, 2007; Kilamo, 2010; Kon et al., 2011; Krogstie, 2008; Kussmaul, 2009; Li et al., 2009; Liu, 2005; Lundell et al., 2007; Marmorstein, 2011; Martínez, 2009; Megias et al., 2009; Meiszner et al., 2009; Meneely et al., 2008; Morelli & de Lanerolle, 2009; Morelli, Ellis, de Lanerolle, Damon, & Walti, 2007; Morelli et al., 2009; Nachbar, 1998; Nandigam et al., 2008; O'Hara & Kay, 2003; Papadopoulos et al., 2012; Robles et al., 2008; Santore, Lorenzen, Creed, Murphy, & Orcutt, 2010; Shockey & Cabrera, 2005; Spinellis, 2006; Stroulia et al., 2011; Toth, 2006; Tucker et al., 2011; Whitehurst, 2009; Xing, 2010; Yamakami, 2012; Zhang & Su, 2007)	50
Management	(Conlon & Hulick, 2005)	1
Design/ Architecture	(Boldyreff et al., 2009; Carrington, 2003; Costa-Soria & Pérez, 2009; Hepting et al., 2008; Kume et al., 2006; Qian & Fu, 2008; Seiter, 2009; Tao & Nandigam, 2006)	8
Quality/ V & V	(Williams & Shin, 2006)	1
Testing	(Carrington, 2003; Sowe & Stamelos, 2007; Sowe et al., 2006; Williams & Shin, 2006)	4
Evolution/ Maintenance	(Buchta et al., 2006; Costa-Soria & Pérez, 2009; Gokhale et al., 2012; Lutfiyya & Andrews, 2000; McCartney, Gokhale, & Smith, 2012; Petrenko et al., 2007)	6
Process	(Qian & Fu, 2008)	1
Development/ Construction	(Allen, Cartwright, & Reis, 2003; Beaufait et al., 2011; Boldyreff et al., 2009; Raj & Kazemian, 2006; Sabin, 2011; Smrithi Rekha et al., 2009)	6

or buying software, contrasting it with the possibility of "downloading" software, encouraging students to analyze such questions.

To assess the scientific contribution of the selected studies, it is important to identify which research methods each paper uses (Table 11). The *Facet 2 - Research Type* clarifies this issue, as we can see in Figure 2. Thirty nine papers fit into the *solution proposal* category, standing for 54.2% of the selected papers. *Experience reports* account for the second most popular category, with 14 studies. Very few studies pay explicit attention to research methodology. Jaccheri and Østerlie (2007) apply the *action research* method , while Krogstie (2008) and McCartney et al. (2012) use the *case study* method. In the first paper, each student assumes both the roles of developer and researcher in a chosen OSP (Jaccheri & Østerlie, 2007). In this report, the student tries to answer his or her own research questions by participating in the project. In the second paper, the author collected, coded and analyzed data from documents, field notes, interviews, across all teams in the course, reporting results in a chronological structure (Krogstie, 2008). In the last work, besides the results, the author describes the qualitative method applied to analyze students' learning of code maintenance and evolution, based on pre- and post-course surveys (McCartney et al., 2012). We did not find studies designed as or evaluated with an *Experiment or Quasi-experiment*. Kon et al. (2011) present a careful *literature review* of the opportunities brought by OSS to software engineering research and education in Brazil. Their scope was papers published in the Brazilian Symposium on Software Engineering (SBES), the main Brazilian venue for software engineering researchers.

Some studies were hard to classify within the *Facet 2 - Research Type*, because

Table 11. Studies classified by *Facet 2 - Research Type*

Category	**Studies**	**#**
Experience report	(Beaufait et al., 2011; Boldyreff et al., 2009; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; German, 2005; Lutfiyya & Andrews, 2000; Martínez, 2009; Morelli et al., 2009; Petrenko et al., 2007; Robles et al., 2008; Sabin, 2011; Smrithi Rekha et al., 2009; Sowe & Stamelos, 2007; Stroulia et al., 2011; Toth, 2006)	14
Case study	(Krogstie, 2008; McCartney et al., 2012)	2
Action research	(Jaccheri & Østerlie, 2007)	1
Survey	(Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Gehringer, 2011; Hislop et al., 2009)	4
Opinion paper	(Conlon & Hulick, 2005; Ellis et al., 2008; Li et al., 2009; O'Hara & Kay, 2003; Spinellis, 2006; Whitehurst, 2009)	6
Solution proposal	(Allen et al., 2003; Auer et al., 2011; Buchta et al., 2006; Budd, 2009; Carrington, 2003; Chen et al., 2008; Costa-Soria & Pérez, 2009; Cousin et al., 2002; Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Ellis, Purcell, & Hislop, 2012; Gokhale et al., 2012; Hepting et al., 2008; Horstmann, 2009; Kilamo, 2010; Kume et al., 2006; Kussmaul, 2009; Liu, 2005; Lundell et al., 2007; Marmorstein, 2011; Megias et al., 2009; Meiszner et al., 2009; Meneely et al., 2008; Morelli & de Lanerolle, 2009; Morelli et al., 2007; Nachbar, 1998; Nandigam et al., 2008; Papadopoulos et al., 2012; Qian & Fu, 2008; Raj & Kazemian, 2006; Santore et al., 2010; Seiter, 2009; Shockey & Cabrera, 2005; Sowe et al., 2006; Tao & Nandigam, 2006; Williams & Shin, 2006; Xing, 2010; Zhang & Su, 2007)	39
Philosophical paper	(Kamthan, 2007; Yamakami, 2012)	2
Report	(de Lanerolle et al., 2008; Ellis & Morelli, 2008; Tucker et al., 2011)	3
Literature review	(Kon et al., 2011)	1

their goals were multiple. For instance, Toth (2006) presents a solution proposal but also discusses learned lessons and approach changes along four iterations. Likewise, Gehringer (2011) states he presents results of a *Survey*, but the paper is more likely to be categorized as an *Opinion Paper*. Even though the author surveyed managers of OSS projects that had previously interacted with classes and students, the paper shows very few details about the survey. Most of the paper is devoted to recommendations for instructors wishing to embed OSS into a computing course.

Concerning the *Facet 3 - Learning Approach*, Figure 2 shows that most selected studies (37.5%) applied the traditional *Project Method*. Some papers explicitly mention other approaches, such *Active Learning* or *Project-Based Learning*, but none of them give detailed explanations on how these approaches are applied. Table 12 shows the studies classified in each category. No instances of *Peer/Group/Team learning* or *Studio-based learning* were found.

In studies classified as *active learning* (Beaufait et al., 2011; Boldyreff et al., 2009; Chen et al., 2008; Jaccheri & Østerlie, 2007), students drive their learning in some way by getting involved in research activities. We found only one initiative where students learn by playing a game (Kilamo, 2010). This paper reports students learning how the open source community works via exercises in the game, which mimic actual development practices in an open source project. *Service learning* is the basis of Liu's pedagogical method, which promotes student learning by engaging students in activities that address human or community needs (Liu, 2005).

Figure 2 presents the six examples of *case-based learning* found in our study. Three of them address the apprenticeship of software design/architecture (Costa-Soria & Pérez, 2009; Kume et al., 2006; Tao & Nandigam, 2006), while the other three focus on general principles of software engineering (Nandigam et al.,

Table 12. Studies classified by *Facet 3 - Learning Approach*

Category	Studies	#
Active learning (general)	(Beaufait et al., 2011; Boldyreff et al., 2009; Chen et al., 2008; Jaccheri & Østerlie, 2007; Li et al., 2009; Meiszner et al., 2009; Raj & Kazemian, 2006)	7
Case-based learning	(Costa-Soria & Pérez, 2009; Kume et al., 2006; Nandigam et al., 2008; Tao & Nandigam, 2006; Toth, 2006; Zhang & Su, 2007)	6
Game-based learning	(Kilamo, 2010)	1
Project-/Problem-/Inquiry-based learning	(Auer et al., 2011; Kussmaul, 2009; Papadopoulos et al., 2012; Sabin, 2011; Sowe & Stamelos, 2007; Sowe et al., 2006; Stroulia et al., 2011)	7
Project method	(Allen et al., 2003; Buchta et al., 2006; Budd, 2009; Carrington, 2003; Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; German, 2005; Hepting et al., 2008; Hislop et al., 2009; Horstmann, 2009; Krogstie, 2008; Lundell et al., 2007; Lutfiyya & Andrews, 2000; Marmorstein, 2011; Martínez, 2009; McCartney et al., 2012; Morelli & de Lanerolle, 2009; Petrenko et al., 2007; Qian & Fu, 2008; Robles et al., 2008; Santore et al., 2010; Seiter, 2009; Smrithi Rekha et al., 2009; Williams & Shin, 2006; Xing, 2010)	27
Other	(Liu, 2005)	1
Not specified	(Shockey & Cabrera, 2005)	1
Does not apply	(Conlon & Hulick, 2005; Cousin et al., 2002; de Lanerolle et al., 2008; Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis & Morelli, 2008; Ellis et al., 2008; Ellis, Purcell, & Hislop, 2012; Gehringer, 2011; Gokhale et al., 2012; Kamthan, 2007; Kon et al., 2011; Megias et al., 2009; Meneely et al., 2008; Morelli et al., 2007, 2009; Nachbar, 1998; O'Hara & Kay, 2003; Spinellis, 2006; Tucker et al., 2011; Whitehurst, 2009; Yamakami, 2012)	22

2008; Toth, 2006; Zhang & Su, 2007). It is worth noting that Costa-Soria and Pérez (2009) also address the specific area of software maintenance.

We also distinguished studies whose goal is to learn SE concepts and principles by using OSP from studies that aim to learn about open source software. We identified 40 studies that addressed only the former goal, 18 that addressed only the latter, while 14 addressed both goals (*Facet 6 - Approach Goal*). Table 13 shows the articles in each category. This shows that regardless the relevance of learning the OSS process, most authors have sought to improve learning in software engineering using OSP.

3.3. *Fitting OSP into Students' Activities*

The map in Figure 3 shows how open source projects have been integrated into students' activities. The goal was to summarize the methods reported by the studies. Thus, we produced the second map as a combination of the *Facet 8 - Control Level* with the *Facet 7 - Curriculum Choice* and *Facet 9 - Project Choice*.

Table 14 shows the results for the *Facet 7 - Curriculum Choice*. Most studies embodied OSP into the course syllabus (38 articles). In a different approach, 10 studies used OSP in extra activities, while five studies used OSP in capstone projects. It is worth noting that six papers described the use of more than one approach: Smrithi Rekha et al. (2009) incorporated OSP in extra activities, capstone project and a regular course; Toth (2006) and Stroulia et al. (2011) used OSP in extra activities and capstone projects; Ellis, Hislop, et al. (2012) and Hislop et al. (2009) included OSP in extra activities and courses; and, Kussmaul (2009) embedded OSP in capstone projects and courses.

The information available in the papers led us to conclude that most courses with embedded open source projects were compulsory. We detected only three papers

Table 13. Studies classified by *Facet 6 - Approach Goal*

Category	Studies	#
Learning SE	(Allen et al., 2003; Boldyreff et al., 2009; Buchta et al., 2006; Costa-Soria & Pérez, 2009; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Ellis & Morelli, 2008; Ellis, Purcell, & Hislop, 2012; Gokhale et al., 2012; Hepting et al., 2008; Hislop et al., 2009; Kon et al., 2011; Kume et al., 2006; Kussmaul, 2009; Li et al., 2009; Liu, 2005; Lutfiyya & Andrews, 2000; Marmorstein, 2011; McCartney et al., 2012; Meiszner et al., 2009; Nachbar, 1998; Nandigam et al., 2008; O'Hara & Kay, 2003; Papadopoulos et al., 2012; Petrenko et al., 2007; Qian & Fu, 2008; Raj & Kazemian, 2006; Santore et al., 2010; Shockey & Cabrera, 2005; Smrithi Rekha et al., 2009; Sowe & Stamelos, 2007; Sowe et al., 2006; Spinellis, 2006; Stroulia et al., 2011; Tao & Nandigam, 2006; Toth, 2006; Williams & Shin, 2006; Xing, 2010; Zhang & Su, 2007)	40
Learning OSS	(Beaufait et al., 2011; Budd, 2009; Conlon & Hulick, 2005; de Lanerolle et al., 2008; Ellis & Hislop, 2011; German, 2005; Horstmann, 2009; Jaccheri & Østerlie, 2007; Kilamo, 2010; Krogstie, 2008; Lundell et al., 2007; Martínez, 2009; Megias et al., 2009; Morelli & de Lanerolle, 2009; Morelli et al., 2007; Robles et al., 2008; Seiter, 2009; Tucker et al., 2011)	18
Both	(Auer et al., 2011; Carrington, 2003; Chen et al., 2008; Cousin et al., 2002; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Ellis et al., 2008; Gehringer, 2011; Kamthan, 2007; Meneely et al., 2008; Morelli et al., 2009; Sabin, 2011; Whitehurst, 2009; Yamakami, 2012)	14

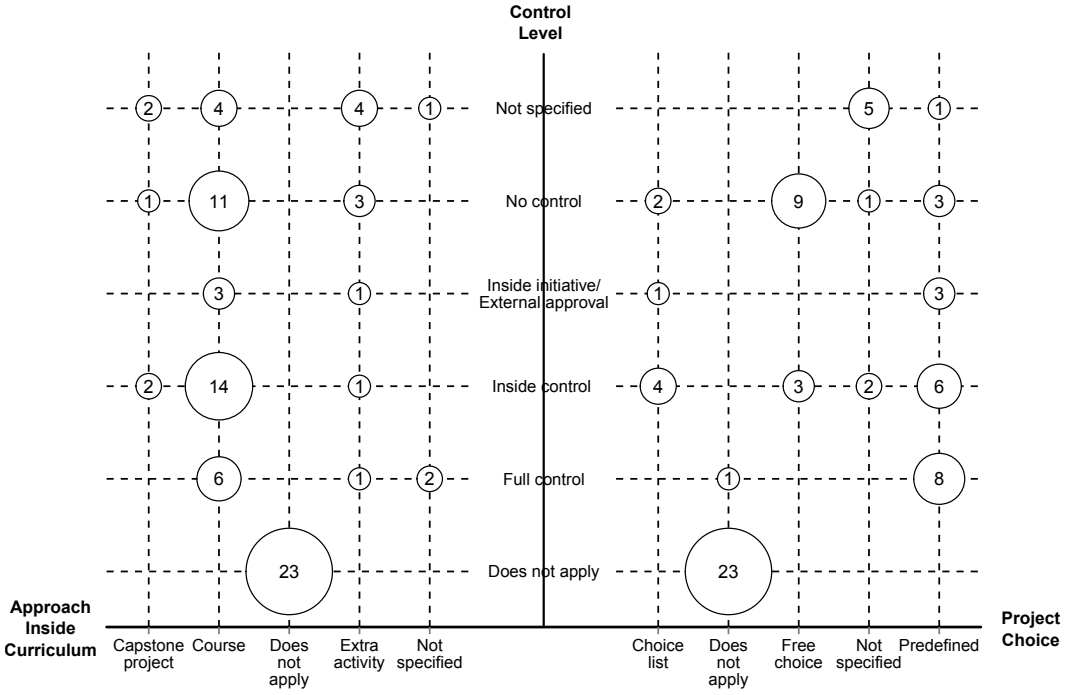

Figure 3. Curriculum Choice *versus* Control Level *versus* Project Choice

to present experiences in elective courses (Horstmann, 2009; Qian & Fu, 2008; Seiter, 2009). Qian and Fu (2008) cover component-based software development using a large number of open source components, Horstmann (2009) and Seiter (2009) address the open source software development process. However, Horstmann describes experiences with both undergraduate and graduate students. We also identified four papers that report experiences only in Masters' courses: one case in the University of Victoria, Canada (German, 2005), another in the University of Skövde, Sweden (Lundell et al., 2007), and the other two in the Universitat

Table 14. Studies classified by *Facet 7 - Curriculum Choice*

Category	**Studies**	#
Extra activity	(Beaufait et al., 2011; Boldyreff et al., 2009; Chen et al., 2008; Ellis, Hislop, et al., 2012; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Hislop et al., 2009; Jaccheri & Østerlie, 2007; Smrithi Rekha et al., 2009; Stroulia et al., 2011; Toth, 2006)	10
Capstone Project	(Kussmaul, 2009; Martínez, 2009; Smrithi Rekha et al., 2009; Stroulia et al., 2011; Toth, 2006)	5
Course	(Allen et al., 2003; Buchta et al., 2006; Budd, 2009; Carrington, 2003; Costa-Soria & Pérez, 2009; Ellis, Hislop, et al., 2012; Ellis, Morelli, de Lanerolle, & Hislop, 2007; German, 2005; Hepting et al., 2008; Hislop et al., 2009; Horstmann, 2009; Kilamo, 2010; Krogstie, 2008; Kume et al., 2006; Kussmaul, 2009; Liu, 2005; Lundell et al., 2007; Lutfiyya & Andrews, 2000; Marmorstein, 2011; McCartney et al., 2012; Meiszner et al., 2009; Morelli & de Lanerolle, 2009; Nachbar, 1998; Nandigam et al., 2008; Papadopoulos et al., 2012; Petrenko et al., 2007; Qian & Fu, 2008; Raj & Kazemian, 2006; Robles et al., 2008; Sabin, 2011; Santore et al., 2010; Seiter, 2009; Smrithi Rekha et al., 2009; Sowe & Stamelos, 2007; Sowe et al., 2006; Tao & Nandigam, 2006; Williams & Shin, 2006; Xing, 2010)	38
Not specified	(Auer et al., 2011; Ellis, Hislop, & Morelli, 2011; Shockey & Cabrera, 2005)	3
Does not apply	(Conlon & Hulick, 2005; Cousin et al., 2002; de Lanerolle et al., 2008; Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis & Morelli, 2008; Ellis et al., 2008; Ellis, Purcell, & Hislop, 2012; Gehringer, 2011; Gokhale et al., 2012; Kamthan, 2007; Kon et al., 2011; Li et al., 2009; Megias et al., 2009; Meneely et al., 2008; Morelli et al., 2007, 2009; O'Hara & Kay, 2003; Spinellis, 2006; Tucker et al., 2011; Whitehurst, 2009; Yamakami, 2012; Zhang & Su, 2007)	23

Oberta de Catalunya, Spain (Martínez, 2009; Robles et al., 2008). All of them cover content about OSS development process and communities. Finally, we found approaches based on extra activities, where OSP were applied with a research perspective, blending research, learning and development (Beaufait et al., 2011; Boldyreff et al., 2009; Chen et al., 2008; Jaccheri & Østerlie, 2007).

With the *Facet 8 - Control Level*, we classified studies in an increasing level of control, according to how much faculty/staff controlled students' activities (see Table 15). Results showed 15 studies with *no control*. In these studies, assignments should contribute to the OSP based on community requests, and contributions should be approved by these communities. In this case, instructors only monitor students' activities. As an example, Marmorstein (2011) proposes a SE course where students learn about software development processes and project management practices by participating in one open source community. In this study, after deciding which community to contribute with, students submit a short proposal for instructor's approval. This document describes the chosen community and contributions they intend to make. This document is used to ensure variety and exposure to tasks with the appropriate level of difficulty. To encourage students to manipulate source code as early as possible, and to initiate communication with developers, students have to submit a resource report summarizing needed development resources and available project documentation. Afterwards, students implement their contributions. Students are assessed not only by the contributions they make (e.g., new feature, bug fixing, bug report, documentation), but also if those are accepted by project developers and become part of the software.

We uncovered four papers (Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Morelli & de Lanerolle, 2009; Raj & Kazemian, 2006) where faculty work together with students to propose features to be built, which, once ready, are submitted to community approval (*inside ini-*

Table 15. Studies classified by *Facet 8 - Control Level*

Category	Studies	#
No control	(Beaufait et al., 2011; Boldyreff et al., 2009; Budd, 2009; German, 2005; Horstmann, 2009; Jaccheri & Østerlie, 2007; Krogstie, 2008; Lundell et al., 2007; Lutfiyya & Andrews, 2000; Marmorstein, 2011; Martínez, 2009; Meiszner et al., 2009; Papadopoulos et al., 2012; Sowe & Stamelos, 2007; Sowe et al., 2006)	15
Inside initiative / External approval	(Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Morelli & de Lanerolle, 2009; Raj & Kazemian, 2006)	4
Inside control	(Buchta et al., 2006; Carrington, 2003; Costa-Soria & Pérez, 2009; Hepting et al., 2008; Kume et al., 2006; Kussmaul, 2009; McCartney et al., 2012; Nandigam et al., 2008; Petrenko et al., 2007; Qian & Fu, 2008; Santore et al., 2010; Seiter, 2009; Tao & Nandigam, 2006; Toth, 2006; Xing, 2010)	15
Full control	(Allen et al., 2003; Auer et al., 2011; Chen et al., 2008; Kilamo, 2010; Liu, 2005; Robles et al., 2008; Sabin, 2011; Shockey & Cabrera, 2005; Williams & Shin, 2006)	9
Not specified	(Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Hislop et al., 2009; Nachbar, 1998; Smrithi Rekha et al., 2009; Stroulia et al., 2011)	6
Does not apply	(Conlon & Hulick, 2005; Cousin et al., 2002; de Lanerolle et al., 2008; Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis & Morelli, 2008; Ellis et al., 2008; Ellis, Purcell, & Hislop, 2012; Gehringer, 2011; Gokhale et al., 2012; Kamthan, 2007; Kon et al., 2011; Li et al., 2009; Megias et al., 2009; Meneely et al., 2008; Morelli et al., 2007, 2009; O'Hara & Kay, 2003; Spinellis, 2006; Tucker et al., 2011; Whitehurst, 2009; Yamakami, 2012; Zhang & Su, 2007)	23

tiative/external approval category). The first three cited papers describe the same project, HFOSS, which we discuss in the *Long-Term Projects* subsection. The other paper discusses the role that OSP can play in computer science courses to enhance student learning (Raj & Kazemian, 2006). In the case reported, students have to i) choose one project from one list, ii) further examine the chosen project and propose improvements based on areas identified by the teacher, iii) work on their proposals, and iv) discuss their contributions with the OSS community.

We located 15 papers with *inside control*. In this case, there is no needed interaction with the community. In general, the instructor uses a project as an independent, internal branch. We illustrate this approach with two examples. In the method proposed by Costa-Soria and Pérez (2009), each student has to reverse engineer a selected open source project, outline the software architecture, identify the concerns of a particular subsystem, isolate one of these concerns, and ultimately, perform changes related to the concern in this subsystem. Santore et al. (2010) report an experience where students work with requirements, design, implementation and tests to produce a graphical user interface to Subversion, an open source version control system. In both examples, no interaction with the OSP community is required.

Lastly, we detected nine studies where the OSP community core are the faculty/staff themselves. Projects were created inside the university and have been supported by it (*full control*). Allen et al. (2003) present an example of this approach. They use Dr.Java, an open source software created in Rice University, to teach extreme programming in a software engineering course. Dr.Java is an integrated development environment (IDE) that aims to facilitate beginners starting to code in Java. During the course reported on this article, students had to answer to bug reports, feature or support requests from users. Besides the project team, all people that use DrJava (e.g., instructors of other institutions, other instructors and students from Rice) participate in the project. The project uses the infrastructure

Table 16. Studies classified by *Facet 9 - Project Choice*

Category	Studies	#
Predefined	(Allen et al., 2003; Auer et al., 2011; Beaufait et al., 2011; Boldyreff et al., 2009; Buchta et al., 2006; Chen et al., 2008; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Hepting et al., 2008; Kilamo, 2010; Krogstie, 2008; Liu, 2005; Morelli & de Lanerolle, 2009; Nandigam et al., 2008; Petrenko et al., 2007; Sabin, 2011; Santore et al., 2010; Shockey & Cabrera, 2005; Stroulia et al., 2011; Tao & Nandigam, 2006; Williams & Shin, 2006)	21
Choice list	(Carrington, 2003; Jaccheri & Østerlie, 2007; Kussmaul, 2009; Lutfiyya & Andrews, 2000; Raj & Kazemian, 2006; Seiter, 2009; Xing, 2010)	7
Free choice	(Budd, 2009; German, 2005; Horstmann, 2009; Lundell et al., 2007; Marmorstein, 2011; Martínez, 2009; McCartney et al., 2012; Papadopoulos et al., 2012; Qian & Fu, 2008; Sowe & Stamelos, 2007; Sowe et al., 2006; Toth, 2006)	12
Not specified	(Costa-Soria & Pérez, 2009; Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Hislop et al., 2009; Kume et al., 2006; Meiszner et al., 2009; Nachbar, 1998; Smrithi Rekha et al., 2009)	8
Does not apply	(Conlon & Hulick, 2005; Cousin et al., 2002; de Lanerolle et al., 2008; Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis & Morelli, 2008; Ellis et al., 2008; Ellis, Purcell, & Hislop, 2012; Gehringer, 2011; Gokhale et al., 2012; Kamthan, 2007; Kon et al., 2011; Li et al., 2009; Megias et al., 2009; Meneely et al., 2008; Morelli et al., 2007, 2009; O'Hara & Kay, 2003; Robles et al., 2008; Spinellis, 2006; Tucker et al., 2011; Whitehurst, 2009; Yamakami, 2012; Zhang & Su, 2007)	24

provided by SourceForge. Three experienced students work as project managers, assisting teachers to determine project priorities, and supervising teams of two to six students. Experienced students are students that had already passed the course and later worked as developers in the project. These students are important to sustain the project and they work on it either as paid research assistants or for credit.

As previously discussed (Section 3.1), finding a suitable project is a great challenge when adopting OSP in computing courses. Regardless of being a difficult task, it must be done by either instructors or students. The *Facet 9 - Project choice* gives us a view of how the selected studies deal with this issue. We identified 21 studies where instructors define the project students will work with, seven studies where instructors provide a list of projects students can choose from, and 12 studies where students can choose any open source project of their interest. Table 16 shows the studies classified in each category.

Elaborating on this facet, we uncovered that Hepting et al. (2008) use a predefined project for their students, although, in the end, authors decided to allow students to select a project that better matches their interests in future editions of the course. In another example, although students could freely choose their projects, they appreciated the instructor's help on this task (Sowe et al., 2006). In the *Choice list* category, Carrington (2003) provides a list of OSP, but encourages students to seek and suggest other projects. In two additional papers, the list is restricted to a specific domain area. In their example, Raj and Kazemian (2006) focus on OSS database systems, while in Seiter's (Seiter, 2009), it is on content management systems.

A careful look at Figure 3 shows that studies with *full control* level are also *predefined* for students. A single paper escapes this rule, an experience where students create an OSS from scratch (Robles et al., 2008). *No control* and *inside control* levels had 15 studies each. It is worth noting that while in most *no control* level studies, students have *free choice* over their projects, in the *inside control* level, students have no choice since projects are *predefined*, or, in some cases, students can choose from a list.

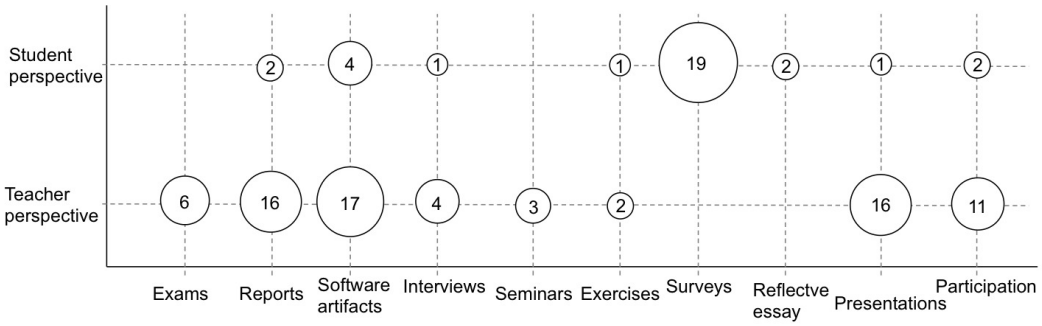

Figure 4. Assessment Perspective *versus* Assessment Type

Still in Figure 3, looking at the *Facet 7 - Curriculum Choice* and *Facet 8 - Control Level*, we realize that *inside control* is slightly more common in regular *courses*, while *no control* is slightly more common in *extra activities*. On the other hand, *full control* is mainly used in regular *courses*.

3.4. *Assessment of Student Learning*

The map in Figure 4 helps to answer the research question on the assessment of student learning (RQ3). The aim was getting an overview of how assessment happens in the selected studies. Thus, we built this map combining the *Facet 4 - Assessment Perspective* with the *Facet 5 - Assessment Type*. Studies are described in Tables 17 and 18, noting that studies could be classified in more than one category in both tables.

We tried to identify from which perspective studies evaluate students' learning (Table 17). We noticed that although assessment is important to evaluate the effectiveness of any learning approach, 14 studies just do not mention it, and only three studies specify criteria to assess students' products. Buchta et al. (2006) present detailed criteria to evaluate students' outcomes, while testing results (e.g., found bugs, reported bugs, fixed bugs) are used in the other two articles (Sowe & Stamelos, 2007; Sowe et al., 2006).

Regarding the *Facet 4 - Assessment Perspective*, Table 17 shows that student perspective (28 studies) was more frequently used than faculty perspective (24 studies). From the studies that mention assessment, 57.6% applied both perspectives (26.4% from all selected papers). Reading the 72 studies, we detected six studies that apply *Peer Review* for grading (Budd, 2009; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Hepting et al., 2008; Papadopoulos et al., 2012; Raj & Kazemian, 2006; Sabin, 2011), although this instrument is not mandatory in last paper. The study described by Sabin (2011) was the only selected study to report self-assessment, which was part of students' final grade. In this work, besides the teacher's assessment, students review their own exercises as well as their peers', and have their work assessed by students of other teams.

Analyzing Figure 4, we see that *software artifacts* (17 papers), *reports* (16 papers), and *presentations* (16 papers) were the main instruments used to grade students (*teacher perspective*), even though other instruments were used as well. On the other hand, *surveys* (19 papers) were the main instrument used to gather student feedback on their perception of their own learning (*student perspective*). For the same purpose, two studies applied *reflective essays*, and one study used *interviews*. Where peer review was applied, other instruments were also used to

Table 17. Studies classified by *Facet 4 - Assessment Perspective*

Category	Studies	#
Student perspective	(Allen et al., 2003; Buchta et al., 2006; Budd, 2009; Carrington, 2003; Chen et al., 2008; Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Hepting et al., 2008; Hislop et al., 2009; Kilamo, 2010; Krogstie, 2008; Kussmaul, 2009; Lundell et al., 2007; Marmorstein, 2011; Martínez, 2009; McCartney et al., 2012; Morelli & de Lanerolle, 2009; Papadopoulos et al., 2012; Petrenko et al., 2007; Qian & Fu, 2008; Raj & Kazemian, 2006; Sabin, 2011; Smrithi Rekha et al., 2009; Sowe & Stamelos, 2007; Sowe et al., 2006; Xing, 2010)	28
Teacher perspective	(Allen et al., 2003; Buchta et al., 2006; Budd, 2009; Carrington, 2003; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; German, 2005; Hislop et al., 2009; Krogstie, 2008; Kussmaul, 2009; Lundell et al., 2007; Lutfiyya & Andrews, 2000; McCartney et al., 2012; Morelli & de Lanerolle, 2009; Nandigam et al., 2008; Papadopoulos et al., 2012; Petrenko et al., 2007; Raj & Kazemian, 2006; Sabin, 2011; Santore et al., 2010; Sowe & Stamelos, 2007; Sowe et al., 2006; Stroulia et al., 2011; Xing, 2010)	24
Product perspective	(Buchta et al., 2006; Sowe & Stamelos, 2007; Sowe et al., 2006)	3
Not specified	(Auer et al., 2011; Beaufait et al., 2011; Boldyreff et al., 2009; Costa-Soria & Pérez, 2009; Horstmann, 2009; Jaccheri & Østerlie, 2007; Liu, 2005; Meiszner et al., 2009; Robles et al., 2008; Seiter, 2009; Shockey & Cabrera, 2005; Tao & Nandigam, 2006; Toth, 2006; Williams & Shin, 2006)	14
Does not apply	(Conlon & Hulick, 2005; Cousin et al., 2002; de Lanerolle et al., 2008; Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis & Morelli, 2008; Ellis et al., 2008; Ellis, Purcell, & Hislop, 2012; Gehringer, 2011; Gokhale et al., 2012; Kamthan, 2007; Kon et al., 2011; Kume et al., 2006; Li et al., 2009; Megias et al., 2009; Meneely et al., 2008; Morelli et al., 2007, 2009; Nachbar, 1998; O'Hara & Kay, 2003; Spinellis, 2006; Tucker et al., 2011; Whitehurst, 2009; Yamakami, 2012; Zhang & Su, 2007)	25

assess the student (*student perspective*).

For studies that used some type of assessment, 70% applied more than one instrument. From these, 66.7% simultaneously applied between four and six instruments. All studies that used both student and faculty perspectives applied at least three instruments, except for two studies (Allen et al., 2003; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007) that did not mention which instruments they applied.

3.5. *Distribution of Publications*

This section reports on how the research community interest on using OSP in software engineering education has evolved in the last years. We present a temporal view of publications, main venues for publication, and the institutions that studied this subject more thoroughly.

Temporal View. Figure 5 shows the distribution of publications over the years. Discussions addressing open source software and software engineering education first show up in 1998, with a few sparse publications over the first years. This rate rises since 2005, showing a peak in 2009 with 15 studies, a slight fall in 2010, and a recovery in 2011 (11 articles). It is worth noticing that we performed the search in October 2012.

Publication Sources. Table 19 presents the distribution of articles by venue, considering venues that published two or more of the selected articles. The complete set of publication venues (each with only article from the 72 selected articles) can

Table 18. Studies classified by *Facet 5 - Assessment Type*

Category	**Studies**	#
Exams	(Carrington, 2003; Morelli & de Lanerolle, 2009; Raj & Kazemian, 2006; Sabin, 2011; Sowe & Stamelos, 2007; Xing, 2010)	6
Reports	(Buchta et al., 2006; Budd, 2009; Carrington, 2003; German, 2005; Krogstie, 2008; Kussmaul, 2009; Lundell et al., 2007; Lutfiyya & Andrews, 2000; Morelli & de Lanerolle, 2009; Nandigam et al., 2008; Papadopoulos et al., 2012; Petrenko et al., 2007; Sabin, 2011; Sowe et al., 2006; Stroulia et al., 2011; Xing, 2010)	16
Software artifacts	(Buchta et al., 2006; Budd, 2009; Carrington, 2003; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Hepting et al., 2008; Hislop et al., 2009; Krogstie, 2008; Kussmaul, 2009; McCartney et al., 2012; Papadopoulos et al., 2012; Petrenko et al., 2007; Raj & Kazemian, 2006; Sabin, 2011; Santore et al., 2010; Sowe & Stamelos, 2007; Sowe et al., 2006; Stroulia et al., 2011; Xing, 2010)	18
Interviews	(Buchta et al., 2006; Hislop et al., 2009; Krogstie, 2008; Petrenko et al., 2007)	4
Seminars	(Budd, 2009; German, 2005; Kussmaul, 2009)	3
Exercises	(Carrington, 2003; Morelli & de Lanerolle, 2009; Sabin, 2011)	3
Surveys	(Buchta et al., 2006; Carrington, 2003; Chen et al., 2008; Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Hislop et al., 2009; Kilamo, 2010; Marmorstein, 2011; McCartney et al., 2012; Morelli & de Lanerolle, 2009; Papadopoulos et al., 2012; Petrenko et al., 2007; Qian & Fu, 2008; Raj & Kazemian, 2006; Smrithi Rekha et al., 2009; Sowe & Stamelos, 2007; Sowe et al., 2006; Xing, 2010)	19
Reflective essay	(Kussmaul, 2009; Lundell et al., 2007)	2
Presentations	(Budd, 2009; Carrington, 2003; Ellis, Morelli, de Lanerolle, & Hislop, 2007; German, 2005; Krogstie, 2008; Kussmaul, 2009; Lundell et al., 2007; Lutfiyya & Andrews, 2000; McCartney et al., 2012; Morelli & de Lanerolle, 2009; Nandigam et al., 2008; Papadopoulos et al., 2012; Sabin, 2011; Sowe & Stamelos, 2007; Sowe et al., 2006; Xing, 2010)	16
Participation	(Buchta et al., 2006; Budd, 2009; German, 2005; Papadopoulos et al., 2012; Petrenko et al., 2007; Raj & Kazemian, 2006; Sabin, 2011; Sowe & Stamelos, 2007; Sowe et al., 2006; Stroulia et al., 2011; Xing, 2010)	11
None	(Allen et al., 2003; Auer et al., 2011; Beaufait et al., 2011; Boldyreff et al., 2009; Costa-Soria & Pérez, 2009; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Horstmann, 2009; Jaccheri & Østerlie, 2007; Liu, 2005; Martínez, 2009; Meiszner et al., 2009; Robles et al., 2008; Seiter, 2009; Shockey & Cabrera, 2005; Tao & Nandigam, 2006; Toth, 2006; Williams & Shin, 2006)	17
Does not apply	(Conlon & Hulick, 2005; Cousin et al., 2002; de Lanerolle et al., 2008; Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis & Morelli, 2008; Ellis et al., 2008; Ellis, Purcell, & Hislop, 2012; Gehringer, 2011; Gokhale et al., 2012; Kamthan, 2007; Kon et al., 2011; Kume et al., 2006; Li et al., 2009; Megias et al., 2009; Meneely et al., 2008; Morelli et al., 2007, 2009; Nachbar, 1998; O'Hara & Kay, 2003; Spinellis, 2006; Tucker et al., 2011; Whitehurst, 2009; Yamakami, 2012; Zhang & Su, 2007)	25

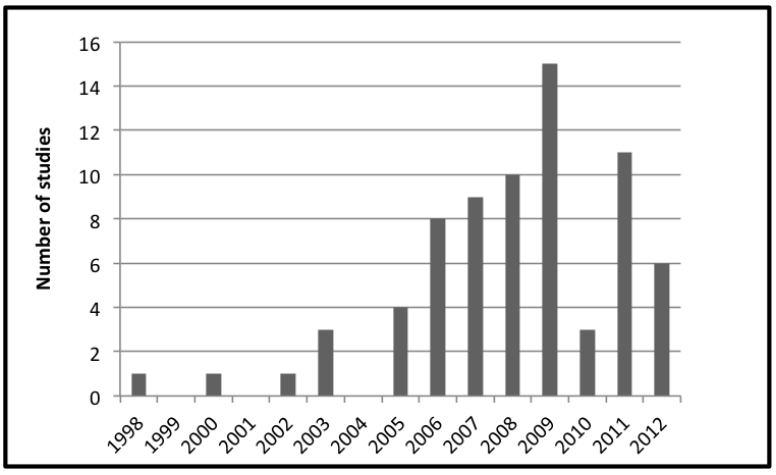

Figure 5. Publications *versus* Year

Table 19. Main Publication Venues

Venue	#
Frontiers in Education Conference, FIE	8
Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE	7
ACM Technical Symposium on Computer Science Education, SIGCSE	5
ASEE Annual Conference and Exposition	3
Free and Open Source Software (FOSS) Symposium	3
ACM SIGCSE Bulletin	2
Conference on Software Engineering Education and Training Workshop, CSEETW	2
IEEE Software	2
International Academic MindTrek Conference, MindTrek	2
International Conference on Computer Supported Education, CSEDU	2
International Conference on Software Engineering, ICSE	2

Table 20. Active Research Communities

Institutions	#
Trinity College, US	12
Drexel University, US	9
Western New England University, US	6
Aristotle University of Thessaloniki, Greece	4
North Carolina State University, US	3
Universitat Oberta de Catalunya, Spain	3
Bowdoin College, US	2
Community Leadership Team, Red Hat	2
Concordia University, Canada	2
Connecticut College, US	2
Grand Valley State University, US	2
Norwegian University of Science and Technology, Norway	2
University of Connecticut, US	2
Wayne State University, US	2
Wesleyan University, US	2

be found in our mapping study website[10]. Some important journals and conferences from the area of computing education have published the articles: Frontiers in Education (FIE) Conference, Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), ACM Technical Symposium on Computer Science Education (SIGCSE), among others. Nonetheless, we also uncovered articles published in more general venues, whose focus is not education (IEEE Computer Journal, Communications of the ACM) or venues focused on specific areas, such as software engineering (IEEE Software, International Conference on Software Engineering - ICSE, Brazilian Symposium on Software Engineering - SBES) or open source software (International Conference on Open Source Systems, Free and Open Source Software Symposium).

One important remark is that studies presented in the ITiCSE and SIGCSE conferences were also published in the ACM SIGCSE Bulletin. To not duplicate results, we decided to count only the conferences. However, the database engines reported two studies only in the SIGCSE Bulletin (Hepting et al., 2008; Santore et al., 2010). The authors of one such study (Santore et al., 2010) reported that it was submitted directly to the Bulletin.

Active Research Communities. To learn which institutions have studied student participation in OSP as an approach to learn SE, we looked at affiliation details in our selected studies. Table 20 summarizes the communities that had at least two selected publications in this mapping study. Overall, we found 68 institutions working on this theme.

According to Table 20, Trinity College, Drexel University and Western New England University produced the largest number of studies. Since all of these studies,

[10]https://sites.google.com/site/dmcnascimento/mapping

including contributions of Bowdoin College, Connecticut College, and Wesleyan University, are related to the Humanitarian FOSS – Free and Open Source Software – Project (see Section 3.6), we identified an important research community, not only for the number of published studies, but also due to the number of involved institutions.

We can highlight another active research community, based on Aristotle University of Thessaloniki (Greece). This institution has studies in collaboration with The Open University (UK) (Meiszner et al., 2009), United Nations University (The Netherlands) (Papadopoulos et al., 2012) and the Technological Education Institute (Greece) (Sowe et al., 2006) (see Section 3.6).

The article of Stroulia et al. (2011) has contributions from University of Alberta and University of Toronto, both in Canada. However, the authors reported that 25 universities participated in the Undergraduate Capstone Open-Source Project (UCOSP), object of their study.

The Free Technology Academy[11] is a virtual campus, supported by the collaboration of Universitat Oberta de Catalunya, in Spain, Instituto Superior de Ciencias do Trabalho e da Empresa, in Portugal, the Open University Netherlands, and the Free Knowledge Institute, in Netherlands (Megias et al., 2009).

Ellis, Hislop, Chua, and Dziallas (2011) and Whitehurst (2009) cite Oregon State University (US) and Seneca College (Canada) as academic institutions that had already recognized the benefits of OSS involvement for student learning. They included open source development in their formal program and created specific labs or centers for this subject. While Whitehurst (2009) add North Carolina State University (US) to this list, Ellis, Hislop, Chua, and Dziallas (2011) cite Rochester Institute of Technology (US) and Rensselaer Polytechnic Institute (US). In our research, we found three articles with contributions from North Carolina State University (Gehringer, 2011; Meneely et al., 2008; Williams & Shin, 2006), one article with contributions from Oregon State University (Budd, 2009) and one from Rochester Institute of Technology (Raj & Kazemian, 2006). We did not find papers related to Seneca College or to Rensselaer Polytechnic Institute. This shows that there can be other initiatives unreported by the scientific community.

On the other hand, there are also initiatives from OSS communities and companies. Whitehurst (2009) cites Google's Summer of Code (GSoC) program, that, along the years, has involved more than 1,500 students in several OSP. Red Hat invited interested teachers to take part in POSSE (Professors' Open Source Summer Experience), a workshop to introduce tools and practices of open source communities, and also to provide support and mentoring during the school year (Ellis, Hislop, Chua, & Dziallas, 2011). Red Hat Community leadership team contributes in two articles as well (Ellis, Hislop, Chua, & Dziallas, 2011; Whitehurst, 2009).

Ellis, Hislop, Chua, and Dziallas (2011) also highlight the existence of communities interested in aiding instructors in teaching and involving students in OSP: TeachingOpenSource.org[12], the SoftHum[13] and HumIT [14] projects.

Finally, we also found the openSE[15], a virtual environment whose goal is to gather academy, open source projects and companies to combine formal and informal learning, and stimulate practical experiences, among other goals. Several

[11] http://ftacademy.org
[12] http://teachingopensource.org
[13] http://www.xcitegroup.org/softhum
[14] http://www.xcitegroup.org/humit
[15] http://www.opense.net

Table 21. Studies with solutions applied in the long term

Study	Objective	Period
(Qian & Fu, 2008)	Describes a course model for teaching Component-Based Software Development (CBSD) using open source components.	For four years before publication in 2008).
(Toth, 2006)	Presents author's experiences with students using and extending open source software engineering tools.	4 times: 2000-2001, 2003-2004, 2004-2005 and 2005-2006.
(Jaccheri & Østerlie, 2007)	Reports authors' experience using principles from action research, blended research, development and OSP.	For five years since 2002.
(Chen et al., 2008)	Describes GamesCrafters, a research and development group that works with an open-source AI architecture.	Since 2001 (study was published in 2007).
(Sabin, 2011)	Discusses the challenges to engage students in OSP proposed by clients.	One experience during Fall 2010 and two experiences during Spring 2011.
(Stroulia et al., 2011)	Reports authors' experience with the Undergraduate Capstone Open-Source Project.	Fall 2008, Winter 2009, Fall 2009, Winter 2010, Fall 2010, and Winter 2011.
(Budd, 2009)	Describes a course in open source development.	2 years.

institutions, such as Sociedade Portuguesa de Inovação, Aristotle University of Thessaloniki, Tampere University of Technology, UNU-MERIT, The Open University, among others, take part in this initiative.

3.6. *Long-Term Projects*

As we read the articles included in the mapping, we tried to capture the maturity of the solution proposal in each study, through how long or how many times it had been applied. Moreover, we also recognized that some studies are strongly related to others. To clarify this relationship between articles, we analyzed institutions, authors and content of the studies.

Table 21 presents studies whose solution proposal was applied twice or more. One can observe that very few cases did so. Hence, most studies identified in this mapping likely applied the proposed solution only once.

In a different thread, Figure 6 shows the cases where more than one published study is related to the same project.

In Figure 6, each linked set of nodes represents one identified project. The largest project, with 15 articles, is the Humanitarian FOSS Project (HFOSS project). As discussed in Section 3.5, the HFOSS project involves various universities and aims to revitalize undergraduate computing courses in the United States by building open source software that benefits non-profit organizations (de Lanerolle et al., 2008). The project began in 2006 (first publication in 2007). Since this year, many students and faculty have engaged in developing various OSS systems to benefit humanity (Tucker et al., 2011). In our search, we found some papers that describe the project and its status (de Lanerolle et al., 2008; Ellis & Morelli, 2008; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Ellis et al., 2008; Morelli et al., 2007, 2009; Tucker et al., 2011), three studies over the years that reported results from students' perspective on what they learned when they took part in an HFOSS project (Ellis, Hislop, & Morelli, 2011; Ellis, Hislop, et al., 2012; Hislop et al., 2009), and four articles with general proposals, not specifically related to humanitarian open source but for any open source project (Ellis & Hislop, 2011; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis,

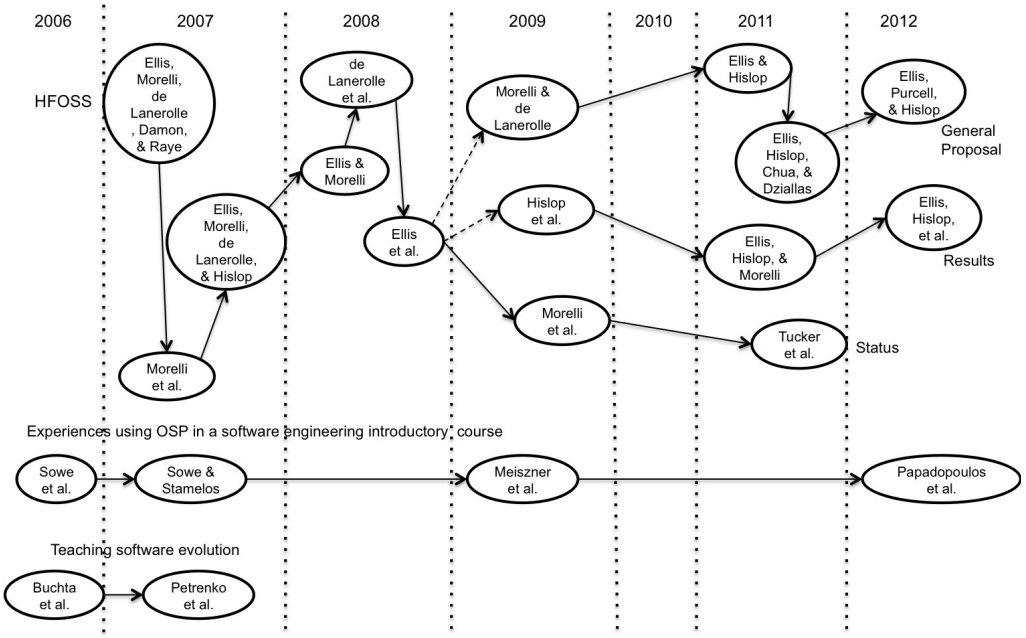

Figure 6. Long Term Projects

Purcell, & Hislop, 2012; Morelli & de Lanerolle, 2009).

As described in Section 3.5, Aristotle University of Thessaloniki, in collaboration with other universities, has developed another long-term project. The first published study was in 2006, where Sowe et al. (2006) propose a framework for teaching software testing using OSS in a SE introductory course. However, in this case, only volunteering students participated. In the second paper, Sowe and Stamelos (2007) extended the first paper with more details, including the results of two applied surveys. In 2009, Meiszner et al. (2009) report not only the pilot project of 2005/2006, but also other experiences of 2006/2007, where students could test or develop an OSS, and of 2007/2008, where students should use only OSS to either test, develop or write a requirement specification document for an OSS system. Nevertheless, the focus of this latter study, as part of the FLOSSCom project, is to propose a hybrid learning environment, with the same tools usually found in OSS communities. Thus, in the 2008/2009 experiment, students not only practiced with OSP, but also participated in the process of knowledge construction. In the last selected study, Papadopoulos et al. (2012) show details of the instructional method applied since 2008/2009, and report results of students' opinion about the method, over the previous three years.

Petrenko et al. (2007) extend the work of Buchta et al. (2006). Both used OSP to teach software evolution, following a specific software change process model. While Buchta et al. (2006) mentioned that the project began in Fall 2004, Petrenko et al. (2007) compared results between offered courses in Fall 2004 and Fall 2005. Building from the experience in this long-term project, Rajlich, one of the authors of these papers, published a book intended to aid teaching software engineering from a software change perspective (Rajlich, 2012).

4. Discussion and Limitations

The results previously reported are discussed below, together with the limitations of our research.

4.1. *Discussion*

Subject of interest. Open source software has played an important role in software industry, with competitive solutions in various domains comparing to proprietary software. This caused a growth of interest in this subject not only in academia, but also in government and industry. In industry, some companies may sometimes see OSS as a rival. For example, Microsoft has once argued against adopting GNU/Linux (Lessig, 2010). Other companies have become OSP participants (Carrington, 2003). Others have even changed their business models (Megias et al., 2009). IBM Corporation, once an important advocate of intellectual property rights for computer programs, presently contributes to the development of Linux and other OSP. Furthermore, it has also released the Eclipse IDE as an open source software, supporting and extending it (Samuelson, 2006). Governments have encouraged their agencies to use OSS (Auer et al., 2011; Lessig, 2010), because of their quality, low cost and the independence they provide. Finally, academia sees OSS both as a new software development process[16], as a source of examples of large software that solves real world needs, and as accessible real development environments that students can take part in.

Various universities have published articles on either teaching open source or using OSP to learn SE concepts and principles, mostly in the latest seven years. We found several studies from the United States, Canada and Europe (mainly from Greece, Spain, Norway and the Netherlands, and some from the United Kingdom, Sweden, Finland and France). We also uncovered papers from Latin America (Brazil and Puerto Rico), Asia (Japan and China) and Oceania (Australia).

Studies are very diverse in aim. Some use OSP to attract and motivate students in computing courses, while others provide guidance on introducing OSP in the curriculum. Yet others propose platforms, web applications and learning approaches.

In summary, the results presented in this mapping study provide evidence that the research community is interested in the topic, for its relevance and broad range.

Need for more rigorous research methods. One of our main findings is that most studies are not deeply concerned with research methods: among the relevant selected papers, we may cite two *case studies*, one *action research*, but no *experiments*, which seems to be in sharp contrast with the recent growth of evidence-based software engineering. Software engineering research is slowly adopting increasing scientific rigor in the latest years. Moreover, software engineering education is an interdisciplinary area. As such, it can strongly benefit not only from SE research methods, but also from research methods from areas such as sociology, anthropology, pedagogy and communication. Thus, it seems relevant to identify which research methods are more appropriate in this intersection, in order to achieve better results in this interdisciplinary area.

[16] Robles et al. (2008) mention "software engineering in libre software environments", while German (2005) uses the term "open source software engineering".

Learning approach. Given the large variety of OSS systems, with different sizes, domains and complexity, we believe they are an important source of examples to teach software design, architecture and quality (Brown & Wilson, 2012). Nonetheless, we found very few studies to describe these issues in, say, a *case-based learning* approach. Only three reported this type of experience: two of them related to software design and architecture, and one of them, to software evolution.

No selected study focuses on learning the areas of requirements or configuration management, despite the large use of configuration management and issue tracking tools in OSP. We believe that these specific areas may benefit from an active engagement with OSP and their associated tools.

Meiszner et al. (2009) point out some learning features of the OSP experience: self-learning, project-/problem-/inquiry-based learning, collaborative learning, and reflective practice. However, very few studies cite these learning approaches, and none of them provides details on how to design and implement pedagogical practices that result in effective learning of SE skills.

Fitting OSP into the Academic System. We developed three facets to describe how OSP fit into the existing academic framework in the selected studies. We identified some trends after analyzing the results.

First, *Facet 7 - Curriculum Choice* showed that most studies either use OSP in regular SE courses or in graduate courses on open source software. *Capstone projects* and *non-curricular activities* account for less than half of the studies. Then, we noticed a balance between *inside control* and *no control* course designs in the *Facet 8 - Control Level*. Finally, *Facet 9 - Project Choice* showed that most experiences use *predefined* open source projects to teach software engineering.

Combining *Facet 8 - Control Level* and *Facet 9 - Project Choice*, we detected three particular trends: (i) *full control* and a *predefined* project, (ii) *no control* and *free choice* projects, and (iii) *inside control* and either a *predefined* project or a *choice list* of some OSP.

For each category in each facet, we discuss below the benefits and drawbacks regarding issues such as: formal structure of educational systems, student motivation, student overload, instructor effort, sustainability of the initiative, and learning objectives.

In (*Facet 7 - Curriculum Choice*), we suggest that instructors should consider some issues before choosing a project: (i) a regular course has time constraints ("term-based") and should cover specific content for its duration; (ii) less time constraints are present in capstone projects, where advisors and students decide together on the content; and (iii) non-curricular activities provide free choice of goals and generally impose no time constraints.

In Section 3.1, we assert that project choice is a challenge. Some criteria have been proposed to help cope with this task, as in the framework provided by Ellis, Purcell, and Hislop (2012). Besides these criteria, one also has to decide who chooses the projects, whether students or faculty. Whoever does, the following issues should be considered: (i) even with appropriate criteria, choosing a project demands effort; (ii) project choice influences on how much support teachers should provide students; (iii) project choice by students may positively influence their motivation.

Some studies assert that teachers should help students whenever they need. Instructors should manage students' expectations, and provide guidance as well as hands-on assistance (Raj & Kazemian, 2006). The level of support the instructor can provide depends on his or her knowledge of the project. Xing (2010) highlights that teachers should be familiar with the projects, while Gehringer (2011) claims

that they should be personally involved in the project development. Moreover, Budd (2009) mentions barriers to the existence of more courses on open source development: few instructors have previous experience with OSS, and there is a lack of background resources. According to this author, there are trade books, but no textbooks. As a result, both using OSP and teaching OSS demand additional effort from instructors (Xing, 2010). Various studies mention the need for teaching assistants (Buchta et al., 2006; Petrenko et al., 2007; Sowe & Stamelos, 2007) or mentors in the university or in the community (Budd, 2009; Chen et al., 2008; Ellis, Hislop, Chua, & Dziallas, 2011; Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Gehringer, 2011; Morelli et al., 2009; Sabin, 2011; Stroulia et al., 2011). Petrenko et al. (2007) acknowledge that the effort decreases on second course offerings, when projects are reused and technology does not change.

Student motivation is very important for learning. Motivation can come either from the student's interest in participating in a particular OSP, or from contributing to a non-profit organization in a project that benefits humanity. However, motivation also decreases when students feel overwhelmed or when no one recognizes their work.

Each OSP imposes a learning curve on students (Meneely et al., 2008; Xing, 2010; Zhang & Su, 2007). The learning curve may be related to software domain, size, design, technologies, quality, and (the scarcity of) documentation. The challenge is admitting students with different backgrounds, but not letting them feel overwhelmed. To ease this problem, it is important to create an environment that promotes collaborative learning.

Jaccheri and Østerlie (2007) remind that motivation and interest are important issues for anyone participating in open source communities. Letting students choose one project from their own interest increases motivation, hence, the potential for learning in the *free choice* category. However, as we have already seen, no project is appropriate to every learning objective, thus, in this category, the effort required to find an adequate project is transferred to students. Depending on student experience, some students may feel overwhelmed with this task and may appreciate support on project selection (Papadopoulos et al., 2012). On the other hand, when students freely choose their projects, the ability of an instructor to support students' activities decreases substantially, since it is hard to maintain a working knowledge of each chosen project.

According to Gokhale et al. (2012), it is very difficult for students to find appropriate projects, or even unlikely that they select projects with similar level of complexity across teams. Therefore, teachers should provide a pool of projects, from which students select a project of interest (*Choice list* category). The task of preparing the list requires effort in advance from instructors, before the course starts.

In the other extreme of the *project choice* range (*Facet 9*), lies the *predefined project* category, the most frequent choice in the selected studies. Predefined projects avoid the possibility of project choice by students. However, they allow the instructor to fully support students, since he or she has a deeper understanding of the chosen OSP. The instructor can also align students' assignments to content and duration of the course. Raj and Kazemian (2006) state two benefits when instructors use only one or two OSP. First, the teacher can hold class discussions on OSP design and code, both before and after changes are performed. Second, a sustainable development effort may derive, when instructors use the code resulting from previous offerings of the course as a starting point. These authors point out

that previous students can also be invited to present their experience to current students.

Categories of the *Facet 8 - Control Level* represent how much control of students' activities instructors have. Hence, they are directly related to aligning students' assignments with course content and duration, and to the creation of a sustainable development effort.

Auer et al. (2011) state that when faculty lead an OSP (*full control* category), the project can be steered in a way that educational goals are considered. Instructors can provide adequate support, documentation, high quality design and code, comprehensive unit tests, and a development model, all aligned to learning goals and student assignments. Full control also allows a sustainable development effort. Student work is not discarded. Newbies reuse and extend code from previous students. Students with more experience in the project can work as the community core team, project managers or teaching assistants. The main challenge of this choice is to create and sustain a "real" community. One needs to attract experienced developers to the team and users to report bugs and request new features. Auer et al. (2011) assert that if teachers and students alone participate in the OSP, there is no improvement to the typical way of teaching and learning SE. The key to attract other members to the community is software domain. Robles et al. (2008) state the importance of low barriers for new users, i.e., the system should be easy to install and use, there should be a minimum level of activity in the project. Some studies suggest that other teachers and students from the same institution can work as users (Allen et al., 2003; Auer et al., 2011).

With *inside control*, the instructor creates an internal branch of the OSP, aligning all assignments to the course goals. It requires a deeper understanding of the project by the instructor. Thus, this category is usually associated with a *predefined* project, and the instructor is able to provide support for students. Frequently, students have to understand design and code, develop enhancements and perform tests. If the instructor decides to continue the branch in a future course offering, his/her previous effort is reduced, and student work may not be discarded. These features of *inside control* make it adequate to *case-based learning*. All examples we found of *case-based learning* (Costa-Soria & Pérez, 2009; Kume et al., 2006; Nandigam et al., 2008; Tao & Nandigam, 2006; Toth, 2006) but one are also examples of the *inside control*. The exception was a single study that proposed a web application to facilitate case creation (Zhang & Su, 2007).

Inside initiative / external approval is appropriate when there is full integration between faculty and community. Otherwise the risk of wasting students' efforts is large. HFOSS is an example where this integration occurs (Ellis, Morelli, de Lanerolle, Damon, & Raye, 2007; Ellis, Morelli, de Lanerolle, & Hislop, 2007; Morelli & de Lanerolle, 2009).

The main issue with the *no control* choice is that instructors may fail to enforce good development practices. Students should contribute to open source projects, attending to requests from their users. This choice allows for a fully "real" environment, where students answer to real demands, participate in the community development process, and interact with users and external developers. This environment may foster the development of professional skills related to communication, problem solving, and confidence, among others. As their contributions will be evaluated by a third-party, students usually try to do their best.

When this approach is associated with the *free choice* category, the instructor only monitors students' activities. When it exists, mentoring comes from the com-

munity. While some OSP are interested in getting student involvement (Lundell et al., 2007), others are not friendly to coach novices frequently (Ellis, Purcell, & Hislop, 2012). One challenge of this approach is the time constraints imposed by academic calendars. It is difficult for students with little or no support to make significant contributions within a time limit and with simultaneous work in other courses. Furthermore, lack of experience to code may lead students to contribute only with documentation and bug reports (German, 2005; Lundell et al., 2007), even though coding is a relevant skill required in most SE courses. Therefore, teachers should reflect on how to deal with this problem. Papadopoulos et al. (2012) decided to extend the official lecturing period, allowing students to submit their assignment results at a later time.

When students submit their work to a OSS community, in general, one of three situations may happen: (i) the contribution is accepted and added to the project; (ii) the community asks for some changes or requires additional work; and (iii) the contribution is not accepted, for different reasons. Each situation may have a differnt impact on student motivation and learning, sometimes requiring teacher intervention to better deal with the situation.

Combining *no control* and *free choice* seems to be an interesting approach to learn OSS. Students have a real feeling of the OSS ecosystem, how the community works, how the development process is organized, what challenges there are. Learning begins with project selection, when students analyzing features of each candidate project. It usually includes students experiencing the whole open source process, making contributions, and using tools such as version control systems and bug trackers.

Assessment. Despite the scarce references to learning assessment in the selected papers, some studies assessed the experiences from both the *teacher* and the *student perspective*. They also used various different instruments. The main issues with the *teacher perspective* to assessment is the absence of clear definitions of criteria to assess students' products, performed tasks, and expected skills and attitudes. Therefore, it is important to state that student assessment deserves more thorough work.

Ellis, Morelli, de Lanerolle, and Hislop (2007) point out that students can perform various types of contributions. Since they have different backgrounds and previous experiences, they fulfill different roles and perform different contributions. Therefore, the authors recognize that grading is not an easy task. They suggest the need to establish a set of metrics for each role a student plays.

Learning how to solve complex problems, and knowing how to work in teams are relevant skills that are typical requirements in SE education. Thus, students should develop some skills such as communication and leadership. Peer assessment is one important instrument to approach this need. Morelli and de Lanerolle (2009) sustain that students should assess their peers in conjunction with teacher assessment in courses on OSP, even though, the adoption of this practice is still a challenge.

In any active learning approach, students are responsible to conduct their own learning. A formative evaluation that includes self, peer and faculty assessment can play an important role in this process. According to Ellis, Purcell, and Hislop (2012), the iterative development process present in OSP, where any artifact can be reviewed by different people, and in multiple times, provides an intrinsic and valuable environment of formative evaluation.

All those issues represent research opportunities to be more thoroughly explored in the future.

4.2. *Limitations*

Results of this study must be interpreted within the following limitations: (a) publications that were available after October 2 and 15, 2012 were not accounted for; (b) first list of studies obtained may be subject to the limitations of the automated search engines of each digital library used (IEEE, ACM, Scopus, Springer, Science Direct and Engineering Village); (c) only studies written in English were selected; (d) only studies whose full text were available were considered; (e) "snowballing" was performed only for the references of relevant selected papers; (f) the classification of each study was performed by only one reviewer.

5. Conclusion

This work presents a mapping study that summarizes and categorizes information on how open source projects have been used in the context of software engineering education. The goal was not to verify the use of open source software as a tool or computer environment, but to identify initiatives where students participate in the development effort of open source software, because this allows students to deal with real projects, often large and complex, such as the ones they will find when working in industry.

Results show that there are important research communities interested in this topic, especially in the United States, Canada and Europe. These communities have produced diverse and interesting initiatives over the years with two general goals: learning SE concepts and principles by using OSP, followed by learning to produce open source software.

Some trends and issues emerged from a detailed analysis of the studies: i) solution proposals are the main research approach; ii) very few papers focus on specific software engineering areas; iii) the traditional project method is the main learning approach; iv) most studies use previously chosen OSP in regular courses; v) there is a balance between *inside* and *no control* approaches; and iv) very few papers use criteria to evaluate students' learning based on either outcomes or developed skills. We also found three main combinations of OSP use: a) *full control* and *predefined* projects, b) *no control* and *free choice* projects, and c) *inside control* with no or almost no project choice for students. These trends and issues provide future directions for research.

We plan to perform an exploratory mixed-methods research using OSP in an undergraduate course in software engineering, in order to obtain new insights with an experience with this approach. We will experiment with a combination of *inside control* and a *predefined* project in a software evolution course, using a combination of different learning approaches. We also plan to investigate better methods to assess students' learning in this context.

Acknowledgment

The authors would like to thank Rodrigo Souza for composing the bubble plot diagrams, and Kenia Cox, Thiago Almeida and Wendell Sampaio for their participation as reviewers in the step of paper screening.

References

Allen, E., Cartwright, R., & Reis, C. (2003, January). Production Programming in the Classroom. *ACM SIGCSE Bulletin*, *35*(1), 89–93.

Auer, L., Juntunen, J., & Ojala, P. (2011, September). Open Source Project as a Pedagogical Tool in Higher Education. In *Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments (MindTrek '11)* (pp. 207–213). New York, New York, USA: ACM Press.

Barg, M., Fekete, A., Greening, T., Hollands, O., Kay, J., Kingston, J. H., & Crawford, K. (2000). Problem-Based Learning for Foundation Computer Science Courses. *Computer Science Education*, *10*(2), 109–128.

Beaufait, M. P., Chen, D., Dietrich Jr., C. B., Dietrich, C., & Vanhoy, G. M. (2011). Transition from Undergraduate Research Program Participants to Researchers and Open Source Community Contributors. In *ASEE Annual Conference and Exposition.* Vancouver: ASEE.

Boldyreff, C., Capiluppi, A., Knowles, T., & Munro, J. (2009). Undergraduate Research Opportunities in OSS. *Open Source Ecosystems: Diverse Communities Interacting*, 340–350.

Brown, A., & Wilson, G. (Eds.). (2012). *The Architecture of Open Source Applications.* lulu.com.

Buchta, J., Petrenko, M., Poshyvanyk, D., & Rajlich, V. (2006, September). Teaching Evolution of Open-Source Projects in Software Engineering Courses. In *22nd IEEE International Conference on Software Maintenance* (pp. 136–144). IEEE.

Budd, T. A. (2009). A Course in Open Source Development. In *Integrating FOSS into the Undergraduate Computing Curriculum, Free and Open Source Software (FOSS) Symposium.* Chattanooga. Retrieved from `http://www.cs.trincoll.edu/~ram/hfoss/Budd-FOSS-Course.pdf`

Budgen, D., Turner, M., Brereton, P., & Kitchenham, B. (2008). Using Mapping Studies in Software Engineering. In *Proceedings of PPIG 2008* (Vol. 2, pp. 195–204). Lancaster University.

Carrington, D. (2003). Teaching Software Design with Open Source Software. In *33rd Annual Frontiers in Education Conference (FIE)* (Vol. 3, pp. S1C–9–S1C–14). IEEE.

Chen, Y., Roytman, A., Fong, P. C., Hong, J., Garcia, D. D., & Poll, D. E. (2008). 200 Students Can't Be Wrong! GamesCrafters, a Computational Game Theory Undergraduate Research and Development Group. In *AAAI Spring Symposium - Technical Report.*

Conlon, M. P., & Hulick, F. W. (2005). Is There a Role for Open Source Software in Systems Analysis ? In *Proceedings of ISECON* (Vol. 22, pp. 1–7).

Costa-Soria, C., & Pérez, J. (2009, August). Teaching Software Architectures and Aspect-Oriented Software Development Using Open-Source Projects. *ACM SIGCSE Bulletin*, *41*(3), 385.

Cousin, E., Ouvradou, G., Pucci, P., & Tardieu, S. (2002). Picolibre: a Free Collaborative Platform to Improve Students' Skills in Software Engineering. In *IEEE International Conference on Systems, Man and Cybernetics* (Vol. vol.4, pp. 564–568). IEEE.

de Lanerolle, T. R., Morelli, R. A., Danner, N., & Krizanc, D. (2008). Creating an Academic Community to build Humanitarian FOSS: A Progress Report. In *the 5th International ISCRAM Conference* (pp. 337–341). Washington. Retrieved from `http://www.hfoss.org/uploads/images/ISCRAM2008\_deLanerolle\_etal.pdf`

Ellis, H. J. C., & Hislop, G. W. (2011, June). Courseware: Student Learning via FOSS Field Trips. In *Proceedings of the 16th Annual joint Conference on Innovation and Technology in Computer Science Education (ITiCSE'11)* (p. 329). New York, New York, USA: ACM Press.

Ellis, H. J. C., Hislop, G. W., Chua, M., & Dziallas, S. (2011, October). How to Involve Students in FOSS Projects. In *41st Annual Frontiers in Education Conference (FIE)*

(pp. T1H–1–T1H–6). IEEE.

Ellis, H. J. C., Hislop, G. W., & Morelli, R. A. (2011, June). A Comparison of Software Engineering Knowledge Gained from Student Participation in Humanitarian FOSS Projects. In *Proceedings of the 16th Annual joint Conference on Innovation and Technology in Computer Science Education (ITiCSE'11)* (p. 360). New York, New York, USA: ACM Press.

Ellis, H. J. C., Hislop, G. W., Rodriguez, J. S., & Morelli, R. A. (2012). Student Software Engineering Learning via Participation in Humanitarian FOSS Projects. In *ASEE Annual Conference and Exposition.*

Ellis, H. J. C., & Morelli, R. A. (2008, April). Support for Educating Software Engineers Through Humanitarian Open Source Projects. In *21st IEEE-CS Conference on Software Engineering Education and Training Workshop* (pp. 1–4). IEEE.

Ellis, H. J. C., Morelli, R. A., de Lanerolle, T. R., Damon, J., & Raye, J. (2007, March). Can Humanitarian Open-Source Software Development Draw New Students to CS? *ACM SIGCSE Bulletin*, *39*(1), 551–555.

Ellis, H. J. C., Morelli, R. A., de Lanerolle, T. R., & Hislop, G. W. (2007, July). Holistic Software Engineering Education Based on a Humanitarian Open Source Project. In *20th Conference on Software Engineering Education & Training (CSEET'07)* (pp. 327–335). IEEE.

Ellis, H. J. C., Morelli, R. A., & Hislop, G. W. (2008, October). Work in Progress: Challenges to Educating Students within the Community of Open Source Software for Humanity. In *38th Annual Frontiers in Education Conference (FIE)* (pp. S3H–7–S3H–8). IEEE.

Ellis, H. J. C., Purcell, M., & Hislop, G. W. (2012, February). An Approach for Evaluating FOSS Projects for Student Participation. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education - SIGCSE '12* (pp. 415–420). New York, New York, USA: ACM Press.

Gehringer, E. F. (2011, October). From the Manager's Perspective: Classroom Contributions to Open-source Projects. In *41st Annual Frontiers in Education Conference (FIE)* (pp. F1E–1–F1E–5). IEEE.

German, D. M. (2005). Experiences Teaching a Graduate Course in Open Source Software Engineering. In *the First International Conference on Open Source Systems* (pp. 326–328).

Gokhale, S. S., Smith, T., & McCartney, R. (2012, June). Integrating Open Source Software into Software Engineering Curriculum: Challenges in Selecting Projects. In *First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex)* (pp. 9–12). IEEE.

Hepting, D. H., Peng, L., Maciag, T. J., Gerhard, D., & Maguire, B. (2008, June). Creating Synergy Between Usability Courses and Open Source Software Projects. *ACM SIGCSE Bulletin*, *40*(2), 120–123.

Hislop, G. W., Ellis, H. J. C., & Morelli, R. A. (2009, August). Evaluating Student Experiences in Developing Software for Humanity. *ACM SIGCSE Bulletin*, *41*(3), 263–267.

Horstmann, C. S. (2009). Challenges and Opportunities in an Open Source Software Development Course. In *Integrating FOSS into the Undergraduate Computing Curriculum, Free and Open Source Software (FOSS) Symposium.* Chattanooga. Retrieved from `http://www.hfoss.org/symposium09/documentdl/papers/HFOSS\_Symp-paper24.pdf`

Jaccheri, L., & Østerlie, T. (2007, May). Open Source Software: A Source of Possibilities for Software Engineering Education and Empirical Software Engineering. In *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (pp. 1–5). IEEE.

Kamthan, P. (2007). On the Prospects and Concerns of Integrating Open Source Software Environment in Software Engineering Education. *Journal of Information Technology*

Education, *6*, 45–64.

Kilamo, T. (2010, October). The Community Game: Learning Open Source Development through Participatory Exercise. In *Proceedings of the 14th International Academic MindTrek Conference on Envisioning Future Media Environments (MindTrek '10)* (pp. 55–60). New York, New York, USA: ACM Press.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. (EBSE 2007-001). Retrieved from `http://www.citeulike.org/user/akom/article/3955888`

Kitchenham, B., Mendes, E., & Travassos, G. H. (2007, May). Cross versus Within-Company Cost Estimation Studies: A Systematic Review. *IEEE Transactions on Software Engineering*, *33*(5), 316–329.

Kon, F., Meirelles, P., Lago, N., Terceiro, A., Chavez, C., & Mendonca, M. (2011, September). Free and Open Source Software Development and Research: Opportunities for Software Engineering. In *25th Brazilian Symposium on Software Engineering* (pp. 82–91). IEEE.

Krogstie, B. R. (2008). Power Through Brokering: Open Source Community Participation in Software Engineering Student Projects. In *Proceedings of the 13th International Conference on Software Engineering (ICSE'08)* (pp. 791–800). ACM Press.

Kume, I., Nitta, N., & Takemura, Y. (2006). A Method for Creating Teaching Materials of Practical Object-Oriented Methods Education. In *Learning by Effective Utilization of Technologies: Facilitating Intercultural Understanding, Proceeding of the 14th International Conference on Computers in Education, ICCE 2006.*

Kussmaul, C. (2009). Software Projects Using Free and Open Source Software: Opportunities, Challenges, & Lessons Learned. In *ASEE Annual Conference and Exposition.*

Lessig, L. (2010). Open Source Baselines: Compare to What? In R. W. Hahn (Ed.), *Government policy toward open source software* (p. 114).

Li, W., Zhang, S., & Li, Z. (2009, December). Open Source Movement and Computer Science Education Innovation. In *International Conference on Information Engineering and Computer Science* (pp. 1–4). IEEE.

Liu, C. (2005). Enriching Software Engineering Courses with Service-Learning Projects and the Open-Source Approach. In *Proceedings of the 27th International Conference on Software Engineering (ICSE)* (pp. 613–614). IEEe.

Lundell, B., Persson, A., & Lings, B. (2007). Learning Through Practical Involvement in the OSS Ecosystem: Experiences from a Masters Assignment. In J. Feller, B. Fitzgerald, W. Sacchi, & A. Sillitti (Eds.), *Open Source Development, Adoption and Innovation* (pp. 289–294). Springer.

Lutfiyya, H. L., & Andrews, J. H. (2000). Experiences with a Software Maintenance Project Course. *IEEE Transactions on Education*, *43*(4), 383–388.

Marmorstein, R. (2011, June). Open Source Contribution as an Effective Software Engineering Class Project. In *Proceedings of the 16th Annual joint Conference on Innovation and Technology in Computer Science Education (ITiCSE'11)* (pp. 268–272). New York, New York, USA: ACM Press.

Martínez, J. J. M. (2009, November). Learning Free Software Development from Real-World Experience. In *International Conference on Intelligent Networking and Collaborative Systems* (pp. 417–420). IEEE.

McCartney, R., Gokhale, S. S., & Smith, T. M. (2012, September). Evaluating an Early Software Engineering Course with Projects and Tools from Open Source Software. In *Proceedings of the 9th Annual International Conference on International Computing Education Research (ICER '12)* (pp. 5–10). New York, New York, USA: ACM Press.

Megias, D., Tebbens, W., Bijlsma, L., & Santanach, F. (2009, August). Free Technology Academy: A European Initiative for Distance Education about Free Software and Open Standards. *ACM SIGCSE Bulletin*, *41*(3), 70–74.

Meiszner, A., Moustaka, K., & Stamelos, I. (2009). A Hybrid Approach to Computer Science Education a Case Study: Software Engineering at Aristotle University. In

Proceedings of the 1st International Conference on Computer Supported Education (CSEDU) (pp. 39–46). Retrieved from http://pt.scribd.com/doc/10933440/A-HYBRID-APPROACH-TO-COMPUTER-SCIENCE-EDUCATION

Meneely, A., Williams, L., & Gehringer, E. F. (2008, August). ROSE: A Repository of Education-Friendly Open-Source Projects. *ACM SIGCSE Bulletin*, *40*(3), 7–11.

Morelli, R. A., & de Lanerolle, T. R. (2009, March). Foss 101: Engaging Introductory Students in the Open Source Movement. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)* (Vol. 41, pp. 311–315). New York, New York, USA: ACM Press.

Morelli, R. A., Ellis, H. J. C., de Lanerolle, T. R., Damon, J., & Walti, C. (2007). Can Student-Written Software Help Sustain Humanitarian FOSS? In *4th International Conference on Information Systems for Crisis Response and Management (ISCRAM)* (pp. 41–44). Delft.

Morelli, R. A., Tucker, A., Danner, N., de Lanerolle, T. R., Ellis, H. J. C., Izmirli, O., ... Parker, G. (2009, August). Revitalizing Computing Education through Free and Open Source Software for Humanity. *Communications of the ACM*, *52*(8), 67–75.

Nachbar, D. (1998, March). Bringing Real-World Software Development into The Classroom. In *Proceedings of The 29th Technical Symposium on Computer Science Education (SIGCSE '98)* (Vol. 30, pp. 171–175). New York, New York, USA: ACM Press.

Nandigam, J., Gudivada, V. N., & Hamou-Lhadj, A. (2008, October). Learning Software Engineering Principles Using Open Source Software. In *38th Annual Frontiers in Education Conference (FIE)* (pp. S3H–18–S3H–23). IEEE.

Nascimento, D. M. C., Chavez, C., Bittencourt, R. A., Cox, K., Almeida, T., Sampaio, W., & Souza, R. (2013). Using Open Source Projects in Software Engineering Education: A Systematic Mapping Study. In *43rd Annual Frontiers In Education Conference (FIE).* Oklahoma City.

O'Hara, K. J., & Kay, J. S. (2003, February). Open Source Software and Computer Science Education. *Journal of Computing Sciences in Colleges*, *18*(3), 1–7.

Papadopoulos, P. M., Stamelos, I. G., & Meiszner, A. (2012). Students' Perspectives on Learning Software Engineering with Open Source Projects: Lessons Learnt After Three Years of Program Operation. In *4th International Conference on Computer Supported Education CSEDU* (pp. 313–322).

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008, June). Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08)* (pp. 68–77).

Petrenko, M., Poshyvanyk, D., Rajlich, V., & Buchta, J. (2007, November). Teaching Software Evolution in Open Source. *Computer*, *40*(11), 25–31.

Prince, M. (2004). Does Active Learning Work? A Review of the Research. *Journal of Engineering Education*, *93*(3), 223–231.

Qian, K., & Fu, X. (2008, April). Teaching Component-Based Software Development. In *21st IEEE-CS Conference on Software Engineering Education and Training Workshop* (pp. 13–15). IEEE.

Raj, R., & Kazemian, F. (2006). Using Open Source Software in Computer Science Courses. In *36th Annual Frontiers in Education Conference (FIE)* (pp. 21–26). IEEE.

Rajlich, V. (2012). *Software Engineering: The Current Practice.* CRC Press.

Reichlmayr, T. J. (2006). Collaborating with Industry: Strategies for an Undergraduate Software Engineering Program. In *International Workshop on Summit on Software Engineering Education (SSEE'06)* (pp. 13–16). New York, NY, USA: ACM.

Robles, G., Caballe, S., & González-Barahona, J. M. (2008). Teaching Software Development in Community-Driven Software Projects- A Practical Experience. In *Free Knowledge Free Technology. The SELF Conference.* Barcelona.

Sabin, M. (2011, October). Free and Open Source Software Development of IT Systems. In *Proceedings of the Conference on Information Technology Education (SIGITE '11)*

(pp. 27–31). New York, New York, USA: ACM Press.

Samuelson, P. (2006). IBMs Pragmatic Embrace of Open Source. *Communications of the ACM*, 21–25.

Santore, J., Lorenzen, T., Creed, R., Murphy, D., & Orcutt, R. (2010, January). The Software Engineering Class Builds a GUI for Subversion. *ACM SIGCSE Bulletin*, *41*(4), 82–84.

Seiter, L. M. (2009). Computer Science and Service Learning: Empowering Nonprofit Organizations through Open Source Content Management Systems. In *Integrating FOSS into the Undergraduate Computing Curriculum, Free and Open Source Software (FOSS) Symposium.* Chattanooga.

Shockey, K., & Cabrera, P. J. (2005). Using Open Source to Enhance Learning. In *6th International Conference on Information Technology Based Higher Education and Training* (pp. F2A–7–F2A–12). IEEE.

Smrithi Rekha, V., Adinarayanan, V., Maherchandani, A., & Aswani, S. (2009). Bridging the Computer Science Skill Gap with Free and Open Source Software. In *International Conference on Engineering Education (ICEED)* (pp. 77–82).

Software Engineering Body of Knowledge (SWEBOK). (2013). Retrieved 04/02/2013, from `www.swebok.org/`

Sowe, S. K., & Stamelos, I. (2007). Involving Software Engineering Students in Open Source Software Projects: Experiences from a Pilot Study. *Journal of Information Systems Education (JISE)*, *18*(4), 425–435.

Sowe, S. K., Stamelos, I., & Deligiannis, I. (2006). A Framework for Teaching Software Testing using F/OSS Methodology. In *Open Source Systems* (Vol. 203, pp. 261–266).

Spinellis, D. (2006, September). Open Source and Professional Advancement. *IEEE Software*, *23*(5), 70–71.

Stroulia, E., Bauer, K., Craig, M., Reid, K., & Wilson, G. (2011, May). Teaching Distributed Software Engineering with UCOSP: the Undergraduate Capstone Open-Source Project. In *Proceeding of The Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD '11)* (pp. 20–25). New York, New York, USA: ACM Press.

Tao, Y., & Nandigam, J. (2006). Work in Progress: Open Source Software as the Basis of Developing Software Design Case Studies. In *36th Annual Frontiers in Education Conference (FIE)* (pp. 27–28). IEEE.

Toth, K. (2006, November). Experiences with Open Source Software Engineering Tools. *IEEE Software*, *23*(6), 44–52.

Tucker, A., Morelli, R. A., & de Lanerolle, T. R. (2011, October). The Humanitarian FOSS Project: Goals, Activities, and Outcomes. In *IEEE Global Humanitarian Technology Conference* (pp. 98–101). IEEE.

Whitehurst, J. (2009). Open Source: Narrowing the Divide between Education Business and Community. *EDUCAUSE Review*, *44*(1), 70–71.

Williams, L., & Shin, Y. (2006). Work in Progress: Exploring Security and Privacy Concepts through the Development and Testing of the iTrust Medical Records System. In *36th Annual Frontiers in Education Conference (FIE)* (pp. 30–31). IEEE.

Xing, G. (2010, April). Teaching Software Engineering Using Open Source Software. In *Proceedings of the 48th Annual Southeast Regional Conference on (ACM SE '10).* New York, New York, USA: ACM Press.

Yamakami, T. (2012). Re-engineering Software Education: OSS-aware Software Education in the Era of Utilizing External Resources. *International Conference on Advanced Communication Technology, ICACT*.

Zhang, H., & Su, H. (2007, July). A Collaborative System for Software Engineering Education. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)* (Vol. 2, pp. 313–318). IEEE.