A Comprehensive Review and Synthesis of Open Source Research

Altay Aksulu
Schulich School of Business, York University, aaksulu@schulich.yorku.ca

Michael R. Wade
Schulich School of Business, York University, mwade@schulich.yorku.ca

Follow this and additional works at: https://aisel.aisnet.org/jais

Recommended Citation
DOI: 10.17705/1jais.00245
Available at: https://aisel.aisnet.org/jais/vol11/iss11/6

This material is brought to you by the AIS Journals at AIS Electronic Library (AiSeL). It has been accepted for inclusion in Journal of the Association for Information Systems by an authorized administrator of AIS Electronic Library (AiSeL). For more information, please contact elibrary@aisnet.org.
The open source movement has grown steadily and matured in recent years, and this growth has been mirrored by a rise in open source related research. The objective of this paper is to pause and reflect on the state of the field. We start by conducting a comprehensive literature review of open source research, and organize the resulting 618 peer-reviewed articles into a taxonomy. Elements of this taxonomy are defined and described. We then draw on a number of existing categorization schemes to develop a framework to situate open source research within a wider nomological network. Building on concepts from systems theory, we propose a holistic framework of open source research. This framework incorporates current research, as represented by the taxonomy, identifies gaps and areas of overlap, and charts a path for future work.

**Keywords:** Open Source Software (OSS), Open Source Research (OSR), OSR Literature, OSR Taxonomies, OSR Frameworks.
A Comprehensive Review and Synthesis of Open Source Research

1. Introduction

While many of the concepts behind the open source movement, like peer production, shared code, and software as a public good have been around since the beginning of the computing era, use of the term “open source” is a relatively recent phenomenon (see Weber, 2004 for a chronological history). According to most accounts, the term was first used in February 1998 as a consequence of the decision by Netscape to allow free access to its browser’s source code (Raymond, 2001a; Searls, 2005). The term was subsequently appropriated by the Open Source Initiative, in part to counter the lingering misconceptions about the Free Software Foundation.¹ While there is no single definitive definition of open source, it is generally accepted to mean software that allows for the modification of source code, is freely distributed, is technologically neutral, and grants free subsidiary licensing rights (Perens, 1999). In many respects, the term open source is seen as a certification mark (Neumann, 1999) requiring free, independent, and indiscriminate redistribution of software, source code and reuse licenses.² However, the open source phenomenon has come to represent more than merely a process for managing software development projects. Indeed, the term has been used to describe a wide range of collaborative ventures well beyond software development (Butcher, 2009; Pénin and Wack, 2008; Stanford and Mikula, 2008; Watson et al., 2008b; Hutchinson, 2008; Pitt et al., 2006; Shah, 2005). As with any new and fast developing topic area, it is useful to pause and reflect on the progress it has made. The purpose of this paper is to present a comprehensive taxonomy of existing open source research, to develop a framework to organize this research, and to chart a course for future work.

We believe that open source research (OSR) has advanced considerably over the last decade and despite the field’s ever-changing nature, a snapshot to determine the “state of the field” can add substantial value. With many top-tier journals publishing OSR, a large body of knowledge has already been accumulated (see Crowston et al., 2010 for a recent review). We believe that the time is right to examine the extent of this research and organize it into a taxonomy. A research taxonomy can provide several potential benefits. First, it can provide an organized repository of the existing literature. This repository allows interested researchers to quickly and efficiently zoom in on particular articles or subcategories. Second, it can present a clear picture of the breadth and depth of an emerging field. This picture can help to identify research trends as well as any potential gaps in the literature. Finally, a taxonomy can provide a stepping stone towards the establishment of a comprehensive theory or framework, one that can help to explain and predict current trends and guide future research. As Vogel and Wetherbe (1984) put it, “taxonomies help to focus research, clarify representation in the literature, define standards, and spot trends or gaps in the research. By categorizing research efforts, taxonomies help provide a measure of order that would go wanting in their absence.” (Vogel and Wetherbe, 1984, p. 4).

2. Building a taxonomy of open source research

Our first objective was to conduct an extensive search of open source research across different academic areas. Building on Vogel and Wetherbe (1984), we sought to ensure that our taxonomy was comprehensive, parsimonious and useful. These three standards have been used to evaluate previous taxonomy schemes, including information systems research (Järvinen, 2000), operations research education (Reisman, 2004), and supply chain management (Capar et al., 2004).

In order to be comprehensive, a taxonomy should cover the breadth and depth of a field’s high quality work in an unbiased fashion. To be parsimonious, a taxonomy must avoid unnecessary categories. Nonetheless, in practice this goal may not be easy to achieve since there exists a notable tension

---

¹ Free Software Foundation is a United States-based donor supported charity founded in 1985 to promote computer user freedom and to defend the rights of all free software users. More information can be found at http://www.fsf.org/.
² The complete criteria can be found at http://www.opensource.org/docs/osd. For the purposes of our research, any software project that cannot be classified into one of more than sixty open source licenses listed by the open source initiative (http://www.opensource.org/licenses) is considered to be proprietary software.
between comprehensiveness and parsimony. The more comprehensive a taxonomy becomes, the more compromises have to be made around parsimony. Thus, a delicate balance has to be maintained to provide meaningful and reasonably distinct categories. Finally, to be useful, taxonomy categories should be both near mutually exclusive and collectively exhaustive, and offer reasonably distinct and meaningful categorizations of the literature to prevent confusion and to facilitate widespread use.

We are aware of two existing OSR bibliographies. These resources provide lists of open source research, but do not attempt to organize the resulting work into a taxonomy. The first bibliography is maintained by Joseph Keller and appears on the University College Cork website. This resource includes 316 entries and is current up to the end of 2006. The second source, the Massachusetts Institute of Technology's (MIT) Free/Open Source Community lists and hosts 227 OSR studies, most of which are working papers. This resource provides coverage up to the beginning of 2009. A third resource, hosted by the Vienna University of Economics and Business Administration and maintained by Stefan Koch, is no longer available.

While we were not able to identify a published taxonomy of OSR during our initial review, a recent literature review (presently in press) focusing on empirical OSS papers has been provided by Crowston et al. (2010). This study relied on papers that were collected in two waves. In the first wave, paper collection involved the above-mentioned MIT repository, special journal issues, conference tracks, document databases and key article reference lists whereas the second wave focused on peer-reviewed journals. Wave one produced 586 papers, less than half of which were journal articles, and only 138 were considered to be empirical. Wave two resulted in an additional 55 empirical articles from peer-reviewed journals, bringing the total number of empirical articles that were reviewed to 193.

2.1. Research methodology

Our data collection procedure was unbiased to the extent that it was not guided by an a priori framework (although a number of such frameworks exist as we will demonstrate later), nor was it restricted to certain academic disciplines or journals. We wanted to conduct a thorough search of OSR and then organize the resulting data into a taxonomy. It was not our intent at this stage to normatively determine gaps or areas of duplication, or conduct any forward-looking analysis. We merely intended to provide a snapshot of the current state of OSR. In the sections below, we briefly elaborate on the guiding principles that shaped the data collection and analysis processes. In particular, the data collection section talks about the selection of our data sources, assessment of their comprehensiveness, as well as the criteria for inclusion and exclusion. The data analysis section provides detailed information on our coding and refinement processes, including how codes were affixed to articles as well as the identification of common themes and patterns.

2.1.1. Data collection

Our preliminary mode of data gathering involved examining (Wolcott, 1994) major research archives. We selected ProQuest as our primary research portal. As the world’s largest repository of academic research, ProQuest provides archives of sources such as newspapers, periodicals, dissertations, and aggregated databases of many types. Its content is estimated at 125 billion digital pages and is accessible through most university library Internet gateways. Our original intention was to capture academic research on the open source phenomenon including, but not limited to, OSS. Nonetheless, our initial search efforts using such terms as “open phenomenon” or “open paradigm” uncovered few results. We therefore had to rely on commonly used keywords like “open source” and “open systems” that may have biased the results towards a software focus.

Using ProQuest’s advanced search functionality, we created a query that searched for documents where the subject included the terms “open source software” or “open systems” and the document abstract had the term “open source.” We selected all available databases under the multiple databases option, which includes ABI/INFORM Global, ABI/INFORM Trade & Industry, ProQuest European/Asian Business, and ProQuest Research Library, and did not limit the search to a specific time period. We only limited search results to documents published in English and where publication
type was scholarly journals. As of June 9, 2009 this initial search returned 524 articles that appeared in more than 190 journals. Each document was captured and, in aggregate, constituted the core raw material for our analysis.

Despite the comparatively high number of journals covered, the ProQuest multiple database archives appeared to have omitted a few key journals that frequently publish OSR. In an effort to provide additional coverage and develop a more thorough sample, we used a randomly selected sample of OSR articles and conducted a backward and forward citation analysis (Webster and Watson, 2002). For the backward tracking, we looked at the references provided in the sample set and searched for journal omissions. For the forward tracking, we used Google Scholar to identify articles that cited the sample set and investigated the journal coverage. Through this quality check, we were able to identify a few additional journals that were omitted from the initial pool. For example, the addition of one such journal, First Monday, resulted in around 70 additional OSR articles. Following this exercise, we were able to expand the data pool to 618 articles. This number is substantially larger than previous OSR bibliographies, and comparable to reviews and bibliographies in related fields.

2.1.2. Data analysis – coding, refining, patterning

Our main aim in analyzing the data was to discover the underlying concepts and look for overarching themes to organize these into a taxonomy. To accomplish this goal, we selectively borrowed techniques from qualitative analysis methodology and followed a systematic process of interpretation and analysis. This process involved conceptualization and reduction of data, elaboration of categorical properties and dimensions, and semantic filtering through prepositional statements. We followed a multistage content analysis process (Glasser and Strauss, 1967; Miles and Huberman, 1984) that is appropriate for a research area at the ad-hoc classification level (Webster and Watson, 2002). The particular process we followed is depicted in Figure 1. We believe that OSR can benefit from a taxonomy in advance of any serious grounded theory efforts. We use inductive analysis to mean that the descriptive codes, interpreted categories and patterns came from the data rather than being imposed by theory prior to data collection and analysis (Patton, 1980).

In order not to bias the results, we did not begin the coding process with a pre-defined list of categories, instead allowing the categories to emerge and evolve throughout the analysis (Glasser and Strauss, 1967). We felt that this emergent process would be more suited to our intended approach.

Stage 1A – Descriptive coding

This multistage and highly iterative process started with one researcher reviewing the entire data pool. In an effort to take advantage of the more specialized nature of newer articles, the articles were reviewed in reverse-chronological order, starting with the most recent and moving backwards in time. Each paper’s abstract was reviewed first, before the paper was scanned. When scans were not sufficient in giving clues about the topical content, papers had to be read in their entirety. In the first pass, our aim was to highlight recurring content classes and themes. As each paper was reviewed, we asked ourselves “What is this paper about?” and “What are the fundamental concept or concepts...
that are being addressed in this paper?”

This early process involved little or no interpretation as we simply affixed descriptive labels to papers such as “Developer Motivations,” “OSS Licensing,” or “OSS Reliability.” Nonetheless, instead of linking these descriptive codes to small segments of text, we attached them to complete studies focusing on the totality of the paper as opposed to line-by-line or segment-by-segment coding. Several factors, including the title of the study, the publishing journal, the ProQuest abstract, the abstract of the original article (if different from the ProQuest abstract or when the article originated from a supplemental source), as well as the primary author’s other known work, affected what code or codes were assigned to each article.

Stage 1B – Interpretive coding

The coding process was highly iterative. When existing codes were expanded or merged, this resulted in a re-categorization of all previous articles where the same codes were utilized. At this point, we ventured beyond descriptive codes into interpretive coding. For example, while we originally had different categories such as “Collaboration,” “Knowledge Sharing,” and “Code Reuse,” later on we noticed that all these codes were conceptually similar and generally appeared together. We then went back to all articles where those codes were used to check the validity of this assumption at its source. Following this process, we created a “Collaboration and Knowledge Sharing” code to accommodate them (the higher level pattern was not created until the next stage). Similarly, it was not until such descriptive codes as “OSS Reliability,” “OSS Usability,” “OSS Maintainability,” and “OSS Performance” were in place that we noticed they were different aspects of an overall quality discussion, and we created the “Software Quality” code after reassessing all those articles to make sure they fit into this unifying view.

In some cases, we created additional codes. For example, under the “Software Quality” umbrella, it became clear to us that articles addressing the topics of software testing and security were conceptually different enough to deserve their own labels but not to deserve a separate high level code. We therefore created the “Testing and Bug Fixes” and the “OSS Security” sub-categories under the “Software Quality” code. Due to the emergent and descriptive nature of this iterative process, the pace of new code generation was very high in the beginning but slowed down as the existing code base became larger. Saturation was achieved about half way through the coding process, after which point few new codes were required. A total of 111 labels were identified after one pass through the complete article set.

Stage 2 – Pattern coding

In the next stage, we organized the existing labels into emergent patterns. During this stage, we were able to identify seven explanatory patterns. These patterns are: conceptual; performance metrics; legal and regulatory; OSS production; OSS applications; OSS diffusion; and beyond software (a description of each category is provided in Appendix A). The process of assigning codes to patterns was highly iterative. For example, the “Legal and Regulatory” pattern code grouped such labels as “OSS Licensing,” which investigated and compared various OSS license types; “OSS Intellectual Property Rights,” which looked into diverse intellectual property issues such as patents, trademarks and copyrights; “OSS Legal Issues,” which assessed legal issues and risks concerning OSS in general; as well as “OSS Standards and Regulation,” which took a closer look into the interactions between standards and regulation as well as their impact on OSS community evolution and diffusion.

At this stage, using the higher level of abstraction provided by the pattern codes, we were also able to further consolidate the labels. For instance, creation of the “OSS Applications” pattern, which grouped articles that were written on discipline or area-specific OSS applications, allowed us to bundle a number of existing labels. Hence, we were able to combine such labels as “Information Management” and “Knowledge Management,” “Imaging” and “Plotting,” “Planning” and “Optimization,” as well as “Programming Languages,” “Scripting Languages,” “Modelling Languages” and “Markup Languages” to create natural groupings that were not as obvious prior to this stage. Once all similar labels had been identified and merged, the total number of descriptive codes was reduced to 88 and each code
belonged to one of the 7 pattern codes.

**Stage 3 – Check coding**

In the final stage, the codes and patterns were independently assessed by two naive coders. Two doctoral students were trained in the coding framework (Miles and Huberman, 1994) and then given identical samples of raw data in the form of all article abstracts and were also provided with a one-page summary sheet that included the 88 descriptive codes and the 7 patterns. In other words, the emergent coding scheme that was the result of the preceding analysis stages became the provisional “start list” of codes for the co-analysts.

The coders were then asked to go through the raw data and decide which code or codes best described the content of each article by asking themselves the same questions that guided our earlier analysis: “What is this paper about?” and “What are the fundamental concept or concepts that are being addressed in this paper?” The co-analysts were told they were not restricted to the start list of codes and if they felt an article required a new code or a number of new codes they were free to expand the list. We also asked them to report on whether they thought patterns grouped similar codes (to validate internal consistency) and whether each pattern was reasonably distinct (to validate discreteness). During the validation process, we refrained from giving any feedback to co-analysts in order not to bias the results. Once all the raw data were coded, we grouped the results in a spreadsheet and calculated intercoder reliability (Miles and Huberman, 1994). The initial reliability figures for the two coders were 68 percent and 69 percent respectively. These results were very close to Miles and Huberman’s upper limit of 70 percent for pre-resolution reliability benchmark.

We then followed a structured approach to resolve differences. We organized codes under their category patterns and for each pattern we compared the assigned codes. In cases where there were disagreements between the original coding scheme and the two validity checks, we followed a majority rules approach. In a small number of cases, none of the coders were in agreement. Each of these cases was resolved by re-checking the codes against the original subject article(s). On occasions when an additional code (or codes) provided better content clarity of the article, we added that code (or codes) to the list of codes associated with the article. During the resolution process, we also had an opportunity to further refine the codes by combining those that were not sufficiently distinct as manifested by repeated disagreements caused by the same set of codes (suggesting they were confusingly similar). For example, we merged the “content management” and “information and knowledge management” codes, and combined “Programming, scripting, modeling and markup languages,” “Object oriented software,” and “Integrated development environments” codes under the “Software Development and Engineering” code, among many others. As a result of the validation process, we were able to reduce the number of codes from 88 to 57, which are reflected in Table 1. The table shows the groupings and the categories, along with the number of instances within each category. The total number of counts or instances was 1,355, which is greater than the total number of articles, 618. This difference is accounted for by the fact that many papers contained topics that fell into multiple categories. A detailed description of each pattern and category along with the complete list of articles included within the category is provided in Appendix A.
Table 1: A Taxonomy of Open Source Research

<table>
<thead>
<tr>
<th>PATTERN</th>
<th>CODE</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCEPTUAL</td>
<td>OSS DESCRIPTIVE</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>OSS BENEFITS/DRAWBACKS</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>OSS VISION/ROADMAP</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>OSS RESEARCH CATEGORIZATION / RESEARCH AGENDA</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>OSS VERSUS PROPRIETARY</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>BUSINESS/ECONOMIC MODELS&amp;STRATEGIES/POLICIES FOR OSS</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>PATTERN SUB-CATEGORY TOTAL</td>
<td>216</td>
</tr>
<tr>
<td>PERFORMANCE METRICS</td>
<td>SOFTWARE QUALITY</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE QUALITY – TESTING and BUG FIXES</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE QUALITY – OSS SECURITY</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE DEVELOPMENT – OSS CODE EFFICIENCIES</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>DEVELOPMENT TEAM PERFORMANCE</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>OSS SUCCESS</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>PATTERN SUB-CATEGORY TOTAL</td>
<td>137</td>
</tr>
<tr>
<td>LEGAL AND REGULATORY</td>
<td>OSS LICENSING</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>OSS INTELLECTUAL PROPERTY RIGHTS</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>OSS LEGAL ISSUES</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>OSS STANDARDS AND REGULATION</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>PATTERN SUB-CATEGORY TOTAL</td>
<td>162</td>
</tr>
<tr>
<td>OSS PRODUCTION</td>
<td>PROCESS</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>COMMUNITIES</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>TEAM FORMATION</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>GOVERNANCE</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>TEAM/PROJECT LEADERSHIP</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>INDIVIDUAL AND TEAM LEARNING</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>INNOVATION</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>ROLE OF VOLUNTEER USERS / DEVELOPERS</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>COLLABORATION AND KNOWLEDGE SHARING</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>USER AND DEVELOPER MOTIVATIONS</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>ROLE OF COMMERCIAL CORPORATIONS</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – ROLE OF LICENSING AND IP</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>SELF-ORGANIZATION (PRODUCT AND COMMUNITY EVOLUTION)</td>
<td>20</td>
</tr>
</tbody>
</table>
### Table 1: A Taxonomy of Open Source Research

<table>
<thead>
<tr>
<th>OSS APPLICATIONS</th>
<th>PATTERN SUB-CATEGORY TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TELECOMMUNICATIONS, NETWORKING AND ARCHITECTURE</td>
<td>11</td>
</tr>
<tr>
<td>EDUCATION</td>
<td>37</td>
</tr>
<tr>
<td>LIBRARIES, ARCHIVES, DATABASES AND REPOSITORIES</td>
<td>56</td>
</tr>
<tr>
<td>CONTENT, INFORMATION &amp; KNOWLEDGE MANAGEMENT SYSTEMS</td>
<td>32</td>
</tr>
<tr>
<td>IMAGING, PLOTTING AND VISUAL</td>
<td>10</td>
</tr>
<tr>
<td>SECURITY AND CYBERCRIME</td>
<td>8</td>
</tr>
<tr>
<td>SUPPLY CHAIN MANAGEMENT AND OPTIMIZATION</td>
<td>9</td>
</tr>
<tr>
<td>DESKTOP AND SERVER OPERATING SYSTEMS</td>
<td>27</td>
</tr>
<tr>
<td>GAMING AND SIMULATIONS</td>
<td>5</td>
</tr>
<tr>
<td>SOFTWARE DEVELOPMENT AND ENGINEERING</td>
<td>34</td>
</tr>
<tr>
<td>ACADEMIC AND COMMERCIAL RESEARCH</td>
<td>22</td>
</tr>
<tr>
<td>BIOMEDICAL AND HEALTH SCIENCES</td>
<td>25</td>
</tr>
<tr>
<td>BUSINESS, PROFESSIONAL AND SOCIAL SCIENCES</td>
<td>9</td>
</tr>
<tr>
<td>NATURAL SCIENCES</td>
<td>10</td>
</tr>
<tr>
<td>PUBLIC SECTOR AND E-GOVERNMENT</td>
<td>9</td>
</tr>
<tr>
<td>PATTERN SUB-CATEGORY TOTAL</td>
<td>338</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OSS DIFFUSION</th>
<th>PATTERN SUB-CATEGORY TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSS ADOPTION – GENERAL</td>
<td>16</td>
</tr>
<tr>
<td>OSS ADOPTION – BARRIERS</td>
<td>13</td>
</tr>
<tr>
<td>OSS ADOPTION – DECISION FACTORS</td>
<td>15</td>
</tr>
<tr>
<td>OSS IMPLEMENTATION – GENERAL</td>
<td>14</td>
</tr>
<tr>
<td>OSS IMPLEMENTATION – IMPLEMENTATION COMMUNITIES AND NETWORKS</td>
<td>8</td>
</tr>
<tr>
<td>OSS IMPLEMENTATION – GOVERNMENTS / NATIONS</td>
<td>22</td>
</tr>
<tr>
<td>PATTERN SUB-CATEGORY TOTAL</td>
<td>304</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BEYOND SOFTWARE</th>
<th>PATTERN SUB-CATEGORY TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN PARADIGM</td>
<td>31</td>
</tr>
<tr>
<td>OPEN INNOVATION</td>
<td>28</td>
</tr>
<tr>
<td>OPEN KNOWLEDGE FLOWS</td>
<td>25</td>
</tr>
<tr>
<td>OPEN STANDARDS</td>
<td>8</td>
</tr>
<tr>
<td>OPEN EDUCATION</td>
<td>12</td>
</tr>
<tr>
<td>USER OR CO-PRODUCTION OF GOODS AND CONSUMER IMPLICATIONS</td>
<td>6</td>
</tr>
<tr>
<td>PATTERN SUB-CATEGORY TOTAL</td>
<td>88</td>
</tr>
</tbody>
</table>

| GRAND TOTAL | 1355 |

### 2.2. Evolution of open source research

During the coding process, we discovered some insights into the development of the field. For instance, we noticed that much of the early OSR was exploratory in nature. While some studies
provided a general overview in a largely descriptive manner (O’Reilly, 1999; Blau, 1999; Cass, 2001; Brethauer, 2002, Krishnamurthy; 2003), other studies looked at various isolated topics ranging from potential benefits or advantages of open source software (Kogut and Metiu, 2001; Torvalds and Diamond, 2001; Fuggetta, 2003; Ringle, 2004) compared to proprietary software (Neumann, 1999; Raymond, 2001b; West, 2003; Fuggetta, 2003; Paulson et al., 2004), to quality (Bollinger et al., 1999; Neumann, 1999; McConnell, 1999; Zhao and Elbaum, 2003; Huntley; 2003; Samoladas et al., 2004), and to lessons and strategies for traditional enterprises (O’Reilly, 1999; Ousterhout, 1999; Gannon, 2000; West, 2003; Hawkins, 2004).

As the field started to mature, specific knowledge islands started to emerge. Certain topic areas began to garner a substantial amount of attention and interest within the field, most notably licensing (Välimäki and Borsalino, 2003; de Laat, 2004; Wacha, 2005; Gandel and Wheeler, 2005; Lerner, 2005; Carver, 2005; Gambardella and Hall, 2006), developer motivations (Hertel et al., 2003; Bonaccorsi and Rossi, 2003; Zeitlyn, 2003; Bitzer and Schröder, 2005; Bagozzi and Dholakia, 2006; Shah, 2006), open innovation (Kogut and Metiu, 2001; von Hippel, 2001; von Krogh et al. 2003; von Hippel et al., 2003; Grand et al., 2004), and open source governance (Franck and Jungwirth, 2003; Bonaccorsi and Rossi, 2003; Demil and Lecocq, 2006; Shah, 2006). As can be seen in the taxonomy, these areas constitute some of the more popular categories of OSR.

While it is useful to assess the progress of a field’s development, a research taxonomy is silent about whether or not the extant research is appropriate or productive. The taxonomy tells us only where the field is, not where it should be, or where it needs to go next. It can perhaps provide an indication of areas of overlap, but it says little about gaps. The categories may be mutually exclusive or near exclusive, but there is no way to know if they are appropriate, or exhaustive. To address these normative issues, it is necessary to develop a framework or roadmap of where the field should go, and then evaluate the taxonomy in light of this framework. A number of organizing frameworks for open source research have been put forward in the literature. In the following section, we will critically evaluate these frameworks in light of the taxonomy.

3. Open source research frameworks

We identified nine attempts to provide OSR frameworks or research agendas in the literature aiming to define and direct future research efforts. They are: Feller and Fitzgerald (2000), Lerner and Tirole (2001 and 2005), Rossi (2004), Nelson et al. (2006), Niederman et al. (2006a,b), von Krogh and von Hippel (2006), Scacchi (2007) and Jin et al. (2007). While the objectives of these papers varied, to some degree they all sought to categorize extant research and propose an agenda for OSR. In Appendix B, we have interpreted each of these classification schemes in light of the taxonomy. This cross-mapping exercise involved in-depth evaluation of reference articles to match the paper’s categories with the taxonomy codes based on detailed content analysis (Holsti, 1969; Krippendorff, 1980).

A summary of this analysis is presented in Table 2. The table shows that existing frameworks and reviews have provided a strong and valuable level of insight on particular aspects of the field and addressed certain niche areas within OSR well. The table also shows that the previously proposed research agendas helped with the progress of OSR as they successfully guided the research efforts in the field. The research guidance provided by these agendas likely has encouraged and directed research efforts and resulted in actual research, as exemplified by the diverse content categories of the proposed OSR taxonomy. However, Table 2 makes it clear that none of these past categorization schemes covered the totality of the field. To be fair, most of these papers were written with a different purpose in mind. Nevertheless, and with the exception of Crowston et al. (2010) which focuses on empirical OSS papers, we are not aware of any published work that attempts to capture the depth and breadth of open source research in a taxonomy, nor one that offers an over-encompassing research framework to guide future research efforts.

While it could be argued that the taxonomy presented in this paper is comprehensive, it does not have the prescriptive or explanatory quality of the frameworks. It also lacks the level of parsimony...
required to make a framework useful on a practical level. Thus, in the following section, we propose a holistic framework to organize and guide open source research.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCEPTUAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS DESCRIPTIVE</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS BENEFITS/DRAWBACKS</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS VISION/ROADMAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS RESEARCH CATEGORIZATION / RESEARCH AGENDA</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>OSS VERSUS PROPRIETARY</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>PERFORMANCE METRICS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFTWARE QUALITY</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFTWARE QUALITY – TESTING and BUG FIXES</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFTWARE QUALITY – OSS SECURITY</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFTWARE DEVELOPMENT – OSS CODE EFFICIENCIES</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>DEVELOPMENT TEAM PERFORMANCE</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>OSS SUCCESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>LEGAL AND REGULATORY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS LICENSING</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>OSS INTELLECTUAL PROPERTY RIGHTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>OSS LEGAL ISSUES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>OSS STANDARDS AND REGULATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>OSS PRODUCTION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>PROCESS</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>COMMUNITIES</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>TEAM FORMATION</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td>±</td>
<td>±</td>
</tr>
</tbody>
</table>
### Table 2: Taxonomy Codes And Previous Research Categorizations

<table>
<thead>
<tr>
<th>Category</th>
<th>√</th>
<th>√</th>
<th>√</th>
<th>√</th>
<th>√</th>
<th>√</th>
<th>√</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOVERNANCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEAM/PROJECT LEADERSHIP</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDIVIDUAL AND TEAM LEARNING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>INNOVATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>ROLE OF VOLUNTEER USERS / DEVELOPERS</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>COLLABORATION&amp;KNOWLEDGE SHARING</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>USER AND DEVELOPER MOTIVATIONS</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>ROLE OF COMMERCIAL CORPORATIONS</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROLE OF LICENSING AND IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>SELF-ORGANIZATION (PRODUCT AND COMMUNITY EVOLUTION)</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>APP OSS APPLICATIONS – (all sub-categories)</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS DIFFUSION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS ADOPTION – GENERAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS ADOPTION – BARRIERS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS ADOPTION – DECISION FACTORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS IMPLEMENTATION – GENERAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS IMPLEMENTATION-IMPLEMENTATION COMMUNITIES AND NETWORKS</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>OSS IMPLEMENTATION – GOVERNMENTS / NATIONS</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>BEYOND SOFTWARE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN PARADIGM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN INNOVATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN KNOWLEDGE FLOWS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>OPEN STANDARDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN EDUCATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>USER OR CO-PRODUCTION OF GOODS AND CONSUMER IMPLICATIONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. Proprietary and open source systems through the lens of systems theory

Over the last decade, the term “open source systems” has become widely used in both academia and industry. Today, research on OSS commonly appears in respected journals, and has started to make its way into the college and university curricula. Furthermore, demand for OSS practitioners is on the rise. As our taxonomy has shown, open source research bridges many different topics and disciplines. Our literature review covered 618 papers published in more than 190 journals across multiple academic disciplines. Hence, we believe the time is right to establish a comprehensive framework to define and direct future research efforts in what has become an increasingly diversified field of inquiry.

In order to build an appropriate guiding framework, we turned to Nolan and Wetherbe who faced a similar task in the late 1970s when they set out to categorize the emerging management information systems field (Nolan and Wetherbe, 1980). They decided to draw on systems theory, primarily for its ability to cope with high levels of ambiguity and complexity. We believe that systems theory represents a valid starting point in tackling open source research. In this section, we start with a brief introduction of systems theory before presenting a stylized view of proprietary and open systems using this theoretical lens. Considering a variety of possible interactions among these cases and using a hybrid systems approach that makes up most real world systems, we will then propose a holistic framework for OSR.

4.1. Systems Theory

Systems theory was proposed in the 1930s by Ludwig von Bertalanffy (1934, 1950, 1962, 1968, 1972) as a counterpoint to the dominant reductionist approach of the day. Instead of considering entities as being made up of parts that can be analyzed and interpreted in isolation, systems theory argues that entities should be seen holistically in relation to interactions among the parts and their relationship within the whole. Systems theory attempts to incorporate the complexity inherent in most relationships in such a way that the whole can be greater than the sum of the parts. The theory has been developed and extended over time by several prominent scholars, such as Kenneth Ewart Boulding (1956), William Ross Ashby (1947, 1962), Charles West Churchman (1968), and Fremont Ellsworth Kast and James E. Rosenzweig (1972).

According to systems theory, all systems, whether they are mechanical, biological, or social, are composed of interrelated parts or elements that are referred to as subsystems or components (Kast and Rosenzweig, 1972, p. 450). Thus, a system can be defined as “a set of elements standing in interrelation among themselves and with the environment.” (von Bertalanffy, 1972 p. 417). Whereas systems are made up of lower-order subsystems, they themselves are parts of higher-order “supra-systems” in a hierarchy.

Systems can be considered as either open or closed. Open systems have permeable boundaries allowing them to exchange information, energy, or material with their environments. Such exchanges are restricted in closed systems that tend to have impermeable or barely permeable boundaries (Kast and Rosenzweig, 1972, p. 450). Von Bertalanffy defined open systems as, “…systems exchanging matter with the environment as every ‘living’ system does” (von Bertalanffy, 1972, p. 412). This definition makes it possible to view systems as dynamic transformational models taking various inputs from their environment and transforming them into outputs, that are, in turn, transported back into the environment. Systems theory posits that closed systems are subject to positive entropy changes over time, resulting in eventual failure due to lack of resource transformation. The theory views open systems as capable of positive as well as negative entropy changes, making it possible for the system to attain a dynamic equilibrium or steady state, raising the chances of survival.

4.2. Proprietary and Open Source Systems

In systems theory terminology, both proprietary and open source systems can be viewed as open systems. These systems, which may also be conceptualized as different ends of a software development continuum, take multiple inputs from the environment, transform them into outputs, and...
reintegrate those outputs into the environment. In the case of software development projects, the inputs include human effort, process know-how, and information technologies (IT). These inputs become part of a sub-system with the purpose of creating outputs such as human development and growth, elevated process knowledge, and software applications, all under the influence of the surrounding environment. This multi-dimensional transformation relies on effective interactions among the people, processes and the technology sub-system, set within a larger organizational system context consisting of management issues, development methodologies, project activity planning, scheduling, process standards and metrics, resource management, continuous learning, and a host of other contingent variables (Nambisan and Wilemon, 2000). 3

When we talk about people in the context of proprietary and open source systems, we are referring to the commitment of human participants in a project, including all explicit and tacit knowledge applied within the project’s context. Processes are written descriptions of explicit knowledge. They may represent strategies, policies, procedures, rules and regulations, as well as standards. Finally, technology refers to a product or a module of a product such as a system component or a piece of software.

4.2.1. Proprietary systems

At one end of the continuum lie proprietary systems. While proprietary systems are open systems, since they interact with the external environment, the majority of the activity happens within the boundaries of the system, which in turn exists within the boundaries of the firm. This occurs mainly because proprietary systems are designed for a specific, well-defined purpose, such as to produce a marketable technology product that is then sold on the open market. While people and process inputs are measured and tracked in most proprietary system environments, the same cannot be said for outputs, which are rarely monitored (Mayo, 2001). The main purpose of a proprietary system is to create a technology output, and all other outputs – like enhanced technology skills of the developers or new processes – while desirable, are subsidiary to the technology output. Figure 2 shows a graphical representation of a proprietary system environment.

Figure 2: Proprietary System

3 While Nambisan and Wilemon (2000) define people, processes and technology as dimensions, we prefer the term ‘subsystem’ to maintain a link to systems theory.
There are several characteristics of proprietary systems that are worthy of note. These systems tend to be externally defined, have artificial lifespans, operate largely within organizational environments, and are tied to predefined constraints, such as limitations in scope, time, and cost (Brooks, 1995; Royce, 1998; Bittner and Spence, 2006). Once they outlive their design purpose, for example when a system’s technology outputs no longer produce sufficient financial rewards, proprietary systems are often terminated, and their subsystems discarded or reallocated (Seacord et al., 2003).

The Interaction between a proprietary system and its environment is relatively limited (Alexy and Henkel, 2007 and 2009). Indeed, its immediate environment is defined by organizational boundaries. Such organizational boundaries are generally fixed and rigidly defined. However, as open systems, proprietary systems permit both negative and positive feedback loops, although the timing and frequency of these recursive factors is often predetermined (Lehman et al., 1996). In the case of positive feedback, this usually corresponds to major output milestones and is transmitted intermittently; whereas negative feedback is generally tied to deviations from a prescribed project trajectory (Bittner and Spence, 2006).

Proprietary transformation subsystems (people, processes and technology) are purposefully held together for the duration of a project. Thus, even under suboptimal conditions, the transformation subsystem will be required to produce a technology output through external organizational pressure.

Finally, the adoption of technology outputs of a proprietary system typically occurs outside the organizational boundaries. Since neither people nor process outputs are as closely measured or tracked, any possible adoptions within the organization would be very hard to measure (Battisti and Stoneman, 2005).

4.2.2. Open source systems

At the other end of the continuum lie open source systems. Unlike proprietary systems that are designed for well-defined purposes, the objectives of open source systems are often loosely defined and allow contributors great freedom to work on areas they find interesting (Lerner and Tirole, 2005). While there may be specifically stated objectives at the outset, these systems tend to mature over time in an unpredictable manner, reflecting and representing the changing objectives of system participants (Nakakoji et al. 2002; Lee and Davis, 2003; Shah, 2006). As a direct consequence of this change, open source systems are not fixated on technology outputs only, and typically produce people and process outputs with equal emphasis. For example, it is not unusual for a contributor within an open source project to be motivated by personal development potential, even if there is no tangible code contribution.

Open source systems also differ from proprietary systems in that they typically do not have an externally defined lifespan. Rather, an open source system survives as long as at least one actor/person is able and willing to maintain it (Shah, 2006). In addition to flexible lifespans, open source systems tend to have permeable boundaries that allow for rapid expansion or contraction. Whereas proprietary systems operate primarily within one organizational environment, open source systems operate under multiple environmental layers, allowing for numerous opportunities for multi-level interactions between the system and the surrounding environment. We have graphically represented an open source system in Figure 3. The inner layer concerns the core system where the bulk of the technology transformation occurs. Within this layer, the human component includes a core team of user-developers who are associated with a specific open source system (that usually manifests itself as an open source project). In contrast with the relatively uniform and stable organizational environment associated with a proprietary system, the core project environment in an open source system usually allows for more diversity and creative instability, allowing much higher levels of interaction and contributing to elevated levels of innovation (Kogut and Metiu, 2001; David and Rullani, 2008; Vujovic and Ulhøi, 2008). Freedom from temporal and spatial input limitations, physical conformity requirements, adherence to hierarchical processes and strictly enforced technology policies, and lack of standards and regulations, all contribute to this flexibility. It also provides better support for an environment that is bound by ideology, communication and trust,
resulting in innovative communities that can achieve high levels of effectiveness (Mockus et al. 2002; Stewart and Gosain, 2006; Kidane and Gloor, 2007; Hossain and Zhu, 2009).

![Figure 3: Open Source System](image)

Furthermore, open source systems operate under one additional environmental layer that would normally be absent from traditional proprietary systems. This layer, which we call the intra-project environment, includes a project’s wider developer and user communities. Unlike core developers, code contributions by user-developers within this layer may be relatively rare (Mockus et al. 2000; Kogut and Metiu, 2001; Mockus et al., 2002). In fact, the contribution of many users in the intra-project environment may be limited to testing and bug reporting only. Indeed, the majority of user-developers may offer no contribution other than promoting the use of project outputs. Note that the expanded open source system involving the intra-project environment operates under the same system rules as the core open source system. In other words, it also contributes people, technology and process inputs that would be transformed through relevant subsystems into corresponding system outputs.

Few of the traditional operational constraints hold an influence over open source systems. Scope, time and cost factors, which largely drive proprietary system development efforts, are mostly absent from open source systems, rendering Brooks’s Law insufficient (Koch, 2004). Factors contributing to failure of proprietary software projects, including inaccurate estimation techniques and assumptions, poor progress monitoring, or ineffective project management become less relevant in open source systems where massive parallel development, lack of time pressure, and frequent releases reduce the incidence of development problems (Brooks, 1995). However, open source projects are not a panacea. When assessed with traditional development metrics, due to the lack of project structure and redundant development efforts, the output of an open source system may be less efficient and more time-consuming than for a proprietary system (Johnson, 2001; Gasser and Ripoche, 2003). Indeed, a reliable and useful technology output may never be produced. It is worth reiterating, however, that the output of an open system includes people and processes in addition to technologies, such that even if the system fails to produce a tangible technology output, it may still enhance the skills of the people involved and/or produce a process advancement or innovation.
Like proprietary systems, open source systems permit negative as well as positive feedback loops (Raymond, 1999; Järvensivu et al., 2006). However, instead of producing positive feedback at well-defined intervals corresponding to major project milestones or negative feedback on predefined system deviations, open source systems produce constant and near real-time feedback, both positive and negative, on subsystem activity in a highly iterative fashion (Schmidt and Porter, 2001; Bonaccorsi and Rossi, 2003; Zhao and Elbaum, 2003). System outputs from one subsystem may become immediate inputs into another system. In Figure 3, we have attempted to model this permeable, iterative, and flexible structure.

In an open source system, the concept of adoption is not limited to technology outputs that occur outside the system’s boundaries. Adoption of technology, people or process outputs can occur both inside as well as outside the core project environment.

While pure proprietary and open source systems serve illustrative purposes well, most real world systems fall between the two extremes, in what can be described as hybrid forms. The idea of hybrid forms, especially as alternative business models, has already generated some interest in the academic community (West, 2003; Fitzgerald, 2006; Shah, 2006). Table 3 presents a description of proprietary and open source systems as endpoints along a continuum, across a number of different dimensions.

### Table 3: Proprietary, Open Source and Hybrid Systems

<table>
<thead>
<tr>
<th>System</th>
<th>Pure Proprietary</th>
<th>Hybrid Forms</th>
<th>Pure Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Well-defined from the start. Later changes usually indicate lack of planning or changing external environment. Main purpose is to produce a technological output. People and process outputs are treated as secondary only.</td>
<td>◄◄ Continuum ►►</td>
<td>Looseley defined at project start. MATures over time in an unpredictable manner. System represents varied objectives to different participants resulting in equal weighting of people, process and technology outputs.</td>
</tr>
<tr>
<td>Lifespan</td>
<td>Artificially and externally determined. Often financially bound. System will die as soon as it fails to serve its intended purpose.</td>
<td>◄◄ Continuum ►►</td>
<td>Organic and internally determined. System lives on as long as it serves a purpose or maintains involvement by developer-users.</td>
</tr>
<tr>
<td>Boundaries</td>
<td>Fixed and meticulously defined. Expansions or contractions possible but due to “unforeseen circumstances.”</td>
<td>◄◄ Continuum ►►</td>
<td>Flexible and undefined. Expansions and contractions are possible and expected.</td>
</tr>
<tr>
<td>Operational Constraints</td>
<td>Multiple factors. Operates under triangle of constraints: scope, cost, time.</td>
<td>◄◄ Continuum ►►</td>
<td>Few if any. Traditional project management practices are insufficient.</td>
</tr>
<tr>
<td>Feedback</td>
<td>Possible and desired, but intermittent. Usually after major milestones or upon deviations from preset limits only.</td>
<td>◄◄ Continuum ►►</td>
<td>Highly iterative. Constant and real time. Outputs become immediate inputs within the system, and externally.</td>
</tr>
</tbody>
</table>
4.3. A holistic framework for open source research

Even though Table 3 is a useful portrayal of ideal types and hybrid forms as a collection of composite attributes, it only shows an end state and does not explain how hybrid forms come into existence operationally. In this section, we develop a holistic, multi-dimensional framework to help explain the inner workings of real world hybrid systems.

While there are tangible differences between pure proprietary and open systems, as we have suggested above, linkages exist between and among all types of systems, and it is through these linkages that hybrid forms emerge. Open source systems tend to have more permeable boundaries, and thus interact to a greater degree with other systems than proprietary systems (Alexy and Henkel, 2007 and 2009). However, both types of systems co-exist in a highly interactive inter-project environment. This “network of systems” constitutes the basis of a unifying framework for OSR.

Three dimensions define the main elements of this framework: First, there are four environmental layers consisting of organizational, core-project, intra-project, and inter-project components; second, there is an iterative procedural stage, consisting of inputs, transformational activities, and outputs; and finally, there is the transformation subsystem consisting of people, processes, and technology.

Within each environmental layer, the subsystems of people, processes and technology guide the iterative input-transformation-output procedure. These elements, along with all possible combinations of multi-way interactions make up the proposed holistic framework. While a great many potential interactions can be conceptualized, for illustrative purposes we only focus on a few here to show how hybrid systems emerge.

First, the people input into an open source system can be provided by a proprietary system (Figure 4) or another open source system. In the case of a proprietary system, this input can come in the form of a formal commitment, for example, when a commercial company participates in an open source project (O’Mahony, 2007; Dahlander, 2007; Dahlander and Wallin, 2006; Lerner et al., 2006; Mustonen, 2005; Edelsohn et al., 2005; Grand et al., 2004; de Joode, 2004), or it can be informal, such as when a commercial developer spends part of his or her paid work time contributing to an open source project (Lakhani and Wolf, 2005). For example, Oracle Corporation is involved either directly or indirectly in more than 700 open source community projects.\(^4\) Similarly, IBM reports that more than 600 of its developers work on open source community projects.\(^5\)

---

4 http://oss.oracle.com/
5 http://www-03.ibm.com/linux/ossstds/oss/ossindex.html
Second, the people input into a proprietary system can be provided by an open source system (Figure 5). This can happen when a developer proves his or her ability on an open source project and subsequently receives an employment offer from a proprietary company. In fact, job market signaling as an extrinsic open source developer motivation factor has been extensively studied in the literature (Lerner and Tirole, 2000; Kogut and Metiu, 2001; Lerner and Tirole, 2002; Bonaccorsi and Rossi, 2003).

Third, a proprietary system can receive process inputs from an open source system (Figure 6). Sharing of best practices has always been common among open source projects (Johnson, 2006; Hemetsberger and Reinhardt, 2006; Haefliger et al., 2008). Recently, however, there has been an increase in the number of proprietary systems taking in process outputs of open source systems as inputs into their own processes (Riehle et al., 2009). Having seen the success of the open source software development life cycle, many proprietary companies have started considering adaptation of this methodology into their production processes (Ajila and Wu, 2007). For example, borrowing concepts from the process of open source development, Amazon’s “Mechanical Turk” website is built on the premise that an on-demand, flexible and scalable workforce can be assembled from among millions of online user developers.

http://www.mturk.com
Fourth, a technology output from an open source system can become a technology input into either a proprietary system (O’Reilly, 1999; Stone, 2002; Voth, 2003) or another open source system (Obrenovic and Gasevic, 2007; Haefliger et al., 2008). In the case of another open source system, this may show itself in the form of code module or component reuse; whereas in the case of a proprietary system (Figure 7), it may be modules of code or a complete software application. The former scenario is very common and well-researched in the open source literature (Ajila and Wu, 2007; Obrenovic and Gasevic, 2007; Haefliger et al., 2008). For example, Haefliger et al. (2008) identified 55 reused components comprising 2,975 reuse incidents in a sample of six open source projects, signifying that code reuse is extensive among OSS projects. The latter scenario is also common when proprietary companies create a business model around enhancement and support of commercially promising open source projects. For example, both Novell Inc. and Canonical Ltd. provide commercial services and enhancements on SuSE and Ubuntu Linux operating system distributions respectively (Lohr, 2003; Hastings, 2005).

Fifth, a technology output from a proprietary system can become a technology input into an open source system (Capek et al., 2005; Hawkins, 2004). This can occur when a large piece of proprietary code is opened up by the owner of the software, as in the case where IBM released Eclipse, resulting in the creation of the Eclipse Foundation, or when Sun Microsystems made the source code for Star
Office Suite available, leading to the emergence of OpenOffice.org (Vaughan-Nichols, 2003; Müller-Prove, 2007). Figure 8 illustrates such a scenario.

Figure 8: Inter-Project Environment (Hybrid technology subsystems)

Although we focused on very simplified examples for illustrative purposes and only showed a few sample interactions involving one subsystem at a time and happening at a selected environmental layer (inter-project), real world systems would likely include interactions at several environmental layers (organizational, core, intra or inter-project), involve many transformation subsystems (people, processes and technology), and concern a variety of procedural stages (input, transformation, output). Indeed, the value of the framework lies in its capability to conceptualize these multi-level interactions (Table 4).

Table 4: Holistic OSR Framework

<table>
<thead>
<tr>
<th>Environmental Layer</th>
<th>Transformation Subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HYBRID FORMS</td>
</tr>
<tr>
<td></td>
<td>Iterative Procedural Stage</td>
</tr>
<tr>
<td></td>
<td>People</td>
</tr>
<tr>
<td>Input</td>
<td>Transform</td>
</tr>
<tr>
<td>Organizational</td>
<td></td>
</tr>
<tr>
<td>Core-project</td>
<td></td>
</tr>
<tr>
<td>Intra-project</td>
<td></td>
</tr>
<tr>
<td>Inter-project</td>
<td></td>
</tr>
</tbody>
</table>
5. Application of the open source research taxonomy to the holistic framework

The holistic framework of OSR outlined above represents an attempt to use systems theory to capture the depth and breadth of open source research. This framework is purely conceptual in that it presents a normative view of the field – the role of the open source movement within the larger environmental context. The taxonomy presented earlier, on the other hand, is a comprehensive summary of existing OSR. In a sense, the framework describes where the field should be, while the taxonomy describes where the field is today. Thus, there is significant utility in comparing one with the other. In the following section, we attempt to provide examples of how the taxonomy codes fit in the framework and identify gaps where research may be lacking.

For purposes of clarity and brevity, we only describe in detail the cross-mapping of a subset of taxonomy codes onto the framework. For the selected subset of codes, we looked at each of the articles identified with those codes and re-assessed them in light of the three dimensions of the framework. We then marked all interactions where evidence was present (Table 5).

For example, the “OSS Production – Team/Project leadership” code (Case 1) had been partly created to capture studies investigating skills required to assume leadership positions in open source projects. Since project leaders are selected from among core project teams, these studies focused on core project teams and evaluated individual inputs to arrive at skills that mattered for leadership. Thus, we marked the interactions at the core-project environmental layer alongside the people transformation subsystem at the input stage with “C1.” Similarly, when the “OSS Production – User and developer motivations” code (Case 2) was assessed, we considered the characteristics of research concerning the motivations of open source community participants and noted that such studies investigated individual motivations shaping people inputs at core, intra as well as inter-project environments. Thus, the people transformation subsystem at the input stage was marked for three relevant environmental layers with “C2.”

Evaluation of articles under each of the three sub-categories of the “Software quality” code (Case 3) showed that despite addressing many attributes of quality in all environmental layers, those articles used a very narrow definition of quality that is restricted to technology outputs mostly in the form of completed OSS applications. We therefore marked interactions at all four environmental layers as well as one concerning outputs at the technology transformation subsystem level with “C3.” Finally, articles slotted in the “OSS Production – Individual and team learning” code (Case 4) focused on core (for developer learning) as well as intra (mainly for user learning) environmental layers and concerned the event of learning involving human inputs, how those inputs are transformed and manifested in the form of changed behavior (human outputs). Hence, interactions at two environmental layers and at all three procedural stages of people subsystems were marked with “C4.”
We extended the process described above to include each taxonomy code such that all the codes were individually mapped on to the OSR framework. One important outcome of this process was that we were able to identify a number of gaps within the present OSR literature.

At a general level, three of the environmental layers – organization, core project, and intra-project – are reasonably well covered in the literature. The fourth environmental layer – inter-project – however, has been less thoroughly examined. When it comes to the iterative procedural stage, inputs and outputs have been covered more extensively than transformations. Finally, within the transformational subsystem level, people and technology have been researched more often than processes. These results are perhaps not surprising, since the areas that have not received as much research attention are those that are inherently complex, longitudinal, and multivariate. For example, research examining the inter-project environment must make observations or collect data from multiple projects; and at the iterative procedural stage, inputs and outputs are much easier to observe and measure than transformations. We can conclude from this analysis that much of the low-hanging fruit of OSR has been picked. What remains to be conducted is more challenging, yet arguably more valuable research. Below, we identified a number of specific areas that appear to be under-researched based on our analysis of current OSR.

### System performance and project success

Most OSR has focused on technology outputs (e.g., level and frequency of code contributions). Development of multi-dimensional frameworks involving all transformation subsystems and covering all procedural stages would likely lead to the emergence of better performance indicators. This is an area that may have important implications for the future of open source projects. As noted in Table 3, traditional performance metrics may under-represent the value of open source projects as they overlook people and processual outputs. Furthermore, when it comes to inputs, most OSR focuses on people (e.g., number of developer or user participants that are then linked to factors affecting...
participation, like type of licenses). As a result, many open source projects are deemed to be inefficient at best, or failures at worst. Consideration of other procedural stages for all transformation subsystems would help construct more comprehensive success criteria. For example, a project that has failed to produce a technology output may not be seen as a failure if its participants developed skills helping them to create future successes, or if the project managed to create reusable process improvements.

**Software licensing and intellectual property rights**

The present body of OSR mostly covers two endpoints of the intellectual property rights spectrum: licensing of end products (technology outputs), and the effect of licensing and the strength of intellectual property regimes on community participation (people inputs). However, we argue that many areas that fall between these two extremes are worthy of consideration. For example, patenting of processes and methods is a relatively recent phenomenon and its potential impacts on the future of open source may provide an interesting research topic. Business methods and processes patent applications have grown by more than 2,000% over the past decade, yet we know very little about how this change might affect the open source movement (O’Mahony and Ferraro, 2004, p.6).

**The effect of standards on outcome variables of interest**

The open source movement has evolved, developed, and thrived in a relatively flexible and unstructured environment. Open source development projects have been characterized by far fewer standards and fixed methodologies than proprietary projects. This unstructured environment has allowed the open source community to produce a huge number of useful outputs. However, the lack of standards and structure can also be damaging. For example, certain best practices may never become widely accepted standards as they tend to stick with the originator and be seen as a collection of personal idiosyncrasies. Similarly, it may be hard to assess a certain developer skill set without any certification standard. There is evidence that this situation is changing. For example, open source certification standards are being developed, and are starting to spread within the open source community. Nevertheless, the open source community lags far behind the proprietary community in this regard. For example, there is currently no open source equivalent to ITIL or ITSM. There is very little research on this issue.

**Adoption and implementation**

The majority of the current OSR has focused on software development, and the ecosystem that surrounds it. By contrast, there is a paucity of research on what happens after an open source product or service has been produced. Yet, adoption, acceptance, use, continuance behavior, and discontinuance are important topics, particularly in organizational research. In other words, the current research is missing a great swath of the traditional IS life cycle. The present open source adoption research is very preliminary and narrowly focused on external adoption of technology outputs by end users, either at the individual or organizational level. There is almost no OSR on internal adoption of technology, people or process outputs. For example, we know very little about the adoption of processes in the form of best practices, or about the factors affecting and shaping the intensity of code reuse. Similarly, our knowledge is lacking on the adoption of people in the form of personnel movements between and among organizational forms. We also know little on the structure, inner workings, or impacts of OSS implementation communities and networks.

This is an important area of research since the unidimensional view of adoption of open source products and services by organizations projects a misleading picture of OSS adoption. This view shows that OSS adoption is still far behind the adoption of proprietary systems, except in a few niche areas, such as web server or other, behind-the-scenes infrastructure software. The proposed multidimensional view of adoption covering all transformation subsystems at all environmental layers will likely present a more realistic view of OSS embeddedness in various organizational forms.

6. Limitations and conclusions

No piece of research is perfect and this paper is no exception. While we attempted to conduct a
A comprehensive search of the research literature for instances of OSR, it is possible that we missed important contributions. For example, not all academic research is indexed by ProQuest, and despite our attempts to expand this source, some contributions may not have been captured by our search string. We did not collect work presented at conferences, that might have provided an important source of new ideas. We also did not capture dissertations, non-peer reviewed papers, or research that was not published in English. All these factors may have led to coverage error. However, we argue that the 618 research articles that we collected and analyzed in the paper constitute a representative and useful list of extant open source research. Through a systematic and iterative reduction of taxonomy categories, we also believe we were able to establish a proper balance between comprehensiveness of content and parsimoniousness of the taxonomy.

The processes of summarizing, coding, and categorizing the articles is inherently subjective, and may be influenced by researcher bias. While we took a number of precautions, and followed a systematic and structured process, it is possible that the codes and categories that we developed may not be consistent across the population of open source researchers. For example, the reliance on a single coder during the early stages of the research may have led to a bias. However, a number of checks and balances, including the introduction of independent coders later in the process, may have alleviated this concern. Similarly, the optimum point at which to mark the end of the iterative reduction of taxonomy categories is, at its root, a subjective judgment. On the one hand, further reduction of taxonomy categories would result in an ever more parsimonious taxonomy. On the other hand, this reduction process risks creating overlapping categories that could be questioned on the basis of non-exclusivity.

We propose that the taxonomy presented here along with the categorization schemes and the holistic framework represent a starting point for a future conversation about the value, direction, and efficacy of open source research. In the spirit of the open source movement, we intend to share our database of articles and categories with the open source community, so that it can become a living resource. We hope that this resource can grow in size as new research is conducted and added to the dataset. We also hope that this work can become increasingly valuable to the open source research community, as codes and categories are refined.

The open source movement has grown rapidly since its inception a little over ten years ago. Open source research has also grown and proliferated across many different research areas and disciplines. Due to this proliferation across multiple domains, it has become difficult to grasp the totality of what has been done, where the gaps are, and generally what the form and character of the field is. As OSR grows, it is useful to take a snapshot of how the field has developed, and, if necessary, to guide its future growth.

In this paper, we conducted an unbiased and comprehensive review of current open source research. This review resulted in the collection of 618 peer-reviewed research articles that we organized into a taxonomy. This taxonomy consisted of 57 code categories organized into 7 higher-level groupings. In total, we categorized 1,355 unique instances of each code from the full article set. This taxonomy represented our analysis of the current OSR state-of-the-field. We then drew on existing conceptual work and systems theory to develop a holistic framework to situate open source research within the larger environmental context. Finally, we overlayed the taxonomy on to the framework in order to propose a sample list of gaps and areas of overlap that can guide future work in OSR. In summary, we feel that open source research has evolved into a vibrant and productive field that has a great deal to offer both research and practice. By summarizing the field and offering a guiding conceptual framework, we hope to continue this trend into the future.
References


Aksulu & Wade/Open Source Research


Feb. 14, 2010).


Aksulu & Wade/Open Source Research


Cooperation, Innovation and Technologie (CITE'03), Troyes, France.


Feb.14, 2010).


Aksulu & Wade/Open Source Research


Aksulu & Wade/Open Source Research


Aksulu & Wade/Open Source Research


Aksulu & Wade/Open Source Research


Journal of the Association for Information Systems Vol. 11 Special Issue pp. 576-656 November 2010


Sloan Management Review, 42(4), pp. 82-86.


and Innovation, 15(2), pp. 223-231.


Appendices

Appendix A: Pattern/Code Descriptions and Related Articles

Table A: Code Descriptions and Related Articles

Appendix B: Extant Categorizations of Open Source Research

Table B1: Feller and Fitzgerald (2000)
Table B2: Lerner and Tirole (2001)
Table B3: Rossi (2004)
Table B4: Lerner and Tirole (2005)
Table B5: von Krogh and von Hippel (2006)
Table B6: Nelson et al. (2006)
Table B7: Niederman et al. (2006a,b)
Table B8: Scacchi (2007)
Table B9: Jin et al. (2007)
### Appendix A: Pattern/Code Descriptions and Related Articles

#### Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
<th>Articles</th>
</tr>
</thead>
</table>
### Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Articles that compare OSS with proprietary software from both process and product perspectives. Such comparisons include feature, attribute, design structure, or overall process comparisons as well as elaborations on competitive factors, risk, legal, and financial aspects. Proprietary in this sense sometimes means traditional software development practices or simply closed source.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Articles in this category propose economic models to explain OSS, devise business models to explore how OSS can be taken advantage of, and/or talk about firm or nation level strategies and policies involving OSS.</th>
</tr>
</thead>
</table>
**Table A: Code Descriptions and Related Articles**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFTWARE QUALITY - TESTING and BUG FIXES</td>
<td>Articles in this category represent a subset of software quality. Specifically, studies in this category focus on defect handling and bug fixing processes within OSS teams. Some articles focus on performance (defined as the ratio of submitted bugs to resolved ones) whereas some others zoom in on the bug fixing process and practices as an indication of underlying social patterns or as a surrogate for organizational learning. OSS bug tracking tools are evaluated against proprietary ones.</td>
<td>Au et al. (2009), Ayewah et al. (2008), Crowston and Scozzi (2008), Kidane and Gloor (2007), Long and Siau (2007), Nizovtsev and Thursby (2007), Louridas (2006), Falcioni (2005), Gyimothy et al. (2005), Louridas (2005), Remillard (2005), Glance (2004), Koru and Tian (2004), Huntley (2003), Bollinger et al. (1999), McConnell (1999).</td>
</tr>
</tbody>
</table>
Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Articles</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFTWARE QUALITY - OSS SECURITY</td>
<td>Articles in this category represent a subset of software quality with a particular focus on security. The dilemma surrounding public disclosure of vulnerabilities is discussed and the security of OSS applications is evaluated and compared with that of proprietary systems. The rationale for OSS use in security appliances is elaborated on.</td>
<td>Masri and Podgurski (2008), Hoepman and Jacobs (2007), Nizovtsev and Thursby (2007), Boulanger (2005), Ye et al. (2005), Pashalidis and Fleury (2004), Hansen et al. (2002), Payne (2002), Swift (2001).</td>
</tr>
<tr>
<td>OSS SUCCESS</td>
<td>Studies in this category investigate one or many determinants of OSS success and the relationships among them. Various definitions of success as well as factors contributing to the success of OSS projects are evaluated. Aspects of OSS projects that are considered as successful are looked at.</td>
<td>Beecher et al. (2009), Lee et al. (2009), Méndez-Durón and García (2009), Subramaniam et al. (2009), Crowston and Scozzi (2008), Frank (2008), Sohn and Mok (2008), Bayersdorfer (2007), Comino et al. (2007), Schweik and English (2007), Grewal et al. (2006), Stewart and Gosain (2006), Stewart et al. (2006).</td>
</tr>
<tr>
<td>Pattern</td>
<td>LEGAL AND REGULATORY: This pattern hosts articles that talk about licensing of OSS</td>
<td></td>
</tr>
</tbody>
</table>
# Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Articles</th>
</tr>
</thead>
</table>
Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
<th>Articles</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Articles</th>
</tr>
</thead>
</table>
### Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Category</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COMMUNITIES</strong></td>
<td>Articles that look at the definitions of the community construct as well as the underlying social, cultural, and economic elements of OSS communities. Sponsored and autonomous communities are differentiated and their norms, value systems, and common practices are investigated. Community governance structures as well as coordination practices are assessed. The effect of community portals on collaboration and the relationship between communities and commercial firms are looked at.</td>
</tr>
<tr>
<td><strong>TEAM FORMATION</strong></td>
<td>Articles examining the formation of OSS teams and how individuals make decisions about what teams to join. Strategies and processes by which new people join are evaluated and interactions between team members are investigated. Studies looking at the geographical location or network embeddedness of developers involved in OSS projects are also included in this category.</td>
</tr>
<tr>
<td><strong>GOVERNANCE</strong></td>
<td>Studies in this category look at how communities producing collective goods govern themselves. Various definitions of governance are investigated and dimensions and types of governance are categorized. Community hierarchies, conflict management, and decision making mechanisms are examined.</td>
</tr>
<tr>
<td>TEAM / PROJECT LEADERSHIP</td>
<td>Articles that investigate individual and project-level factors leading to the emergence of project leaders among OSS community participants. Studies that talk about the link between skills of OSS leaders and its influence on OSS success are also included in this group.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INDIVIDUAL AND TEAM LEARNING</td>
<td>Articles in this category are concerned with individual and collective learning practices in OSS projects. Factors affecting learning process and adaptive learning mechanisms are investigated.</td>
</tr>
<tr>
<td>Table A: Code Descriptions and Related Articles</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>COLLABORATION AND KNOWLEDGE SHARING</strong></td>
<td></td>
</tr>
<tr>
<td>This category looks at collaboration and its role in creation and dissemination of knowledge. Knowledge-sharing activities in OSS communities are compared with those at traditional organizations, and potential benefits of OSS style collaboration in organizations are discussed. Articles investigating the task assignment and work allocation mechanisms in collaborative efforts and ones assessing factors affecting collaboration are included in this category.</td>
<td></td>
</tr>
<tr>
<td><strong>USER AND DEVELOPER MOTIVATIONS</strong></td>
<td></td>
</tr>
<tr>
<td>Articles in this group investigate the dynamics that affect the motivation of participants in OSS communities. Intrinsic and extrinsic motivations including personal attributes, attitudes, behavioral patterns, and job related factors and how they affect individual developers’ selections of projects/project preferences are evaluated. Implications for commercial firms are discussed.</td>
<td></td>
</tr>
<tr>
<td><strong>ROLE OF COMMERCIAL CORPORATIONS</strong></td>
<td></td>
</tr>
<tr>
<td>Articles focusing on corporations’ motivations to engage in open source development endeavors. Firm efforts in commercialization of OSS as well as indirect effects on emergence of OSS projects are assessed. Various scenarios of firm participation including sponsored communities as well as code donations are discussed. Potential link between organizational sponsorship and OSS success is investigated.</td>
<td></td>
</tr>
</tbody>
</table>
## Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Software Development - Use of OSS Components</th>
<th>Description</th>
<th>Articles</th>
</tr>
</thead>
</table>


| Pattern | OSS APPLICATIONS: This pattern hosts a bundle of articles that focus on area or discipline specific OSS applications. |

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Articles</th>
</tr>
</thead>
</table>
### Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Related Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SUPPLY CHAIN MANAGEMENT AND OPTIMIZATION</strong></td>
<td>Articles focusing on OSS applications for corporate supply chain management such as OSS enterprise resource planning and customer relationship management systems, electronic data interchange and operational planning and optimization applications.</td>
<td>Christou and Ponis (2009), Brydon and Vining (2008), Aldous and Lintott (2007), Fei and Olson (2007), Bruce et al. (2006), Serrano and Sarriegi (2006), Nau et al. (2005), Wüstner et al. (2005), Curtis and Funderburg (2003).</td>
</tr>
<tr>
<td>Table A: Code Descriptions and Related Articles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| ACADEMIC AND COMMERCIAL RESEARCH | Articles focusing on OSS applications that help researchers organize and share digital information for research purposes such as:  
- OSS citation document and webpage management software,  
- Collaborative OSS tools that help develop new research approaches such as OSS social computing applications that would allow the creation of new research communities, and  
### Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th><strong>Pattern</strong></th>
<th><strong>OSS DIFFUSION</strong></th>
<th><strong>Articles</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code</strong></td>
<td><strong>Description</strong></td>
<td><strong>Articles</strong></td>
</tr>
<tr>
<td>NATURAL SCIENCES</td>
<td>Articles focusing on OSS applications for natural sciences such as chemistry, physics, GIS, astronomy, and environmental sciences.</td>
<td>Rey (2009), Schweik et al. (2009), Benton et al. (2008), Bullung and Remmel (2008), Hand (2008), Sykora and Leahy (2008), Guha et al. (2006), Rzepa et al. (2006), Ramli et al. (2005), Anselin et al. (2004).</td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
<td>Articles</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OSS Implementation - Communities and Networks</td>
<td>Articles that elaborate on the importance of user support community groups both for implementation and for long-term viability of OSS. Implications for commercial firms aiming to use OSS in hybrid form are discussed. Some articles in this category investigate the role of collaborative business networks in OSS implementation. Focus on both commercial and non-profit networks are possible.</td>
<td>Hamel and Schweik (2009), Ven and Mannaert (2008), Feller et al. (2007), Jin et al. (2007), Sowe et al. (2006), Shah (2006), Yu et al. (2006), Hess (2005).</td>
</tr>
</tbody>
</table>
### Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>OSS IMPLEMENTATION - GOVERNMENTS / NATIONS</th>
<th>Code Description</th>
<th>Articles</th>
</tr>
</thead>
</table>

### Pattern

BEYOND SOFTWARE: This important pattern groups articles that explore implications of OSS over and beyond the software development domain. In general, these categories investigate the possibility of the applicability of OSS style organization and work practices to other domains and their implications for users.

### Code Description

Studies in this category investigate how various OSS approaches can influence non-software areas. These studies look at wider socio-political effects of OSS and see OSS as a broader icon for openness and collaboration and key to understanding future forms of organizations. The primary focus is placed on benefits that can be derived from the OSS philosophy and translation of OSS approaches to corporate management that emulate OSS style and governance practices.
# Table A: Code Descriptions and Related Articles

| OPEN STANDARDS | Studies in this category assess linkages between open standards, proprietary and OSS development and argue that open source is one (but not the only) way to enforce and exploit open standards. Some articles approach open standards as a business strategy whereas some others develop a pragmatic concern for the importance of open standards for long-term access to information. | Fisher et al. (2008), Cerri and Fuggetta (2007), LaMarca (2006), Simon (2005), Smith (2005), Wüstner et al. (2005), Wheeler (2004a), Coyle (2002). |
### Table A: Code Descriptions and Related Articles

<table>
<thead>
<tr>
<th>Category</th>
<th>Articles/Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OPEN EDUCATION</strong></td>
<td>Articles that investigate the possibility of a new model of education based on the OSS movement. OSS is discussed as it relates to educational platforms, and efficacy of open source education software is assessed. Some articles propose the OSS model as an opportunity to revolutionize future administrative systems while some others look at known cases of open education such as MIT’s OpenCourseWare. Chumney and Zhou (2008), Luyt (2008), Watson et al. (2008b), Landsberger (2007), Currie (2005), Hepburn (2004), Wheeler (2004a), Baldi et al. (2003), Keats (2003), Moore (2002), Newmarch (2001), Werry (2001).</td>
</tr>
</tbody>
</table>
### Appendix B: Extant Categorizations of Open Source Research

#### Table B1: Feller and Fitzgerald (2000)

<table>
<thead>
<tr>
<th>Feller and Fitzgerald (2000) Category</th>
<th>Matching Categories from The Taxonomy of Open Source Research</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What (Transformation):</strong></td>
<td></td>
</tr>
<tr>
<td>• What defines a software project as OSS?</td>
<td>OSS DESCRIPTIVE</td>
</tr>
<tr>
<td>• What types of projects tend to be OSS?</td>
<td>OSS LICENSING</td>
</tr>
<tr>
<td>• OSS APPLICATIONS – (All sub-categories)</td>
<td>OSS APPLICATIONS – (All sub-categories)</td>
</tr>
<tr>
<td><strong>Why (Weltanshauung/World View):</strong></td>
<td></td>
</tr>
<tr>
<td>• What are the technological motivations for OSS development?</td>
<td>SOFTWARE QUALITY</td>
</tr>
<tr>
<td>• What are the economic motivations for OSS development?</td>
<td>SOFTWARE DEVELOPMENT – OSS CODE EFFICIENCIES</td>
</tr>
<tr>
<td>• What are the socio-political motivations for OSS development?</td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS</td>
</tr>
<tr>
<td></td>
<td>OSS STANDARDS AND REGULATION</td>
</tr>
<tr>
<td></td>
<td>OSS VERSUS PROPRIETARY</td>
</tr>
<tr>
<td><strong>When and Where (Environment):</strong></td>
<td></td>
</tr>
<tr>
<td>• What are the temporal dimensions of OSS development?</td>
<td>OSS PRODUCTION – PROCESS</td>
</tr>
<tr>
<td>• What are the spatial/geographic dimensions of OSS development?</td>
<td>OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING</td>
</tr>
<tr>
<td><strong>How:</strong></td>
<td></td>
</tr>
<tr>
<td>• How is the OSS development process organized?</td>
<td>OSS PRODUCTION – PROCESS</td>
</tr>
<tr>
<td>• What tools are used to support the OSS model?</td>
<td>OSS PRODUCTION – GOVERNANCE</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE QUALITY (all sub-categories)</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING</td>
</tr>
<tr>
<td><strong>Who (Client, Actor, Owner):</strong></td>
<td></td>
</tr>
<tr>
<td>• What are the characteristics of the individual developers contributing to OSS projects?</td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td>• What are the characteristics of the companies distributing OSS products?</td>
<td>OSS PRODUCTION – ROLE OF VOLUNTEER USERS/DEVELOPERS</td>
</tr>
<tr>
<td>• What are the characteristics of the users of OSS products?</td>
<td>OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – ROLE OF VOLUNTEER USERS/DEVELOPERS</td>
</tr>
</tbody>
</table>

Notes: Feller and Fitzgerald (2000) drew on Zachman’s information systems architecture (Zachman, 1987) and Checkland’s CATWOE framework from Soft Systems Methodology (Checkland, 1981), to propose a framework to organize and analyze open source research. The main elements of the proposed framework along with its linkages to the taxonomy codes are shown in Table B1.
### Table B2: Lerner and Tirole (2001)

<table>
<thead>
<tr>
<th>Lerner and Tirole (2001) Categories</th>
<th>Matching Categories from The Taxonomy of Open Source Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why do programmers participate?</td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS</td>
</tr>
<tr>
<td>What constitutes a good project and good leadership?</td>
<td>OSS SUCCESS OSS PRODUCTION – DEVELOPMENT TEAM PERFORMANCE OSS PRODUCTION – TEAM/PROJECT LEADERSHIP</td>
</tr>
<tr>
<td>Why do software vendors participate?</td>
<td>OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS OSS PRODUCTION – COMMUNITIES SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS</td>
</tr>
<tr>
<td>Opening proprietary code</td>
<td>OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS OSS PRODUCTION – GOVERNANCE BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS</td>
</tr>
<tr>
<td>Legal aspects</td>
<td>OSS LICENSING OSS LEGAL ISSUES OSS INTELLECTUAL PROPERTY RIGHTS</td>
</tr>
<tr>
<td>Sociological aspects</td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS OSS PRODUCTION – SELF-ORGANIZATION (PRODUCT AND COMMUNITY EVOLUTION)</td>
</tr>
</tbody>
</table>

Notes: *Lerner and Tirole (2001)* were concerned with exploring the incentives for programmers and software vendors to participate in open source software projects, and built their framework to explore this topic area. Table B2 shows the matching of topics that have been discussed or proposed for further research by the authors to the taxonomy codes established in this study.
Notes: Rossi (2004) provided a comprehensive review on the production side of free and OSS research with particular attention to the topics of intellectual property and governmental policies. After elaborating on topics that have already been explored in extant research, Rossi provided directions for future research suggesting a number of research worthy areas that included institutional–individual motivation complementarity, long-term view on voluntary community governance, and impact of commercial involvement as well as licensing issues for publicly funded software and extension of open source model to other domains.
<table>
<thead>
<tr>
<th>Lerner and Tirole (2005) Categories</th>
<th>Matching Categories from The Taxonomy of Open Source Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Motivates Open Source Contributors?</td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS</td>
</tr>
</tbody>
</table>
| How Do Commercial Firms Work and Compete with Open Source? | OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS
                      OSS PRODUCTION – COMMUNITIES
                      SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS
                      BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS |
| How Does the Legal System Affect Open Source? | OSS LICENSING
                      OSS LEGAL ISSUES |
| What is the Relative Quality of Open Source Software? | SOFTWARE QUALITY
                      OSS VERSUS PROPRIETARY
                      SOFTWARE DEVELOPMENT – OSS CODE EFFICIENCIES |
| What are Appropriate Public Policies Toward Open Source? | BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS
                      OSS IMPLEMENTATION – GOVERNMENTS / NATIONS |
| How Will Software Patents Affect Open Source? | OSS INTELLECTUAL PROPERTY RIGHTS
                      OSS LEGAL ISSUES |
| Can Open Source Work Beyond Software? | BEYOND OSS – OPEN PARADIGM
                      BEYOND OSS – OPEN INNOVATION
                      BEYOND OSS – OPEN KNOWLEDGE FLOWS |
| Can Firms Realize the Benefits of Open Source in Other Ways? | OSS BENEFITS/DRAWBACKS
                      OSS VISION/ROADMAP
                      BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS
                      OSS INTELLECTUAL PROPERTY RIGHTS
                      OSS LICENSING
                      OSS STANDARDS AND REGULATION |
| Open Source and Academia | BEYOND OSS – OPEN KNOWLEDGE FLOWS
                      BEYOND OSS – OPEN PARADIGM
                      BEYOND OSS – OPEN EDUCATION
                      OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS
                      OSS INTELLECTUAL PROPERTY RIGHTS |

Notes: *Lerner and Tirole (2005)* continued their exploration of the motivation for open source contributions with a third study in 2005. In this study, they focused on the interaction between commercial and open source software, as well as a set of relevant legal systems. Their categories are presented in Table B4.
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Motivations for contributions</strong></td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE DEVELOPMENT – OSS CODE EFFICIENCIES</td>
</tr>
<tr>
<td><strong>Governance, organization, and innovation process</strong></td>
<td>OSS PRODUCTION – GOVERNANCE</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – TEAM FORMATION</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – INNOVATION</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – SELF-ORGANIZATION (PRODUCT AND COMMUNITY EVOLUTION)</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – PROCESS</td>
</tr>
<tr>
<td></td>
<td>OSS IMPLEMENTATION – IMPLEMENTATION COMMUNITIES AND NETWORKS</td>
</tr>
<tr>
<td></td>
<td>BEYOND OSS – OPEN INNOVATION</td>
</tr>
<tr>
<td><strong>Competitive dynamics</strong></td>
<td>OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS</td>
</tr>
<tr>
<td></td>
<td>BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS</td>
</tr>
<tr>
<td></td>
<td>OSS VERSUS PROPRIETARY</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – SELF-ORGANIZATION (PRODUCT AND COMMUNITY EVOLUTION)</td>
</tr>
</tbody>
</table>

**Notes:** Von Krogh and von Hippel (2006) developed a framework to organize the submission to a special issue of Management Science on Open Source Software. Table B5 summarizes the proposed framework categories and matching taxonomy codes from the literature review.
Table B6: Nelson et al. (2006)

<table>
<thead>
<tr>
<th>Nelson et al. (2006) Categories</th>
<th>Matching Categories from The Taxonomy of Open Source Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiation Phase:</td>
<td></td>
</tr>
<tr>
<td>• Why are open source projects started in the first place?</td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS</td>
</tr>
<tr>
<td>• What incentives regulate the initiation phase?</td>
<td></td>
</tr>
<tr>
<td>Ongoing Project Phase:</td>
<td></td>
</tr>
<tr>
<td>• What motivates individuals and organizations to participate in an</td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS</td>
</tr>
<tr>
<td>ongoing open source project?</td>
<td>OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS</td>
</tr>
<tr>
<td>• How can individuals and organizations participate? What are the</td>
<td>BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS</td>
</tr>
<tr>
<td>roles they play and the quality of contributions they make?</td>
<td>OSS PRODUCTION – PROCESS</td>
</tr>
<tr>
<td>• Which coordinating and communication mechanisms aid or hinder</td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td>open source projects?</td>
<td>OSS PRODUCTION – TEAM FORMATION</td>
</tr>
<tr>
<td>Adoption/Deployment Phase</td>
<td>OSS PRODUCTION – GOVERNANCE</td>
</tr>
<tr>
<td>Coordination Mechanisms in an OSS Project</td>
<td>OSS PRODUCTION – ROLE OF VOLUNTEER USERS / DEVELOPERS</td>
</tr>
<tr>
<td>Impact of OSS Projects</td>
<td>OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING</td>
</tr>
<tr>
<td>Participation Alternatives</td>
<td>OSS SUCCESS</td>
</tr>
</tbody>
</table>
| NOTES: Nelson et al. (2006) categorized open source software research in three major phases and three additional research domains. A number of leading questions were provided. Table B6 lists the proposed phases and additional domains and maps them to the taxonomy categories.
Table B7: Niederman et al. (2006a,b)

<table>
<thead>
<tr>
<th>Niederman et al. (2006a,b) Categories</th>
<th>Matching Categories from The Taxonomy of Open Source Research</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The Software Artifact:</strong></td>
<td></td>
</tr>
<tr>
<td>• Artifact type: Contrasting open source and proprietary artifact characteristics</td>
<td>OSS VERSUS PROPRIETARY</td>
</tr>
<tr>
<td>• License type: Precursors to the choice of license type, effect of license type on diffusion and use of software</td>
<td>OSS LICENSING</td>
</tr>
<tr>
<td>• Quality of product (bugs, security)</td>
<td>OSS PRODUCTION – ROLE OF LICENSING AND IP</td>
</tr>
<tr>
<td></td>
<td>OSS ADOPTION – BARRIERS</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE QUALITY – OSS SECURITY</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE QUALITY – TESTING and BUG FIXES</td>
</tr>
<tr>
<td><strong>The Individual:</strong></td>
<td></td>
</tr>
<tr>
<td>• Developer: Motivations for participation</td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS</td>
</tr>
<tr>
<td>• User: Choice of project, adoption decisions</td>
<td>OSS ADOPTION – DECISION FACTORS</td>
</tr>
<tr>
<td><strong>The Team/Project/Community:</strong></td>
<td></td>
</tr>
<tr>
<td>• Organization governance: Mixtures of paid and volunteer developers</td>
<td>OSS PRODUCTION – GOVERNANCE</td>
</tr>
<tr>
<td>• Mechanics for artifact creation: Processes for modularizing projects, “assigning” work tasks, for evaluating and integrating new code. Communication processes and patterns.</td>
<td>OSS PRODUCTION – ROLE OF VOLUNTEER USERS / DEVELOPERS</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – PROCESS</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE DEVELOPMENT – OSS CODE EFFICIENCIES</td>
</tr>
<tr>
<td><strong>The Organization:</strong></td>
<td></td>
</tr>
<tr>
<td>• Developer/distributor/users: Business models for developers and distributors of open source software. Total cost of ownership for investing in open source. Mixtures of open source and proprietary software over a whole MIS department.</td>
<td>BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS</td>
</tr>
<tr>
<td></td>
<td>SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td></td>
<td>OSS IMPLEMENTATION – IMPLEMENTATION COMMUNITIES AND NETWORKS</td>
</tr>
<tr>
<td><strong>The Society:</strong></td>
<td></td>
</tr>
<tr>
<td>• Influence on society: Diffusion of the open source “philosophy” to other areas such as licensing of intellectual property. Governmental policies regarding the use of open source versus proprietary software.</td>
<td>BEYOND OSS – OPEN PARADIGM</td>
</tr>
<tr>
<td></td>
<td>OSS IMPLEMENTATION – GOVERNMENTS/NATIONS</td>
</tr>
<tr>
<td></td>
<td>BUSINESS/ECONOMIC MODELS AND STRATEGIES/POLICIES FOR OSS</td>
</tr>
<tr>
<td></td>
<td>OSS VERSUS PROPRIETARY</td>
</tr>
</tbody>
</table>

Notes: Niederman et al. (2006a,b) present a two-part study of open source research. In Part 1, the authors frame open source software “in a larger context” and propose a multi-level framework for the investigation of the open source software domain. In Part II, selected theories from information systems and other disciplines are introduced as being potentially relevant to open source research. Table B7 maps the research issues by levels of analysis.
### Table B8: Scacchi (2007)

<table>
<thead>
<tr>
<th>Scacchi (2007) Categories</th>
<th>Matching Categories from The Taxonomy of OSR</th>
</tr>
</thead>
</table>
| Individual Participation in FOSSD Projects | OSS PRODUCTION – USER/DEV. MOTIVATIONS  
OSSE PRODUCTION – GOVERNANCE  
OSSE PRODUCTION – COMMUNITIES  
OSSE PRODUCTION – TEAM FORMATION |
| Resources & Capabilities Supporting FOSSD:  
- Personal software development tools and networking support  
- Beliefs supporting FOSS Development  
- FOSSD informalisms  
- Competently skilled, self-organizing, and self-managed software developers  
- Discretionary time and effort of developers  
- Trust and social accountability mechanisms | OSS PRODUCTION – ROLE OF VOLUNTEER USERS / DEVELOPERS  
OSSE PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING  
OSSE PRODUCTION – USER/DEV. MOTIVATIONS  
OSSE PRODUCTION – COMMUNITIES  
OSSE PRODUCTION – PROCESS  
OSSE PRODUCTION – GOVERNANCE  
OSSE PRODUCTION – SELF-ORGANIZATION |
| Cooperation, coordination, and control in FOSS projects | OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING  
OSSE PRODUCTION – PROCESS  
OSSE PRODUCTION – GOVERNANCE |
| Alliance formation, inter-project social networking and community development | OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING  
OSSE PRODUCTION – SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS  
OSSE PRODUCTION – INDIVIDUAL AND TEAM LEARNING  
LEGAL AND REGULATORY – OSS LICENSING  
OSSE PRODUCTION – COMMUNITIES  
OSSE IMPLEMENTATION – IMPLEMENTATION COMMUNITIES AND NETWORKS  
OSSE PRODUCTION – SELF-ORGANIZATION |
| Community development and system development | OSS PRODUCTION – COMMUNITIES  
OSSE PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING  
OSSE PRODUCTION – SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS  
OSSE PRODUCTION – SELF-ORGANIZATION  
PERFORM. METRICS – SOFTWARE QUALITY  
OSSE PRODUCTION – INNOVATION |
| FOSS as a multi-project software ecosystem:  
- Co-evolving socio-technical systems for FOSS | OSS PRODUCTION – COMMUNITIES  
OSSE PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING  
OSSE PRODUCTION – SOFTWARE DEVELOPMENT – USE OF OSS COMPONENTS  
OSSE PRODUCTION – SELF-ORGANIZATION  
PERFORM. METRICS – SOFTWARE QUALITY  
OSSE PRODUCTION – INNOVATION |
| FOSS as a Social Movement | OSS PRODUCTION – COMMUNITIES  
LEGAL AND REGULATORY – OSS LICENSING  
OSSE PRODUCTION – ROLE OF COMMERCIAL CORPORATIONS |

Notes: Focusing on free and open source software development (FOSSD) and evolution, Scacchi (2007) provides a comprehensive review of selected empirical studies along a variety of dimensions, talks about their limitations and identifies potential contributions FOSSD can make to traditional software engineering practices. Table B8 summarizes topical categories examined by Scacchi and the overlapping OSS taxonomy codes.
**Table B9: Jin et al. (2007)**

<table>
<thead>
<tr>
<th>Jin et al. (2007) Categories</th>
<th>Matching Categories from The Taxonomy of Open Source Research</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Creation of OSS User Communities:</strong></td>
<td>OSS ADOPTION – GENERAL</td>
</tr>
<tr>
<td>• How do new users, especially technically disadvantaged users, learn about OSS alternatives to proprietary software?</td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td>• How are OSS user communities created?</td>
<td>OSS PRODUCTION – USER AND DEVELOPER MOTIVATIONS</td>
</tr>
<tr>
<td>• What are the incentives for participating in OSS user communities?</td>
<td></td>
</tr>
<tr>
<td><strong>Characteristics of OSS User Communities:</strong></td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td>• What is the structure of OSS user communities?</td>
<td>OSS PRODUCTION – PROCESS</td>
</tr>
<tr>
<td>• How do user communities coordinate their physical and virtual activities?</td>
<td>OSS IMPLEMENTATION – IMPLEMENTATION COMMUNITIES AND NETWORKS</td>
</tr>
<tr>
<td><strong>Contributions by members of OSS User Communities:</strong></td>
<td>OSS IMPLEMENTATION – GENERAL</td>
</tr>
<tr>
<td>• What do OSS users contribute to the community by using free software?</td>
<td>OSS PRODUCTION – GOVERNANCE</td>
</tr>
<tr>
<td>• What do OSS users contribute to the community beyond their use of free software?</td>
<td>OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING</td>
</tr>
<tr>
<td>• What contributions can OSS user communities make to other users?</td>
<td></td>
</tr>
<tr>
<td><strong>Change and Evolution of OSS User Communities:</strong></td>
<td>OSS PRODUCTION – ROLE OF VOLUNTEER USERS / DEVELOPERS</td>
</tr>
<tr>
<td>• How will OSS user communities change as they grow larger and more successful?</td>
<td>OSS PRODUCTION – COMMUNITIES</td>
</tr>
<tr>
<td>• How will the character of OSS user communities change over time?</td>
<td>OSS PRODUCTION – COLLABORATION AND KNOWLEDGE SHARING</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – INDIVIDUAL AND TEAM LEARNING</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – INNOVATION</td>
</tr>
<tr>
<td></td>
<td>OSS PRODUCTION – SELF-ORGANIZATION (PRODUCT AND COMMUNITY EVOLUTION)</td>
</tr>
</tbody>
</table>

Notes: Jin et al. (2007) point out that most open source research has looked at development rather than use of open source software, and thus propose a framework establishing four areas of investigation to guide open source software usage research. The four areas of investigation along with several guiding research questions are summarized and mapped against the taxonomy codes in Table B9.
About the Authors

ALTAY AKSULU is a doctoral candidate in Operations Management and Information Systems at the Schulich School of Business in Toronto, Canada. His research focuses on the adoption and use of open source technologies and systems in an organizational context. Altay is an aeronautical engineer and has an MBA degree with a concentration in information systems. Prior to his academic career, Altay worked for 15 years as an Information Technology professional in both private and public sector enterprises in North America, Europe, and the Middle East.

MICHAEL R. WADE is Professor of Innovation and Strategic Information Management at IMD in Lausanne, Switzerland. Previously, he was an Associate Professor and Academic Director of the Kellogg-Schulich Executive MBA Program at York University, Toronto, Canada. He received a Ph.D. from the Ivey Business School at the University of Western Ontario. Professor Wade has worked extensively with public and private sector organizations to further an understanding of the strategic use of information systems for sustainable competitive advantage. He has lived and worked in seven countries across four continents and consulted for top international organizations like Cisco Systems, Shell, IBM, and Credit Suisse. His research has appeared in journals such as MIS Quarterly, Strategic Management Journal, and the Communications of the ACM. He is a Senior Editor at the Journal of the AIS and an Associate Editor at ISR. Professor Wade has co-authored six books in the areas of Information Systems, eCommerce, and Management Theory.