Organizational Social Structures for Software Engineering

DAMIAN A. TAMBURRI, PATRICIA LAGO, and HANS VAN VLIET, VU University Amsterdam

Software engineering evolved from a rigid process to a dynamic interplay of people (e.g., stakeholders or developers). Organizational and social literature call this interplay an Organizational Social Structure (OSS). Software practitioners still lack a systematic way to select, analyze, and support OSSs best fitting their problems (e.g., software development). We provide the state-of-the-art in OSSs, and discuss mechanisms to support OSS-related decisions in software engineering (e.g., choosing the OSS best fitting development scenarios). Our data supports two conclusions. First, software engineering focused on building software using project teams alone, yet these are one of thirteen OSS flavors from literature. Second, an emerging OSS should be further explored for software development: social networks. This article represents a first glimpse at OSS-aware software engineering, that is, to engineer software using OSSs best fit for the problem.

Categories and Subject Descriptors: D2.9 [**Software Engineering**]: Management—*Software process models*

General Terms: Management, Human Factors

Additional Key Words and Phrases: Organizational social structures, social context, users, cultural implications, social adaptivity, user perspective, information trust, knowledge management, governance, organizational decision-making, software organizations, social networks, social structures, communities, software practice

ACM Reference Format:
Tamburri, D. A., Lago, P., and van Vliet, H. 2013. Organizational social structures for software engineering. ACM Comput. Surv. 46, 1, Article 3 (October 2013), 35 pages.
DOI: http://dx.doi.org/10.1145/2522968.2522971

1. INTRODUCTION

1.1. Vision and Goals

Social interaction has evolved as a consequence of globalization [Martinelli 2007]. The impact of this evolution is deep and spread to all aspects of society (infrastructure, services, etc.) [Langhorne 2001]. For example, dynamic and unpredictable global demands in businesses transformed single supply chains into value-creating supply networks of corporate partnerships [Jetter et al. 2009]. Moreover, cloud computing rendered software and data increasingly pervasive, inexpensive, and globally accessible [Armbrust et al. 2010; Kshetri 2010]. Previous research also suggests the keyword "social" is rapidly growing in interest for software engineering [Chard et al. 2010; Herbsleb and Mockus 2003]. Indeed, from a social perspective, software engineers, stakeholders, and end-users form an Organizational Social Structure (OSS). Literature shows that quality support to developers' OSS influences project success [Cataldo and Nambiar 2009a; Cataldo et al. 2009] and final product quality [Nagappan et al. 2008; Cataldo and Nambiar 2009b, 2012]. Better OSS support would ease, for example, early detection of socio-technical incongruences (e.g., software risks) [Cataldo et al. 2008;

Authors' addresses: D. A. Tamburri (corresponding author), P. Lago, and H. van Vliet, VU University Amsterdam, De Boelelaan 1081a, The Netherlands; email: d.a.tamburri@vu.nl.

DOI: http://dx.doi.org/10.1145/2522968.2522971

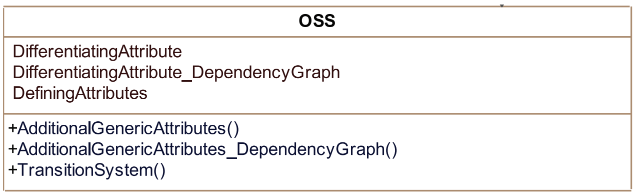

OSS
DifferentiatingAttribute
DifferentiatingAttribute_DependencyGraph
DefiningAttributes
+AdditionalGenericAttributes()
+AdditionalGenericAttributes_DependencyGraph()
+TransitionSystem()

Fig. 1. Key concepts within our study and their relations.

Lyytinen et al. 1998]. However, in the discipline of software engineering there is still little understanding on OSSs. Our goal in this work is to present the state-of-the-art in OSSs and provide instruments allowing practitioners to identify, select, analyse, or support the exact social structure they need, or will be part of, as software engineers. We found that each OSS type can be modelled by a set of distinctive attributes and their dependencies. These models allow reasoning on OSS types, to evaluate their fitness for software development purposes. These purposes include: (a) in software process monitoring, analyse the current OSS status (i.e., attribute values) for timely detection of socio-technical incongruences; (b) in postmortem analysis, identify the OSS attributes that led to failure, to define best practices.

We discuss the state-of-the-art in OSS and its implications, by referring to real-life development scenarios or previous and related research in software engineering.

1.2. Terminology

An *OSS* is the set of interactions, patterned relations, and social arrangements emerging between individuals part of the same endeavour [Wenger et al. 2002]. To clarify the critical concept of OSSs, let us consider Global Software Engineering (GSE) as an example. GSE is a software engineering practice entailing project teams spread across many timezones, for example, to achieve round-the-clock productivity [Herbsleb and Mockus 2003]. The people taking part in GSE process necessarily interact with others (stakeholders, colleagues, superiors, etc.) during this process. The emerging web of relations, dependencies, collaborations, and social interactions they will be part of is a set of nontrivial patterns (therefore a "structure") which entails people (therefore "social"). These people act as an organised whole, focusing on a particular goal (therefore "organizational").

Figure 1 provides an overview of our contributions to define OSS and support OSS-related decisions. In what follows, in *italic*, we explain each contribution. Univocal identification of OSSs is possible through a single *DifferentiatingAttribute* for every type. Also, each type is further characterised by *DefiningAttributes*, whose value is unique to it. In addition, we provide means to determine which attributes are influenced by the selection of a certain OSS type (i.e., through a *Graph* that plots dependencies between differentiating attribute and others). Moreover, we offer a set of patterns through which key OSS types can be combined (i.e., through an OSS *TransitionSystem*), and according to which certain OSS types can transit to other types, under certain circumstances (e.g., some attributes change value). Finally, we offer a set of attributes (and their dependencies) which are applicable (yet not required) to all OSSs. These are *GenericAttributes* that can further configure the chosen OSS to fit exactly the domain and problem at hand.

1.3. Structure of the Article

The rest of the article is structured as follows: Section 2 compares our study to others which investigate OSS attributes and characteristics in practice, to understand their impact. Section 3 explains the methods we used to attain our results as well as the primary studies we analysed. Section 4 presents our results. Section 5 discusses results usage, implications, and threats to validity. Section 6 concludes the article. The Appendix contains links to online material.

2. STUDYING OSS IN SOFTWARE ENGINEERING CONTEXTS

There are two immediate applications of our contributions in practice: (a) supporting software engineering research in the study of OSSs in practice; (b) supporting software engineering practitioners in OSSs-related decisions. To show the relevance of these activities, we investigated related work in software engineering. Looking at top conferences in software engineering (ICSE and FSE) and data mining, we found many studies that implicitly study OSSs.

First of all, studies on socio-technical congruence [Cataldo et al. 2008] could benefit from our contributions. Socio-technical congruence is the degree to which technical and social dependencies match, when coordination is needed [Cataldo et al. 2009]. Authors in Cataldo et al. [2009] present socio-technical congruence in formal terms and empirically investigate its impact on product quality. Similar works (e.g., Bird et al. [2009b] and Kwan et al. [2011]) strongly motivate the study of OSSs and their influence on socio-technical congruence. In Bird et al. [2009b] authors use social network analysis to investigate coordination among groups of developers with socio-technical dependencies. In our work we do not focus on such specific aspects of OSSs (e.g., member social dependencies [Kwan et al. 2011] for socio-technical congruence). Our scope is limited to providing a clear picture of what types of OSSs can be encountered, for example, while studying socio-technical congruence. This could help in understanding incongruences, their impact (e.g., through OSS attributes dependencies) and finding appropriate preventive governance (e.g., if formality of structure inhibits collaborativeness, changing community type may help loosen formality during software development).

In addition, organizational decision-making could benefit from our contributions. For example, offshoring is an organizational decision. Understanding if the current organizational layout of a company is performant (or even compatible) with offshoring is essential to “go global” [Cusick and Prasad 2006; Cataldo and Nambiar 2009a]. Organizational decisions change the OSS layout, and can be used to plan and support offshoring [Cataldo and Nambiar 2012]. We analyzed evidence in Cusick and Prasad [2006] and found that authors are truly suggesting that offshoring attempts should assume a “Formal Network” OSS type. In addition local sites should be governed by means of “Formal Groups”; this is consistent with findings from Cataldo and Nambiar [2012] and Cusick and Prasad [2006]. In this article we characterize OSS types, allowing practitioners to use them as reference (e.g., while applying results from Cusick and Prasad [2006]). It is out of our scope to provide an exhaustive treatment of OSSs’ application in software engineering.

Finally, many works similar to Cusick and Prasad [2006] are available in literature (e.g., Bird et al. [2009a], Turhan et al. [2009], and Meneely and Williams [2009]) which investigate the influence of organizational decisions on collaborations and product quality aspects (both in open- and closed-source ecosystems [Pinzger et al. 2008; Tosun et al. 2009; Datta et al. 2011; Witten et al. 2001]). Literature also shows that collaboration and social structure are critical in multisite development and linked to final product quality [Herbsleb and Mockus 2003; Nagappan et al. 2008; Cataldo and Nambiar 2009b; Bird et al. 2009a]. These studies motivate the need for OSSs in software engineering. Similar works could draw from our results a clear picture of possible organizational decisions, their mutual dependencies, and impact. Known attributes’ dependencies could support decision-making, by showing which attributes change with a certain decision. For example, management decisions suggested in Cusick and Prasad [2006] may positively influence observed aspects but negatively influence OSS lifecycle.

3. APPROACH AND PRIMARY STUDIES

3.1. Research Approach

To attain our results, we conducted a Systematic Literature Review (SLR) of OSSs using grounded theory [van Niekerk and Roode 2009]. We set out to answer two research questions:

(1) What types of OSSs can be distinguished in literature?
(2) What attributes can be identified for each type?

The approach can be organized in three steps: *(a)* create the set of articles to review (i.e., primary studies); *(b)* conduct the review; *(c)* analyze the data.
Step (a) was carried out through a systematic search protocol [Kitchenham et al. 2008]. The protocol is divided in three stages: (i) elaborate the search string; (ii) apply the string on chosen search engines; (iii) extract primary papers from search results by filtering out through exclusion criteria.

For stage (i) of the systematic search, the search string was determined by extracting relevant keywords directly from the research questions. To maximize the match between the research questions and the search string we consulted with an external expert in software engineering and organizations research. The following search terms were selected:

(1) “organization” $\vee$ “organizational” $\vee$ “company” $\vee$ “enterprise” $\vee$ “firm” $\vee$ “structure” $\vee$ “group”; (2) “social structure” $\vee$ “social network” $\vee$ “informal network” $\vee$ “informal group”; (3) “knowledge sharing” $\vee$ “knowledge management” $\vee$ “knowledge exchange” $\vee$ “knowledge transfer”; (4) “community of practice” $\vee$ “communities of practice” $\vee$ “knowledge community” $\vee$ “knowledge communities” $\vee$ “network of practice” $\vee$ “networks of practice”. The preceding terms were combined in the following search string.

$$[4 \vee (1 \wedge 2 \wedge 3)] \wedge [1 \wedge 2 \wedge 3 \wedge 4]$$

In stage (ii) of the systematic search, the string was applied to the following scholarly search engines: *Software Engineering* - ACM Digital Library, SCOPUS, IEEE Xplore Digital Library, Science Direct SCOPUS, SpringerLink and Wiley InterScience; *Knowledge Management* - EBSCO electronic library, JSTOR knowledge storage, ProQuest-ABI/Inform; *Multidisciplinary* - ISI Web of Sciences. Finally, during stage (iii) of the systematic search, the initial results were screened against inclusion and exclusion criteria. We selected 122 peer-reviewed documents as primary studies and later added 21 with subsequent searches. An overview is present online (see the Appendix). These documents span a wide array of fields: from cognitive ergonomics to social sciences to software engineering.

Table I provides an overview (with rationale) of the criteria we used for screening.

Step (b) and (c) of our SLR were carried out through a hybrid Glaserian-Straussian Grounded Theory (GT) approach [Corbin and Strauss 1990]. GT means to systematically apply a series of steps to allow a theory to emerge “spontaneously” from data (hence, “grounded”). This makes GT valid since the resulting theory is emergent from data rather than confirmed (or disproven). Each phase in GT is self-contained, incremental, and iterative [Corbin and Strauss 1990]. Our approach is structured as follows.

(1) *Open Coding* - (4 phases). (1) Pilot study: 28 primary studies were randomly selected to generate an initial set of codes. A second pass on the pilot papers was applied at the end of coding with the final list of codes, to minimize inconsistency;

Table I. Inclusion/Exclusion Criteria with Rationale

Inclusion Criteria	*Exclusion Criteria*
A study that is mainly about social structures in relation to any organization or practice. *Rationale: we are interested in types and attributes of organizational social structures. This implies that studies that are about organizational social structures are relevant to our research questions. For instance, studies that discuss community of practices within an organization.*	A study that is not about social structures in relation to an organization. *Rationale: Studies that do not investigate social structures in relation to an organization (i.e. a company or an enterprise) are beyond the scope of this research and should be excluded. For instance, a study that is about an online social networking application like Facebook without any relation to an organization.*
One of the main objectives of the study is to present a type of organizational social structure or a specific attribute of an organizational social structure. *Rationale: if one of the objectives of a study is to present a type or attribute, it is relevant to our research questions. For instance, a study that introduces an emerging type of social structure for project-oriented organizations or a study that discusses boundary-crossing attribute of networks of practice.*	A study which is marginally related to organizational social structures. *Rationale: For example, studies that focus on technologies related to organizational social structures or studies that are mainly related to statistics or social network analysis rather than social structures themselves.*
A study that is in form of a scientific paper (i.e. it must be peer reviewed and published by a scientific publisher). *Rationale: a scientific paper guarantees a certain level of quality and contains reasonable amount of contents. For instance, a journal paper.*	A study that does not discuss any specific type or attributes of organizational social structures. *Rationale: if a study does not discuss types and attributes of organizational social structures and only discusses them in general, it has no value to our research questions and should be excluded. For instance, studies about the importance of organizational social structures.*

(2) develop initial theory: based on the pilot study, an initial theory was generated; (3) constant comparison: the pilot study generated an initial set of 229 codes. These were organized into a hierarchy of codes based on emerging relations between concepts. Thus structured, the start-up list of codes was used to code the rest of the primary papers. Each paper was analyzed line by line with the list of codes. A code was applied iff it reflected a concept in a paragraph. This process is known as microanalysis [Onions 2006]. (4) constant memoing: along step 3, notes were kept to capture key messages, relations, and observations on the study[1].

(2) *Selective Coding* - (2 phases). (1) Axial coding: comparing the concepts coded led us to inductively generate relations among coded concepts (e.g., OSS types, attributes of OSS types, etc.); (2) aliasing: the definitions of all concepts coded were compared with each other to identify aliases.

(3) *Theoretical Coding* - (3 phases). (1) Data arrangement: we captured every portion of text that was coded with a certain code in a table. Five tables were extracted, each representing a core concept observed in the literature. (2) data modeling: the data was represented in a view, consisting of two diagrams. The first diagram shows the OSS as well as their relations. The second diagram shows all the attributes and relations found (clustered according to core concepts identified); (3) theoretical sampling: the diagrams and all the data at hand were analyzed and sorted, trying to identify recurrent patterns, underlying relations, and hidden meaning. Our observation was aided by standard analysis methods such as weighted frequency analysis (i.e., by analyzing the number of times each type was encountered

[1]The labels used in all the models are in the form "M$< x >.< y >$". The label schema locates the Y-th memo on the X-th table, for traceability (for all the material available online, refer to the Appendix).

Table II. Overview of OSS Types Clustered by Metatype

Community	*Network*	*Group*	*Team*
Community of Practice (CoP); Knowledge Communities (KC); Strategic Communities (SC); Informal Communities (IC); Learning Communities (LC); Problem Solving Communities (PSC);	Network Of Practice (NoP); Formal Networks (FN); Informal Networks (IN); Social Networks (SN);	Work Groups (WG); Formal Groups (FG);	Project Teams (PT);

weighted against the number of papers in which these were found) card sorting (by rearranging the hierarchy of types to let underlying relations show themselves), and conceptual modeling.

4. RESULTS

This section presents the overview of our three main results. As a first result, we found that each OSS type can be uniquely identified with a single key differentiating attribute. Additionally, OSSs are characterized by two key informations. The first information is a dependency graph, made of attributes that depend on the OSS's key differentiating attribute. The second information entails other attributes, whose value is unique to that specific OSS type. Finally, the explicit relations among OSS types are represented in a UML-style metamodel (see Figure 2).

The second result presents a set of attributes, applicable (yet, nonnecessarily) to all OSS types. A UML-style class diagram shows the dependencies among these generic attributes. The third result presents an OSS transition system and the analysis of the patterns which compose it. We found that it can be used in two ways. First, the transition system allows practitioners to predict OSS type changes. Analyzing and comparing the current state of an OSS (i.e., its defining attributes) to the state of others in the system, one can predict the shift between them. Second, these patterns represent empirical evidence of effective OSS combinations.

4.1. OSS Types and their Relations

The literature we analyzed discusses 26 OSS types. However, for the sake of space, we discuss the 13 most relevant OSSs[2]. Table II provides an overview of the 13 types, clustered according to their metatype (i.e., community, network, group, or team). Metatypes represent core categories in our results, found through Grounded Theory. Each OSS is described through a text and a table. The text explains the OSS's general characteristics quoting from relevant papers and comparing with other types. The table contains: (a) the OSS's key differentiating attribute (in bold); (b) the OSS's dependency graph (below the key differentiating attribute); (c) the other defining attributes for the OSS. The types are presented according to their metatype, and ordered by descending relevance in literature.

4.1.1. Communities. All communities are made for sharing. For example, in Communities of Practice (CoPs), people sit in the same location, sharing a practice or passion. In Strategic Communities (SCs), people share experience and knowledge with the aim of achieving strategic business advantage for the corporate sponsors. Communities are generally associated with situatedness, however, we have found explicit indication of this only for CoPs. Finally, all communities are expected to exhibit a certain goal, be it personal, organizational, or both. Here follows the detailed definitions.

[2]Relevance was computed through a weighted bibliometric count, following suggestions in [Librarya 2010].

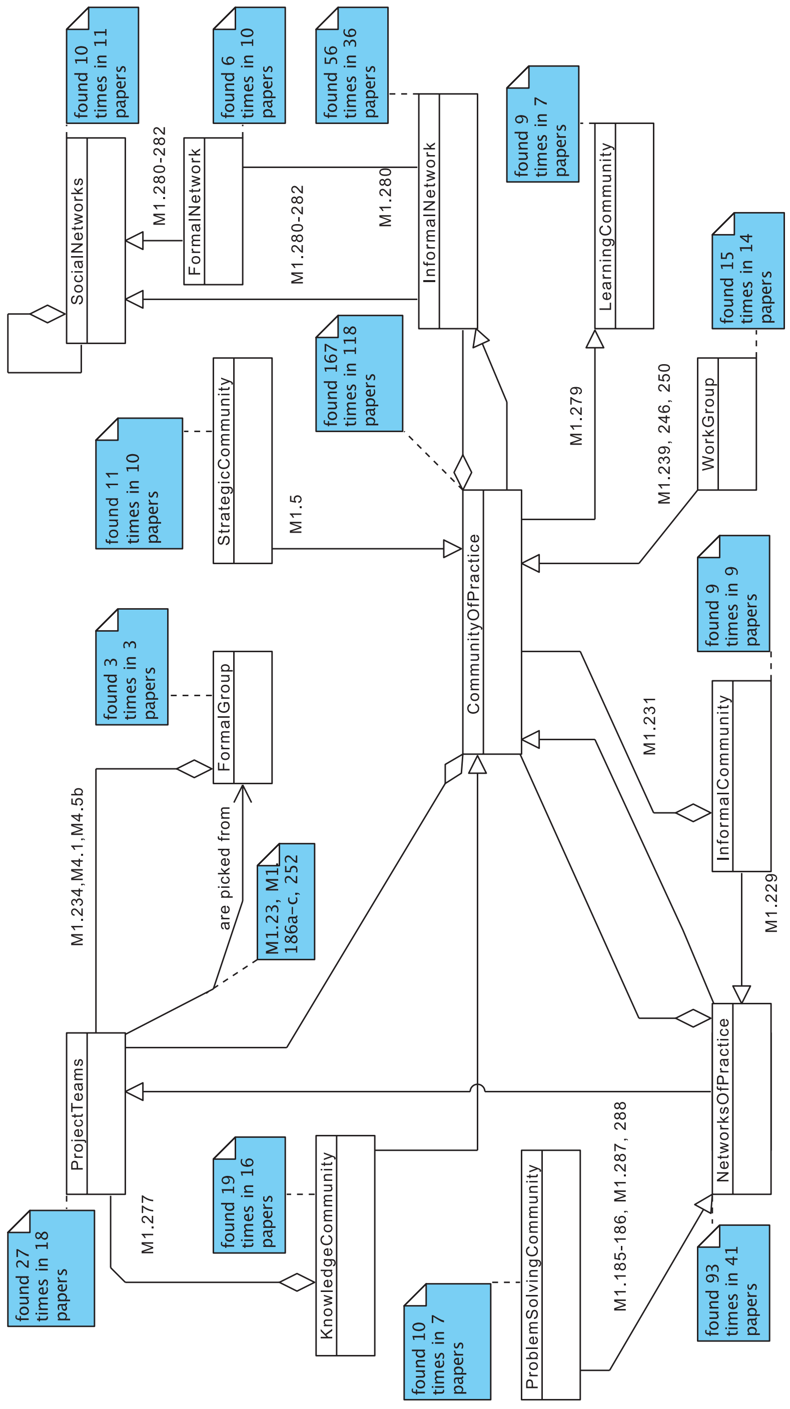


Fig. 2. Metamodel of OSS types.

(1) *Communities of Practice (CoP):* The key differentiating attribute for CoPs is situated sharing, or learning/sharing of a common practice (i.e., the attribute "situatedness" is a differentiator to identify a CoP). For example, the SRII events[3] are gatherings in which multiple CoPs (corporate and academic) meet physically to informally exchange best practices in services science. Defining attributes for CoPs are in Table I. Quoting Wenger and Snyder [2000]

"[CoPs] are groups of people informally bound together by shared expertise and passion for a joint enterprise. Engineers engaged in deep-water drilling, for example, consultants who specialize in strategic marketing, or frontline managers in charge of check processing at a large commercial bank."

A CoP consists of groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting frequently and in the same geolocation. As such, CoPs serve as scaffolding for organizational learning in one specific practice. They were found in 118 papers but many other papers use this fundamental type to define and characterize other types, as extensions or specializations of it (e.g., see the generalization link between CoPs and NoPs in Figure 2). Within traditional software engineering, CoPs are constantly being exploited (most of the time unknowingly) by single (or more) corporation(s) to aggregate their employees together and allow them to exchange ideas and gather best practices. Literature presents evidence that CoPs are indeed managed by a leadership circle with egalitarian or loosely stratified forms [Klein et al. 2005]. This therefore differentiates CoPs from LCs and KCs, since literature stresses for them the presence of strong leadership and its inclination towards the accumulation and dissemination of culture [Blankenship and Ruona 2009; Ruikar et al. 2009]. The presence of egalitarian forms suggests that a CoP looks much like a graph of peers, where nodes are equally shaped and characterized.

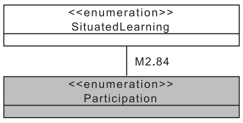
(2) *Knowledge Communities (KC).* The key differentiating attribute for KC is the visibility of its contents and results produced (i.e., the attribute "Visibility" is a differentiator to identify a KC). For example, global software engineering efforts can be represented as KCs, since within them every branch of the development (both in terms of knowledge exploited, and results produced) should be visible to all others [Richardson et al. 2010]. Defining attributes for KCs are in Table IV. Quoting from Dickinson [2002]

"Virtual knowledge communities are organized groups of experts and other interested parties, who exchange knowledge on their field of expertise or knowledge domain in cyberspace within and across corporate and geographical borders. Virtual knowledge communities focus on their knowledge domain and over time expand their expertise through collaboration. They interact around relevant issues and build a common knowledge base."

Essentially KCs are groups of people with a shared passion to create, use, and share new knowledge for tangible business purposes (e.g., increased sales, increased product offer, clients profiling, etc.). The main difference with other types is that KCs are expected (by the corporate sponsors) to produce actionable knowledge (knowledge which can be put to immediate action, e.g., best practices, standards, methodologies, approaches, problem-solving-patterns, etc.) into a specific business area. What is interesting is that these are expected to produce actionable knowledge, even if corporate sponsors do not formally state the expectations. On the contrary, management practices are usually put in place to make sure this expectation comes true. Moreover, they are not limited to use electronic communication and collaboration means (such as NoPs) but rather they can colocate meetings or

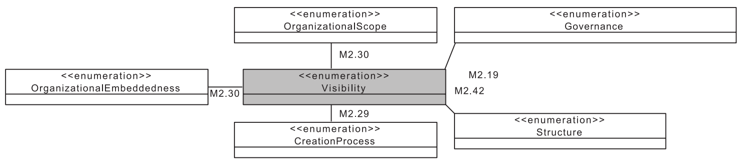
[3]www.theSrii.org.

Table III. CoP Defining Attributes

Attribute	***in Communities of Practice***
"Situatedness"	**For CoPs situated learning or practice is expected, informal and collective, in that members learn through reciprocity within the (localized) CoP.**
"Situated Learning dependency graph"	 <<enumeration>> SituatedLearning M2.84 <<enumeration>> Participation
"Reciprocity"	According to literature, reciprocity of knowledge exchange is common practice in CoPs, since members carry out knowledge activities with each other, almost exclusively.
"Shared Repository"	Literature shows the enforced presence of a shared repository in every CoP, aimed at containing and easing the distribution of knowledge.
"Members Official Status"	Literature indicates that there is no official status for CoP members other than participation. A professional spontaneously participates within the CoP and is accepted as a member.
"Shared Practice"	Literature shows clearly that CoPs are indeed based around the sharing of a common practice.
"Membership Creation Process"	Literature indicates two possible ways for membership creation to happen in CoPs, namely: Self Selection, i.e. autonomous self-appointment; Promotion, i.e. new members are elected by promotion from organizational sponsors or CoP management.
"Health"	For CoPs, literature indicates this attribute to be proportional to good governance, i.e. to the correct maintenance of the CoP through governance practices.
"Knowledge Activities'	Within a CoP literature stresses all possible knowledge activities can be carried out.
"Perceived Competitiveness"	Literature identifies CoPs as non-competitive.
"Creation Process"	CoPs' creation process is spontaneous and emergent.
"Visibility"	CoPs are invisible according to literature. Therefore only visible to who knows of its existence already
"Communication Media"	Literature suggests CoPs are commonly using all identified media (e.g. electronic, paper, publicity, etc.).
"Organizational Goal(s)"	Four organizational goals were identified in literature for CoPs, namely: Decrease Learning Curve, Prevent Employee Turnover, Manage Information Flow, Produce Actionable Knowledge.
"Cognitive Distance"	Literature identifies cognitive distance for CoPs as high.
"Contract Value"	Literature indicates contract value for CoPs is limited in that organizational sponsors have little or no expectations from CoPs.
"Context Openness"	Within a CoP, egalitarianism is enforced thereby strengthening the equality of communication.
"Size"	A CoP is indicated in literature as being big, i.e. with a total number of elements between 100 and 1000.
"Orientation"	A CoP can be employed in both the possible orientations, namely, strategic and operational.

workshops to devise or explore new ideas (much like CoPs). Literature also stresses the presence of management and its key role in fostering community visibility [Lee and Williams 2007]. This suggests that a KC, in contrast to a CoP, looks much like a mixed hierarchy/heterarchy in which members and leadership have clear-cut distinctions and roles [Dickinson 2002]. Another remarkable difference between KCs and CoPs is that CoPs need situatedness to exist, while KCs are often associated

Table IV. KC Defining Attributes

Attribute	*in Knowledge Communities*
"Formal Status"	Literature identifies knowledge communities as aggregates of people which have a formal and acknowledge status for the organizational sponsor.
"Visibility"	**Knowledge communities need increased visibility (and management practices to maintain it) to make their practices and results known to the organizational context.**
"Visibility dependency graph"	 <<enumeration>> OrganizationalScope <<enumeration>> Governance <<enumeration>> OrganizationalEmbeddedness M2.30 M2.30 <<enumeration>> Visibility M2.19 M2.42 M2.29 <<enumeration>> CreationProcess <<enumeration>> Structure
"Management Practices"	Since the organizational sponsor has precise expectations from the KC, it applies a combination of management practices to its operational effectiveness.

with virtual spaces, for example, as described in Rosso [2009] for the Apache WebServer knowledge network. Given these differences, the topology of a KC would also differ from that of a CoP, since its nodes would neither be constrained to a single geolocation nor be of a single type. This difference also suggests that while CoPs may be more frequent within the boundaries of a single corporation, KCs may be transversal in nature, encompassing multiple experts from multiple (partner) organizations [Dickinson 2002].

(3) *Strategic Communities (SC).* The key differentiating attribute for SCs is the contract value (i.e., the attribute "Contract Value" is a discriminator to identify an SC) that should be maintained (e.g., by generating ad hoc best practices) or generated (e.g., by analyzing strategic market sections ripe for growth). In software engineering, for example, SCs are commonly associated with mission-critical systems that need 24/7 availability, for example, ESA, the European Space Agency, uses so-called "tiger teams" to investigate and solve specific software problems during mission engineering; tiger teams are selected from a loosely specified (cross-organizational) strategic community of experienced practitioners in software engineering and research: their focus is to support specific missions, safe-guarding the people and infrastructure involved (i.e., the contract value is human, monetary, and technological). Defining attributes for SCs are in Table V. Quoting from Blankenship and Ruona [2009].

"Strategic communities are formalized structures that consist usually of a limited number of experts within a single organization. These share a common, work-related interest into producing unexpected ideas to achieve strategic advantage [Hustad 2010]. These communities are intentionally created by the organization to achieve certain business goals."

SCs consist of meticulously selected people, experts in certain sectors of interest to a corporation or or a set of organizational partners tied with formal nondisclosure agreements. These try to proactively solve problems within strategic business areas of the organizational sponsor. Compared to the others, SCs indeed seem very similar to KCs in that they are targeted towards obtaining business advantage. Similarly to WGs, SCs differ from KCs in their level of granularity, formality, and openness of membership as well as contract value [Hansen 1999]. KCs act on wider business sector (pro project) than SCs, and their membership is usually open but subject to management practices (e.g., evaluation). SCs, on the other hand,

Table V. SC Defining Attributes

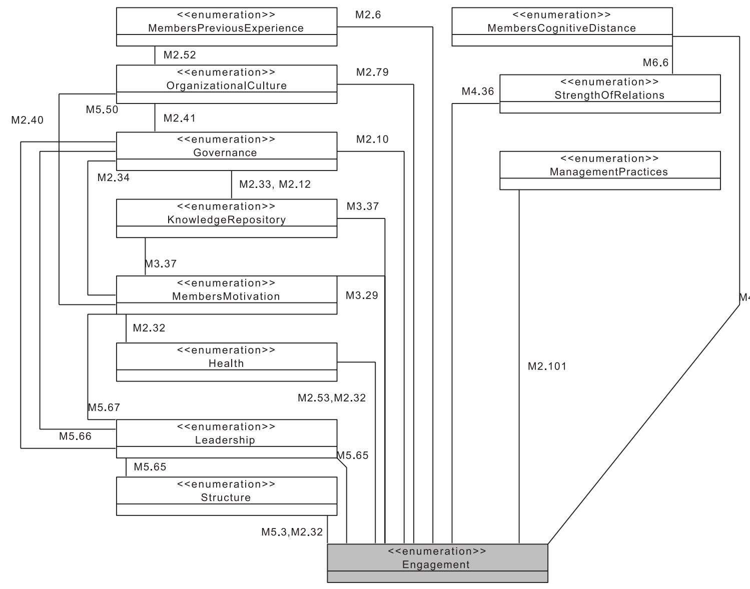
Attribute	***in Strategic Communities***
"Formal Status"	Literature shows that SCs have a strongly formal acknowledged status by the organizational sponsor. Indeed SCs are specifically created by the sponsor to pursue a specific business goal.
"Organizational Sponsor"	In line with their formal status, strategic communities are fathered by the sponsors they are trying to support. The organizational sponsor provides fostering support after fathering the organization.
"Members Previous Experience"	People appointed to a SC need to have a strong background in order to be selected for participation or membership.
"Personal Goal"	Every member shares with the other a deep interest on tackling a common work-related issue. This issue is personal since it inhibits the work productivity of the single individuals (e.g. tackle the geographical distance of two sites which need to collaborate).
"Contract Value"	**Since the SC is needed to tackle a work-related issue, the business gain from its success is strongest. As a consequence, the contract value invested by the organizational sponsor is highest.**
"Contract Value dependency graph"	-

concentrate on very tight business missions, commanded by an organizational sponsor. They are expected to produce strategic advantage through innovation or best practices [Erik Andriessen 2005]. Also, they are different from most other communities since the members are appointed based exclusively on experience, personal success, career titles, and anything that can justify their status of experts [Ruuska and Vartiainen 2003]. Given the right social network analysis tools, one could identify a strategic community by investigating the cliques of experts communicating in a certain business sector. Given that membership is enforced through management, the node types in SCs would be extremely similar if not identical. Finally, sometimes in their lifecycle, SCs are similar to (and sometimes identified with) ICs [Erik Andriessen 2005]; reasons can vary (e.g., SCs could be dormant, waiting for organizational objectives to fulfill).

(4) *Informal Communities (IC).* The key differentiating attribute for ICs is the high degree of member engagement (i.e., the attribute "Member Engagement" is a discriminator to identify an IC). An example in software engineering can be seen in the Agile Movement, whose success is decided explicitly by the contributions of its members (i.e., their engagement in actively disseminating and using "agile" practices). In addition, open-source communities considered in Witten et al. [2001] and Meneely and Williams [2009] also match our definition of an IC, adding to it some feats of CoPs. Defining attributes for ICs are stated in Table VI. According to Dickinson [2002] the IC term "was coined in 1968 by the Internet pioneers J.C.R. Licklider and R.W. Taylor who, when wondering what the future online interactive communities would be like, predicted that they would be communities of common interest, not of common location made up of geographically separated members. Their impact will be great both on the individual and on society."

ICs are usually sets of people part of an organization, with a common interest, often closely dependent on their practice. They interact informally, usually across unbound distances, frequently on a common history or culture (e.g., shared ideas, experience etc.). The main difference they have with all communities (with the exception of NoPs) is that their localization is necessarily dispersed so that the community can reach a wider "audience". Loosely affiliated political movements (such as Greenpeace) are examples of ICs: their members, disseminate their vision (based on a common idea, which is the goal of the IC). Also, the success of the IC

Table VI. IC Defining Attributes

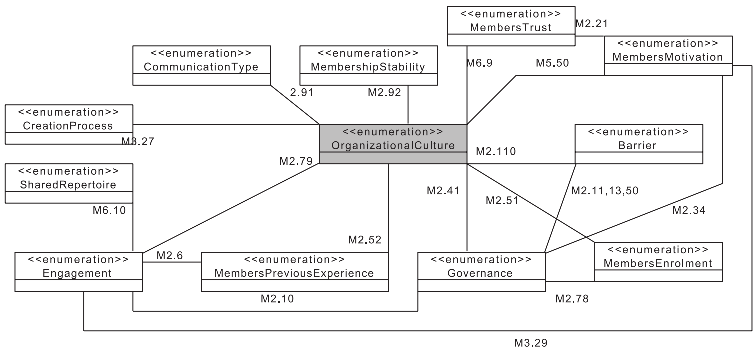
Attribute	***in Informal Communities***
"Members Engagement"	**The engagement of members in informal communities is kept high since there is no formal arrangement linking members to the community. High engagement acts as a glue for the community.**
"Members Engagement dependency graph"	 <<enumeration>> MembersPreviousExperience <<enumeration>> MembersCognitiveDistance <<enumeration>> OrganizationalCulture <<enumeration>> StrengthOfRelations <<enumeration>> Governance <<enumeration>> ManagementPractices <<enumeration>> KnowledgeRepository <<enumeration>> MembersMotivation <<enumeration>> Health <<enumeration>> Leadership <<enumeration>> Structure <<enumeration>> Engagement M2.6 M2.52 M2.79 M6.6 M4.36 M5.50 M2.41 M2.40 M2.10 M2.34 M2.33, M2.12 M3.37 M3.37 M3.29 M4.29 M2.32 M2.101 M2.53,M2.32 M5.67 M5.66 M5.65 M5.65 M5.3,M2.32
"Personal Goal"	The informal community promotes mutual learning. The practice held within the community itself favors individuals part of the community to share their own insight with others and learn, in turn.

is exclusively tied to members' engagement since their effort is what drives the community to expand, disseminate ideas, gather members, and so on. One characteristic which also differentiates ICs from other communities is the assumption of self-organization. This makes them very similar to networks rather than communities [Mentzas et al. 2006]. From a topological perspective, informal communities would exhibit self-similarity, as a consequence of self-organization [Hustad 2010]. However, nodes would be egalitarian (much like in CoPs) and highly dispersed (much like in NoPs). We weren't able to determine if ICs can be seen as a mid-stage in the transition from a CoP (purely situated) to an NoP (purely geodispersed).

(5) *Learning Communities (LC).* What differentiates this particular type from others is its explicit goal of incrementing, tracking and maintaining the organizational culture of an organization (the attribute "Organizational Culture" is differentiator when identifying an LC). A perfect example of an LC within software engineering is the BORLAND Academy[4]. BORLAND Academy and similar institutions are used by the organization's practitioners to learn about its current practices, methods, processes, and tactics, as well as refine these over time as their product portfolio evolves. Defining attributes are contained in Table VII. LCs are, quoting Blankenship and Ruona [2009]

[4]Recently discontinued after BORLAND's acquisition by MicroFocus.

Table VII. LC Defining Attributes

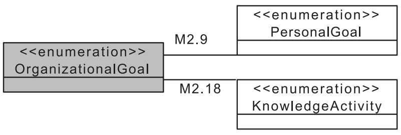
Attribute	*in Learning Communities*
"Leadership"	in learning communities leadership must be strong to maintain motivation, and steer the learning practices.
"Organizational Culture"	**Learning practices promoted by leadership increase the organizational culture which the learning community is maintaining. The organizational culture maintained by a learning community is as strong as the community's lifespan.**
"Organizational Culture dependency graph"	 <<enumeration>> MembersTrust <<enumeration>> CommunicationType <<enumeration>> MembershipStability <<enumeration>> MembersMotivation <<enumeration>> CreationProcess <<enumeration>> OrganizationalCulture <<enumeration>> Barrier <<enumeration>> SharedRepertoire <<enumeration>> Engagement <<enumeration>> MembersPreviousExperience <<enumeration>> Governance <<enumeration>> MembersEnrolment M2.21 M6.9 M5.50 2.91 M2.92 M3.27 M2.79 M2.110 M6.10 M2.41 M2.51 M2.11,13,50 M2.34 M2.52 M2.6 M2.10 M2.78 M3.29

"structures that provide space for learning and sharing knowledge. Much of the workplace learning community literature is situated within the education literature, where the structure is referred to as a Professional Learning Community (PLC). Stoll, Bolam, McMahon, Wallace, and Thomas (2006) agreed that although there is no universal definition for PLCs, there is international consensus that a PLC is a Ògroup of people sharing and critically interrogating their practice in an ongoing, reflective, collaborative, inclusive, learning-oriented, and growth-promoting way".

LCs provide a space for pure learning and explicit sharing of actionable knowledge (i.e., skills). In a learning community, the leadership is expected to steer the community's practices and membership is subject to approval and tied to the learning objectives given to the member [Bogenrieder and Nooteboom 2004]. Each developed or exchanged practice must become part of the organizational culture. Topics of learning are important for both business sponsors and participants (e.g., personal development skills, management skills, time optimization skills, etc.) [Ruuska and Vartiainen 2003]. The nature of discussed topics makes it extremely different from other communities: most other communities have either personal or organizational goals, while LCs have both and pursue them equally. Maintaining and transmitting organizational culture is the chief aim of an LC and this makes it even more specific than PSCs. From a topological perspective, LC would appear as directed structures (e.g., directed graphs) since learning usually takes place one way only, that is, from learning manager to learner [Ruuska and Vartiainen 2003].

(6) *Problem Solving Communities (PSC).* The key differentiating attribute for PSCs is their goal, that of solving a specific problem in the scope of an organization or corporate sponsor (i.e., the attribute "Organizational Goal" is a differentiator for PSCs). For example, during the Apollo 11 mission, NASA adopted specific groups of action, to solve issues which impeded or limited the mission success, as well as to establish critical problem-solving practices [Riley and Dolling 2011]. These groups of action could be considered as PSCs. Defining attributes are contained in Table VIII. PSCs

Table VIII. PSC Defining Attributes

Attribute	***in Problem Solving Communities***
"Membership Official Status"	a peculiar characteristic of problem solving communities is their potential of having both formally participating members and informal/occasional participants as needed. The community, since its targeted to a specific problem or issue, upholds the participation of (occasional) professionals which can help or consult with the problems at hand.
"Organizational Goal"	**problem solving communities are bent to resolving specific issues of relevance to their organizational sponsors. They are explicitly created to address a specific goal (i.e. solving a specific problem).**
"Organizational Goal dependency graph"	 <<enumeration>> PersonalGoal M2.9 <<enumeration>> OrganizationalGoal M2.18 <<enumeration>> KnowledgeActivity

consist generally of many geographically and organizationally dispersed employees of the same discipline. They focus on mitigating a specific, well-defined hazard (e.g., communities of anticontamination engineers). Quoting from Hustad [2010]

"The problem-solving network is a distributed network of practice which meets the criteria of an expert group. The network provides resources in terms of help-desk functions where participants of the network support other colleagues by giving them special advice as regards particular business problems. In addition, participating in this kind of network ensures collaborative learning among the participants of the network."

In comparison to other types, a PSC can be seen as a specific instance of a Strategic Community focused on a particular problem. One would expect this community to be formal in nature. Contrarily we found that they emerge as informal, since informality aids immediate problem-solving processes [Allen et al. 2007], since the goal in PSCs is not achieving business advantage, but rather solving a critical problem (either immediate, demanding, often occurring, or disastrous). Much like KCs and ICs, PSCs use both face-to-face digital technologies (e.g., forums) to enact problem-solving [Uzzi 1997]. From a topological perspective, a PSC would appear very similar to an IC where organizations acknowledge nodes' membership and provide problems to be solved (i.e., the organization is an active peer in the PSC). This characteristic makes the nature of PSCs very peculiar, since it's the only community type in which organizational sponsors are themselves members, they possess the problems, and other members interact with them in egalitarian forms. Additional research should be invested in representing PSCs with a 2-mode network [Rosso 2009] and studying this condition further.

4.1.2. Networks. Networks suggest the presence of digital or technological support tools, however, we found this explicitly only for NoPs. All network types are used by an organization to increase reachability, either through formal means (in FNs) or through informal ones (in INs) or through customized forms of boundary spanning (e.g., in NoPs). Here follow detailed definitions.

(7) *Networks of Practice (NoP).* The one differentiating attribute for NoPs is informality in geolocalized practice (i.e., the attribute "geodispersion" is a differentiator to identify an NoP). In global software engineering many (virtual) teams collaborate together through the Internet across timezones, with specific networks (e.g., VPNs) and with strong management and governance policies in place. However, the social

structure in GSE is not fully specified nor supported and matter of current research Tamburri et al. [2012]. NoPs' defining attributes are in Table IX. Quoting from Hustad [2010].

"NoP comprises a larger, geographically dispersed group of participants engaged in a shared practice or common topic of interest [...] CoPs and NoPs share the characteristics of being emergent and self-organizing, and the participants create communication linkages inside and between organizations that provide an "invisible" net existing beside the formal organizational hierarchy".

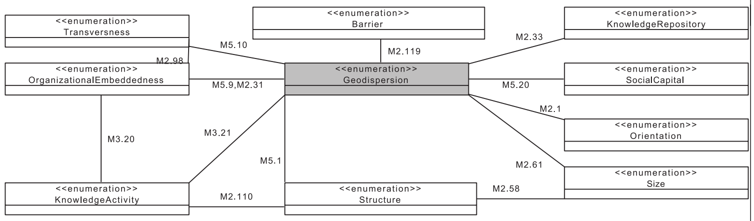
An NoP is a networked system of communication and collaboration that connects CoPs (which are localized). In principle anyone can join it without selection of candidates (e.g., OpenSource forges are an instance of NoP [Parreiras et al. 2004]). NoPs have a high geodispersion, that is, they can span geographical and time distances alike. The high geodispersion increases their visibility and the reachability by members. An unspoken requirement for entry is the expected IT literacy of members. IT literacy must be high since the tools needed to take part in NoPs are IT based (e.g., microblogs, forums, hang-outs, etc.). NoPs are much like CoPs, since they share repositories of knowledge through their members. Different from CoPs, NoPs can be seen as IT-enabled global networks, since their chief aim is to allow communication (and collaboration) on the same practice through large geographical distance. In the scope of software engineering, NoPs have been used massively within global software engineering (mostly unknowingly or with limited support). For example, Bird et al. [2009b] discuss socio-technical networks in software engineering. Based on our definitions such networks sit at the intersection between SNs and NoPs, since they address a specific practice but show a strong social connotation. Works similar to that in Bird et al. [2009b] could benefit from a more detailed overview of both SNs and NoPs, for example, to understand which of their characteristics are supporting failure prediction. Moreover, socio-technical congruence, as expressed in Cataldo et al. [2008], could be studied monitoring the organizational attributes that influence collaboration: this could lead to better understanding of what hinders global developers' productivity.

(8) *Informal Networks (IN).* The key differentiating attribute for INs is the type of interaction that binds its members (i.e., the attribute "members interaction" is a differentiator to identify an IN). Informal networks have existed in software engineering since its very beginning: all people participating within any software process collaborate and interoperate within a web of social ties that could be defined as an IN. Also, in academia, the informal and loosely coupled set of research communities can be considered a world-wide informal network. Defining attributes for INs are contained in Table X. Simple networks emerge based on the relationships that individuals form with others. They are the building blocks from which other social structures may emerge. Anyone can join an IN, since there are no formal subscription processes: membership is often based on collective acceptance of a certain individual (e.g., establishment of friendship between flat-mates). Moreover, INs are essential foci for information exchange. Quoting Vithessonthi [2010].

"The literature on social networks suggests that an informal social network can play a key role in enhancing organizational learning since social networks can be a source of information (Liebeskind et al., 1996). For example, Liebeskind et al. (1996) have found that social networks tend to extend the scope of learning and help the integration of knowledge possessed by two firms due to their collaboration".

As compared to the other types, INs can be seen as looser networks of ties between individuals that happen to come in contact in the same context. The driving force of the informal network is the strength of these ties between members. Finally, an IN differs from other types since it does not use governance practices but its success

Table IX. NoP Defining Attributes

Attribute	***in Networks of Practice***
"Boundary Spanning"	Literature identifies boundary spanning as a very common practice in NoPs. NoP's have larger geographical dispersion and therefore larger context with which to interact.
"Members Motivation"	Literature identifies members' motivation as high in NoPs.
"Members Selection Process"	Selection for the entitlement to a Network of Practice is said to be driven by entry requirements.
"Support Tool"	Given their nature, literature identifies many support tools that characterize a NoP. These are: Digital Communication Platforms, Persistent Computer Networks as well as Interactive on-line communities.
"Communication Openness"	Communication is identified as Open for NoPs. Therefore even professionals which are not strictly part of the network as members, might make use of its resources (e.g. online community forums which can be consulted by anyone).
"Management Practices"	Strong management practices are needed to maintain a NoPs.
"Knowledge Type"	Literature conveys that NoPs are particularly good to harvest tacit knowledge in the minds of the members (e.g. posting a quesiton on on-line forums and everybody answering with their own insight).
"Geodispersion"	**Literature suggests that the whole NoP is geodispersed, i.e. every node can be distant from others in both time and space.**
"Geodispersion dependency graph"	 <<enumeration>> Transversness <<enumeration>> Barrier <<enumeration>> KnowledgeRepository M5.10 M2.98 M2.119 M2.33 <<enumeration>> OrganizationalEmbeddedness M5.9,M2.31 <<enumeration>> Geodispersion M5.20 <<enumeration>> SocialCapital M2.1 <<enumeration>> Orientation M3.20 M3.21 M5.1 M2.61 <<enumeration>> Size <<enumeration>> KnowledgeActivity M2.110 <<enumeration>> Structure M2.58

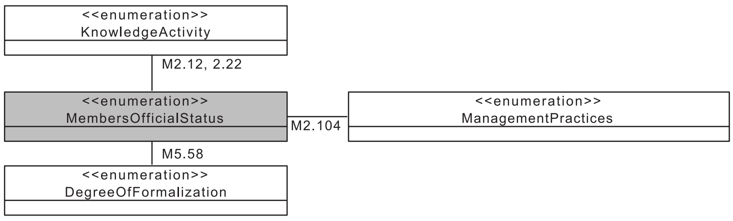
is solely based on the emergent cohesion its members have (therefore its success depends on the type of people that the IN comes into contact with and assimilates, rather than its internal dynamics).

(9) *Formal Networks (FN).* In formal networks memberships and interaction dynamics are explicitly "made" formal by corporate sponsors (i.e., the attribute "Membership Official Status" is differentiator when identifying an FN). Conversely, although formally acknowledged by corporate sponsors, FGs are (commonly) informal in nature, and are grouped for governance or action. An example in software engineering is the OMG (Object Management Group): it is a formal network, since the interaction dynamics and status of the members (i.e., the organizations which are part of OMG) are formal; also, the meeting participants (i.e., the people that corporations send as representatives) are acknowledged formally by their corporate sponsors. An even more interesting example of an FN is the structure emerging between corporate partners with offshoring agreements. They constitute an FN (e.g., as discussed in Cusick and Prasad [2006]) since both partners need to agree on their membership status through formal agreements and interoperation procedures. Moreover, each site needs to be governed with clear role definitions and responsibilities. In addition staff is usually managed as FGs as suggested in Cusick and Prasad [2006], this implies that FNs can be seen as virtual counterparts connecting local FGs, for example, in offshoring partnerships. We did not find any indication of inheritance between FGs and FNs. Defining attributes are contained in Table XI. FNs are, according to Allen et al. [2007],

Table X. IN Defining Attributes

Attribute	*in Informal Networks*
"Members Interaction"	**interaction in informal networks is intended as a social and informal interaction between individuals**
"Members Interaction dependency graph"	-
"Critical Success Factor"	a critical success factor in informal networks is a strong set of ties between the whole network and its members. This can keep motivation and participation high.
"Organizational Sponsor"	the organizational sponsor, where present. should limit its interactions with the community to mere participation. Its interference should be reduced to an essential minimum only.
"Connectedness"	The connectedness of participants depends on the degree of informality and organizational embeddedness the network has to an organizational sponsor. Therefore the connectedness in an informal network is custom.
"Members Cohesion"	cohesion of members is based on mutual need. if strength of ties is kept high then as a consequence mutual need is high. Therefore members cohesion is also high.
"Trust In The Context"	trust in the context is assumed to be present in an informal network. given the informality of definition trust is assumed and not maintained.
"Trust In Partner"	the same goes for the trust each member retains on its connections. Since it's an informal establishment, trust is assumed, not maintained or guaranteed in any way.
"Organizational Embeddedness"	embeddedness practices should be limited to encouragement of activities.

Table XI. FN Defining Attributes

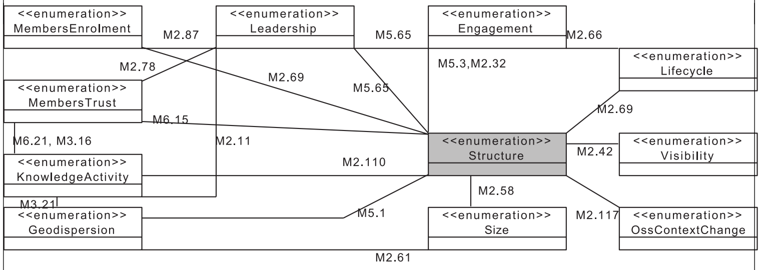
Attribute	*in Formal Networks*
"Membership Official Status"	**the status of formal networks' members is officially and formally recognized by the organizational sponsor.**
"Membership Official Status dependency graph"	 <<enumeration>> KnowledgeActivity M2.12, 2.22 <<enumeration>> MembersOfficialStatus M2.104 <<enumeration>> ManagementPractices M5.58 <<enumeration>> DegreeOfFormalization
"Creation Process"	Creating a membership in a formal network requires an invitation or some kind of formal appointment by the organizational sponsor. Also, the organizational sponsor usually nominates management for the formal network or uses management teams internal to the organization.

"formal [social] networks are those that are prescribed and forcibly generated by management, usually directed according to corporate strategy and mission".

Within FNs, members are rigorously selected and prescribed. They are forcibly acknowledged by management of the network itself. Direction is carried out according to corporate strategy and its mission is to follow this strategy.

(10) *Social Networks (SN).* In social sciences, the concept of social networks is often used interchangeably with OSSs. Since every OSS type can be defined in terms of SNs, there is no distinctive difference with other types; rather, SNs can be seen as a supertype for all OSSs. To identify the presence of an SN (or OSS) it is sufficient to be able to split the structure of an observable set of organizational

Table XII. SN Defining Attributes

Attribute	*in Social Network Sites*
"Goals"	A social network has two main goals: the construction of shared knowledge and the harnessing of internet-based technologies to enable socially aware networking
"structure"	**The structure of a social network can be divided into two: macrostructure details regard the highest level of abstraction of a social network while microstructure details characterize the nature, type and attributes of the social ties of single "nodes" in the social network**
"structure dependency graph"	 <<enumeration>> MembersEnrolment <<enumeration>> Leadership <<enumeration>> Engagement <<enumeration>> Lifecycle <<enumeration>> MembersTrust <<enumeration>> KnowledgeActivity <<enumeration>> Structure <<enumeration>> Visibility <<enumeration>> Geodispersion <<enumeration>> Size <<enumeration>> OssContextChange M2.87 M5.65 M2.66 M2.78 M2.69 M5.65 M5.3,M2.32 M2.69 M6.15 M6.21, M3.16 M2.11 M2.110 M2.42 M2.58 M3.21 M5.1 M2.117 M2.61

patterns (e.g., organization interactions, teams' interactions, etc.) into macrostructure (i.e., structure of social ties and interactions in the large [Johnsen 1985]) and microstructure (i.e., structure of social ties and interactions at the single social-agent level [Fershtman and Gandal 2008]). The defining attributes for SNs are contained on Table XII. Of particular interest is the definition present in Hatala and Lutta [2009]

"Social structure can be viewed as a set of actors with the additional property that the relational characteristics of these networks may be used to interpret the social behavior of the individuals involved".

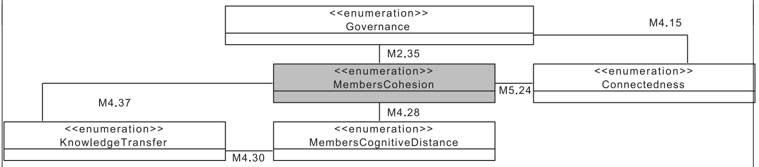
SNs represent the emergent network of social ties spontaneously arising between individuals who share, either willingly or not, a practice or common interest on a problem. SNs act as a gateway to communicating communities. With the advent of internetworking technologies, they are now explicitly supported by technologies and massively used in software engineering (e.g., LinkedIn, Academia.Edu, Google+, etc.).

4.1.3. Groups. Groups are tightly knit sets of people or agencies that pursue an organizational goal. Cohesion in groups can be an activation mechanism (to increase engagement, as in WGs) or a way to govern people more efficiently (e.g., in FGs). Here follow detailed definitions.

(11) *Workgroups (WG).* The key differentiating attribute for WGs is the cohesion of their members: they need to work in a tightly bound and enthusiastic manner to ensure success of the WG (i.e., the attribute "Members Cohesion" is a differentiator to identify a WG). The IFIP WG 2.10 on software architecture[5] is a WG, since its effort is planned and steady (i.e., cohesive), as well as focused on pursuing the benefits of certain organizational sponsors (e.g., IEEE, in the case of IFIP). Table XIII contains their defining attributes. WGs are defined as groups of individuals who work together on a regular basis to attain goals for the benefit of (potentially multiple) corporate sponsors. In Soekijad et al. [2004],

[5]http://www.softwarearchitectureportal.org/.

Table XIII. WG Defining Attributes

Attribute	***in Work Groups***
"Membership Formality"	The formality of a WG is established and acknowledged by an explicit organizational sponsor which establishes both the operative goals of the workgroup and the expected outcomes of each effort track it must be involved in.
"Members Cohesion"	**Cohesion practices within the WG must be well defined to allow its productivity to remain high. Besides establishing goals for the WG, the organizational sponsor should also define requirements that the WG must adhere to, in order to operate exactly on the issues it is targeted to issue. Both requirements and goals maintain the cohesion of members high.**
"Members Cohesion dependency graph"	 <<enumeration>> Governance M4.15 M2.35 <<enumeration>> MembersCohesion <<enumeration>> Connectedness M5.24 M4.37 M4.28 <<enumeration>> KnowledgeTransfer <<enumeration>> MembersCognitiveDistance M4.30

"three characteristics [of WGs] are central: [they have] multiple goals, multiple compositions, and multiple results".

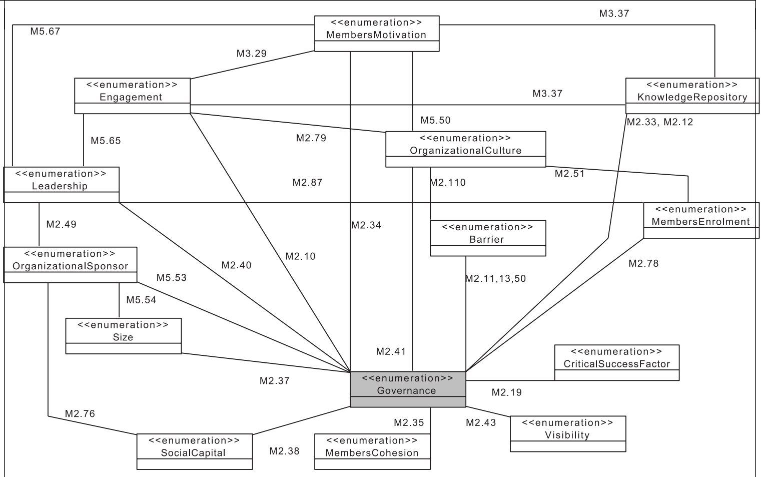
WG's goals can be multiple and spanning an array of organizational factors. Therefore, the expected or produced benefits can be wide as well. What's also fundamental is that a WG is always acknowledged and supported by organizational sponsor(s). Differentiating it from other types is its level of granularity: PTs act on specific projects (i.e., domain-specific, well-specified problems, with a clear set of goals and complementary sets of skills) while KCs focus on specific business areas and goals. A WG has a wider agenda and uses experts with similar skills and interests to tackle strategic issues, for example, developing practices that can span multiple business areas or observing the enterprise to drive organizational changes. Software engineering practitioners have been using WGs to generate standards and evaluate practices for standardizations. Although these examples suggest that WGs cannot be used to develop software, in Pinzger et al. [2008] authors observe that software developers should "[...] follow a well-defined goal and keep their work focused". Using the observation in Pinzger et al. [2008] as a rule would make development groups more similar to WGs than textbook PTs.

(12) *Formal Groups (FG).* The key differentiating attribute for FGs is in their governance practices, which must be declared upon creation of the formal group (i.e., the attribute "Governance" is determinant to identify this type). Literature refers explicitly to formal groups as sets of project teams with a particular mission (or also, groups of people from which project teams are picked). Examples of formal groups in software engineering are software taskforces, such as the IEEE Open-Source Software Task Force[6]. Defining attributes are contained in Table XIV. FGs are exemplified in Hustad [2007] as

"[groups of] teams and/or working groups [. . .]. numerous different definitions of diversity have been put forth; however, they generally distinguish between two main sets of characteristics [for FGs]: (1) diversity of observable or visible detectable attributes such as ethnic background, age, and gender; (2) diversity with respect to nonobservable, less visible or underlying attributes such as knowledge disciplines and business experiences".

[6]http://ewh.ieee.org/cmte/psace/CAMS_taskforce/index.htm.

Table XIV. FGs Defining Attributes

Attribute	***in Formal Groups***
"Organizational Goal"	Formal groups are appointed within an organization and always carry an organizational goal (e.g. corrdination, performance, evaluation, etc.)
"Members Official Status"	The official status of members is formal since the organization appoints each member upon joining the organization or division
"Governance"	**A formal group always establishes diverse governance practices. The most common are Routinization (e.g. fixed hang-outs or meetings, stand-up status reports, etc.); emotional management (e.g. counseling of less performant engineers, etc.); control (e.g. micro-management of development tasks, etc.); Scientific Management (e.g. management of the methods and approaches, development and promotion of best practices, etc.).**
"Governance dependency graph"	 <<enumeration>> MembersMotivation <<enumeration>> Engagement <<enumeration>> KnowledgeRepository <<enumeration>> OrganizationalCulture <<enumeration>> Leadership <<enumeration>> MembersEnrolment <<enumeration>> Barrier <<enumeration>> OrganizationalSponsor <<enumeration>> Size <<enumeration>> CriticalSuccessFactor <<enumeration>> Governance <<enumeration>> Visibility <<enumeration>> SocialCapital <<enumeration>> MembersCohesion M5.67 M3.37 M3.29 M3.37 M5.50 M2.33, M2.12 M5.65 M2.79 M2.51 M2.87 M2.110 M2.49 M2.34 M2.40 M2.10 M2.78 M5.53 M2.11,13,50 M5.54 M2.41 M2.37 M2.19 M2.76 M2.35 M2.43 M2.38

FGs are comprised of people which are explicitly grouped by corporations to act on (or by means of) them (e.g., governing employees or ease their job or practice by grouping them in areas of interest). Each group has a single organizational goal, called mission (governing boards are groups of executives whose mission is to devise and apply governance practices successfully). In comparison to Formal Networks, they seldom rely on networking technologies, on the contrary, they are local in nature. Finally, it is very common for organizations to have these groups and extract project teams out of them.

4.1.4. Teams. Teams are specifically assembled sets of people with a diversified and complementary set of skills. They always pursue an organizational goal with clear-cut procedures and activities. Finally, all project teams exhibit a longevity which is bound to a project or product. Here follow detailed definitions.

(13) *Project Teams (PT).* The key differentiating attribute for PTs is their longevity, tied to a specific project (i.e., the attribute "longevity" is a differentiator to identify PTs). Their defining attributes are in Table XV. In Lindkvist [2005], the author provides a general definition of PTs with the following words:

"[PTs are] temporary organizations or project groups within firms [that] consist of people, most of whom have not met before, who have to engage in swift socialization and carry out a prespecified task within set limits as to time and costs. Moreover, they comprise a mix of individuals with highly specialized competences,

Table XV. PT Defining Attributes

Attribute	***in Project Teams***
“Longevity”	**Longevity of a project team should be limited to the project for which it was selected and assembled. The team has the chance to be reunited again for similar projects, based on performance.**
“Longevity dependency graph”	-
“Knowledge Activities”	The knowledge activities within a project teams are limited to the use in practice of what knowledge is available already or from the context (or by means of an OSS to which the project team is made a member of)
“Proficiency Diversity”	The proficiency diversity when selecting team members should be maintained as complementary. All project members should complete each other in terms of skills.
“Organizational Slack”	The organization or sponsor reserves (or should reserve) no operational slack to the project team. This means that the deadlines assigned to a certain team for a certain workpackage should be strict and ultimate.
“Organizational Goal”	The organizational goal of the OSS emerging in teams is that of delivering the software (piece) it is being used to develop
“Critical Success Factors”	Four critical success factors are stated as being at stage in project teams: creative problem solving by its members; a social network for the team and the team alone; a weak set of ties between the members of the team; a technological gate-keeper which mediates the team’s communication with the rest of the world for technical, operational or organizational issues.
“Size”	The size of a project team should be extremely small (i.e. never more than 5–10 elements) and localized.
“Creation Process”	The creation process of project teams should be formal and operated by the organization or organizational sponsor.
“Members Cohesion”	Milestone-based and social-closeness based cohesion practices should be adopted to properly nurture the teams’ operation.
“Members Previous Experience”	Previous experience of members should be cross-functional.

making it difficult to establish shared understandings or a common knowledge base”.

PTs are made by people with complementary skills who work together to achieve a common purpose for which they are accountable. They are enforced by their organization and follow specific strategies or organizational guidelines (e.g., time-to-market, effectiveness, low cost, etc.). Their final goal is delivery of a product or service which responds to the requirements provided. Compared to the other OSSs, PTs are the most formal type, comparable (in terms of formality) only to the formal groups type (of which it is an instance and from which they are picked). PTs are defined as strict and single-minded aggregates of people, (closely) collaborating on well-defined reification tasks (i.e., tasks which produce a tangible artifact which justifies their effort). Within software engineering, PTs are constantly used as the basic logical unit for software production.

4.2. OSS Types’ Relations

The 13 types are also reported in the UML-style metamodel in Figure 2. This shows all the relations found among them. The note on top of each type carries the number of publications in which it was found.

The two most general types in the metamodel are SNs and PTs. These two types appear at the top of the diagram. Our metamodel shows that INs and FNs are sibling types, deriving from SNs.

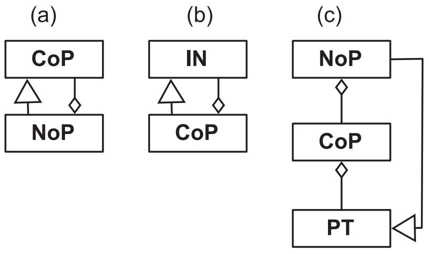


Fig. 3. OSS patterns found.

This indicates that research currently approaches OSSs from the two perspectives of SNs or PTs. The contact point between organizational research and social network analysis (or research in social networks) is in the CoP type, which inherits and aggregates INs. CoPs generalize four types: KCs, NoPs, WGs, and SCs. LCs are at the same abstraction level as INs (probably because learning is "informal" itself), while ICs seem an intermediate type between CoPs and NoPs since they aggregate CoPs and are more specific than NoPs.

PTs are a "plug-in" type to CoPs since they are instances of FGs and aggregate into FGs (e.g., for governance), CoPs (e.g., communities of interest or interest groups), or NoPs (in case of virtualization of teams, as in global software engineering). Finally, PSCs are specific instances of NoPs (e.g., focused on a specific set of problems).

4.3. OSS Patterns and Transitions

The OSS metamodel (see Figure 2) exhibits the recurring patterns shown in Figure 3. According to pattern, (a) NoPs aggregate multiple CoPs, which can be further specialized into NoPs. Similarly, according to pattern (b), CoPs aggregate multiple INs, which can be further specialized into CoPs. Finally, pattern (c) shows that NoPs can be made of CoPs, CoPs themselves can be made of PTs and finally, NoPs can be a specific type of PTs (i.e., virtual teams).

4.4. Additional Generic Attributes

In addition to OSS defining attributes (contained in the definition of specific OSS types), we found 15 attributes which are applicable to all OSSs. The following enumeration distinguishes between attributes internal to OSSs (beginning with "OSS::") and attributes which belong to the environment or context of an OSS (beginning with "Context::").

(1) *OSS::lifecycle*: This consists of the (planned) steps that the OSS is expected to go through during its lifespan. The Software Process' intermediate steps constitute the lifecycle of the software engineering OSS.
(2) *OSS::lifespan*: This is the projected length of time within which the OSS is considered operational. The planned time-to-market of a software being engineered (in addition to its operational expectance-of-life) is the lifespan of a software engineering OSS.
(3) *OSS::Goals*: These are the aims that the OSS is going to work towards. These are either emergent or enforced through an organizational sponsor. The delivery of a software product meeting, all the stakeholders' requirements, and within their specific constraints constitutes the goal of a software engineering OSS.
(4) *OSS::Barriers*: These are impediments, physical or otherwise, which hinder the operations of OSSs. Governance practices can tackle these barriers through

barrier mitigation mechanisms. Time and distances in software engineering are instances of OSS barriers.

(5) *OSS::Governance Practices*: These are activities which create, decide upon, steer, or enforce organizational issues in OSSs. Deciding brand and structure of workbenches for every developer and arranging together available developers into development units are governance practices in software engineering OSSs.

(6) *OSS::Critical Success Factors*: These are factors which are necessary to achieve successfully the goals established for the OSS within the boundaries of its projected lifespan and following the enforced governance practices. 24/7 availability in fault-tolerant systems is an instance of critical success factors in software engineering of critical systems OSSs.

(7) *OSS::Management Practices*: These are practices which are specifically aimed at the management of resources, physical or otherwise, which are part of the OSS (e.g., history, version-tracking, inventory, etc.). Deciding which team should carry out which task is a management decision. Using round-the-clock productivity as a guideline for this decision is a management practice in global software engineering OSSs.

(8) *OSS::Knowledge Repositories*: These are the types of repositories found in literature. Different types may be employed together to achieve a compounding effect, increasing capabilities. A wiki containing documents and codebases for a software engineering efforts is a Knowledge Repository in software engineering OSSs.

(9) *Context::Barrier Mitigation Mechanisms*: These mechanisms are used as part of governance practices by the management of OSSs in order to mitigate barriers to its proper function. Risk engineering practices in software engineering OSSs are barrier mitigation mechanisms.

(10) *Context::Trust*: These are the types of trust discussed in literature, which are specifically relevant to OSSs. Privacy maintenance and measurement between outsourced and outsourcer partners are indicative of the level of trust between partners in software engineering OSSs.

(11) *Context::Openness*: This is the degree to which the context of an OSS and the OSS itself are open to information transfer and interchange. Information interchange between external communities and the software engineering project team in a project is its degree of openness (e.g., fully open in Open-Source communities).

(12) *Context::Changes*: These are events which alter the context of an OSS and to which the OSS reacts, either explicitly or implicitly. Test engineers turn-over is a context change in a software engineering OSS.

(13) *Context::Boundary Crossing Practices*: These are practices that OSSs adopt to exchange information among each other, through boundary objects, for example, their communication or interacting protocols count as boundary crossing practices.

(14) *Context::Boundary Objects*: These are the objects that can be used to actuate knowledge transfer between and across OSSs. Technological gateways such as people which explicitly interchange design information between two development sites are instances of boundary objects in software engineering OSSs.

(15) *Context::Organizational Culture*: These are the practices that an OSS uses in order to pursue internal integration (i.e., governance of members, resources, etc.) as well as external adaptations (actions on the context, knowledge transfer, etc.). Allowing 1 hour sleep after lunch to increase software designers' productivity in the afternoon is part of the organizational culture of Google's (software engineering) OSS.

Given their number, all the possible attribute values we found are available online (for a link, see the Appendix).

In addition, Figure 4 captures attribute dependencies in a UML class diagram. The diagram can be used to choose additional attributes as needed, based on the impact each

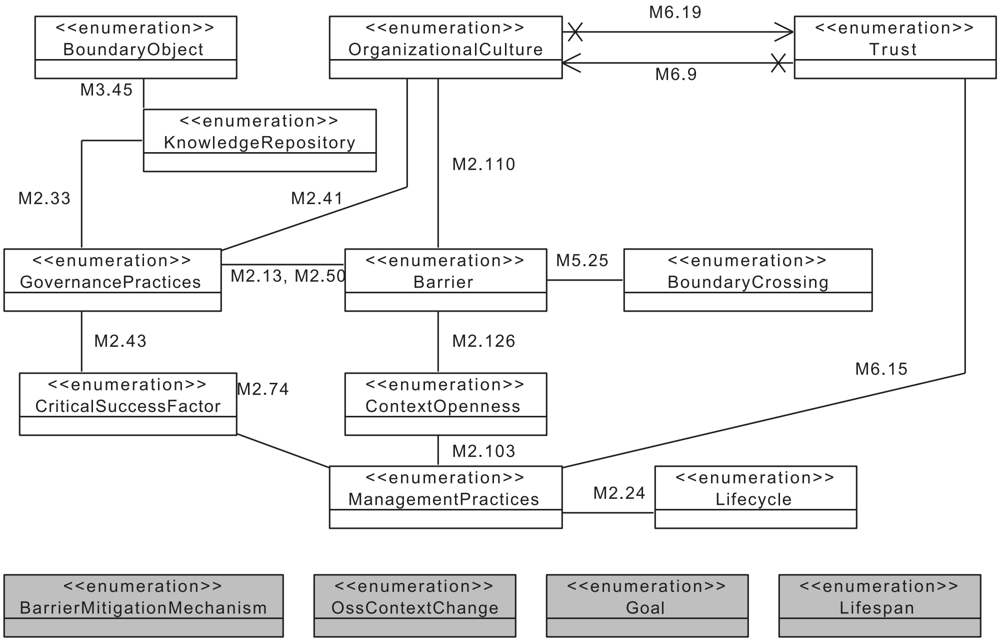

<<enumeration>> BoundaryObject
<<enumeration>> OrganizationalCulture
<<enumeration>> Trust
M6.19
M6.9
M3.45
<<enumeration>> KnowledgeRepository
M2.110
M2.33
M2.41
<<enumeration>> GovernancePractices
M2.13, M2.50
<<enumeration>> Barrier
M5.25
<<enumeration>> BoundaryCrossing
M2.43
M2.126
M6.15
<<enumeration>> CriticalSuccessFactor
M2.74
<<enumeration>> ContextOpenness
M2.103
<<enumeration>> ManagementPractices
M2.24
<<enumeration>> Lifecycle
<<enumeration>> BarrierMitigationMechanism
<<enumeration>> OssContextChange
<<enumeration>> Goal
<<enumeration>> Lifespan

Fig. 4. OSS additional generic attributes' dependencies.

attribute has on the other ones. On the diagram, the labels (following the same scheme defined in Section 4.1) allow traceability on raw data (available online). Specifically, 4 out of 15 attributes are not dependent on any of the 15, and they appear at the bottom of Figure 4 in grey. In all 14 relations are present.

(1) "BoundaryObject" is dependent on "KnowledgeRepository" since the choice of a boundary object influences the presence (or operation of) a knowledge repository. For example, a boundary object can often be used as a knowledge repository (e.g., a forum or wiki, such as in open-source forges).
(2) "KnowledgeRepository" is dependent on "GovernancePractices" since the presence of a repository influences governance, such as by limiting which practices can be applied and how.
(3) "GovernancePractice" is dependent on "CriticalSuccessFactors" of OSSs since governance is the prime activity through which an OSS's success can be established and steered. For example, a corporation can adopt many-eyes phenomena to govern the success of its best-practices development workgroup.
(4) "GovernancePractice" is also dependent on "Barrier" since many governance mechanisms are specifically designed to tackle specific difficulties (e.g., redundancy matrix for employee skills in mission-critical systems).
(5) "GovernancePractice" is also dependent on "OrganizationalCulture" since many governance practices are usually part of the established organizational procedures in a (large) corporation.
(6) "Barrier" is dependent on "OrganizationalCulture" since many barriers are intrinsic of the domain or culture in certain organizations. For example, highly formal communities (e.g., formal methods) tend to exhibit "xenophobic" attitudes towards radically new ideas.
(7) "Barrier" is also dependent on "BoundaryCrossing" and "ContextOpenness". The (im-)possibility of exchanging information with external environments, or in fully closed contexts, can become a barrier. For example, consider security-critical

systems such as Signal-Intelligence processing networks (e.g., the Echelon sensor array): these may be left unaware of critical failures in certain procedures or devices, given their "eyes-only" policies.

(8) "OrganizationalCulture" is dependent on "Trust" and vice versa. For example, the effectiveness of organizational procedures and culture in organizations often depend on the degree of "loyalty" (which is a type of Trust) from its employees. Conversely, the organizational stability of a company depends heavily on how much worthy of trust are its premises and promises.

(9) "Trust" is dependent on "ManagementPractices" since many of these practices are studied specifically to increase and maintain the level of trust in employees (e.g., informal leadership).

(10) "ManagementPractices" are dependent on "Lifecycle" since these practices steer and maintain the lifecycle of an OSS and, conversely, the healthy lifecycle of an OSS depends heavily on the management practices adopted. For example, a business-critical system in need of strict time-to-market constraints needs specific management practices such as extreme coordination, live integration, and testing coordination, etc.

Remarkably, four attributes are not related to any, namely: "Barrier Mitigation Mechanisms", "OSS Context Changes", "Goals", and "Lifespan". Investigating further in the literature, we found they influence (either directly or indirectly) all remaining additional attributes and therefore are not explicitly related to any.

5. USAGE, IMPLICATIONS AND THREATS TO VALIDITY

This section discusses our results, their exploitation, and their implications. Finally, we discuss threats to validity and mitigating mechanisms we adopted.

5.1. Discussion of Results

Several authors suggest that OSSs have three key functions in software engineering: representation of knowledge, management of roles among partners, and communication patterns [Hannoun et al. 2000; Horling and Lesser 2004; Isern et al. 2011; Seidita et al. 2010; Fox 1988]. We used these research segments as parameters to analyze metatypes found. In addition, we discuss the practical exploitation of our results in these research segments. Finally, we discuss the exploitation of our results within the software engineering lifecycle. Table XVI summarizes our observations.

5.1.1. Exploitation in Knowledge Representation. In the domain of knowledge representation and reasoning, the results have great potential for application. First, the 13 OSS types we provide could be used as a map to build or monitor collective-intelligence systems (e.g., ontology- and community-based crowd-sourcing applications [Lin et al. 2009]). Because collective–intelligence is an emergent property of OSSs, practitioners could study their OSS requirements and compare them to our definitions, to determine the best fit OSS type. Moreover, many community-based approaches to ontology engineering and evolution (e.g., Hepp et al. [2006] and Debruyne et al. [2010]) could benefit from the OSS profiles we offer, since these could be used as reference models (e.g., the relations we have found among OSS types could be used as patterns to integrate multiple OSS types). In addition, the OSS definitions we provide could be used as a framework to evaluate the expressivity Tobies [2001] in community-support systems (e.g., social networking technologies designed for specific community types Sintek et al. [2007]).

As a practical example of the previous discussions, consider the Forge Ontology Proposal (FOP)[7]. Forges are collaborative communities of open-source software development. To develop FOP, the community around it could observe real-life forges (e.g.,

[7]https://forge.projet-coclico.org/plugins/mediawiki/wiki/wp2/index.php/Forge_Ontology_Proposal.

Table XVI. Metatypes Comparison according to Key Parameters from Literature

	Knowledge Representation	Roles/Partnership Management	Communication Patterns
Communities	Communities are ideal mechanisms to achieve shared-understanding. This makes them also ideal to study community-based knowledge representations. Works in this direction can already be found, e.g. in [Hepp et al. 2006] and [Debruyne et al. 2010], where authors adopt a community based approach to evolution of ontologies.	Communities are (explicitly or implicitly) dependent on situatedness [Soekijad et al. 2004]. Consequently they have limited effectiveness in global partnerships (e.g. as a result of outsourcing) which force distance into the equation [Herbsleb and Mockus 2003]. This is consistent with results from [Cusick and Prasad 2006].	Communities usually exhibit informal communication, sharing, and focus on members. These characteristics makes them ideal to study communication patterns which are efficient for particular goals. For example, PSCs could be studied to identify communication patterns which increase their effectiveness in solving problems. Moreover, given their focus on members, communities could be observed to design people-centric communication patterns (e.g. for more effective ambient assisted living).
Networks	Networks can be differentiated by level of formality, goals and member-centricity. Studying these parameters and their relations, more accurate and expressive knowledge representations can be devised. For example, semantic tools are now used to support individuals and formal networks alike (e.g. [Oren et al. 2006; Monticolo and Gomes 2011]). Studying these networks, semantic wikis and ontologies could be adapted to the community needs and attributes.	Networks are mechanisms that connect people across many distances (e.g. space, time, culture, biases, etc.). They are ideal mechanisms to support global partnerships, since their attributes can be customized to fit exactly with partnerships' characteristics. For example, a network with high formality could connect a formal alliance between global offshoring partners(e.g. as suggested in [Cusick and Prasad 2006]). Conversely informal networks could connect informal cooperations of professionals, e.g. in global Scrum-of-Scrums scenarios [Karolak 1999].	Our network profiles, are ideal to study communication patterns. For example,aided by social networks analysis (SNA), scholars could identify "success" or "failure" communication patterns by observing networks' attributes and compare them to the expected goals. This could lead to the development of network tuning strategies, to mitigate identified barriers. For example, an integrated Scrum team is indeed a network of practice. Through SNA, scholars could represent the network and verify the presence of attributes critical for NoPs (e.g. communication openness, based on our definition). If these attributes are not evident, there is a problem.
Groups	Groups are tightly knit and interrelated sets of people [Hustad 2007; Cummings 2004]. Hence, their attributes restrict their uses to the study of knowledge representations for specific domain areas. For example, studying the IFIP Workgroup 2.10 on software architecture (e.g shared understanding, cohesion practices, formality, etc.) practitioners could design adaptable knowledge representation mechanisms for software architectures.	Again, groups are commonly associated with tightness of relations. This makes them ideal for governance of collocated people in global partnerships (e.g. groups in a single site, as suggested in [Cusick and Prasad 2006]).	The study of groups, their attributes and relation could be useful to identify successful communication and collaboration patterns. For example, managers could profile their managed groups and use the attributes as meters to assess the effectiveness of governance strategies.

Table XVI. Continued

	Knowledge Representation	Roles/Partnership Management	Communication Patterns
Teams	Teams are instances of groups which are time-bound and focused on a specific objective (e.g. the delivery of a product). We found that these attributes make them unreliable sources of information for knowledge representation since each instance is different and cannot be generalized.	Our literature comes from many fields, including software engineering. In software engineering, project-teams have very-well defined connotation but are configured ad-hoc to match the development problem. A better definition of teams would not be beneficial for roles and partnerships management, since teams still need to be configured ad-hoc, to match the development problem (e.g. to include offshoring).	A better definition of teams' attributes is beneficial to develop communication and collaboration patterns that increase project success rate. For example, studying the attribute values that arise when project teams and software process are combined successfully, can lead to development of successful communication and collaboration patterns.

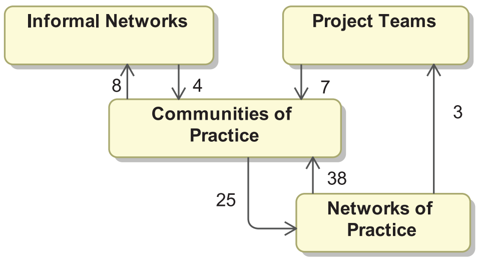


Fig. 5. OSS transitions.

SourceForge) and identify the communities present, using our OSS models as a reference. By putting together the type requirements and attributes of communities found, the FOP initiative could develop a collaborative ontology based on empirical evidence. In addition, FOP authors could validate its applicability in practice by comparing it to practical examples of the communities it was inspired from (e.g., a CoP or NoP in software development).

5.1.2. Exploitation in Partnerships or Communication Management. We analyzed further in the primary studies the patterns introduced in Section 4.3. We found that their usage can be twofold.

The first usage helps companies understanding, triggering, and supporting organizational change (e.g., new partnerships or change in role). We observed that the specialization association between OSS types corresponds to a transition of a certain OSS (e.g., a CoP) into a more specialized OSS (e.g., an NoP). For example, connecting multiple local CoPs on a distributed network corresponds to adding the attribute "geolocalization", and hence specializing CoPs into an NoP. This indicates some kind of transition of an existing OSS into another type, possibly due to evolving organizations, partnerships, business strategies, or customers. We made the same observation for the remaining two patterns: in each pattern the aggregating type transforms into the specializing one.

The three patterns can be combined together, since they involve a common type, CoPs. The diagram in Figure 5 merges the transitions together (the transition label indicates the number of times it was found in literature). We observed a practical example of this transition system in one of our industrial research partners, the Architecture Group at Logica[8], which includes practicing software architects from diverse project teams.

The second usage of the OSS patterns in Figure 3 concerns how, based on empirical evidence, OSS types can be combined together effectively. We found the aggregation patterns in Figure 3 can be both within and across organizations. The former is when specific community types (e.g., CoPs) are combined with other types (e.g., NoPs) to increase communication and organizational efficiency. The latter is when organizations combine (pieces of) communities (e.g., indoor software architecture CoPs) to partner on specific projects. Therefore, a company can identify its OSS type (by analyzing its attributes) and find (through the patterns) other OSSs that can be aggregated effectively.

For example, Philips NL[9] was a pioneer of collaboration with open-source (social)NoPs, but the process was long and costly [Torkkeli et al. 2007]. After the community was up and running, Philips NL commonly used combined expertise from its internal CoPs and its digital NoP to join collaborations on specific projects. In Philips NL's scenario, three communities are involved: (a) the indoor software development community (CoP); (b) the open-source community sponsored by Philips (NoP); and (c) the project teams involved in cross-organizational collaborative projects (PTs). Philips' scenario matches pattern "(c)": using details of key types involved. Philips NL could have planned the "expansion" pattern "(c)", to speed up the process, make it explicit, or support it.

5.1.3. Exploitation in Lifecycle Management. There are many scenarios in which our results can be used during the software lifecycle.

A first possibility is during process planning, where practitioners can compare the development problem with OSS types. For example, one of the goals could be to identify the best-fit OSS.

In addition, developing software practitioners can analyze the development OSS. One of the goals could be to understand and possibly correct the OSS status (attribute values).

The way in which OSS types can be defined suggested to us a sequential process that can be used in both scenarios. In step 1, practitioners identify the type which best fits the development problem. In steps 2 and 3, defining attributes and dependency graphs are used to tailor the type to fit the development problem. In step 4, the types metamodel is used to understand relations for selected types. In step 5, additional generic attributes are used to enrich the OSS to better support its domain (e.g., using governance to overtake certain domain barriers). Finally, in step 6, the transition system is used to monitor the OSS type, predicting and supporting its state changes. For example, in the first scenario, let us suppose that organization X needs to develop a large and concurrent ballistic-control *system* (i.e., a critical system). X requires a formal "MembersOfficialStatus", since personnel background should be certified for security reasons. In step 1, the best-fit OSS type for X's efforts is FN since its "MembersOfficialStatus" is fixed to a formal value. In step 2, the ballistic-control FN is characterized with another attribute, "CreationProcess". Official status of FN members is formally acknowledged and therefore the creation process should respond to precise and formal semantics. Organization X should support these semantics explicitly (e.g., by specific ad hoc ontologies backed by automated security checks on personal and professional background). In step 3, management of ballistic-control FN uses FNs' dependency graph to choose the appropriate "KnowledgeActivity", "DegreeOfFormalization", and "ManagementPractices".

[8]http://www.logica.com.
[9]http://www.philips.nl.

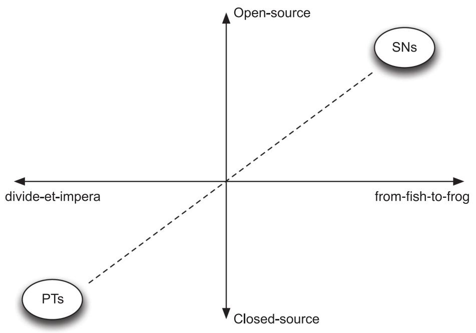


Fig. 6. OSS classification meter.

As a consequence of "MembersOfficialStatus" , some values might not be applicable. For example, the ballistic-control system should be engineered through self-contained and independent teams in the FN. It follows that the "KnowledgeActivity" attribute is set to status-only, hence no cross-team collaboration is possible, and the project must be developed in self-contained work units. In step 4, management of ballistic-control FN uses types relations to exclude informal networks from the development loop. Informal networks would constitute a security risk and are organizationally incompatible with informal networks. In step 5, management of the ballistic-control FN further enrich it for its domain selecting emotional management as an additional governance practice. Step 6 is not applicable since we have not found any patterns linking FNs to other types. Nevertheless, during the software process, the ballistic-control FN could accidentally move to another type in the transition system (e.g., a CoP, due to changing military policies). The OSS transition system becomes useful to understand and support further type shifts.

5.2. Observations

Two key observations were made on our results. The text following the observations explains their implications.

First Observation. Project teams are well-defined OSS types, usually tailored for specific projects, but act according to clear-cut operational dynamics. According to our results, project teams are merely one of 13 OSSs observed in literature. It is remarkable how traditional software engineering has considered project teams almost exclusively. The remaining 12 flavors were left relatively unexplored for the purpose of building software. Additional research should be carried out in the usage of the remaining 12 types for software engineering. Indeed some of them could prove very efficient for specific domains. For example, the dynamics within problem-solving communities and their focus on solving problems for immediate benefits could be efficient to develop safety-critical systems (e.g., health-care software) since these systems should maintain the benefit and well-being of their users. The OSS definitions and the selection mechanisms we have provided could be used as a map to explore this uncharted segment of software engineering.

Second Observation. Project teams and social networks are nongeneralizable supertypes (see Figure 2). These two types sit at the base of organizational practice. Based on how project teams and social networks are defined in literature, we observed that they represent the two extremes of an ideal classification meter for organizational social structures in software engineering. We have produced this ideal meter in the form of a two-axis diagram in Figure 6. The figure shows Project Teams (PTs) at the bottom left, since they typically operate through divide-et-impera approaches, as part of closed-source software development projects. Moreover, the figure shows Social Networks (SNs) at the top right, since they are typically used explicitly in open-source communities to develop software (e.g., projects part of the Linux foundation[10]) also, they are evolutionary in nature, that is, follow a more Darwinian, from-fish-to-frog approach, in which the collective pulls autonomously the single engineering tasks and the fittest contribution "survives".

Two implications follow this observation. First, when approaching software engineering, practitioners immediately associate project teams to the process of developing software. To our knowledge, no structured process nor software engineering approach has been suggested using explicitly social networks as the basic unit of production for software, in spite of their successful use in OSS research and practice. Additional research should be carried out in "social-network-centric software engineering". In this field, domain-specific social networks are used to carry out software engineering, rather than project teams.

Second, our data was insufficient to classify the remaining 11 OSS types through the diagram in Figure 6. Such a classification could uncover the underlying relation between the two dimensions in the diagram, for example, to understand how far these two dimensions can benefit from each other. Also, it would be interesting to investigate (in organizational research) which areas on the diagram have been charted so far and with which benefits, for example, to support their further exploration for software engineering. The diagram in Figure 6 as well as the results presented in Section 4 could be used to bootstrap such research.

5.3. Threats to validity

Based on the taxonomy in Wohlin et al. [2000], there are four potential validity threat areas, namely: external, construct, internal, and conclusion validity.

External Validity concerns the applicability of the results in a more general context. Since our primary studies are obtained from a large extent of disciplines, our results and observations might be only partially applicable to the software engineering discipline. This may threaten external validity. To strengthen external validity, we organized feedback sessions. We analyzed follow-up discussions and used this qualitative data to fine-tune our research methods and applicability of our results. In addition, we prepared a bundle of all the raw data, all models drawn, all tables, and everything that we used to compose this article so as to make it available to all who might want to further their understanding on our data (for links see the Appendix). We hope that this can help in making the results and our observations more explicit and applicable in practice.

Construct Validity and *Internal Validity* concern the generalizability of the constructs under study, as well as the methods used to study and analyze data (e.g., the types of bias involved). To mitigate these threats, we adopted formal grounded-theory methods; these were conceived to avoid bias by construction [van Niekerk and Roode 2009; Corbin and Strauss 1990; Haig 1995]. To ensure internal and construct validity even further, the initial set of codes for grounded theory was developed by an external researcher and checked against another external reviewer who is not among the authors and not belonging to the software engineering field. In addition we applied grounded theory in two rounds: (a) first the primary studies were split across a group of students, to apply grounded theory; (b) in the second round one of the authors reexecuted a blind grounded theory on the full primary studies set. When both rounds were finished, both grounded theories were analyzed evenly to construct a unique theory.

[10]http://www.linuxfoundation.org/.

When disagreement between the two samples was found, a session was organized with students, researchers, and supervisors to examine the samples and check them against literature.

"Conclusion Validity" concerns the degree to which our conclusions are reasonable based on our data. The logical reasoning behind our conclusions is dictated by sound analysis of the data through grounded theory and other analysis methods which try and construct theory from data rather than confirming initial hypotheses, as explained in Haig [1995] and Schreiber and Carley [2004]. Moreover, all the conclusions in this article were drawn by three researchers and double-checked against data, primary papers, or related studies.

6. CONCLUSIONS

We conducted a systematic literature review based on grounded theory, to obtain a definition and comparison of OSSs. With reference to Section 3.1, we set out to answer the following research questions.

(1) What types of OSSs can be distinguished in literature?
(2) What attributes can be identified for each type?

We answered our first research question with 13 OSS types elicited from literature. For each type we provided: (a) a single differentiating attribute; (b) a dependency graph; (c) a set of attributes unique to that type. Also, we compared each type with others. Moreover, we identified 15 additional attributes (and their dependencies) commonly/generally applicable to all types. This last result, in addition to the defining attributes in every type, answer research question 2. Finally, we provided an OSS transition system (between key OSS types) made of patterns through which OSS types can be combined together.

Our evidence and discussions support two key conclusions. On one hand, we observed how project teams are not the only OSS that could be used to develop software. We conclude that additional research should be carried out in exploring other OSSs for the purpose of building software. Our results could be used as a starting point for such research.

On the other hand, in this article we have described how to decide on the OSS best fitting a certain development problem (e.g., critical systems, as discussed in Section 5) rather than selecting project teams by default. Consequently, this article might serve as the first attempt to consider engineering software *with an OSS-aware approach*, that is, selecting and supporting the organizational structure which fits exactly with the software development problem. Indeed, the primordial stages of this discipline can be found in the emergence of unconventional OSSs such as described in Yamauchi et al. [2000], Northrop et al. [2006] or, even more recently, in Kazman and Chen [2010]. In presence of such emergent OSSs, traditional software engineering is no longer applicable [Northrop et al. 2006]. We argue that the new models, presented in reply of this software engineering disability (e.g., the "Metropolis" model in Kazman and Chen [2010]) are hybrids of our 13 OSS flavors (e.g., SNs and NoPs, in the case of "Metropolis"). *Therefore we conclude that this article serves as a first rudimental compass in this unexplored research segment, namely "OSS-aware software engineering".*

APPENDIX

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

The authors would like to thank Mirella Sangiovanni for the care in correcting this document and the anonymous reviewers for their invaluable contributions.

REFERENCES

ALLEN, J., JAMES, A. D., AND GAMLEN, P. 2007. Formal versus informal knowledge networks in r&d: A case study using social network analysis. *Res. Des. Manag. 37*, 3, 179–196.

ANDRIESSEN, J. H. E. 2005. Archetypes of knowledge communities. In *Communities and Technologies*, Springer, 191–213.

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I., AND ZAHARIA, M. 2010. A view of cloud computing. *Comm. ACM 53*, 50–58.

BIRD, C., NAGAPPAN, N., DEVANBU, P., GALL, H., AND MURPHY, B. 2009a. Does distributed development affect software quality? An empirical case study of windows vista. In *Proceedings of the 31st International Conference on Software Engineering (ICSE'09)*. 518–528.

BIRD, C., NAGAPPAN, N., GALL, H., MURPHY, B., AND DEVANBU, P. 2009b. Putting it all together: Using socio-technical networks to predict failures. In *Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE'09)*. 9–119.

BLANKENSHIP, N. AND RUONA, N. 2009. Exploring knowledge sharing in social structures: Potential contributions to an overall knowledge management strategy. *Adv. Devel. Human Res. 11*, 3, 290.

BOGENRIEDER, I. AND NOOTEBOOM, B. 2004. Learning groups: What types are there? A theoretical analysis and an empirical study in a consultancy firm. *Organization Stud. 25*, 2, 287–313.

CATALDO, M., HERBSLEB, J. D., AND CARLEY, K. M. 2008. Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity. In *Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'08)*. ACM Press, New York, 2–11.

CATALDO, M., MOCKUS, A., ROBERTS, J. A., AND HERBSLEB, J. D. 2009. Software dependencies, work dependencies, and their impact on failures. *IEEE Trans. Softw. Engin. 35*, 6, 864–878.

CATALDO, M. AND NAMBIAR, S. 2009a. On the relationship between process maturity and geographic distribution: An empirical analysis of their impact on software quality. In *Proceedings of the 7th Joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'09)*. ACM Press, New York, 101–110.

CATALDO, M. AND NAMBIAR, S. 2009b. Quality in global software development projects: A closer look at the role of distribution. In *Proceedings of the 4th IEEE International Conference on Global Software Engineering (ICGSE'09)*. 163–172.

CATALDO, M. AND NAMBIAR, S. 2012. The impact of geographic distribution and the nature of technical coupling on the quality of global software development projects. *J. Softw. Evolut. Process. 24*, 2, 153–168.

CHARD, K., CATON, S., RANA, O., AND BUBENDORFER, K. 2010. Social cloud: Cloud computing in social networks. In *Proceedings of the 3rd International Conference on Cloud Computing*.

CORBIN, J. AND STRAUSS, A. 1990. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociol. 13*, 1, 3–21.

CUMMINGS, J. N. 2004. Work groups, structural diversity, and knowledge sharing in a global organization. *Manag. Sci. 50*, 3, 352–.

CUSICK, J. J. AND PRASAD, A. 2006. A practical management and engineering approach to offshore collaboration. *IEEE Softw. 23*, 5, 20–29.

DATTA, S., SINDHGATTA, R., AND SENGUPTA, B. 2011. Evolution of developer collaboration on the jazz platform: A study of a large scale agile project. In *Proceedings of the 4th India Software Engineering Conference (ISEC'11)*. ACM Press, New York, 21–30.

DEBRUYNE, C., REUL, Q., AND MEERSMAN, R. 2010. Gospl: Grounding ontologies with social processes and natural language. In *Proceedings of the 7th International Conference on Information Technology: New Generations (ITNG'10)*. S. Latifi, Ed., 1255–1256.

DICKINSON, A. M. 2002. Knowledge sharing in cyberspace: Virtual knowledge communities. In *Proceedings of the 4th International Conference on Practical Aspects of Knowledge Management (PAKM'02)*. 457–471.

FERSHTMAN, C. AND GANDAL, N. 2008. Microstructure of collaboration: The 'social network' of open source software. CEPR Discussion Papers 6789, C.E.P.R. Discussion Papers.

FOX, M. S. 1988. An organizational view of distributed systems. In *Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, 140–150.

HAIG, B. 1995. Grounded theory as scientific method. Philosophy of education (on-line). http://jan.ucc.nau.edu/~pms/cj355/readings/Haig%20Grounded%20Theory%20as%20Scientific%20Method.pdf.

HANNOUN, M., BOISSIER, O., SICHMAN, J. S., AND SAYETTAT, C. 2000. Moise: An organizational model for multi-agent systems. In *Proceedings of the 7th International Ibero-American Conference and the 15th Brazilian Symposium on Advances in Artificial Intelligence (IBERAMIA-SBIA'00)*. M. C. Monard and J. S. Sichman, Eds., Lecture Notes in Computer Science, vol. 1952, Springer, 156–165.

HANSEN, M. 1999. The search-transfer problem: The role of weak ties in sharing knowledge across organization subunits. *Admin. Sci. Quart. 44*, 1, 82–111.

HATALA, J.-P. AND LUTTA, J. G. 2009. Managing information sharing within an organizational setting: A social network perspective. *Perform. Improv. Quart. 21*, 4, 5–33.

HEPP, M., BACHLECHNER, D., AND SIORPAES, K. 2006. Ontowiki: Community-driven ontology engineering and ontology usage based on wikis. In *Proceedings of the International Symposium on Wikis (WikiSym'06)*. ACM Press, New York, 143–144.

HERBSLEB, J. AND MOCKUS, A. 2003. An empirical study of speed and communication in globally distributed software development. *IEEE Trans. Softw. Engin. 29*, 6, 481–94.

HORLING, B. AND LESSER, V. 2004. A survey of multi-agent organizational paradigms. *Knowl. Engin. Rev. 19*, 4, 281–316.

HUSTAD, E. 2007. Managing structural diversity: The case of boundary spanning networks. *Electron. J. Knowl. Manag. 5*, 4, 399–409.

HUSTAD, E. 2010. Exploring knowledge work practices and evolution in distributed networks of practice. *Electron. J. Knowl. Manag. 8*, 1, 69–78.

ISERN, D., SNCHEZ, D., AND MORENO, A. 2011. Organizational structures supported by agent-oriented methodologies. *J. Syst. Softw. 84*, 2, 169–184.

JETTER, M., SATZGER, G., AND NEUS, A. 2009. Technological innovation and its impact on business model, organization and corporate culture - Ibm's transformation into a globally integrated, service-oriented enterprise. *Bus. Inf. Syst. Engin. 1*, 1, 37–45.

JOHNSEN, E. C. 1985. Network macrostructure models for the davis-leinhardt set of empirical sociomatrices. *Social Netw. 7*, 3, 203–224.

KAROLAK, D. W. 1999. *Global Software Development: Managing Virtual Teams and Environments*, 1st ed. IEEE Computer Society Press, Los Alamitos, CA.

KAZMAN, R. AND CHEN, H.-M. 2010. The metropolis model and its implications for the engineering of software ecosystems. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research(FoSER'10),* G.-C. Roman and K. J. Sullivan, Eds., ACM Press, New York, 187–190.

KITCHENHAM, B., PEARLBRERETON, O., BUDGEN, D., TURNER, M., BAILEY, J., AND LINKMAN, S. 2008. Systematic literature reviews in software engineering a systematic literature review. *Inf. Softw. Technol. 51*, 1, 7–15.

KLEIN, J. H., CONNELL, N. A. D., AND MEYER, E. 2005. Knowledge characteristics of communities of practice. *Knowl. Manag. Res. Pract. 3*, 2, 106–114.

KSHETRI, N. 2010. Cloud computing in developing economies. *IEEE Comput. 43*, 10, 47–55.

KWAN, I., SCHROTER, A., AND DAMIAN, D. 2011. Does socio-technical congruence have an effect on software build success? A study of coordination in a software project. *IEEE Trans. Softw. Engin. 37*, 3, 307–324.

LANGHORNE, R. 2001. *The Coming of Globalization: Its Evolution and Contemporary Consequences*. Palgrave, London and New York.

LEE, S. H. AND WILLIAMS, C. 2007. Dispersed entrepreneurship within multinational corporations: A community perspective. *J. World Bus. 42*, 4, 505–519.

LIBRARYA, U. 2010. Bibliometrics - An introduction. Ph.D. thesis, HinweisŁ.

LIN, H., DAVIS, J., AND ZHOU, Y. 2009. Integration of computational and crowd-sourcing methods for ontology extraction. In *Proceedings of the 5th International Conference on Semantics, Knowledge and Grid*. 306–309.

LINDKVIST, L. 2005. Knowledge communities and knowledge collectivities: A typology of knowledge work in groups. *J. Manag. Stud. 42*, 6, 1189–1210.

LYYTINEN, K., MATHIASSEN, L., AND ROPPONEN, J. 1998. Attention shaping and software risk - A categorical analysis of four classical risk management approaches. *Inf. Syst. Res. 9*, 3, 233–255.

MARTINELLI, A. 2007. Evolution from world system to world society? *World Futures 63*, 5–6, 425–442.

MENEELY, A. AND WILLIAMS, L. A. 2009. Secure open source collaboration: An empirical study of linus' law. In *Proceedings of the ACM Conference on Computer and Communications Security*. E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds., ACM Press, New York, 453–462.

MENTZAS, G., APOSTOLOU, D., KAFENTZIS, K., AND GEORGOLIOS, P. 2006. Inter-organizational networks for knowledge sharing and trading. *Inf. Technol. Manag. 7*, 4, 259–276.

MONTICOLO, D. AND GOMES, S. 2011. Wikidesign: A semantic wiki to evaluate collaborative knowledge. *Int. J. eCollaboration 7*, 3, 31–42.

NAGAPPAN, N., MURPHY, B., AND BASILI, V. 2008. The influence of organizational structure on software quality: An empirical case study. In *Proceedings of the International Conference on Software Engineering.* 521–530.

NORTHROP, L., FEILER, P., GABRIEL, R. P., GOODENOUGH, J., LINGER, R., LONGSTAFF, T., KAZMAN, R., KLEIN, M., SCHMIDT, D., SULLIVAN, K., AND WALLNAU, K. 2006. Ultra-large-scale systems – The software challenge of the future. Tech. rep., Software Engineering Institute, Carnegie Mellon. June. http://www.sei.cmu.edu/library/assets/ULS_Book20062.pdf.

ONIONS, P. E. W. 2006. Grounded theory applications in reviewing knowledge management literature. In *Proceedings of the Leeds Metropolitan University Innovation North Research Conference 1962*. 1–20.

OREN, E., VLKEL, M., BRESLIN, J., AND DECKER, S. 2006. Semantic wikis for personal knowledge management. In *Database and Expert Systems Applications*, S. Bressan, J. Kng, and R. Wagner, Eds., Lecture Notes in Computer Science, vol. 4080, Springer, 509–518.

PARREIRAS, F. S., DE OLIVEIRA, E., SILVA, A. B., BASTOS, J. S. Y., AND BRANDAO, W. C. 2004. Information and cooperation in the free/open source software development communities: An overview of the brazilian scenario. In *Proceedings of the 5th National Meeting on Education and Research on Information Science (V CINFORM'04).*

PINZGER, M., NAGAPPAN, N., AND MURPHY, B. 2008. Can developer-module networks predict failures? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT'08/FSE).* ACM Press, New York, 2–12.

RICHARDSON, I., CASEY, V., BURTON, J., AND MCCAFFERY, F. 2010. Global software engineering: A software process approach. In *Collaborative Software Engineering*, I. Mistrik, J. Grundy, A. van der Hoek, and J. Whitehead, Eds., Springer, 35–56.

RILEY, C. AND DOLLING, P. 2011. *NASA Apollo 11 Manual: An Insight into the Hardware from the First Manned Mission to Land on the Moon*, 1st ed. Haynes Online. http://www.haynes.com/products/productID/563.

ROSSO, C. D. 2009. Comprehend and analyze knowledge networks to improve software evolution. *J. Softw. Maint. Evolut. Res. Pract. 21*, 3, 189–215.

RUIKAR, N., KOSKELA, N., AND SEXTON, N. 2009. Communities of practice in construction case study organisations: Questions and insights. *Constr. Innov. Inf. Process Manag. 9*, 4, 434–448.

RUUSKA, I. AND VARTIAINEN, M. 2003. Communities and other social structures for knowledge sharing: A case study in an internet consultancy company. In *Communities and Technologies*. Kluwer, 163–183.

SCHREIBER, C. AND CARLEY, K. M. 2004. Going beyond the data: Empirical validation leading to grounded theory. *Comput. Math. Organization Theory 10*, 2, 155–164.

SEIDITA, V., COSSENTINO, M., HILAIRE, V., GAUD, N., GALLAND, S., KOUKAM, A., AND GAGLIO, S. 2010. The metamodel: A starting point for design processes construction. *Int. J. Softw. Engin. Knowl. Engin. 20*, 4, 575–608.

SINTEK, M., VAN ELST, L., SCERRI, S., AND HANDSCHUH, S. 2007. Distributed knowledge representation on the social semantic desktop: Named graphs, views and roles in nrl. In *Proceedings of the 4th European Semantic Web Conference*. 594–608.

SOEKIJAD, M., INTVELD, M. A. A. H., AND ENSERINK, B. 2004. Learning and knowledge processes in inter-organizational communities of practice. *Knowl. Process Manag. 11*, 1, 3–12.

TAMBURRI, D. A., DI NITTO, E., LAGO, P., AND VAN VLIET, H. On the nature of the gse organizational social structure: An empirical study. In *Proceedings of the 7th IEEE International Conference on Global Software Engineering.*

TOBIES, S. 2001. Complexity results and practical algorithms for logics in knowledge representation. CoRR cs.LO/0106031. http://cds.cern.ch/record/504396?ln=en.

TORKKELI, M., VISKARI, S., AND SALMI, P. 2007. Implementing open innovation in large corporations. http://www.bibsonomy.org/bibtex/244f0ce478ae3df0644368637ba17ebea/luise_k.

TOSUN, A., TURHAN, B., AND BENER, A. 2009. Validation of network measures as indicators of defective modules in software systems. In *Proceedings of the 5th International Conference on Predictor Models in Software Engineering (PROMISE'09).* ACM Press, New York, 5:1–5:9.

TURHAN, B., MENZIES, T., BENER, A. B., AND DI STEFANO, J. 2009. On the relative value of cross-company and within-company data for defect prediction. *Empirical Softw. Engin. 14*, 5, 540–578.

UZZI, B. 1997. Social structure and competition in interfirm networks: The paradox of embeddedness. *Admin. Sci. Quart. 42*, 1, 35–67.

VAN NIEKERK, J. C. AND ROODE, J. D. 2009. Glaserian and straussian grounded theory: Similar or completely different? In *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT'09),* B. Dwolatzky, J. Cohen, and S. Hazelhurst, Eds., ACM Press, New York, 96–103.

VITHESSONTHI, N. 2010. Knowledge sharing, social networks and organizational transformation. *Bus. Rev. Cambridge 15*, 2, 99–109.

WENGER, E., MCDERMOTT, R. A., AND SNYDER, W. 2002. *Cultivating Communities of Practice: A Guide to Managing Knowledge*. Harvard Business School Publishing. http://hbswk.hbs.edu/archive/2855.html.

WENGER, E. C. AND SNYDER, W. M. 2000. Communities of practice: The organizational frontier. *Harvard Bus. Rev. 78*, 1, 139.

WITTEN, B., LANDWEHR, C., AND CALOYANNIDES, M. 2001. Does open source improve system security? *IEEE Softw. 18*, 5, 57–61.

WOHLIN, C., RUNESON, P., HOST, M., OHLSSON, M. C., REGNELL, B., AND WESSLE, N, A. 2000. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers.

YAMAUCHI, Y., YOKOZAWA, M., SHINOHARA, T., AND ISHIDA, T. 2000. Collaboration with lean media: How open-source software succeeds. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'00).* ACM Press, New York, 329–338.

Received March 2012; revised October 2012; accepted October 2012