

**Oskar Jarczyk**

*Department of Social Informatics*

*Polish-Japanese Institute of Information Technology, 2013*



**Agent-organized evolution of virtual teams in multi-dimensional social networks**

**Abstract**

Multi – agent systems consists of a complex networks of agents located in a discrete cyber space. We try to use multi-agent systems to simulate the process of emerging virtual teams of open source developers. Such process of *emergent* exists in real life, in web portal like SourceFourge or GitHub, were developers join teams according to social and personal motivations. They interact based on environmental knowledge and communication constraints. Success of an open source developers teams depends on many characteristics described in various literature. This research will focus on the distribution of characteristics of emerged teams and the time periods of team evolution. Teams are virtual because they rather exist in mentioned internet portals (“cyber space”) and their members are registered users of those portals. We want to find effective network structures and propose several strategies of team formation.

**Categories and Subject Descriptors**

I.2.11 [Distributed Artificial Intelligence]: Multi-agent Systems;

**General terms**

Simulations

**Keywords**

*Multi–agent simulations, multi-dimensional social networks, OSS (open source software),*

*team emergence, dynamic team formation, OSSD (open source software development)*

1.1

**Introduction**

1.2

Including both virtual and classically defined teams, organizations recognized as being successful usually consists of teams which are highly dependent upon a structure that fosters productive and efficient behaviour at both the individual and the organizational levels. Thus in many applications of multi-agent systems, groups of agents must coordinate effectively in order to solve problems, allocate tasks across a distributed organization, collectively distribute knowledge and information, and achieve collective and personal goals.

1.3

The organizational structure of a multi-agent system dictates the interactions among the agents, and can play a significant role in the overall performance of the agent network (Gaston, 2005). An OSSD cooperation model in which programmers develop free and open software is a promising way of creating high quality computer software (Raymond, 2001). In this simulation of OSSD society we want to allow individual agents to organize their local network connectivity and adaptation mechanisms. Hahn *et.al* (2005) proved that prior collaborative ties have a profound effect on developers’ project joining decisions. The results of their research suggest that it is not only the perceived expected benefits of joining the project that is salient. In addition to such perceived benefits, developers seem to be concerned about the process of realizing those potential benefits. For instance, most of the motivations, as prescribed by the prior literature, seem to focus on extrinsic and intrinsic benefits. Extrinsic benefits are benefits derived from the outcome (*e.g.*, development of software specific to one’s needs, increase in reputation after participating in a successful software project, learning effects *etc.*); whereas intrinsic benefits relate to those that are attainable by virtue of participation in the project itself (e.g., enjoyment, affiliation, community identification etc.).

1.4

Several open source projects have achieved extraordinary success and are among most often choose and used software globally. However, many open source projects have reported few or no development activities at all. Many flounder at the beginning, while others survive, but with little momentum behind them [13]. Contributing to open source software development requires particular skills in the field of programming and other computing skills. OSS projects are naming the technical skills required to participate with help of collaboration portal, and developers tend to self-asses the skills possessed by them.

1.5

The role of prior collaborative ties seems to be related to reducing uncertainties in the process of project participation. In fact, software development is not only a production process but also a social process that heavily involves inter-personal communication and coordination (Curtis *et al.* 1988, Robey and Newman 1996, Sawyer *et al.* 1997, Sawyer and Guinan 1998). Moreover, in the OSSD context, the difficulty inherent in the social process becomes even greater than in traditional software projects since members of an OSSD project are typically from geographically dispersed locations, have diverse cultural backgrounds, and have limited (if not any) face-to-face interactions.

1.6

**Related work**

1.7

Gaston *et al.* [1] studied the team formation problem in a setting where networked agents form teams in a decentralized manner. Agents are able to locally rewire their social network. Anagnostopoulos *et.al.* (2012) considered a setting in which people possess different skills and compatibility among potential team members is modelled by a social network. Authors focused on proposing first online algorithms that assemble teams to deal with tasks. *Balanced Social Task Assignment* problem was introduced with online competitive algorithms. Scholars analysed previous work connected with team formation, i.e.: scheduling with load balancing (Graham *et.al*. 1972) (scheduling of jobs on a set of machines with the goal of minimizing the maximum load on a machine), matching people to tasks (a matching problem for which several systems have recently been proposed, for instance easychair.org, linklings.com and softconf.com).[3] The problem of providing efficient solutions in the bidding model has been addressed by Mehlhorn (2009). Team formation with coordination costs: Lappas et al. (2009) introduce the problem of team formation in social networks. The objective is to minimize the coordination cost, for example, in terms of diameter or weight of the minimum spanning tree for the team. This problem has been extended to cases in which potentially more than one member possessing each skill is required, and where density-based measures are used as objectives [8,9].

1.8

Scholars Jin, Girvan, Newman in their paper “The structure of growing social networks” (2001) proposed two models of growth of social networks which have properties of fixed number of vertices, limited degree, clustering and decay of friendship. Simulation of those models shown emerging societies and existing clustering. Researchers made a conclusion that complex and reasonable patterns of social networks, and evolution of them, can emerge from simple rules, furthermore, general form of those patterns is not influenced by micro details of the rules.

1.9

**Problem definition**

2.1

During research we will try to simulate an evolution (emergence) of virtual teams in social networks. Portals for OSS group work (mentioned SourceForge and GitHub, but also BitBucket and similar) are also multi-dimensional social networks because of their features which allow for interaction occurring between users, i.e. friends list and discussion boards. They are also social constructs which can be described as COINs (Collaborative Innovation Networks). Users (persons) are mostly programmers, developers, designers and experts in the field of software and computer technology. A user and team have dynamic characteristics, which are changing over time. A user is moving forward his collaboration path while he gains experience and knowledge, and a team evolves within its characteristics, structure, needs for new and different users. Persons will get in touch with other people in the network and will create a bond of acquaintance, which can be lost and regained.

2.2

There are several problems which we consider to evaluate:

2.3

* Distribution of characteristics of created teams: sizes and comparing required competencies with competencies of a team

It is deemed it's expected to draw a hierarchical tree or dendrogram showing emerged team structures. Moreover, we plan to introduce clustering coefficient as function of time for different parameters. Because of a dynamic aspect of team formation, virtual team emerging is affected by streams of events. Characteristics will be compared to choose best strategy for optimal team emergence.

Evolution of the network under different strategies can be visualized with spatial layouts of the agents (assuming that we used a continuous space in simulation) and their positions in clusters under different time-frames preferentially mile-stones.

2.4

* Time of the process: of team emergence from the starting point to the state of being a complete team by the definition of required competences

The additional purpose of this dissertation is to find recommendations for improving team emergence process. Evaluating and supporting the team formation can lead to better team performance, software quality of OSS created by those virtual teams, and finally learning process of team members. Paradoxically, fast team emergency doesn't mean that will be high quality and competencies matrix will be most optimal for current project evolution stage.

2.5

* Learning rate: How quickly does a chosen strategy increases performance

Performance can be changed over time as a result of local network adaptation. Number of adaptation changes over time proportionally to changes in structure (Gaston, figure no 3).

2.6

* Stability: Does the strategy lead to a stable network topology

It can be disputed if stability is beneficial for organizational performance (Gaston, p. 1 pnt. 2). A power law distribution is observed in structure changes of open source software teams evolution (Wu, 2006).

2.7

* Global structure: What are the properties of structure of created teams

There is interest in analysing centrality/decentrality of teams. While there is no leadership included in our model, still it’s possible to analyse web of acquaintance inside teams.

2.8

**Problem of completeness**

A complete virtual team have a state of equilibrium which allows for the most efficient work, moreover, further changes in the team structure won't be as dynamic and intensive as they were in previous stages of emergence of this particular team. Capability maturity model created by Software Engineering Institute (SEI) defines 5 stages of *software maturity* (which starts at stage 3). Zhang (2007) analysed *Open Source Software* maturity model based on linear regression and Bayesian analysis. It is believed that team during state of completeness will produce software with optimal quality. Team size will be fluctuating sinusoidal with low growing tendency. Work on issues will focus much less on bug-fixing, reporting bug fixing will happen more rarely.

2.9

**Competencies**

Open source software developers have *competencies* valuable and searched by team and team members. Power with which a person is attracted to a team is a *social vector* calculation. Social vectors are a systematic and mathematical way to define and quantify social relationships between any entities. When attributes of vectors have different importance, a *weighted social vector* can be used.

|  |  |  |  |
| --- | --- | --- | --- |
| **competency** | ***k1*** | ***k{1+i..j-1}*** | ***kj*** |
| **man-hour** | *x1* | *x2* | *x3* |

Table 1: Sample model of competency matrix, stricte competencies required by a team

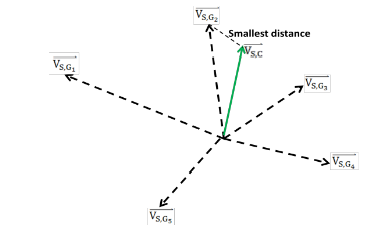


Figure 1: Calculation of best-fit for competency fulfillment

2.10

**Competencies - a real data from GitHub**

Simulation will be using a set of competencies which exists in GitHub portal as labels for repositories. They were extracted with a Google Big Query tool from a “githubarchive.timeline” public dataset and results saved to a CSV comma separated file. Extraction scripts are stated in Illustration no 2,3. Sample of *size* 6000 entities was gathered.

The distribution and percentage existence of competencies in real data projects will allow us to implement probability mechanism of predicting required competencies.

2.11

**Capability Maturity Model**

Capability Maturity Model for Software is a model created by Software Engineering Institute (SEI) and is made for evaluating the process of software production. CMM assess practices during production. Model have 5 levels of degree - starting from chaotic (nothing is supported and/or controlled), ending on strict, disciplined process covering all needed current aspects.

Levels of CMM:

Initial - software created chaotically, without any formal procedures, or with some scrambled practice - not defining process.

Repeatable – basic technics of project tracking are used – watching cost, schedule and functionality. There are technics used which allow to re-implement solutions from previous successful projects.

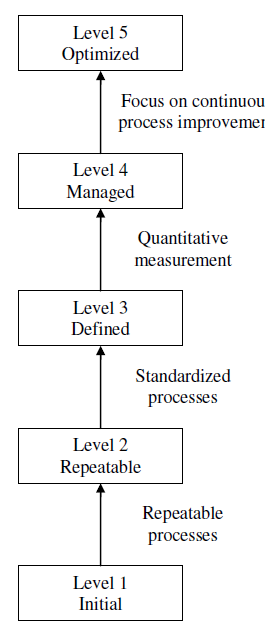
Defined – production process is defined and described, all activities are documented in procedures and functions.

Managed – in projects thoroughly metrics are used to describe the process and software quality.

Optimizing – practice of iterating successful process’ are used through monitoring process, implementing improvements and project management.

CMM also includes key practice areas, goals, common features, and key practices (Zhang, 2007). Moreover, there are maturity models created especially for Open Source Software, which are Open Source Software Maturity Models, Navica/Golden Open Source Maturity Model (OSMM), CapGemini Open Source Maturity Model, finally the Business Readiness Model.

Zhang in his dissertation used linear regression model and Bayesian analysis to evaluate open source software maturity.



Rysunek 1: CMM model and it's stages (levels)

3.1

**Agent-organized network (AON)**

AON exists with explicit set of interactions and interdependencies among the agents in the system, and multiple parties are responsible for the development of the agents (Gaston, 2005)

3.2

There are several considerations for AON model:

3.3

* Perception of team performance

How agent assesses the collective performance of a virtual team? A single agent can check i.e. a team size, list of tasks, required competencies, and age of a team to evaluate overall team state. Agent can assess a team by comparing it to another similar team with structural and performance indexes. It can influence him to move to another team. But in general perspective, a perception of a team performance made by collective audience will be created by checking software quality, current task list and trying to evaluate the stage in CMM where the software can be categorized.

3.4

* Perception of team completeness

Question exists regarding possible ways of checking when a team is in a complete state. We hypothize that when teams reach state of being complete, the rewiring process will be more rare and more predictable than in the previous stages of simulation. A software created by complete team will be mature and almost free from defects, and can be categorize somewhere on stage 4-5 in CMM model. A matrix of required competencies will be covered by overall matrix of skills of team members. Software created by complete team is mature and recognized publicly by high number of downloads in Internet.

3.5

* Adaptation triggers

Agent can adapt himself to better match current team(s), or he can rewire his network and move to other team. During stay in a team an agent modifies his skills matrix and re-assess the teams he is member of. An adaptation trigger in the beginnings of a team life is an important determinant for future structure of a team created from a scratch. For example, if two agents decide to adopt their connectivity at the same time, they could both end in worse position than before. Triggered adaptation must be held in environment of publicly available information (from the perspective of a team). Generally speaking, an agent can decide to adapt basing on both: performance estimates (vide 4.4) and structural requirements (i.e. preferential attachment).

3.6

* Rewiring

The process of rewiring in simulation occurs when an agent decided to remove a connection. Situation happens during decision of leaving a team or unfriending other agent. During rewiring agent can decide to join new team or add other agent to friend list. There many possibilities for specifying rewiring, also referral system can be on the definition.



Rysunek 2: Team state triangle

3.7

We consider people as a set of OSS contributors *P = {pj ; j = 1, 2, . . . , n}*. Each person has a subset of competencies (skills) in her profile, so she is also represented by a point in the competency space: pj 2 S. We use pji = 1 to denote that the jth person has the ith competency, while pji = 0 otherwise. Thus, we have pj = (pj1, pj2, . . . , pjm).

In teams, each task needs to be assigned to team members, optimally to experts which possess required skills for enabling completion of this task. We let *Qj # P* denote the team assigned to the jth task. We use *qj i = 1* to denote that the ith skill is covered by the jth team, so we have qj = *(qj 1, qj 2, . . . , qj m)*. For each team Qj we compute its team profile qj 2 S in the additive skill model [2] that defines the expertise of the team as the (binary) sum of the competencies of each individual:



3.8

Agent is a computer programmer who searches for a team which he can join. In Gaston *et.al* model, an agent could be in one of the 3 states: UNCOMMITED, COMMITTED, or ACTIVE. With every tick of time, an agent can decide to:

- join a team

- leave a team

- work on a task

- improve skills, experience, modify acquaintance list

- stay idle

3.9

Agents have characteristics:

- time of joining the society;

- vector of interests, experience

3.10

Agents are part of multi-dimension social network. Agents can be classified with labels characterizing their type of commitment to team work, i.e.: student, expert, leader, follower, and innovator. Those labels are part of psychological theories and management theories, which describe team work problems. Agents are getting into acquaintance with other agents, therefore making for possibility of inviting other agents to their teams.

4.1

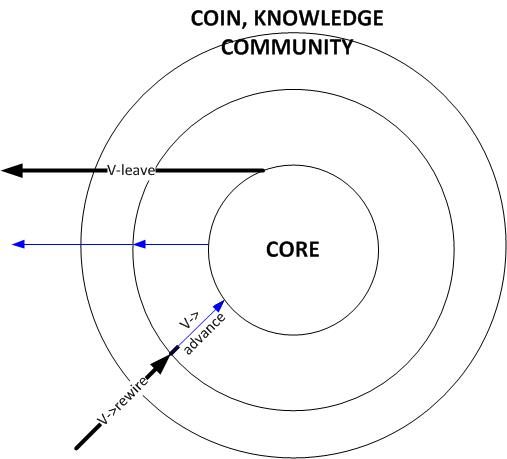
**Team formation strategies**

In social and collaboration networks virtual teams are emerged, characterized by common properties, including their name, set of vectors of demanded competence, their members and the size of a team. Multiple teams are emerging and they are attracting multiple agents to join them. Agents join teams basing on below general classification of heuristics:

1. preferential attachment
   1. continual addition
   2. structure-based adaptation strategy
2. performance based and task assignment strategy
3. homophily/heterophily
4. by invitation

4.2

Agents join the teams for particular period of *x* time, and during this time, they maybe be busy with some activity, or stay idle. A variety of factors may affect the choice of project team to join. Both critical mass theory and expectancy value theories suggest that developers would be influenced by project size since size is a highly visible indicator of the probability of successful outcome and value of the group (Karau and Williams 2000, Markus 1987). However these theories do not explain the decision of developers who choose to join newly formed projects.



Rysunek 3: Rewiring agent in a team on different levels of engagement

4.3

**Continual addition**

Continual addition of vertices and edges to the network as time passes (Jin, 2001). This strategy is the simplest one of mentioned and kind of bare-bones in the model. Continual addition model omits sophisticated social reasoning and works by continual network growth prevailing over rewiring.

[propose here model description]

4.4

**Structure-based adaptation strategy**

In this strategy, the agents adapt their network connectivity based on the notion of *preferential attachment*.

|  |  |
| --- | --- |
| Team size | In standard-preferential attachment, software developers prefer bigger repositories. |
| Project description | Evaluate if project description is similar to description of other teams agent joined |
| Required skills | Try to evaluate if required skills match agent skills |
| Required competencies | Try to evaluate if required competencies match agent competencies |
| Project topic | Evaluate if project topic is similar to topics of other teams agent joined |

Table 2: Preferential attachment matrix

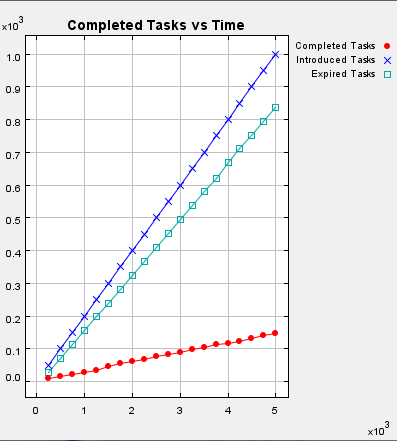
[write here more about centrality/decentrality, comparision, and social behaviour applied]

4.5

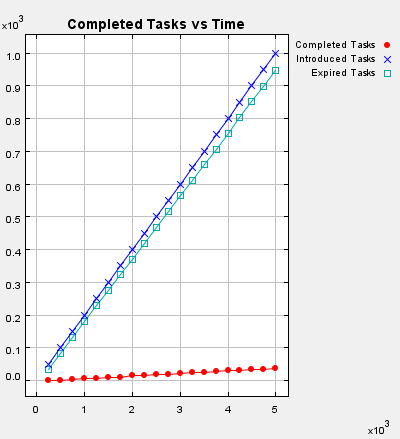
**Performance-based adaptation strategy**

AON strategy based on performance and referrals was analysed and proposed by Gaston *et.al.* During every time tick, agent decides if to rewire when 1) at least one of agents neighbours have valid performance measure 2) when it's performing below the average of neighbour's performance measure. Gaston proposes that the agent can request a referral from its neighbour with the highest performance estimation and similarly, it can refer its neighbour with a high performance estimate.

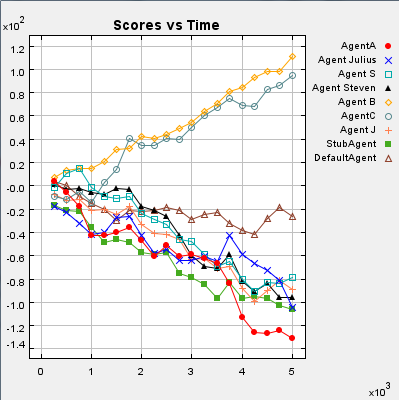
**Hypothesis 1**: Increase in number of skills will suppress team emergence



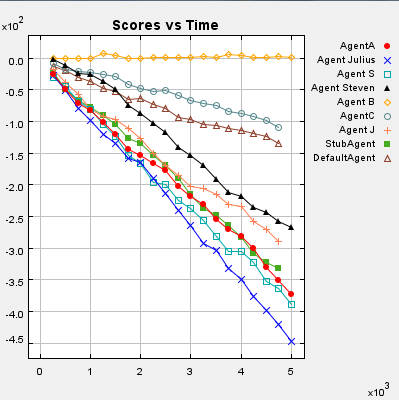
Rysunek 4: Simulation where skills\_no = 12



Rysunek 5: Simulation where skills\_no = 14



Rysunek 6: Agents scores where skills\_no = 12



Rysunek 7: Agents scores where skills\_no = 14

Simulation proves that the more sophisticated skills set and size of issues matrix, the more slow team emergent will be in performance – based strategy.

4.6

**Homophily *versus* heterophily strategy**

Homophily is a word originated from Latin language and means generally a “*love to the same*”. It's a term used widely by biologist and natural sciences, but it was adopted in the field of computer science. (Murthy, 2012) In “Twitter: Social communication in the Twitter age” introduced phenomenon of homophily to analysis of Twitter social media website. Scholar proved that despite the fact that portal “promotes the exchange of different perspective ideas (...)”, users tend to follow tweets and people which are somehow similar to them in at least one field.

Homophily strategy in this simulation will be defined on couple of paradigms, which are stated in table [tab. 2] below.

|  |  |  |
| --- | --- | --- |
| Is An agent a member of Tx team? | Yes | Search for a team similar to Tx and join it |
|  | No | Do nothing |
| Is An agent in an acquaintance with Ux user? | Yes | Join one of his teams. |
|  | No | Do nothing |

Table 3: Homophily strategy paradigms

**Hypothesis 2: Homophily strategy occurs more often than heterophily strategy**

Heterophily strategy is based on a psychological manner of being attracted to opposite.

|  |  |  |
| --- | --- | --- |
| Is An agent a member of Tx team? | Yes | Search for a team different from Tx and join it |
|  | No | Do nothing |
| Is An agent in an acquaintance with Ux user? | Yes | Join one of teams quite different from him. |
|  | No | Do nothing. |

Table 4: Heterophily strategy paradigms

**Question 1: Heterophilly strategy vs. homophily strategy on the comparison of performance, which will be more efficient?**

4.7

**Expert recommendation strategy**

There are known projects regarding recommending new team members to particular projects. Most of them divide information about team and its members to create 3DSN or some other multi-dimensional social network with basic characteristics - acquaintance relations, trust and knowledge. In most cases used there are used content-based, collaborative or hybrid algorithms. On market already exist ready solutions for recommendations systems, i.e.: Expert Seeker, Small Blue, ContactMap, ReferralWeb (K.Wagner 2011).

Members of social network will recommend somebody basing on their relationship with this person and their skills. It is very unlikely that somebody whom we don’t know at all will be recommended to a team, but there are and can be exceptions implemented.

**Hypothesis 3**: Recommending and/or enforcing wiring new members to a team doesn't always improves emergence.

|  |  |  |
| --- | --- | --- |
| S1: Is this person in my friends list? | Yes | Check step 2 |
|  | No | Analyze his experience, team membership |
| S2: Does he possess required skills? | Yes | Recommend him |
|  | No | Skip recommendation |

Table 5: Expert recommendation algorithm

Input:

F: set of friends

for all Fi in ΣFi..1 do

begin

[propose here algorithm]

end

5.1

**Simulation tool**

Simulation is created in Repast Simphony 2.0 framework and coded in Java language. The simulation time is 10.000 ticks. A NetworkBuilder Repast class is used to create a non-spacious environment, but a graph of nodes and their relations.

**Simulation parameters**

Simulation parameters defined in *parameters.xml* file with their interpretation are described in table 1.

|  |  |
| --- | --- |
| Simulation parameters for Repast Simphony model | |
| numNodes | Number of start nodes, which is a number of persons – agents. |
| numTeams | Start number of teams. |
| percStartMembership | Percentage of people is already assigned to the team. |
| allowMultiMembership | Can a person be a member of more than one team? |
| avgStartMembership | In average, to how many teams a person will be assigned in the start of a simulation. |
| numSteps | Number of steps which the simulation will last. |

Table 6: AON Repast Simphony 2.0 simulation parameters

**Future plans**

Future plan is to implement web-application for dynamic re-modeling of team emergence basing on updated data from GitHub portal. It would be possible by bridging Repast modules or by using Flora research project (<https://bitbucket.org/siosonel/flora>).

**References**

List of dissertations, publications and others texts used in this research.

[1] "Agent-Organized Networks for Dynamic Team Formation" Matthew E. Gaston Marie desJardins 2005  
[2] "Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties" Jungpil Hahn  
[3] "Online Team Formation in Social Networks", Aris Anagnostopoulos 2012  
[4] "Learning and Predicting Dynamic Behavior with Graphical Multiagent Models" Quang Duongy Michael P. Wellmany Satinder Singhy Michael Kearns  
[5] R. L. Graham. Bounds on multiprocessing anomalies and related packing algorithms. In AFIPS, 1972.  
[6] K. Mehlhorn. Assigning papers to referees. In ICALP, 2009.  
[7] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In KDD, 2009.  
[8] A. Gajewar and A. D. Sarma. Multi-skill Collaborative Teams based on Densest Subgraphs. In SDM, 2012.  
[9] C.-T. Li and M.-K. Shan. Team Formation for Generalized Tasks in Expertise Social Networks. In SocialCom, 2010.  
[10] Raymond, E. S. 2001. The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary. O'Reilly and Associates, Sebastapol, CA.  
[11] The Structure of Growing Social Networks, Emily M. Jin, Michele Girvan, M.E.J. Newman, Santa Fe Institute 2001  
[12] Open Source Software Evolution and Its Dynamics, Jingwei Wu, 2006  
[13] Thomas and Hunt. Open Source Ecosystems, IEEE Software (21:4), 2004, pp. 89-91.