Walukiewicz, Stanislaw

Conference Paper

The Virtual Production Line as a model for the innovative global economy

56th Congress of the European Regional Science Association: "Cities & Regions: Smart, Sustainable, Inclusive?", 23-26 August 2016, Vienna, Austria

Provided in Cooperation with:
European Regional Science Association (ERSA)

Suggested Citation: Walukiewicz, Stanislaw (2016) : The Virtual Production Line as a model for the innovative global economy, 56th Congress of the European Regional Science Association: "Cities & Regions: Smart, Sustainable, Inclusive?", 23-26 August 2016, Vienna, Austria, European Regional Science Association (ERSA), Louvain-la-Neuve

This Version is available at:
https://hdl.handle.net/10419/174649

The Virtual Production Line as a model for the innovative global economy

Stanisław Walukiewicz

Systems Research Institute, Polish Academy of Sciences

ul. Newelska 6, 01-447 Warsaw, Poland

phone: +48 22 3810 129, mobile: +48 793 802 334, fax: +48 22 3810 105

Stanislaw.Walukiewicz@ibspan.waw.pl

Abstract

By an example of the Linux system, I introduce a concept of the Virtual Production Line (VPL) as an extension of an assembly line, brought in by Henry Ford a hundred years ago, and demonstrate its usefulness in an analysis of open source software development, in particular, and in the global innovative economy, in general. The line is virtual, since it does not exist physically and developers may be located in different parts of the world, exchanging information via Internet. They join their efforts in fixing certain bug(s) or extending some parts of the codebase. So, there exists the one to one correspondence between the Linux project and its VPL. Then, motivation and governance in the Linux project can be studied as formal and informal relations (social capital) among developers working on a given VPL. The paper proposes the VPL as a new model for an analysis and evaluation of creative labor done by Linux developers on the corresponding VPL, and generalizes its findings to the case of any creative activity.

It is useful to divide a work (labor) into routine and creative and define a creative work as a here and now negation of a routine one In the innovative global economy, I claim people will mostly do creative labor working under projects on corresponding Virtual Production Lines. Then, the success of a given project depends on social capital of the experts working on its VPL. Social capital, in turn, can be modeled and measured as cognitive and emotive proximities among these experts. The paper provides new measures for cognitive and emotive proximity and suggests generalizations of its theoretical findings. The main conclusion says that the Linux system can be considered as an example of a completely new way of organization and evaluation of creative labor in the innovative global economy. I call it the Linux world.

ACM Categories: K.6.1. Project and People Management; K.6.3. Software Management

Keywords: Innovative global economy, open source software, Linux system, human and social capital, cognitive and emotive proximities, project management, open innovation.

1. Introduction

The main aim of this paper is to demonstrate how the Linux system is changing our approach to innovation, research and to a market economy, in general. The Linux system, in short Linux, is a (very big) computer program, a result of an open source style work of thousands of developers-volunteers for more than two decades, widely used by commercial companies, government organizations and individual users. The Linux Foundation publishes regularly White Paper (http://www.linuxfoundation.org) under the same precise title *Linux Kernel Development. How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It*, further called the Report, where it presents development of the main parts of Linux. According to the February 2015 Report, the updated versions of the kernel had almost 20 million lines, and if we print 100 lines per page, then it takes almost 200 books, having 1,000 pages each. From the very beginning of Linux, two research questions: (1) why do individuals participate in the Linux projects fixing bugs or extending the codebase, and (2) how their work is organized and managed, are considered in numerous papers and books, reviewed in the paper. The paper investigates these two questions from a new theoretical perspective, based on an idea of the Virtual Production Line (Walukiewicz, 2006).

It is useful to divide a work (labor) into routine and creative, and define a creative work as a here and now negation of a routine one; for instance, using computers and the Internet was creative in the scientific community (here) some 25 years ago (then), but it is a routine activity in 2016 (now). Scientists who do not use computers and the Internet are exceptions that prove the previous sentence here (scientific community) and now (2016). A work of a blue collar on an assembly line is a commonly used example of a routine labor. This paper demonstrates that each bug fixing or a codebase extension can be considered as an example of a successful or not creative activity, done by at least two developers on a corresponding Virtual Production Line (VPL), discussed in Section 3, where among others, we study similarities and differences between the VPL and its classical counterpart. The line is virtual, since it does not exist physically and developers may be located in different parts of the world. They join efforts in fixing certain bugs or extending some parts of a codebase, and exchange information via Internet. So, there exists the one to one correspondence between a Linux project and its VPL. Then, motivation and governance in a Linux project can be studied as formal and/or informal relations among developers working on the corresponding VPL. In other words, this paper proposes the VPL as a new model for an analysis and evaluation of creative labor done by Linux developers on the VPL, and generalizes these findings to the case of any creative activity in the innovative global economy.

In the next section, we draw from the relevant literature on open source software to formulate our main assumption saying that Linux developers do not receive any tangible (financial) gratification for their contributions. Their gratifications are mostly of intangible character, discussed in Section 4, which can be measured by a utility of cognitive and emotive proximities between them. Today, many Linux developers work in software companies like Red Hat, IBM or Microsoft, as the Reports indicate, but we need this assumption for methodological reasons to present the idea of the Virtual Production Line in Section 3. Our main result is presented in Section 5, where we show that the Linux system can be considered as a constantly growing tree of Virtual Production Lines. Generalizing this result in Section 6, we lift up our main assumption to demonstrate usefulness of our model in an analysis of any creative activity in a market economy here and now. Among others, we show that Linux can be considered as an example of a completely new way of organization and evaluation of creative labor, what in fact is a new way of life. I call it the Linux world.

2. The Linux System and the Main Assumption

For a long time, the open source software was called a "free software" and the word "free" has traditionally led commercial software vendors to think "no revenue," and customers of those firms to think "no support" (see Bretthauer, 2001 for the history of the "open source software" term). Thus for a long time, the most in the commercial world saw free software as irrelevant, and free-software developers as idealistic and naïve. Similarly, many free-software advocates have seemed commercial software companies as interested only in short-term profits at the expense of the long-term interests of users and the software development community as a whole. Today these two communities are interlinked closely; open source software has a growing impact on a market economy and on software industry, in particular, and vice versa. The year 2001 should be considered as a turning point in relations between these two communities when Dell Computer, IBM and Oracle have all announced allegiance to Linux products and have offered support services to their customers who operate in a Linux environment (Anonymous, 2001 a, b and c; Gallivan, 2001). The growing success of systems like Android, Apache or Mozilla, to mention a few, demonstrates the potential importance of the open source software to businesses, as both users and for-profit producers of software (Andres, 2002; Curtis *et al*., 1998; Hansen & Kautz, 2004; Joshi *et al*., 2007; Sawyer *et al*., 2008; Williams, 2010). The Linux operating system, developed under the open source model, is a firm base for such a success.

The beginning of the Linux operating system is commonly associated with Linus Torlvalds, who in 1991, then a Finish student of computer science, wrote his first version of the UNIX kernel in the C language. He worked in a bazaar style often distributing early versions of the kernel and widely using solutions of his predecessors. He was one of the most intensive users of the Internet at that time. Raymond in his famous essay *The Cathedral and the Bazaar* called him "a damn fine hacker" (Raymond, 2012, p. 8). In common opinion he is considered as a founder of the GNU/Linux system that in short is called Linux (system). The source of the word "Linux" is unclear: while some software developers see it as a

combination of "Linus" and "UNIX," the others see it as a recursive acronym of "**L**inux **I**s **N**ot **U**NI**X**." And in fact it is not, since UNIX has now a historical value, only, while the Linux is a world class computer system with a very bright perspective. For instance, Hecker, 1999 describes eight business models for incorporating open source software into a strategy of a software company and Munga et al., 2009 present Red Hat and IBM case studies as examples of successful companies (see also Koenig, 2004; Lee & Mendelson, 2008; Lee, 2006; Ghobadi & Mathiassen, 2015 for more general applications).

I think the origin of this success is in the observation, called further the Raymond's claim, which at the first glance seems to be counterintuitive, but the Linux system proves it:

The developer who uses only his or hers own brain in a closed project is going to fall behind the developer who knows how to create an open evolutionary context in which bug-spotting and improvements get done by hundreds of people (Raymond, 2012, p. 20 and 21).

Creating "an open evolutionary context" is a way of organizing of creative work for "hundreds of people" where social capital plays a key role (Walukiewicz, 2008). Obviously, these people use ICT technology, so in fact we have an application of the idea of the Virtual Production Line to be discussed in the next section.

To present that idea we will assume that these "hundreds of people" work without any financial (monetary, tangible) gratification. In fact, their gratification is of intangible character, to be discussed in Section 4, which is much more complex than a simple monetary wages paid workers for their routine labor on an assembly line. So with our main assumption, we are back to the time of a very beginning of open source software, when "hundreds of people" worked for free contributing to the development of the Linux system. As said nowadays, open source software has a big impact on a market economy as a whole, therefore in Section 6 we lift up this assumption and consider the Virtual Production Line as a model for analysis and evaluation of creative labor in a market economy.

3. The idea of the VPL

First, we present systems approach to an analysis of an assembly line, a predecessor of the Virtual Production Line, and next describe the VPL in the same way. The idea of an assembly line was presented by Frederick Winslow Taylor (1856-1915) in his book *Principles of Scientific Management* and Henry Ford (1863-1947) was the first who applied it in motor industry in 1913.

3.1. The Classical Production Line

Before the Ford invention, cars were manufactured in so-called production circles, where a few highly skilled craftsmen produced a car from beginning to end using raw materials and

parts. The division of labor in the production circle was flexible and craftsmen could easily substitute for one another. The number of highly skilled craftsmen in a community only limited the productivity of this method.

Henry Ford was the first, who put the following observation into practice. If we partition, i.e. divide without overlapping, a complex car manufacturing process into a fixed number of simple operations (jobs) done by blue-collar workers on a line (conveyor belt) as shown in Figure 1, then its productivity will increase and the problem of a limited number of highly skilled craftsmen should be solved. Since then, the idea of the assembly line was subsequently applied in many production and service processes. If we have many production/service lines manned by people or robots, we will combine them for the purpose of our analysis into one production/service line called the Classical Production Line (CPL) pictured in Figure 1.

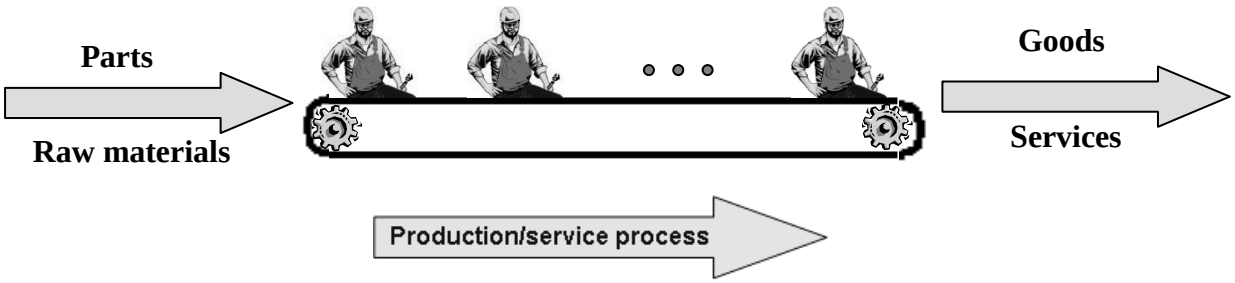

Figure 1. Classical Production Line (CPL)

Let us assume that a worker, say the second one in Figure 1, has increased his skills (human capital) and now can do the assigned operation in half the previous time. Does this have any impact on the organization/productivity of the production process concerned? Assuming that he is not a bottleneck, the answer is no. His extra skills may be used in the design and implementation of another production process on another CPL, but not on the one at hand, since its organization is fixed. Thus, we have shown that the CPL does not allow any self-organization and, roughly speaking, workers are there to work, not to think. So, we can define the CPL as a partition of a complex production/service process into a fixed number of simple operations described to the smallest detail. Such a partition is fixed over time and does not allow any self-organization.

Figure 2 illustrates this definition. The production process, represented by a (full) circle, is partitioned into n operations (jobs) $J_1, J_2, \ldots, J_n$. Since the organization of the CPL is rigid, the number of operations is constant over the relevant production process. Although it may be a bit of an exaggeration, we can say that highly skilled craftsmen are not needed in the automotive industry where organization is a priority. Because of excellent organization, manual workers can produce sophisticated cars.

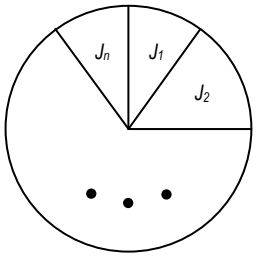

Figure 2. The rigid organization of the CPL

Ford combined moving assembly lines with mass routine labor to make building cars much cheaper and quicker – thus turning the automobile from a rich man's toy into transport for masses. He was the first to demonstrate in practice the importance of constant improvement in organization and technology. He introduced the eight-hour workday, for which he paid five dollars, two times more than his competitors did. He would pay $1,000 in cash to an ordinary worker for a relatively small innovation. The results of his policy were breathtaking: his company produced 189,000 cars in 1914 and more than two million in 1923 – a ten-fold increase in nine years! His innovative management cut the time needed to produce a car from 750 to 93 minutes (Womack et al., 1990, p. 28). In common opinion, Henry Ford is considered as a father of the assembly line, one of the greatest achievements in management science and economics, what has completely changed our life. In Walukiewicz , 2015 Chapter 1, I call it the first big jump in productivity.

3.2. The Virtual Production Line

Now, we turn back to software production under the open source software model with our assumption that developers do not get any tangible (monetary) gratification for their contributions to the Linux system. Any successful or not attempt to fix bug(s) or extend some part(s) of the codebase is a creative work because it is new (original), done for the first time here (by a team of developers) and now (presently). For obvious reasons, we consider "serious" (large) attempts, further called the (Linux) projects. The lions part of the Linux projects discussed in the literature is successful, while unsuccessful attempts form usually a base for successful projects. So, under the Linux project a team of developers tries to solve a corresponding creative problem connected with fixing bugs or extending the codebase.

A remark on the number of people in a project team working on the VPL is needed. We follow Ghobadi and Mathiassen, 2015, who in their study of knowledge sharing divide an open source team into four groups of managers, developers, testers and user representatives, called further experts. So team of experts working on the given VPL is divided into four

abovementioned groups, and this, as usual, is a division, not a partition of labor, since, for instance, a manager is typically working also as a developer or tester.

The idea of a Virtual Production Line is pictured in Figure 3 where a number of experts with their laptops are connected via Internet in their attempt to solve a given creative problem under the Linux project. Since there is no material representation of the VPL (experts may be located in different parts of the world), we denote it in Figure 3 using a dashed line in contrast to Figure 1.

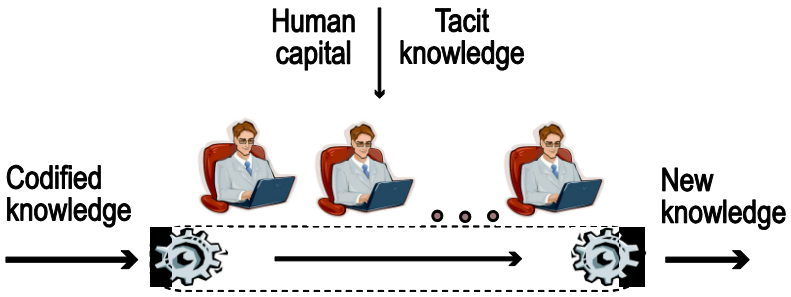

Figure 3. A Concept of the Virtual Production Line (VPL)

On the VPL, experts combine their human capital, mostly their tacit knowledge (competences, experiences, talents, etc.) with codified knowledge, contained in existing software, data bases, libraries, so forth, in their attempt to solve a problem which, at the beginning, may not be well defined and usually is described in a fussy way, but which, due to their efforts, called jointly self-organization, gets usually more and more clear-cut and distinctive (see Figure 4). In other words, experts on VPL do not only work manually (punch the keys), but – first of all – think.

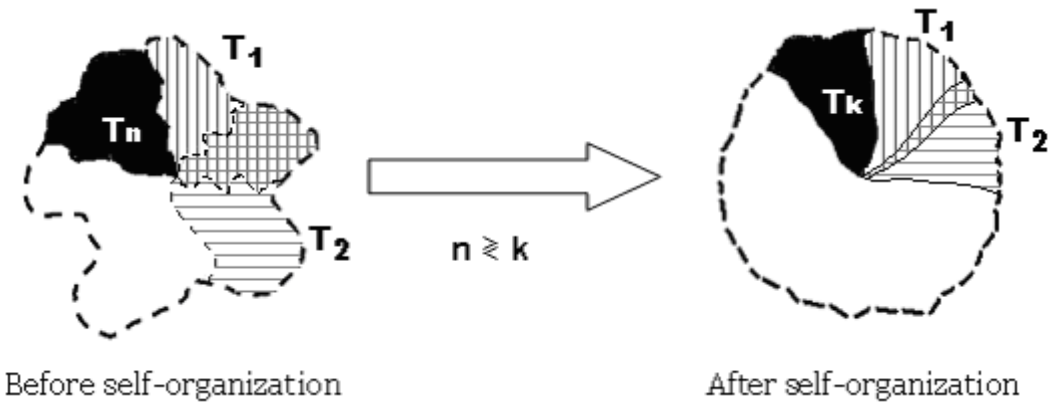

Figure 4. The VPL as a flexible division of labor and self-organization

In Figure 4, we see that at the beginning of the creative process, the problem is not well defined – we mark it with a dashed line along the perimeter. Tasks often overlap each other and their limits are not well delineated – we denote it with a waved line. After self-organization the problem is usually much better defined (it is almost a circle), the overlapping of tasks are substantially smaller and their limits are almost straight lines. If the problem is initially divided into n tasks $T_1, T_2, \ldots, T_n$, then after self-organization, it is divided into k tasks, $T_1, T_2, \ldots, T_k$, where k can be equal, bigger or smaller than n. We conclude that the VPL allows a flexible division of labor while the CPL is based on a rigid (stiff) partition of labor (see Figure 2) where production or servicing process is well defined to the smallest detail. It is a circle, where the jobs $J_1, J_2, \ldots, J_n$ do not overlap each other, and the limits between them are straight lines as Figure 2 shows. I definitely do not claim that any problem, once defined on a relevant VPL will be solved. I am far from that. I only say that a team on the VPL works in its attempt to solve a given creative problem. Figure 4 shows that, in general, after self-organization we do not have a full circle, what corresponds to a solved problem. In Walukiewicz, 2010, I describe the proof of the famous Fermat's Last Theorem as the VPL. Definitely, there were neither computers nor the Internet at that time, but mathematical journals and ordinary letters substituted them very well. Saying it differently, I present the solution of the world's greatest mathematical problem as a chain of self-organizations of the VPL corresponding to that problem. This fascinating story is described in full details by Singh, 1998.

So finally, we define the Virtual Production Line (VPL) as a division of a creative process into more or less precisely described tasks combined with modern ICT. The division into tasks, as well as its number, may be changed throughout the process by actions of experts involved in it. Such a modification is called a self-organization of the VPL.

Three remarks are necessary at this point:

First, on the VPL in contrast to the CPL, we have a division, not a partition of labor (compare Figures 4 with 2). This means that some tasks on the VPL can overlap each other as shown in Figure 4. Such overlapping can be interpreted as a consideration of a given task from different perspectives or from a different science disciplines point of view. It is a natural, basic approach used in any rational analysis. So, it is very natural that tasks on the VPL overlap each other in contrast to the fact that jobs on the CPL are disjoint, form a partition, not a division of a given process as shown in Figure 2.

Second, on the CPL operations or jobs are executed linearly in a fixed, well prescribed order: The second operation is executed only after the first one has been finished. The third job is executed only after the second one has been finished, and so on. Thus, any CPL runs linearly. On the VPL the situation is much more complex because, in general, neither the tasks are executed in a given, already prescribed (planned) order, nor they are executed sequentially. Obviously some task may be executed at the same time (in parallel) and after a self-organization the previous order of tasks may be completely changed, for instance turn upside down. Moreover, we cannot guarantee that a given task will be finished (with its success) because the VPL is an attempt to solve a given creative problem, and the considered

task, in particular. So, any success in such an attempt cannot be granted. Thus in general, the VPL runs nonlinearly. In the next subsection, we show that this nonlinearity forms the essence of self-organization.

Third, on a CPL we have a linear assignment of workers to operations, which means that each worker is assigned to one operation (task) only and each operation is executed by exactly one worker. A mathematician would say at this point that there is one to one correspondence between workers and operations. It can be pictured as a partition a given production process (full circle in Figure 2) into a fixed already prescribed number of operations. The problem of the optimal (the best possible) assignment of workers to operations is one of the most important questions in operations research and management science. On the VPL, the situation is much more complex as a given expert may participate in many tasks executed in different time, but also may participate in tasks executed at the same time, say the same day, in parallel. Moreover, if we take a day as a natural time unite, then is quite possible that a given expert works at the same time (day) on different Virtual Production Lines corresponding to problems connected with fixing different bugs or an extension of different parts of the codebase. So, the VPL is a very flexible model for organization of creative labor.

3.3. Self-Organization

Software development is a collaborative process where success depends on effective knowledge sharing (Walz et al., 1993), and this is particularly true in the case of open source software (Cockburn, 2006; Corvera Charaf et al., 2012; Ghobadi and Mathiassen, 2015; Kautz et al., 2007). In contrast to the Classical Production Line, the Virtual Production Line is not a partition (a specific division) of labor alone, but a combination of labor division and self-organization with modern ICT. In this subsection we will discuss these three components in detail and their role in knowledge sharing.

We begin with a simple example pointing out the difference between the VPL and CPL as its classical counterpart. To be more specific, consider a group of banking clerks working with credit applications. If everything in their work is described to the smallest detail by the banking procedures, then they are working on the Classical Production Line. If they may modify their credit decisions depending on, for instance, the number of applications received, the sums asked, etc., then they are working on the Virtual Production Line. Obviously, they both use computers and ICT networks in their work here and now. Since a partition of labor is a special case of a labor division into tasks, then the self-organization alone makes the difference between that CPL and VPL. Thus self-organization is a key component of any VPL.

It follows from our considerations that routine labor is done on the CPL, while creative one, connected with an attempt to solve a given creative problem, on the VPL. Then the following question naturally arises, are people alone able to solve creative problems? We

know that some animals (dogs, monkeys, etc.) can be trained to solve simple creative problems (to find a way home, count, and so forth). So, creative problems can be solved not only by man. But, I claim that a human being only can reformulate a given creative problem, which often is a way to solve it. In fact, a reformulation of a given creative problem is equivalent to the self-organization of the VPL corresponding to it.

When we have a creative problem here and now, then we usually organize a project team, consisting of at least two experts to solve it. Obviously, nowadays the experts divide somehow the labor (project, work) between them and use computers and ICT networks in their work. So they build (form) the corresponding VPL to solve the creative problem at hand. Solving it experts can change the division of the problem into tasks, as well as the number of tasks and the way the tasks are executed (for instance, sequentially or in parallel), which is the essence of a self-organization of the VPL. Since a self-organization (reformulation) is the main component of any VPL and only man is able to reformulate creative problems, then we conclude that only men work on a VPL. So, in contrast to the CPL, the VPL without men working on it does not exist.

Now, we would like to point out the importance of ICT in more general context. This is true insofar as we realize that social capital became a subject of serious studies only in the 90's, when we began to be able to send information electronically in any form (data, picture, voice, movie) to virtually every corner of the world at almost zero cost. I call it the second big jump in productivity (Walukiewicz, 2015 Chapter 1).

In general, we may define the VPL is an instrument (a virtual conveyor belt) that experts use to combine codified knowledge with their tacit knowledge to produce improvements in products, services, technology and management, and in that way contribute to the world's stock of knowledge, both codified and tacit (see Figure 3). On the VPL experts combine their efforts and use their human capital (knowledge, experience, talent, etc.) in their attempt to solve a given creative problem. Since we consider both the Classical Production Line and the Virtual Production Line as methods of analysis of socio-economic reality, then we write their names with capital letters.

4. Proximity

The efficiency of the Linux project depends on formal and/or informal relations (social capital) between experts working on the corresponding VPL. The pair of experts, named *A* and *B*, is a departure point in our study of social capital of the Linux project (team) working on the given VPL because when studying the social capital of a given group of experts we have to consider all possible pairs within that group. Studying formal and/or relations between experts *A* and *B* for obvious reasons we restrict ourselves to relations that are directly connected with the VPL and call them jointly proximity. Proximity literally means nearness (near friends), closeness (close neighbours), contiguity (contiguous countries) and propinquity (propinquity of individuals). The concept of proximity was introduced and developed by the

French proximity school (Rallet & Torre, 1999; Tore & Gilly, 2000; Tore & Rallet, 2005) and further studied by Menzel, 2006. It can be shown (Walukiewicz, 2008 and 2015 Chapter 14) that similarly to four capitals (financial, physical, human and social) we have four proximities: technological, emotive, spatial and organizational. The first two are most important in study a collaboration of experts *A* and *B* and we consider them in turn.

Technological proximity (TP) between experts *A* and *B* describes differences and similarities in the shared knowledge, both codified and tacit, technological distance between them in used methods, terminology, approaches, etc. Therefore technological proximity is also called cognitive proximity. Similarly to the case of human and social capital, we would like know, and if possible measure, how much technological proximity between experts *A* and *B* contribute to their success on open source market, in general, and on the VPL, in particular. To do so, we introduce the utility (measure) of technological proximity which value changes continuously between zero and one, i.e. $0 \leq u(TP,A,B) \leq 1$, as a function of the technological distance between experts *A* and *B*, denoted as $d(TP,A,B)$. In Figure. 5, the horizontal axis is the technological distance between *A* and *B*, while the vertical one is the utility of technological proximity. Then, we suggest the function *f* of $d(TP,A,B)$ to be like the Gauss curve shown in Figure 5.

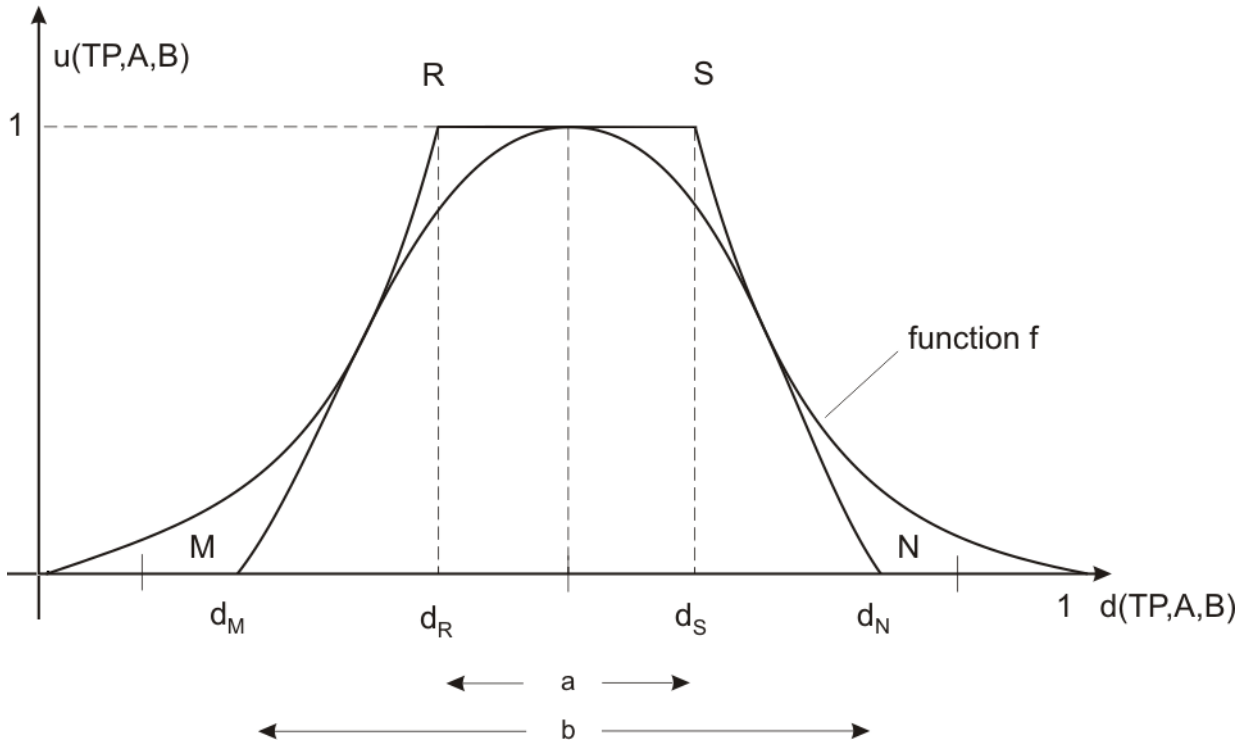

Figure 5. Utility of technological proximity as a function of technological distance

Without loss of generality, we may assume that $0 \leq d(TP,A,B) \leq 1$, where 1 is the largest possible technological distance between experts on the VPL or the Linux project. If experts *A* and *B* know the same about that project, then the technological distance between them is zero

($d(TP,A,B)$ =0), and such a collaboration do not contribute to their success. Therefore in Figure 5, the utility of their collaboration is zero ($u(d(TP,A,B) = 0) = 0$). If one of experts knows everything about that project and the other knows nothing, then the technological distance between them is maximal possible, that is $d(TP,A,B) = 1$, and the utility of their collaboration is, again, zero ($u(d(TP,A,B) = 1) = 0$). From Figure 5, one can read that the maximal utility of technological proximity is for the technological distance equal one half.

The curve f which looks in Figure 5 is, in general case, unknown. Therefore, at the end of this subsection we describe one of possible ways of its approximation by the MRSN trapezium. We suggest approximating the curve f in Figure 5 in such a way that the surface below it equals the surface of the MRSN trapezium with its height equal one. Then, in mathematical terms, we have

$$\int_0^1 f(d)dd = \frac{a+b}{2}.$$

We observe that to define the MRSN trapezium in Figure 5 we have to define its four vertices, that is to define four technological distances d_M, d_R, d_S. and d_N between experts as potential candidates to form a pair (A,B) that can be done, for instance, by an appropriately designed questionnaire.

Emotive proximity (EP) is related to personal relations like trust, emotions, common personal experiences, etc. between two particular experts as individuals. Emotive proximity forms a social environment surrounding any collaboration. It can be measured by the utility of emotive proximity introduced in the way similar to that of the technological proximity (Walukiewicz, 2008 and 2015 Chapter 14).

5. The Main Result

If the Linux project is successful, then its result is called the patch. Each patch should be presented as a logically justified change that can be reviewed for code quality and correctness, and additionally, each patch should, when applied, yield a kernel which still builds and works properly. So, each patch is the Virtual Production Line with a least two experts (a developer and a reviewer/coordinator) where usually self-organization (see Subsection 3.3) is applied many times. All such VPLs are logically connected because they form the Linux system which looks like a tree. Obviously, there are no limits on the development of that system. Therefore our main result states that the Linux system is an infinite set (tree) of Virtual Production Lines.

In other words, the Linux system is one of possible ways of organizing creative labor (work) or research activity (of experts). Our main result states that it is impossible to imagine a day when all, absolutely all computer codes are fine and nothing can be improved or extended. Therefore, our tree of Virtual Production Lines is constantly growing. So, the VPL, in general, and self-organization, in particular offer a new way of studying knowledge sharing

and synergy (Cockburn, 2006; Corvera Charaf et al., 2012; Ghobadi and Mathiassen, 2015; Kautz et al., 2007; Walz et al., 1993).

In the traditional close software business model, a given company provides all (or almost all) of the value to customers, and realizes revenues and profits in return through traditional license fees. In the open source business model, much of the value is not provided by the company as such, but by the outside experts (developers) who are attracted to work 'for free,' without tangible payment. As said in Section 2, nowadays many Linux developers work (for money) at software companies like Red Hat or Microsoft, but for methodological reasons we have assumed that all Linux developers work for free. These 'outside' developers may be motivated by the prospect of working with software that solves important problems for them, the opportunity to increase their own personal knowledge or the ego satisfaction of enhancing their reputation among their peers. The necessary condition for such cooperation is that the company treats its 'outside' developers fairly and provides them with the freedom and the other intangible payments that they do want (and demand). For instance, they should know the terms and conditions under which the company's open source products can be used, modified and redistributed. So, the relation between the 'outside' developers and the company are mostly informal, based on trust as an element of emotive proximity studied in previous section.

6. Generalizations and conclusion

By common understanding a typical research, as an example of creative work, is done in (splendid) isolation lasting months or years with its final result to be evaluated in public on the research market (Walukiewicz, 2012, 2014 and 2015 Chapter 15). So, developers of the open source software have created a completely new Linux market where results of their work are evaluated. This way we have arrived at an important conclusion that the Linux market is the Linux world. This conclusion is similar to a claim that we all are living in a (free) market economy or, in fact, our life is a free market since everything has its own (economic) value defined in appropriate market. In this paper, we have demonstrated why the Linux world not only didn't fly apart in confusion and conflicts, but seemed to go from strength to strength at a speed barely imaginable to developers of the closed source software. In Section 2 we have shown the Linux world is a part of the (free) market economy. We argue that it is a fast growing part.

Open innovation can be considered as a natural extension of open source software. The open innovation paradigm is often contrasted to the traditional or "proprietary" model where internal R&D activities lead to products or services that are developed and distributed by the firm. (Chandler, 1990) In Conant, 2002 this model was summed as "picking a man of genius, giving him money, and leaving him alone." Obviously, getting the ideas from the "man of genius" was only half the challenge, the other half was to exploit those innovations. Here open source software, and Linux provide a variety of examples (Cockburn, 2006; Corvera Charaf et al., 2012; Ghobadi and Mathiassen, 2015; Kautz et al., 2007; Shah, 2006).

As a final note, the Linux market (world) can be considered as a laboratory in the social sciences and in economics in particular – a fascinating laboratory where real people, with all their merits and faults, invest their real time and efforts, but where time between the result publication and its (market) evaluation is shorter than that time of the world economy. We will use this laboratory in forthcoming papers.

References

Andres, H.P. (2002) A comparison of face-to-face and virtual software development teams. *Team Performance Management*, **8**, 39-48.

Anonymous (2001a) Dell and Oracle sign Linux server development pact. *New York Times*, **February 1**.

Anonymous (2001b) Tech firms crowed Linux conference. *New York Times*, **February 1**.

Anonymous (2001c) IBM says Linux is ready for prime-time. *New York Times*, **February 4**.

Bretthauer, D. (2001) Open source software: A history. *UConn Libraries Published Works*. Paper **7**.

Chandler, A.D. (1990) Scale and Scope. Cambridge, MA, Belknap.

Cockburn, A. (2006) Agile Software Development: The Cooperative Game (Agile Software Development Series), Boston, Addison-Wesley Professional.

Conant, J. (2002) Tuxedo Park. New York, Simon & Schuster.

Corvera Charaf, M., Rosenkranz, C. & Holten, R. (2012) The emergence of shared understanding: applying functional pragmatics to study the requirements development process. *Information Systems Journal*, **23**, 115-135.

Curtis, B., Krasner, H. & Iscoe, N. (1988) A field study of the software design process for large systems. *Communications to the ACM*, **31**, 1268-1287.

Gallivan, M.J. (2001) Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies. *Information Systems Journal*, **11**, 277-304.

Ghobadi, S. & Mathiassen, L. (2015) Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal*, doi:10.1111/isj.12053.

Hansen, B.H. & Kautz, K. (2004) Knowledge Mapping: A Technique for Identifying Knowledge Flows in Software Organizations, Berlin, Springer.

Hecker, F. (1999) Setting up shop: the business of open source software. *IEEE Software*, **16**, 45-51.

Joshi, K.D., Sarker, S. & Sarker, S. (2007) Knowledge transfer within information systems development teams: examining the role of knowledge source attributes. *Decision Support Systems*, **43**, 322-335.

Koenig, J. (2004) Open Source Business Strategies. Riseforth Inc.

Kautz, K., Madsen, S. & Nobjerg, J. (2007) Persistent problems and practices in information systems development. *Information Systems Journal*, **17**, 217-239.

Lee, D. & Mendelson, H. (2008) Divide and conquer: competing with free technology under network effects. *Production and Operations Management*, **17**, 12-28.

Lee, J.A. (2006) Government policy toward open source software: the puzzles of neutrality and competition. *Knowledge, Technology & Policy*, **18**, 113-141.

Menzel, M.-P. (2006) Dynamic proximities. Towards a concept of changing relations," *presented at the 5th Proximity Congress, Bordeaux*.

Munga, N., Fogwill, T. & Williams, Q. (2009) Adoption of open source software in business models: a Red Hat and IBM case study. SAICSIT'09 Proceedings of the 2009 Annual Conference of the South African Institute of Computer Scientists and Information Technologists, 112-121, ACM, New York.

Rallet, A. & Torre, A. (1999) Is geographical proximity necessary in the innovation networks in the era of global economy?, *GeoJournal* 49, 373-380.

Raymond, E.S. (2012) The Cathedral and the Bazaar. **Retrieved 18 April**.

Sawyer, S., Guinan, P.J. & Coopider, J. (2008) Social interventions of information systems development teams: a performance perspective. *Information Systems Journal*, **20**, 81-107.

Shah, S.K. (2006) Motivation, governance, and viability of hybrid forms in open source software development. *Management Science*, **52**, 1000-1014.

Singh, S. (1998) FERMAT'S ENIGMA. The Epic Quest to Solve the World's Greatest Mathematical Problem, New York, 1st Anchor Book.

Taylor, F.W. (1911) The Principles of Scientific Management. Harper & Brothers, New York.

Torre, A. & Gilly J.P. (2000) On the Analytical Dimension of Proximity Dynamics, *Regional Studies* 34, 169-180.

Torre, A. & Rallet, A. (2005) Proximity and Location, *Regional Studies* 39 47-59.

Walukiewicz, S. (2006) Systems analysis of social capital at the firm level. *Working Paper WP-1-2006*, Systems Research Institute, Polish Academy of Science, Warsaw.

Walukiewicz, S. (2008) The dimensionality of capital and proximity. *Proceedings of ERSA 2008 Conference*, Liverpool, **August 27-31**.

Walukiewicz, S. (2010) Kapitał ludzki (in Polish: Human Capital), Systems Research Institute, Polish Academy of Science, Warsaw.

Walukiewicz, S. (2012) Kapitał społeczny (in Polish: Social Capital), Systems Research Institute, Polish Academy of Science, Warsaw.

Walukiewicz, S. (2014) The market triple, *CEJOR*, **22**, 713-727.

Walukiewicz, S. (2015) Two-Dimensional Economics. A manuscript, Systems Research Institute, Polish Academy of Science, Warsaw.

Walz, D.B., Elam, J.J. & Curtis, B. (1993) Inside a software design team: knowledge acquisition, sharing, and integration. *Communications of ACM*, **46**, 63-77.

White Paper (2015) "Linux Kernel Development. How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It," (http://www.linuxfoundation.org).

Williams, C. (2010) Client-vendor knowledge transfer in IS offshore outsourcing: insights from a survey of Indian software engineers. *Information Systems Journal*, **21**, 335-356.

Womack, J.P., Jones, D.T. & Roos, D. (1990) The Machine That Changed the World, Rawson Associates.