ISBN 978-83-60810-22-4

Adapting Scrum for Third Party Services and Network Organizations

Lukasz D. Sienkiewicz
Wroclaw University of Economics
Komandorska Street 118/120,
53-345 Wroclaw, Poland
Email: sienkiewicz.lukasz@gmail.pl

Leszek A. Maciaszek
Wroclaw University of Economics
Komandorska Street 118/120,
53-345 Wroclaw, Poland
Email: leszek.maciaszek@ue.wroc.pl

***Abstract*--Large number of scientific publications and press releases demonstrate that organizations are adopting the Scrum software development method with success in almost all areas. Nevertheless, traditional Scrum method is not sufficient for managing work in Network Organizations where Third Party Service providers may know nothing about the Scrum. This paper describes the findings of a field study that explores the Scrum in Network Organizations. We extended Scrum core roles and proposed changes in Scrum artifacts that help in adapting the Scrum method to work in Network Organization where changes and high competition are the cornerstone of the entire process.**

I. Introduction

SELECTING appropriate approach to systems development is crucial for success of the project [1]. Fortiori this is especially important when firms are working as a Network Organization where the environment is turbulent and uncertain. Moreover, the firms that make up that kind of cooperation are likely to be using disparate methods for their internal development projects.

In this paper we propose an adaptation of the Scrum method that suits best for managing development of software applications in Agile environment in a Network Organization using third party internal and external services:

- We refer to the Network Organization and determine how the Third Party Services and the Scrum are interrelated.
- We take a *holonic view* [2][3] on the process and method of software development.
- We propose a Scrum-based software development model that specifically includes some novel and excludes some conventional artifacts and rules.
- We propose a set of metrics (i.e. Key Performance Indicators) that help to control and coordinate proposed model.

The proposed model offers an innovative approach for systems development in Network Organizations. The model has evolved from the Scrum method and has been checked in practice.

II. Network Organization and Third Party Services in Terms of Software Application Development

A. Network Organization

"*A pattern of social relations over a set of persons, positions, groups, or organizations*" [4]

This definition proposed by Lee Douglas Sailer [4] and further elaborated by Marshall van Alstyne [5] is very useful because it emphasizes structure and different levels of analysis. The Network Organization is a collection of autonomous firms or units that behave as a single large entity (i.e. structure), using specific mechanisms to control and coordinate the entire project. The entities that make up that kind of organization are usually legally independent entities (separate companies). However, this is not the rule because some of them may be divisions within the company (sub-organization) that sell to outside customers, or they may be wholly owned subsidiaries providing the third party services to the entire network.

Some advantages of software development in network environment are customization, task basis, and the Structural Embeddedness [6]. These advantages are favouring individual firms and their members. Network Organizations occur in the situation when technology and markets are changing very fast, so consequently, all participants have to coordinate and control the units in some other way, for instance by mechanism design, trust, and Macro-Culture [5].

B. Third Party Services

Network Organizations fall halfway between vertical integration and market disaggregation. They facilitate building packets of services, according to the nature of provided services and relations between them as shown in "Fig. 1".

For the purpose of discussion, we have distinguished two types of third party services provided for software development in Network Organizations:

- Internal Services: time-consuming activities, which are important but additional to the entire project. Usually those are provided by subunits of a large company (e.g. internal UX expertise, internal testing, ICT support, etc.)
- External Services: all those issues that must not be covered by Internal Services and should be handled by other firms (e.g. authorized computer service, external testing service, translations, etc.)

We observe that companies that see their units as separate cost or profit centres (providers of internal services), may encourage the units to sell their services outside the company. The reason is that if the units have to operate within market, they will improve performance, better manage the prices, and of course earn money for the entire organization. The cooperation between services providers, usually establishes a long-term relationship between suppliers (i.e. providers of external services), who may then participate in planning sessions and influence the workload and schedule.

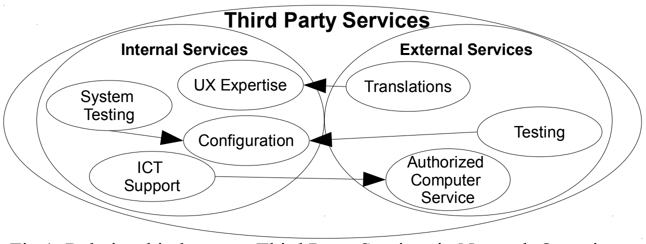

Fig 1. Relationship between Third Party Services in Network Organization

Both internal and external services may involve other third party services (internal or external) so the amount of dependencies, risk and uncertainties may be fast increasing. This in turn may adversely affect software development process by deterioration of quality, changes in the scope, delays in the schedule, and all this may contribute to the project failure.

In the next sections, we will explain the holistic nature of the Scrum. Based on an example of Network Organization and services, we will present the dependencies between key entities in the Scrum process and third party service providers.

III. Agility in Network Organization

A. Agile, Scrum and Engineering Practices

To highlight differences in the impact of Agile, scrum and Engineering Practices in Network Organizations we take advantage of the "*Three Level Framework*" designed by Geary A. Rummler and Alan Brache [7]. We propose the framework that takes the Scrum viewpoint and distinguishes three types of layers:

- Organization Level: all activities that are additional for Scrum (e.g. Human Resources, financial, capability, management, etc.), and identify the organization point of view (i.e. market, competitive advantage, priorities, products and services).
- Process Level: series of steps, rules and artifacts, which are used by the Scrum team to produce the product or service. The goals of this level are developed from customer requirements (i.e. Sprint Planning) and benchmarking information (i.e. during Sprint Review/Retrospective Meeting).

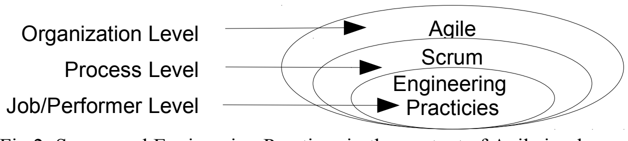

Fig 2: Scrum and Engineering Practices in the context of Agile implementation in Network Organization

- Job/Performer Level: all undertakings and instrumentation essential to achieving the goals of the process (e.g. code review, pair programming, continuous integration, etc.).

For the research presented here, we have assumed that Network Organization is following Agile principles and implements Scrum as a method for managing the software development team (Job/Performer Level in "Fig. 2"). Therefore, all Engineering Practices are considered separately, as not covered by Scrum but used as external or internal services (as discussed next).

B. Scrum Roles – Core and Additional

We would like to clarify that describing all Scrum principles and artifacts is not the subject of this paper, therefore we will not focus on default core Scrum roles (i.e. Product Owner (SM), Scrum Master (SM) and Scrum Team (ST)), but only on additional roles that are essential for further consideration.

In addition to core roles, we consider the groups of people known as: managers [8] and stakeholders (i.e."*... any group or individual, who can affect or is affected by the achievement of the organisation's objectives.*" [9]).

For the sake of proper adaptation of Scrum to work with Third Party Services in Network Organization, we propose another core role, excluded from stakeholders group:

- Third Party Service Provider (S): organization or individual who provide your organization with specialized third party services (e.g. lawyers, accountants, coaches, consultants, translators, internal and external service providers, etc.).

From our point of view, this new role S is crucial for the success of the Scrum performed in Network Organizations. This role should be involved in the entire software development process.

IV. Holistic Nature of Scrum

A. Holons and Holarchies

"*Living systems are organized in such a way that they form multi-levelled structures, each level consisting of subsystems which are wholes in regard to their parts, and parts with respect to the larger wholes*." [10]

The idea of holon entity was introduced by Arthur Koestler in [11]. He coined the term "*holon*" for those entities, which might be simultaneously a part and a whole [2].

We can imagine that each holon has two opposite habitudes (tendencies): an integrative habitude to exist as a part of compound system and a self assertive habitude to preserve its individuality. Those two tendencies are complementary, although they also are opposite. The balance between habitudes is not static, but is adapting based on influence of two complementary tendencies. This makes the whole system open to change and very adaptive.

Thirty years after Koestler's original idea, another philosopher Ken Wilber generalized the idea of holons by highlighting its relative and conceptual nature. In [12], he considered that holon must have four basic characteristics [3]:

- Self-preservation: to maintain own structure, independently of the material that holon is made of.
- Self-adaptation (community): to adapt and link up with other superordinate holons, in order to react biologically, mechanically or intentionally to their stimuli.
- Self-transcendence: the holon has its own characteristics and qualities, which are new and emerging; new properties emerge in superordinate holons and create new classes of holons.
- Self-dissolution: the holons break up along the same vertical lines that they are formed.

Due to their nature, holons are connected to other holons in a typical vertical arborising structure known as a holarchy, which can be viewed as multilayer system, with tree-structure. From the holonic point of view, each member of the organization (e.g. Network Organization), can be considered a holon. It means that each member is a whole if observed as a separate unit and a part if looked at as a member of larger organization. Therefore, the core and ancillary Scrum roles can be interpreted as holons forming a holarchy, when taking into account communication network designed between holons.

B. Holistic View of Scrum and Third Party Services in Network Organizations

"*The holonic view of the world forms a middle ground between atomism and holism and the holonic structures form a middle-ground between network and hierarchic structures. The stratified order of holonic layers resembles a hierarchy of layers and allows flat networks within layer, but it is different from both. The stratified order is not about rigid transfer of control or about free interconnectedness of nodes, but it is rather about the self-organization of complexity and adaptation.*" [13]

Considering Network Organization as a network of intercommunicating elements, we can easily show that the amount of communication paths grows exponentially with addition of new elements [14].

Because network is an overly complex structure, we need some form of hierarchy with some aspects of superiority between elements. Holarchies seem to be the most suitable structures to manage complexity due to their special form of stratified hierarchy without traces of ranking between its elements (holons) and without cycles.

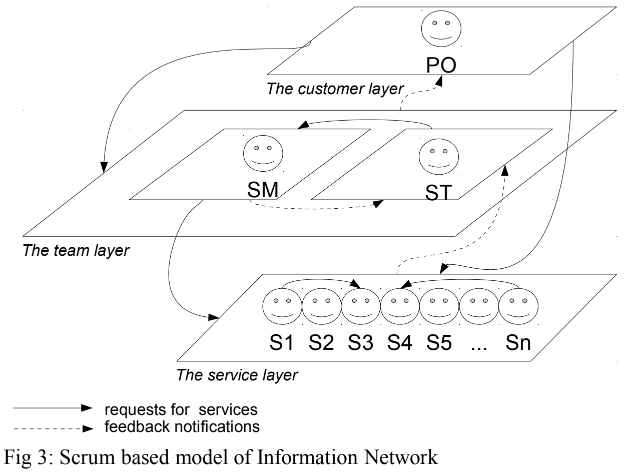

Fig 3: Scrum based model of Information Network

We propose simple three layers model of holarchy ("Fig. 3"), where default core roles(PO, SM, ST) are placed in first two layers and the new core role(S) is placed in the lowest layer. In this model, we skip ancillary roles of Managers and Stakeholders, because of low importance for further consideration.

Layers in a holarchy have an autonomy that enables them to adapt to new circumstances and changes in the environment. All communication that represents request for third party services is downward. Superordinate layers depend on the sub-ordinate layers for third party services, however not vice-versa. The lower layers inform about its state changes by providing the feedback to requesters, possibly but not necessarily from the upper layers.

Wherein our model all dependency relationships between layers are downward, and upper communication is only by providing the feedback, that helps in avoiding cycles of messages and makes communication more efficient.

The proposed model consists of three layers that refer to core Scrum roles:

- The customer layer: with Product Owner (PO) as the main requester. Third party services might be requested directly from this layer, however results (feedback) will be delivered to ST layer.
- The team layer: includes two sub-layers where entities (holons) are Scrum Master (SM) and Scrum Team (ST). ST receives results from S, requested by PO or SM.
- The service layer: this layer represents third party services and third party services providers (S). It is possible that entities in this layer will have interconnectedness between each other (e.g. some S might request services from other S).

The same solution can be used in software development of distributed projects what is shown in "Fig. 4".

The difference occurs in additional interconnectedness between entities from middle-layer (e.g. Scrum of Scrums Meeting may be represented as SM request service from another SM). In addition to model proposed in "Fig. 3" the middle-layer is represented by two multi-entity sub-layers of SM's and ST's with interconnectedness between them. The main information flow has been kept without any changes.

We propose the model, which supports building trust, multi-culture and massive design (section *II.A.*). This is very important when technology and markets are changing very fast. The impact of S on the "*The Team Layer*" results is essential and should not be omitted (e.g. estimates proposed by ST during Sprint Planning should take into account S and dependencies associated with delivering results of third party services - delays in layer S affect results from layer ST).

C. Scrum Artifacts

In this paper we assume that the reader is already familiar with Scrum [1][8][15][16][17][18][19][20][21], therefore we describe only those Scrum artifacts that we propose to change to work better in Network Organizations:

- Task-feasibility instead of time-estimation: we skip using formal time-estimates and try to commit only those User Stories that we are able to deliver before next

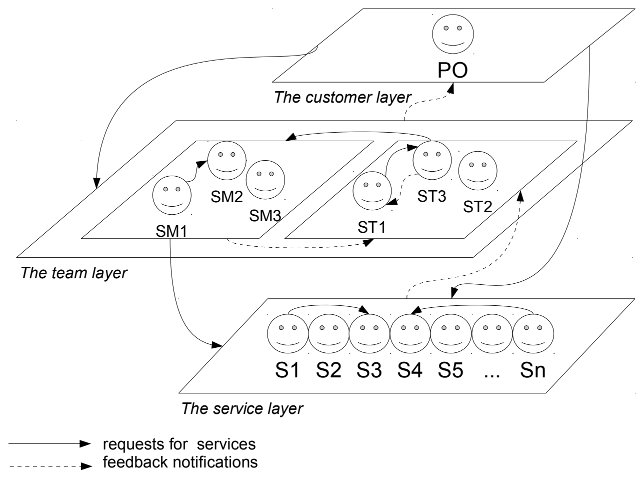

Fig 4: Scrum based model of Information Network in distributed projects

demo session. Through this change, we want to limit commitment, which we are not able to provide.

- Report Meeting instead of Sprint Review Meeting: because regular Sprint Planning session involves many resources (i.e. a lot of participants), we propose to limit participants only to representatives of the customer and the team. In our opinion that kind of meeting should be held more frequently than Sprint Review Meeting (e.g. every week) in order to improve information flow between the customer and the team.
- Planning on demand instead of Sprint Planning Meeting: because we skipped time-estimations we also propose to limit number of Sprint Planning Meetings and hold them only if really needed (i.e. when the customer needs help from the team, because is not able to prioritize Product Backlog without an additional Team's expertise).

We agree with Scrum advocates that planning up-front the Sprint is very important. However we suggest limiting this action only to prioritizing the scope (i.e. Product Backlog), so that the most important User Stories are on top of the Product Backlog list. Of course, the team is committed to delivering proposal of items, feasible for the next Sprint, but without specifying how much time will spend on each item.

Instead of regular Sprint Review Meeting held after each cycle, we propose more frequent Report Meeting (e.g. Weekly Meeting). During this meeting, the team representative is reporting actual status and points out all unsolved issues (i.e. impediments). The shortest but more frequent meetings, with reduced number of participants result in better communication and better overall understanding of project goals.

Skipping time-estimates interferes with regular Scrum Planning Meeting. Therefore, as an alternative, we propose organizing planning sessions on demand, only if required (i.e. planning the scope of next release) instead of time-consuming regular meeting before each cycle.

The proposed changes arise from the fact that we adopted a holistic view (mentioned in section *IV.B.*) that reduces the dependencies between the layers (e.g. no information cycles, stable workload during the Sprint, etc.). In our approach the feedback notifications from S to PO must go through *"the team layer"*, thus ensuring that ST and SM are fully involved in providing deliveries. For instance, the ST will not be able to deliver implementation of new wizard until they get all required translated texts from translators (i.e. S), so this implies that ST and SM must keep an eye on S and their deliveries.

D. Key Performance Indicators

"*A performance indicator can be defined as an item of information collected at regular intervals to track the performance of a system.*" [22]

Within original Scrum we use only one metric (i.e. indicator). This is the time-estimate of the amount of remaining work that needs to be done versus amount of User Stories or Tasks that are set as "*done*" in Sprint Backlog [21]. We propose to use the following KPI's (i.e. Key Performance Indicators) that help better control software development in Network Organization:

- Reliability: to measure if the team is delivering what they said they will. We compare the difference between the amount of committed Story Points (c_i) and delivered Story Points (d_i) like shown in (1). The values might be presented as the percent of reliability calculated per Sprint (R_i).

$$R_i = \frac{c_i}{d_i} * 100\% \qquad (1)$$

- Productivity: to measure project velocity. We measure amount of fixed bugs (b_i) and newly implemented requirements (s_i) like shown in (2). The value of productivity (P_i) should be calculated after each Sprint.

$$P_i = b_i + s_i \qquad (2)$$

- Effectiveness: to measure effectiveness of testing service. The measure includes the amount of defects delivered to the customer. Based on this KPI we can calculate the effectiveness of internal testing service (like shown in (3)), by measuring the ratio between all found defects and those found by external S providing complementary testing. This shows effectiveness (E_i) of software development team and testing services.

$$E_i = \frac{a_i - e_i}{a_i} * 100\% \qquad (3)$$

We believe that the introduced KPI's are crucial to maintaining customer satisfaction. From our point of view, the required data should be collected at the end of each Sprint. We would like to point out that the same KPI's can be measured for S (section *III.B.*) and their findings can be used by the team for increasing customer satisfaction and for coordinating and controlling workload status.

V. Case Study

Our approach to using Scrum in Network Organization is best illustrated by a real life example of two similar projects managed in two different variants of Scrum:

- In the project A: the pure implementation of Scrum.
- In the project B: the Scrum extended to our guidelines.

The company used in this study is a large multi-national organization (over 17000 employees all over the world) specialising in R&D services, telecommunication and mobile solutions. The customer is a large multi-national organization specialising in telecommunication and mobile solutions. The contract between companies was a typical outsourcing service.

A. Structure and Scope of Projects

The project A was a software application, dedicated for care centres for upgrading mobile device software via computer. It was a user-friendly application with an ergonomic presentation layer (UI) and very complex middle-layer to handle hundreds of different variants of mobile devices. The presentation and middle-layer text contents were localised for 40 different languages.

The project B was very similar to the project A, however it was not a stand-alone application, but a part of bigger software application (i.e. software update plug-in) dedicated for end users managing their mobile devices via computer. The plug-in was using the same middle-layer as in project A. The presentation layer was also represented by a user friendly UI localized for 40 languages.

At the time we conducted the case study, the two projects were in maintenance and support phase (i.e. about 70% of workload was bug-fixing and 30% implementation of new functionalities), so the case study did not relate to implementation from the scratch but to maintenance of the existing products.

The projects started with a short initiation phase during which the product backlog was set, general architecture was established, Scrum roles were assigned, and Scrum principles were known for all involved persons.

Iteration length was set at two week interval throughout the project, therefore after first two weeks of project initiation, the team finished the first iteration (i.e. Sprint 0).

In both cases A and B the Scrum Teams were following engineering practices (referred to in section *III.A.).*

B. Roles Assignment

Both projects were using services S (section *III.B.*), such as: translations, complementary external testing service, User eXperience expertise service.

The team composition had not changed since its inception, and for the moment of data collection, the number of persons in software development teams was the same for both projects: three software engineers and one test engineer.

The software development teams and Scrum Masters were co-located (i.e. based in Poland) that allowed building personal relationship between all team members inside their teams.

The customer (i.e. Product Owner) was remotely involved in the project, due to different location (i.e. based in Finland).

C. Meetings

The project A was following all Scrum principles; therefore, all meetings were held as defined.

The project B was performed as we suggest in this paper: instead of Sprint Review was Weekly Meeting and instead of regular Sprint Planning meeting was Planning Meeting on demand.

D. KPI's Results

We measured reliability, productivity and effectiveness values, known as KPI's (introduced in section *IV.D.*) to compare approaches taken by teams A and B. All presented metrics were collected for 13 consecutive Sprints (i.e. each Sprint was two weeks long). We present results of measuring reliability in “Fig.6”, by comparing deviation of the reliability calculated for both projects (i.e. A and B). In both projects expected values were 100% of reliability per each Sprint.

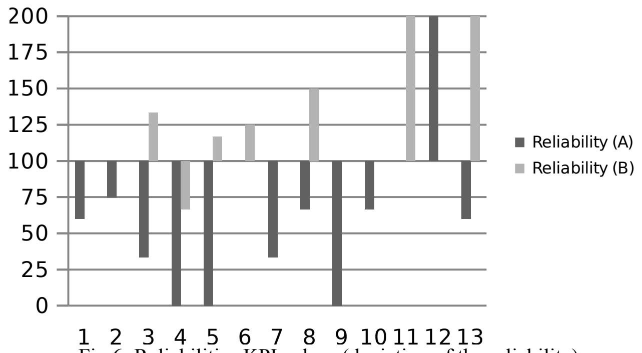

Fig 6: Reliabilities KPI values (deviation of the reliability)

We note that the approach taken by team A, was affected by large number of cases that the team promised but it was not able to deliver committed stories. We also note that our approach was not “the silver bullet”. Therefore, in “Fig.6” the bars are shown to indicate when team promised less than was able to deliver.

The approach taken by team B (i.e. ~46%) was three times more accurate than approach taken by team A (i.e. ~15%).

We checked the productivity of teams A and B. The results are presented in “Fig. 5”. The amount of work realized by the teams (i.e. fixed defects and implemented user stories) was comparable and their values depended on the release scope.

Because the quality was crucial for both projects we measured effectiveness of our internal testing service (i.e. internal S) and compared the amount of found defects to effectiveness of external testing service (i.e. external S). In “Fig. 7” we present the ratio of leaked defects to the number of defects found by our internal testing service.

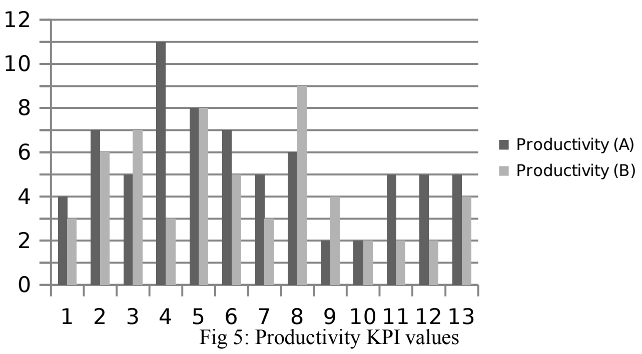

Fig 5: Productivity KPI values

TABLE I.
CUSTOMER SATISFACTION SURVEY – QUESTION FORM

ID	Questions	Project A Scores (1-4)	Project B Scores (1-4)
Project Quality:			
1	What was the quality of project work in general?	3	4
2	How good was schedule estimation accuracy (timeliness)?	3	4
3	How good was effort estimation accuracy?	3	N/A
4	Tasks in the project were professionally accomplished?	3	4
5	The supplier managed the delivery well (delivery management and control)?	3	4
6	Possible changes in project personnel did not have any effect on the delivery?	N/A	4
7	Project's actual total cost corresponds to the expectations in the beginning of the project?	N/A	4
8	Work as a whole correspond to my expectations	3	4
Project Personnel:			
9	Project personnel accomplished their issues/tasks as promised?	3	4
10	Project personnel had a good knowledge of their own professional area?	3	3
11	Project personnel were easy to reach?	4	4
12	Project personnel worked professionally and efficiently?	4	3
13	Project personnel were genuinely interested in solving my issues and/or my problems?	4	4
14	Project personnel had the ability to cooperate?	3	4
15	Project Manager was reliable?	4	4
16	Communication between the project personnel was fluent and there were no information barriers in place?	3	4
17	Project personnel as a whole correspond to my expectations?	3	4
Customer Input:			
18	As the customer, we were able to accomplish our won tasks and obligations as promised?	3	3
19	As the customer, we were able to provide enough time for the delivery?	3	2
20	As the customer, we were satisfied with our won specifications in the beginning of the project?	N/A	3
21	As a customer, we were successful in guiding the third parties (e.g. the subcontractors that were our responsibility)?	4	4
	Average (Questions 1-17):	3,26	3,88
	Average for customer input (questions 18-21):	3,33	3,00
	Overall average:	3,27	3,70

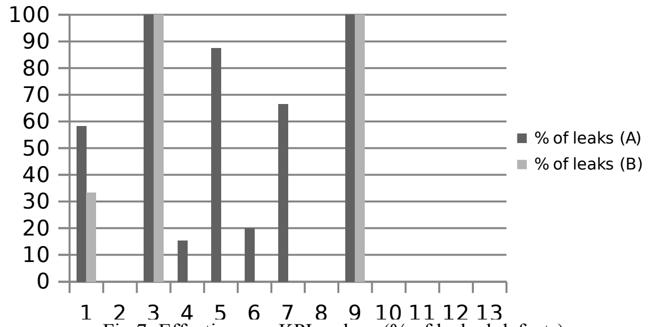

Fig 7: Effectiveness KPI values (% of leaked defects)

We found that internal service testing was more effective in project B than A. This means that the external testing service found more defects in project A than B.

E. Survey Results

After every six months, the customer satisfaction was monitored by internal Customer Satisfaction Survey (CSS).

During the interview the customer evaluates the level of satisfaction in three different areas (questions and areas are presented in Table I) by assigning the score from 1 (i.e. very unsatisfied) to 4 (i.e. very satisfied). The customer was able to comment on all questions, by providing additional feedback.

The scores and comments collected during CSS sessions are a kind of justification for what we have noticed during measuring KPI's. The customer feedback is always welcome and has high importance for further improvements of entire process.

The average results from both CSS's were quite impressive and show that both teams and projects were managed very well.

We note that all sections in CSS were scored higher for team B. However team A also collected high values.

F. Findings

To compare these two projects we used data collected regularly after each Sprint to measure KPI's (section *IV.D.*) and survey data collected after six months period (i.e. Customer Satisfaction Survey) to measure customer satisfaction.

We noticed that switching to "task-feasibility" from "time-estimation" (section *IV.C.*) and taking into account the possibility of delays caused by S (section *III.B.*) influence positively the amount of "*empty promises*".

This finding has been also noticed in high values of CSS in question 2, where the customer was more satisfied from schedule estimation accuracy in project B (e.g. the customer commented that: "*it was very helpful in planning scope of software application releases*".

The customer was disappointed due to bad effort estimations in project A, because that affected the scope (timeliness) of scheduled releases. In the project B the customer was satisfied and got positive feeling that schedule estimations were very good, although time-estimations were not applicable in this approach. The team limited their estimations only to committing that the proposed tasks will be finished and included in next release. Dissatisfaction of the customer was also visible in question 9 in the section "Project Personnel" (i.e. "*Project personnel accomplished their issues/task as promised?*"), where the team A collected lower score than B – probably due to bad estimations.

In both cases (i.e. both projects A and B) the most crucial were quality and timeliness. The key incentives for the client were high visibility (transparency) and an empirical project control that Scrum delivers. Based on the presented results we can conclude that in general the customer satisfaction was higher in the project B.

VI. Related Work

During the last 30 years, many approaches have been proposed to software application development. Starting from "Code and Fix" [1][23], to the waterfall model, spiral model, rapid prototyping, incremental model, extreme programming, scrum, etc. These can be classified as representatives of general life cycle models: heavyweight, middleweight, and lightweight. We cannot say that one model is better than another, because the approach suitable for one project may not be suitable for another. However we can assume, on the basis of the research results and experience [15][16][17][18][19], that the best choice for almost all kinds of software development projects, starting from scratch and executed in continuously changing environment, is an adaptive and flexible life cycle model (i.e. Agile) and a strongly prescriptive method (e.g. Scrum, eXtreme Programming, etc.) [15][17][18]. We agree with Scrum advocates that using time-boxed delivery cycles (i.e. Sprints), visualization of the project scope (i.e. Product Backlog), prescribed roles (e.g. Scrum Master, Product Owner, Scrum Team), essential meetings (e.g. Sprint Retrospective, Sprint Review, Daily Meetings), and following Scrum rules is necessary for project's success.

However, conventional Scrum is not sufficient when daily work results depend on external third party services (e.g. Translation, Testing, Technical Support, etc.) as it is the case in *"Network Environment"*. In contemporary landscape in which Network Organizations are more and more popular and IT projects are realized in inter-cultural environment, distributed services multiply the risks and uncertainties. That makes some Scrum artifacts useless, because of dependencies between third party services and service providers.

In literature, a few researchers have already studied the way to adapt the agile practices in Network Organizations.

We observe that usually Agile software development methods are introduced as a set of principles that need to clarify a lot of different interpretations of Agile Manifesto [24]. Because of that it is very difficult to say what exactly is an Agile methodology and how to adapt it to Network Organization environment, usually based on the Scrum as an example of Agile methods.

The originators of Scrum in software development are Hirotaka Takeuchi and Ikujiro Nonak. In [20] they defined a new approach to software development, called: rugby approach. They presented the whole process as performed by one cross-functional team across multiple phases, where the team tries to "*go the distance as a unit, passing the ball back and forth*" [20].

We consider that Scrum, as one of the representatives of Agile life cycle model, matches up perfectly to principles of "*Manifesto for Agile Software Development*" [17][18][24]. Following the Agile Manifesto principles is possible only because Scrum defines precisely the essential roles, principles and artifacts that make it very prescriptive method for managing software development teams.

We agree with Scrum advocates, but in addition we propose to add one new role (Third Patry Services providers) for adapting Scrum and Third Party Services to Network Organization.

Piero Mella studied the holonic perspective in organization and management. In [3] he examined six different examples of holonic networks in terms of manufacturing systems. This paper was an inspiration to consider the Network Organization as Holonic Network, seen as *"comprised of autonomous firms that are variously located—characterized by different roles and different operations and connected through an holonic network, real or virtual, often oriented, in order to achieve a common objective through the sharing of resources, information, and necessary competencies"* [3].

Dirk S. Hovorka and Kai R. Larsen in [25] present the study that examined the influence of network organization environment on the ability to develop agile adoption practices. They use exploratory case study design to "*investigate the interactions between network structure, social information processing, organizational similarity (homophily), and absorptive capacity during the adoption of a large-scale IT system in two network organization environments*" [25]. They propose Agile Adoption Practice Model (i.e. APM) that proposes interactions within the inter-organizational network that enable Agile adoption practices. We adopted a more detailed approach, and instead of treating Agile life cycle model as a set of good practices, we proposed a more detailed analysis of one selected method of software development (i.e. Scrum) and propose Scrum-based model that suits better Network Organizations.

Wojciech Cellary and Willy Picard presented in [26] the way to achieve agility and pro-activity by introducing the model of Collaborative Network Organizations in its two forms: Virtual Organizations (VO) and Virtual Organization Breeding Environments (VOBE). They presented idea of public administration "*playing a role of Virtual Organization customer on the one hand, and a Virtual Organization member on the other hand*" [26]. This publication was a stimulus for reflection about third party service providers (i.e. S role) as an entity that might be simultaneously a part and a hole in Network Organizations.

VII. Conclusion

In this paper, we proposed a holonic view based on which we adapted Scrum for 3rd part services and the Network Organization.

We developed a new service layer in the holonic structure and recommended new Scrum principles. In an industrial case study, we demonstrated the advantages of our model and method.

We believe that big differences in the level of satisfaction of the customer using our approach were caused by very prescriptive way of working with the pure Scrum.

There is no easy way to adapt Scrum software development method to work in Network Organization; however, we believe that presented results will serve to advance research and help in finding the best solution.

References

[1] V. Guntamukkala, J. H. Wen, M. J. Tarn, "An empirical study of selecting software development life cycle models", *Human Systems Management*, vol. 25, no. 4/2006, pp. 268-278, November 2006.

[2] L. A. Maciaszek, Modeling and Engineering Adaptive Complex Systems, *Challenges in Conceptual Modeling*, in *Proc. Tutorials, Posters, Panels and Industrial Contributions to the 26th International Conference on Conceptual Modeling - ER 2007, CRPIT*, no. 83, Auckland, New Zealand, ed. J. Grundy, S. Hartmann, L. Laender, L Maciaszek, J. Roddick, ACS, November 2007, pp. 31-38

[3] P. Mella, "The Holonic Perspective in Management and Manufacturing", *International Management Review*, vol. 5, no. 1, pp. 19-30, 2009.

[4] L. D. Sailer, "Structural Equivalence: Meaning and Definition, Computation and Application.", *Social Networks*, vol. 1, no. 1, pp. 73-90, 1978.

[5] M. Van Alstyne, "The State of Network Organization: A Survey In Three Frameworks", *Organizational Computing*, vol. 7, no. 3, pp. 88-151, 1997.

[6] M. James, D. R. White, "Structural Cohesion and Embeddedness: A Hierarchical Concept of Social Groups.", *American Sociological Review*, vol. 68, no. 1, pp. 103-127, 2003.

[7] G. A. Rummler, A. P. Brache, *Improving Performance - How to Manage the White Space in the Organization Chart*, California, 350 Sansome Street, Jossey Bass Inc., 1995.

[8] P. Deemer, G. Benefield, The Scrum Primer: An Introduction to Agile Project Managmenet with Scrum., GoodAgile, Version 1.04., 2007.

[9] H. Sharp, A. Finkelstein, G. Galal, "Stakeholder identification in the requirements engineering process", in *Proc. Database and Expert Systems Applications, 1999. Proceedings.*, Florence, Italy, IEEE Computer Society Press, September 1999, pp. 387-391.

[10] F. Capra, The Turning Point. Science Society, and the Rising Culture, New York, USA, Flamingo, 1982, pp. 27.

[11] A. Koestler ,*The Ghost in the Machine*, London, England, Penguin Group, 1967.

[12] K. Wilber ,*A brief history of everything*, Boston, Massachusetts 02115, Shambhala Publications, 2000.

[13] L. A. Maciaszek, "An Investigation of Software Holons - The 'adHOCS' Approach", *Argumenta Oeconomica*, vol. 1-2, no. 19, pp. 1-40, 2007.

[14] L. A. Maciaszek, Architecture-Centric Software Quality Management, *Web Information Systems and Technologies, WEBIST 2008, LNBIP 18*, ed. J. Cordeiro, S. Hammoudi, J. Filipe, Springer-Verflag, Berlin Heidelberg, 2009, pp. 11-26.

[15] A. Cockburn, "Selecting a project's methodology", *IEEE Software*, vol. 4, no. 17, pp. 64-71, July - August 2000.

[16] J. Nandhakumar, J. Avision, "The fiction of methodological development: A field study of information system development", *Information Technology and people*, vol. 2, no. 12, pp. 176-191, 1999.

[17] K. Schwaber, M. Beedle ,*Agile Software Development with Scrum*, Upper Saddle River, New Jersey, USA, Prentice Hall, 2002.

[18] K. Schwaber ,*Agile Project Management with Scrum*, Redmond, Washington, Microsoft Press, 2004.

[19] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. A. Williams, M. V. Zelkowitz, "Empirical Findings in Agile Methods", in *Proc. Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002*, London, England, Springer-Verlag, 2002, pp. 81-92.

[20] T. Hirotaka, N. Ikujiro, The New New Product Development Game, Harvard Business Review, vol. 64, January-February 1986.

[21] N. Zabkar, V. Mahnic, "Using COBIT indicators for measuring Scrum-based software development", *WSEAS Transactions on Computers*, vol. 7, no. 10, pp. 1605-1617, 10 2008.

[22] C. T. Fitz-Gibbon, Bera Dialogues: 2, Performance Indicators, Clevedon, England, Multilingual Matters, 1990, pp. 111.

[23] W. Royce, "Managing the development of large software systems", in *Proc. Proceedings of the 9th international conference on Software Engineering ICSE '87*, Los Angeles, IEEE Computer Society Press, August 1970, pp. 1-9.

[24] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, C. R. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, Manifesto for Agile Software Development, 2001.

[25] D. S. Hovorka, K. R. Larsen, "Enabling agile adoption practices through network organization", *European Journal of Information Systems - Including a special section on business agility and diffusion of information technology*, vol. 15, no. 2, pp. 159-168, April 2006.

[26] W. Cellary, W. Picard, "Agile and Pro-Active Public Administration as Collaborative Networked Organization", presented at the International Conference on Theory and Practice of Electronic Governance ICEGOV'10, ACM, NEW York, 2010, 978-1-4503-0058-2.