African Journal of Business Management Vol. 4(15), pp. 3277-3288, 4 November, 2010
Available online at http://www.academicjournals.org/AJBM
ISSN 1993-8233 ©2010 Academic Journals

Full Length Research Paper

A contribution to software service improvement based on LSP method

Vidan Marković[1] and Rado Maksimović[2]*

[1]Faculty of Sciences, Department for Computer Systems, University of Novi Sad, Novi Sad, Serbia.
[2]Faculty of Technical Sciences, Department for Industrial Engineering and Management, University of Novi Sad, Novi Sad, Serbia.

Accepted 27 September, 2010

With new technologies and various software solutions readily available for utilization in business environment, the main driver for achieving the competitive business advantage is becoming the quality of service. The quality of service expectations is very often managed by established service level agreement (SLA). Behind SLA, there is a real software, infrastructure, organization, culture, and people that maintain it. Therefore, it is very important to take in consideration the complexity of the software solution that is the basis of the given service, probability of an error occurrence, and all costs, and risks that will be associated with operating the service. In this paper, the important elements of the quality of the software that is considered as a service to a business function are analyzed. The proposal for the classification within specific portfolio of software services is given. The logical scoring of preferences (LSP) method is proposed to be utilized for services elementary preferences estimates and overall services comparison within each rank of services. The main objective is in achieving better quality of service within the given rank by recognizing and then emphasizing elements in the services that positively influence overall estimates of the observed services in comparison to the others in the rank. An example of the method utilization on one selected use case is given as well.

Key words: Software quality, software services, software development, improvement.

INTRODUCTION

Higher complexity of software solution brings higher probability of making errors. Making errors in each step of software development life cycle is absolutely expectable. On the other side, testing can never establish correctness of the software. It can only make comparison between the state and behavior of the product against set principles by which someone can recognize a problem. These expectations vary from similar product influences, expected purpose, relevant standards, to user "feelings" that software bugs cost the US economy $59.5 billion annually, and that more than a third of that cost is associated with bad testing performed (Hindi, 2002). That is why, it is so important to develop good test cases, if possible automate them, and run them as often as possible.

The future cost of implemented business software product depends also on the level of maintenance needed to utilize the product in its life cycle. Besides bugs fixing, there is also need to change existing and add new functionalities in the software. The rapid changes in the business are primary cause of the functional changes; however, there are also changes that are added at the later phase only because there was lack of good understanding or communications in the project preparation phase at the very beginning. Sometimes those omitted requirements if proved later to be of the key importance for the system in production could cause disastrous outcomes to the project itself.

This would also emphasize importance of the selected

*Corresponding author. E-mail: rado@uns.ac.rs. Tel: +381 21 4852152.

software process utilization during development phase and the methodology applied in project management.

The problem is in the fact that by the time that required changes get absorbed and defined within the business, the old software system (service) is still operating with the previous state that might not be desirable for the business any more.

To stay competitive, it is very important to constantly improve the software services quality and be able to give an answer to new needs faster (to be more agile).

BACKGROUND - THE QUALITY PERSPECTIVES

There are two main perspectives on quality. The first is an inner product quality that is related to operability of the product that is limited to the defect rate and reliability (normally represented with small “q”), and the over all quality (big “Q”) that is defined by product quality (q), process quality and customer satisfaction. Software quality at first level is linked to lack of “bugs”, and at the second to defect rate (e.g. number of defects per million or thousand lines of code (MLOC or KLOC), per functional unit, etc.), reliability (e.g. number of failures per N hours of operations, mean time to failure, probability of failure-free operations in a specific time), and customer satisfaction (normal or dissatisfied) (Kan, 2003).

More than 15% of software defects are related to requirements errors (Jones, 1992). It is also well known that the errors occurred and not detected in earlier phases of software development do contribute to higher cost in defect fixings later on.

Customer satisfaction is influenced by time to market criteria that have higher weight coefficient with dynamic businesses (for example, fast changes in products portfolios, new organization and processes due to mergers and acquisitions, economic crises, regulation requirements, etc). However, the software process could significantly influence the time to market criteria.

There are principles like the quality improvement paradigm (QIP) that defines software discipline as evolutionary and experimental (Basil, 1994; Basili and McGarry, 1998). This means that there is very little repetition in the software development, which makes the use of statistical control as used in manufacturing sciences extremely hard and dubious. The developers of QIP then take a different approach, for example, the authors of the CMM and a number of other models that are based on the very idea of statistical control of processes (Humphrey, 1989).

The QIP emphasize that all project environments and products are different – missile control software is entirely different thing from game software – and this means that there are certain prerequisites to experience reuse. These prerequisites include capturing and packaging of experiences, explanations on what kind of project and product types they haD applied successfully (and unsuccessfully) to, and how to tailor them to different environments and products (Basili, 1994). The development of reusable experiences is analogous to developing reusable code.

There are a number of paths that one can take to get to the point “B” from point “A”, but there is only one path that is going to be the optimal one (Ravindra, 1993). In the case of the software development processes, the optimal path selection depends on many things including the specifics of the business environment. If the environment does not change that often, it is more static than dynamic, and requirements are very stable, then procedural approach and more traditional processes can give sustainable results, otherwise some agile process (SCRUM, XP, ARUP, etc.) needs to be launched (Marković, 2005).

In the case where simultaneously a number of software services needs to be maintained with strict SLA obligations (Pollard, 2009), then it becomes very important to find a way on how to efficiently control and compare different on-going services and to approximately predict future costs per each new service in operations. By quickly setting defined parameters for new services, one would choose the most appropriate process and development environment with the available human resources (internal and external). Basically, meeting the quality needs for software services does include the principles written by Dr. Deming (Deming, 1982; Shewhart and Deming, 1986), and Total Quality Management (TQM) practice. TQM links customer satisfaction with the quality. TQM system is based on the following: Customer focus, process improvements, human side of quality and measurements and analysis (Juran, 2001). Researchers had used various theories and concepts from many disciplines to explain concepts related to process improvement frameworks. Regardless of the particular flavors of TQM implemented, process definition, control and improvement are always included since it is a core TQM principle (Hackman and Wageman, 1995). The main idea behind process control is that organizations are sets of interlinked processes and improvement of these processes is the foundation of performance improvement (Dean and Bowen, 1994).

The oldest model that can be seen as an improvement action life-cycle model is the PDCA (Plan-Do-Check-Act) model by Shewhart (1931). It was originally devised for improving quality in manufacturing and has its foundation in statistical quality control, that is, controlling the quality by applying metrics on the process. There are many variants of this basic model (for example, Bootstrap material (Kuvaja, 1994)). Some of the process improvements models based on PDCA are:

1. Effective change process model which is more or less an elaboration of the PDCA-cycle, and has been

described in the "Managing the Software Process" by Watts. S. Humphrey (Humphrey, 1989).
2. The AMI approach (Pulford et al., 1996) is essentially a model for implementing a goal-oriented measurement program, but since it is aimed at improvement, it can be considered as a model for process improvement life cycle as well. The AMI method implements the following activities: Assess your project environment (with its objectives and problems) to define primary goals for measurement. The goals are then checked against the assessment. Analyze the primary goals to derive sub-goals and the relevant metrics. The method provides a formal approach for the analysis, which includes a consistency check. “Metricate” by implementing a measurement plan and then process (including verification) the collected raw data into measurement data. Improve, as the participants affected by the goals start to use the measurement data and implement improvement actions.
3. The Pr^{2}imer model had been built from the process improvement consultancy experiences of VTT Electronics, and had been published in Karjalainen et al. (1996). The model draws on AMI-approach and TQM approach, integrating the software process analysis, modeling improvement and measurement techniques into the TQM-based process development.
4. Iteration cycle is the process improvement cycle, described in Culver-Lozo (1995), and has been developed and used at AT&T. The underlying principle of this model is that improvement starts from describing and modeling the process. Improvement activities are planned based on the information collected of how the model had been enacted. The model has three steps: Process definition, process execution, and process improvement.
5. Process improvement paradigm cycle is a process improvement cycle developed and used at Raytheon and has been described by Dion (1993). It is built on the principles from Deming and Juran – that is, that a real process improvement must follow a sequence of steps, starting with making the process visible, repeatable and then measurable. It also draws on Humphrey's effective change process. The model is a three-phase cycle of stabilization, control and change, where projects are the focus of activities.
6. The seven-step improvement process is in a core of continual service improvement (CSI) knowledge area of information technology infrastructure library (ITIL v3). That consist of the following seven steps: Definition of what should be measured, definition of what can be measured, data gathering (who, how, when, integrity), processing data (frequency, format, system, accuracy), analyzing data (relations, trends, according to plan, targets met, corrective actions), presenting and using information, implementation of corrective action) (Boyd, 2007).

TQM principles nicely match the latest version of ITIL v3. ITIL v3 is promoting the need of not just alignment (v2) with business, but integration of the IT services with the business (Van Bon, 2007).

Service management is a set of specialized organizational capabilities for providing value to customers in the form of service. IT service management (ITSM) provides that information systems support to business is offered under contract (SLA) and its performance is managed as a service. In that way, IT service management promises real benefit to the business customer and IT organizations. ITSM is an emerging area for further study (Cater-Steel et al., 2007). Providers of IT services can no longer afford to just focus on technology but should also consider the quality of the services they provide and the relationship with customers. Continual service improvement (CSI) is the part of the core v3 knowledge areas that is focused on PDCA (Van Bon, 2007).

By its nature, ITSM is process-focused, shares common ideas with the process improvement techniques. There are various frameworks developed to assist with the definition, assessment, reporting on and improvement of internal processes, IT, and control in organizations (for example, TQM, Six Sigma, Business Process Management, CobiT, ISO 9001, Balanced Scorecard, PMBOK, Prince 2, ISO 17799, and CMMI) (Ridley et al., 2004). ITIL offers a body of knowledge useful for achieving ISO/IEC 20000 standard requirements. There are materials available to assist auditors including one that maps COBIT to the ITIL framework (Grembergen, 2005).

There are materials written about a need of sharing of domestic and specific knowledge to improve quality of service and business operations (Harris, 2000; Choppin, 1995). There is an increasing importance of focusing on customer-oriented culture through organization structure, on effective communications, and feedback at all levels. Zsidisin et al. (2000) stated that accurate and timely communications are the cornerstone of service quality.

With ITIL implementation, IT governance includes leadership, organizational structures and processes to ensure that the organization’s IT sustains and extends the organization’s strategy (Sallé, 2004). IT governance is getting considered as an integral part of corporate governance, and this leads to ensuring customer satisfaction and picturing holistic nature of the business and IT services as an integral part of the business endeavourers.

MATERIALS AND METHODS

LSP in PDCA cycle

The quality of software could be estimated from different perspectives. The holistic picture of software that is used as a

service to a specific organization could be better obtained if the question of quality is also viewed trough comparison process with other software services for the given users' domains. The estimated relative quality of the service would help in understanding what the current "best practices" in organization are. That is the value that is perceived by users (clients) in relation to the same rank of the portfolio of services from the given services catalogue. Basically, the goal would be to maximize the business value that is driven from the invested efforts in the services development (Van Bon, 2007) and maintenance.

The study proposes the method based on LSP (Dujmović and Nagashima, 2006) that consists of the following steps in PDCA style repeating cycles. These PDCA cycles are mapped into repeating drill-in kaizen (Imai, 1997) cycles from comparison to decision actions that lead to overall IT services quality improvements. These steps are:

1. Identification of the group of the services that belong to the same portfolio category in the service catalog (the same service rank). To fulfill this step it is very important to understand the business and architecture side of the services. The catalogue is based on the business view of provided services, and the architecture complexity of the services, that in the same group could vary from mainframe to Web services.
2. Utilizing LSP method for comparison proposes. This method is based on hierarchical decomposition of criteria (top-down decomposition) till elementary criteria get reached at each level of decomposition. The study would not recommend for the software services comparison to have more than four levels of hierarchy identified for this case, because the dipper drilling down in identifying lower elementary criteria would lead to the "gold plating" mode (no significant differences calculated for comparison proposes). This is just the recommendation for one repeating cycle. However, this does not mean that further research at the lower level is not desirable. On contrary, if there is a business need to understand inner design, implementation, and detail process attributes that make a difference among services in the same service rank, that the previous comparison cycle has not been able to give, then we would recommend taking another cycle, and repeat them until the clear understanding from the result can be reached.

The formula to calculate the estimates of each defined criteria (Dujmović, 2006) is given as:

$$E = \left(\sum_{i=1}^{k} w_i e_i^r\right)^{1/r}, 0 \le w_i \le 1, \sum_{i=1}^{k} w_i = 1,$$
$$e_i \in [0,1], E \in [0,1], k \ge 2 \quad (1)$$

Where coefficient "w" represents weight coefficient associated with comparative importance of each estimated elementary preference that belongs to the same hierarchal group preference. The "r" represents the correlation function that is going to be applied on the specific level. The values of "r" are defined on the basis of the expectation from the combined influence on the estimated preference at the group level (for example, synergy effects). The values for r vary from full conjunction (C, r=-∞) to full disjunction (D, r=+∞). The arithmetic mean (AM) is given at r=1 (Dujmović and Bai, 2006).

3. Identification of the comparison criteria. Typical parameters that get evaluated in comparing the quality of software are: Capability, usability, performance, reliability, "installability", maintainability, documentation, and availability (Kan, 2003). However, as explained above, at each new cycle new set of the group criteria might be identified (for example, drilling in changeability a number of parameters can be indetified such as cohesion and coupling levels, design and implementation patterns, component's encapsulation level, code readability, standards utilization in design and implementation, etc.).
4. Calculating preferences for each service in the selected portfolio rank. For LSP method this means choosing the right coefficients (r,w) for each group level end make calculation till the main preference estimate get reached for the given cycle.
5. Analyzing the results and selecting the best services in the group. This means that certain upper control level (UCL) limit (Ott, 2005) defined for the services in the same service group needs to be "significantly" passed with the best performers (for example by 10% - if 10% means significant difference in the observed service group). The best and worst performers would be analyzed in further details to understand what would be the main contribution to the success, or failure respectively (for example, architecture, development process, people selection, project management methodology, etc.).
6. If it is possible to make clear conclusion and recommendation for service improvements, than conducting services improvements based on the knowledge acquired in the previous step, if not, then continuing with another drill-in cycle.

METHOD UTILIZATION EXAMPLE

The proposed method implementation is going to be demonstrated by following a case study of the selected company that would like to build better awareness of the quality of its IT services, and based on that awareness to improve the overall quality of all services in the services catalogue.

The catalogue of services that was taken from the selected company was firstly analyzed from the business value perspective added by each service from the catalogue. The logical grouping has been made to create different services categories (services ranks) bearing in mind business perspective of the catalogue. The company's management was especially interested in understanding the difference in quality of the core application software services built on different architecture basis, with different teams, and in some cases with partial procurement decisions made for the development tasks. That is why, only those services from the service catalogue that support core business that were built in house or with consulting help had been selected (no third party software, nor the systems software).

The services grouping as a first step of identification were done based on the identified service class's group attributes:

1. Technology group – represented by technical attributes that would better describe the influence of applied technology tools on service development and operations.
2. Complexity group – represents the observed level of complexity in creating solution. More tiers in the solution implementation would in most cases represent more complexity in operating that service.

3. Development process group – represents the possibility to lever the influence on the service by applied development process. Some development processes could create very stable service, but have a problem with low level of flexibility to change.
4. Development team group – team experiences, the skills, team cohesion, in house and outsourcing options do affect the ability for quality maintenance for specific service.
5. Business support domain group – is related to end user profile, the number, the location, and a type of application that is being used (for example, OLTP, reports, etc). In this case study we identified the following values domains for the above group attributes:

a. For technology dependent group attribute TD_i, the study identify two-tier, three-tier and four- tier client server architecture, Web platform on Open Source, Web platform on proprietary (Oracle) platform, and programming languages: Java, VB6, C++, and Oracle PL/SQL.
b. Complexity group attribute C_i, took high, medium and low values.

6. Different services were developed using different development process DP_i. These processes in this case were: Procedural SSA (Structured Systems Analysis), RUP, Agile (Scrum), Hybrid.
7. Development team group TD_i were different for different services. The services were developed and maintained internally (IH), externally (OH), and mixed teams (MX).
8. Business support domain group BD_i was described by values (Yes/No) for the following attributes front-end support, back-end support, internal user's domain, external user's domain, OLTP, reporting facilities.

Based on these group attributes definition, each instance of service class S_i from the catalog was assigned values as the following:

$$S_i = (TD_i, C_i, DP_i, DT_i, BD_i), \quad (2)$$

where:

$TD_i = (td_1, td_2, td_3)$, where $td_1 \in \{2T, 3T, 4T\}$, $td_2 \in \{WO, WP, DC\}$, and $td_3 \in \{J, VB, C, D\}$
$C_i = (c)$, where $c \in \{HI, MI, LO\}$
$DP_i = (dp)$, where $dp \in \{S, R, A, H\}$
$DT_i = (dt)$, where $dt \in \{IH, OH, MX\}$
$BD_i = (bd_1, bd_2, bd_3)$, where $bd_1 \in \{FE, BE\}$, $bd_2 \in \{OL, RE\}$, and $bd_3 \in \{IN, EX\}$

The domains' values are:

TD: 2T – Two-Tier, 3T – Three-Tier, 4T – Four-Tier, WO – Web - Open Source, WP – Web - Proprietary, DC – Desktop Client (Fat Client), J – Java, VB – Visual Basic, C – C++, D – DotNet;
C: HI – High, MI – Medium, LO – Low;
DP: S – SSA, R – RUP, A – Agility, H – Hybrid;
DT: IH – In-House, OH – Outhouse, MX – Mixed;
BD: FE – Front end, BE – Back end, OL – OLTP, RE – Reports, IN – Internal users, EX – External users.

This 'updated' services list is then rearranged by grouping similar services based on the complexity level and business support domain attributes. For the proposes of this case where management wanted to get better understanding on the quality obtained by implementing different processes, different teams (IH, OH, MX) and different architecture (Java, VB, etc.) the study decided to make grouping based on complexity and business domain service attributes and in particular FE/BE (Front End/Back End) and OLTP/RE (On Line Transaction Processing/Reporting). It means that grouping of the services was done based on the following:

$$S_i(C_i, BD_i(BD_1, BD_2)) = S_j(C_j, BD_j(BD_1, BD_2)) \quad (3)$$

The service class attributes value assignments for each service instance in the service catalogue and grouping into the S (H,FE,OL) class rank is shown in Table 2.
All the services in the same service group were then compared. The selected criteria for comparison are based on the hierarchical decomposition till elementary criteria had been reached. In this case, the study used only first and second level of the hierarchical decomposition (Table 1)
The study used the following values for weight coefficients at the first hierarchical level:

$w_1 = w_2 = 0.20$
$w_3 = w_4 = 0.15$
$w_5 = w_6 = 0.10$
$w_7 = w_8 = 0.05$

These values were defined based on the specific preferences requirement at given organization.

In defining the weight coefficient values, it would be advisable to have a team of internal and external consultants who would create more objective metrics for the specific use case scenario.

The maintainability and reliability group preferences are dominantly estimated by detail analysis of the ticket's type, number, and resolution patterns.

The study suggest that the proposed method could be used to create better awareness of the quality that is

Table 1. Hierarchical decomposition.

P_1= Maintainability	P_{11} = Changeability
	P_{12} = Stability
	P_{13} = Testability
P_2 = Documentation	
P_3 = Performance	P_{31} = Processing time
	P_{32} = Throughput
	P_{33} = Resource consumption
P_4= Reliability	P_{41} = Maturity
	P_{42} = Fault tolerance
	P_{43} = Recoverability
P_5 = Usability	P_{51} = Understandability
	P_{52} = “Learnability”
	P_{53} = Operability
P_6 = Capability	
P_7 = “Installability”	
P_8 = Availability	

Table 2. Identified attributes of service class instances grouped in the same rank.

Service code	Product ID	TD	C (H/M/L)	DP	DT	BD
NO00301	P05-19	2T, DC, VB	H	R	IH	FE, OL, IN
ZO00102	P06-06	3T, DC, VB	H	R	IH	FE, OL, IN
ZS00100	P05-21/P08-04-8	3T, DC, VB/3T, WO, J	H	A	MX	FE, OL, EX
OS00103	P05-08	3T, WO, J	H	M	MX	FE, OL, IN
NO00102	P05-24	3T, DC, VB	H	R	IH	FE, OL, IN
NO00101	P05-23	3T, DC, VB	H	R	IH	FE, OL, IN
ZS00102	P05-20	3T, DC, VB	H	R	IH	FE, OL, IN
NO00105	P05-26	3T, DC, VB	H	R	IH	FE, OL, IN
NO00100	P05-00	3T, DC, VB	H	R	IH	FE, OL, IN
OS00105	P05-06	2T, DC, VB	H	S	IH	FE, OL, IN
ZS00101	P06-08	3T, DC, VB	H	S	IH	FE, OL, IN
ZO00100	P05-100	3T, DC, VB	H	S	IH	FE, OL, IN
NO00503	P08-04-1	3T, WO, J	H	A	OH	FE, OL, IN
NO00800	P07-12	3T, WO, J	H	A	IH	FE, OL, IN
NO00106	P05-25	3T, DC, VB	H	R	IH	FE, OL, IN
NO00103	P05-35	3T, DC, VB	H	R	IH	FE, OL, IN

given with each service during the specified time period. However, the length of the period would not have the same meaning for different services placed for different users’ domains. For example, for the Web services, at the beginning, the service would be less known (Menascé, 1998) and small numbers of tickets would be initially triggered. In later phase with exponential growth of service users the situation could be rapidly changed. On the other side, internal service created for the limited and small number of the users that support their core operations would have a different ticket occurrence curve.

To get closer approximation of the quality of the service we would suggest defining a measurement period that is

Table 3. Number of ticket received per type of incident/request.

	2006				2007				2008				2009	
	I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II
ADD	1	0	2	1	3	3	1	0	2	2	1	3	1	0
MAINT	4	20	12	6	5	8	6	6	3	22	15	14	3	7
BCH	1	2	3	0	6	3	4	5	8	15	7	12	15	10
TOTAL	6	22	15	7	14	14	11	11	13	39	23	29	19	17

Table 4. Time to resolution per type of incident/request.

	2006				2007				2008				2009	
	I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II
ADD	24	0	38	128	80.75	112	40	0	64	65	47	547	235	0
MAINT	6	114.26	38	128	80.75	112	40	0	64	65	47	547	235	0
BCH	30	181	4.75	0	23.5	35	42.5	210	213.58	64	72.5	337	82	172
Total	60	295.26	160.75	142.92	499.75	219.5	152.5	273	282.58	1085.5	172	954.5	319.75	225

as long as possible (for example, one year) with all data collected during that time (tickets that had been received). In the example bellow (just for an illustration of the importance of ticket data collection), one service class instance is given that has been in operations for more than three years (Tables 3 to 6, Figures 1 and 2).

The finer granulation on MAINT type of incident/request in our case led to identification of the following lower level maintenance types: Defect (BUG), information request (INF), correction (CORR, not bug related action – small correction of the way how application work, normally lowest level of priority), parameters/rights adjustments (ADJ).Based on all these data collected, it is possible to estimate the quality of services in operations. The study suggest marks from 1 to 5, where 5 is the highest mark (Table 6). The marks were used in LSP calculations (mark 5 could give an estimate value for INF attribute of 1, mark 2 would give 0.2 value of estimate for CORR attribute in maintenance group attribute, etc.), until the final estimate had been reached for each instance of the service in the same service rank (in this case it is obvious that this service has a serious quality issue, but good documentation and training for end users (INF high mark indicator).

AS shown in Table 7, the final estimates and calculations are given for the specific portfolio class of services defined in the service catalogue (service rank S(H,FE,OL)

The classification of the results of the service's preference evaluation could be then further analyzed by grouping them in a few predefined cate-gories for each services rank. In the use case, the study used three different categories, from category marked with C (the lowest) to A (the highest). The marks assignments is based on the prior calculations of the average rank performance preference (S_i (AVG) = 0.855) and "adjusted" utilization of UCL (upper control limit) and LCL (lower control limit) for that rank (in the case, the study defined +/- 5% control limits around the average, UCL = 0.898 and LCL = 0.812). However, by definition (Ott 2005) the calculation of three standard deviations (3σ) would give UCL = 1.033 and LCL = 0.677, and that would mean that all data points in this case are within quality control limits. That is the reason why the study "adjusted" the UCL and LCL values in order to make improvement in services (bring the average service estimate value up). All the service above and under the UCL are the primary candidates for analysis for the overall rank improvements actions.

To better illustrate results for the service rank S

Table 5. Service instance detail maintenance data.

Incident type		2006				2007				2008				
		I	II	III	IV	I	II	III	IV	I	II	III	IV	SUM
BUG	#	1	12	1	2	1	4	0	2	1	13	3	3	43
	Hours	0.5	83.67	4	13	0.5	55	0	57	5	132.5	22	23.5	396.67
INF	#	1	2	1	2	0	1	1	3	0	3	4	2	20
	Hours	0.5	0.75	0.5	0.75	0	0.5	1	4	0	3	3	2	16
CORR	#	2	6	4	1	2	3	5	1	0	4	7	7	42
	Hours	5	29.84	5.5	0.67	83	17	69	2	0	23	26	9	270.01
ADJ	#	0	0	4	0	2	0	0	0	0	2	1	2	11
	Hours	0	0	108	0	312	0	0	0	0	798	1.5	36	1255.5
Total MAINT	#	3	20	10	5	5	8	6	6	1	22	15	14	116
	Hours	6	114.26	118	14.42	395.5	72.5	70	63	5	956.5	52.5	70.5	1938.18

Table 6. Service class instance estimate measurements based on the time to resolution.

Request type	BUG	INF	CORR	ADJ	MAINT
Ticket #	43	20	42	11	116
Time spent (t)	396.67	16	270.01	1255.5	1938.18
Rank average ($\overline{r}_i$)	$\overline{r}_{i,b}$ = 5.68	$\overline{r}_{i,i}$ = 1.25	$\overline{r}_{i,c}$ = 3.05	$\overline{r}_{i,a}$ = 17.9	$\overline{r}_i$ = 5.81
Service average ($\overline{S}_t$)	9.22	0.8	6.43	114.14	16.71
M = $\overline{S}_t : \overline{r}_i$	1.62	0.64	2.11	6.38	2.88
Mark	3	5	2	1	1

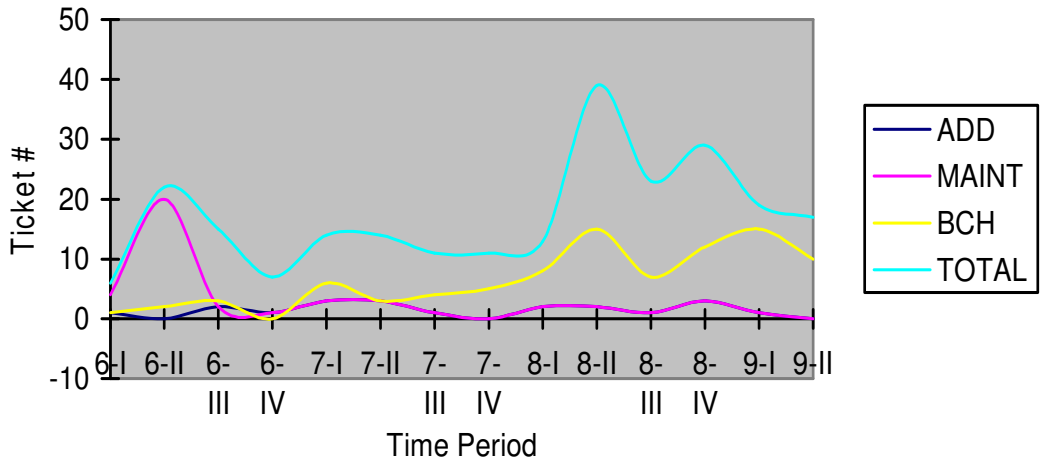

Figure 1. Graphical presentation of the number of ticket received per incident/request type where ADD is an additional functionalities request, MAINT is a maintenance request (for example, bug fixing), and BCH is a request for additional batch process invocation (out of scheduled Bach job utilizations for the given instance of service class).

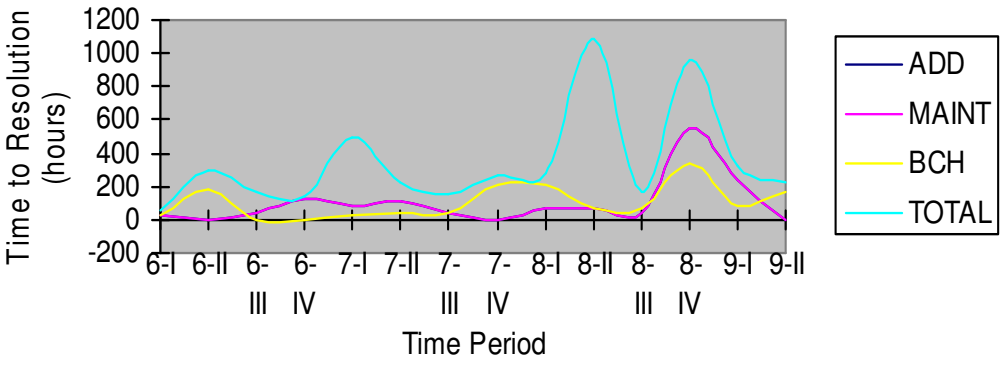

Figure 2. Graphical illustration of time spent on tickets resolution.

Table 7. Final estimates calculations for one service rank based on LSP.

Service code	Product ID	TD	C (H/M/L)	DP	DT	BD	LSP Estimate
NO00301	P05-19	2T, DC, VB	H	R	IH	FE, OL, IN	0.746
ZO00102	P06-06	3T, DC, VB	H	R	IH	FE, OL, IN	0.749
ZS00100	P05-21	3T, DC, VB	H	R	IN	FE, OL, EX	0.791
OS00103	P05-08	3T, WO, J	H	M	MX	FE, OL, IN	0.800
NO00102	P05-24	3T, DC, VB	H	R	IH	FE, OL, IN	0.802
NO00101	P05-23	3T, DC, VB	H	R	IH	FE, OL, IN	0.842
ZS00102	P05-20	3T, DC, VB	H	R	IH	FE, OL, IN	0.858
NO00105	P05-26	3T, DC, VB	H	R	IH	FE, OL, IN	0.864
NO00100	P05-00	3T, DC, VB	H	R	IH	FE, OL, IN	0.875
OS00105	P05-06	2T, DC, VB	H	S	IH	FE, OL, IN	0.885
ZS00101	P06-08	3T, DC, VB	H	S	IH	FE, OL, IN	0.886
ZO00100	P05-100	3T, DC, VB	H	S	IH	FE, OL, IN	0.898
NO00503	P08-04-1	3T, WO, J	H	A	OH	FE, OL, IN	0.902
NO00800	P07-12	3T, WO, J	H	A	IH	FE, OL, IN	0.915
NO00106	P05-25	3T, DC, VB	H	R	IH	FE, OL, IN	0.925
NO00103	P05-35	3T, DC, VB	H	R	IH	FE, OL, IN	0.945

(H, FE, OL), the scatter graph could be used (Figure 3).

Table 8 shows final listing of the instances of the service class belonging to S (H, FE, OL) rank.

After selecting top performers, the common features of the best marked services would need to be analyzed to see whether some pattern exist that is responsible for the excellence in the group. The interest also could be in further analysis of the different TD and DP applied. For example, the service instance from A group with Java and Agile process development (e.g. NO0800), and the services instance from B group VB and RUP (for example, NO0105) could be in further details analyzed for the incidents that occurred in the relatively same period of time. The length of time (the measurement's period) for incidents analysis was taken by the "younger" one in operations and mapped to the measurement time period of the "older" one (Figure 4).

The analysis could be respectively conducted for all representatives of the best and the worst in the service rank to look for the patterns for improvements in all services based on internal best practice and lessons learned (PMI, 2008). This analysis when regularly conducted would trigger improvements actions which would positively influence future decisions in new services development (for example, architecture to apply based on the business domain and level of the complexity, internal development, or outsourcing, or both, agile, or procedural development in relation to other dimensions, etc.). If all curves for the same business portfolio get analyzed in parallel (as the portfolio "summary" curves), this could help

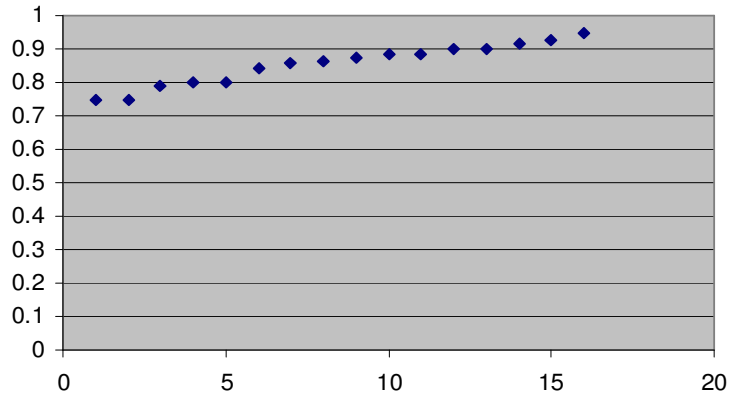

Figure 3. The estimated preferences of the instances of service class within the rank.

Table 8. The final classification on the service quality levels within the service rank.

Service code	Product ID	TD	C (H/M/L)	DP	DT	BD	LSP estimate	Final mark
NO00301	P05-19	2T, DC, VB	H	R	IH	FE, OL, IN	0.746	C
ZO00102	P06-06	3T, DC, VB	H	R	IH	FE, OL, IN	0.749	C
ZS00100	P05-21	3T, DC, VB	H	R	IN	FE, OL, EX	0.791	C
OS00103	P05-08	3T, WO, J	H	M	MX	FE, OL, IN	0.800	C
NO00102	P05-24	3T, DC, VB	H	R	IH	FE, OL, IN	0.802	C
NO00101	P05-23	3T, DC, VB	H	R	IH	FE, OL, IN	0.842	B
ZS00102	P05-20	3T, DC, VB	H	R	IH	FE, OL, IN	0.858	B
NO00105	P05-26	3T, DC, VB	H	R	IH	FE, OL, IN	0.864	B
NO00100	P05-00	3T, DC, VB	H	R	IH	FE, OL, IN	0.875	B
OS00105	P05-06	2T, DC, VB	H	S	IH	FE, OL, IN	0.885	B
ZS00101	P06-08	3T, DC, VB	H	S	IH	FE, OL, IN	0.886	B
ZO00100	P05-100	3T, DC, VB	H	S	IH	FE, OL, IN	0.898	B
NO00503	P08-04-1	3T, WO, J	H	A	OH	FE, OL, IN	0.902	A
NO00800	P07-12	3T, WO, J	H	A	IH	FE, OL, IN	0.915	A
NO00106	P05-25	3T, DC, VB	H	R	IH	FE, OL, IN	0.925	A
NO00103	P05-35	3T, DC, VB	H	R	IH	FE, OL, IN	0.945	A

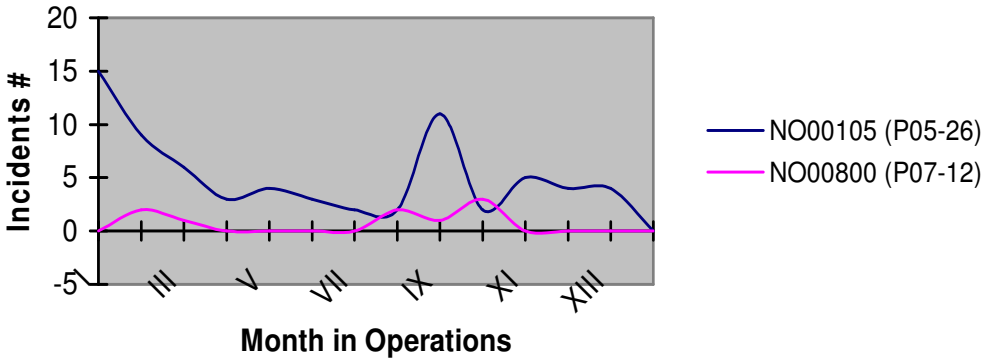

Figure 4. The comparative view on reported incidents in two services from the same rank marked as A (N000800) and B (N000105) for overall quality estimate.

help in better estimation and planning for HR consumption on maintenance tasks for the given portfolio in the future. If there is significant issue with projects' schedule performance indexes (SPI) (Wysocki, 2006) on the given portfolio, the portfolio "summary" curve could help in understanding what the root causes of these issues are.

Organizations that had introduced project or program management offices (PMO) (Rad, 2002) and that need to improve master plan execution could benefit from this analysis as well. The comparison of the quality and efficiency of the software development teams that work together on achieving the master plan targets would help better utilize resource allocations on different projects and tasks. This is especially of interest when the optimum level of outsourcing in development, needs to be mixed with internal development work. The motivations for outsourcing are evolving from a primary focus on cost reduction to an emerging emphasis on improving business performance (McFarlan, 1995; Venkatraman 1997; Nevo, 2007).

The cycled nature in the process of continuous improvements in services would help increasing capability and maturity levels (Chrissis, 2004) of organizations by utilizing this method as an optimizing process that is quantitatively managed.

Conclusions

The study had shown that LSP method could be used in supporting continual quality improvements (CSI) in IT services. CSI is one of the core ITIL v3 knowledge areas that focus on PDCA processes. In the paper, the quality management PDCA cycles has been triggered by results of analysis in LSP utilization on the selected elements of the services in the given group. The cycled drilling-in comparison for related services would be performed until the objective gets met. The related services are those services that belong to the same group (rank).

The study stressed out that the operational behavior of the services in time through type and number of tickets received together with a time needed for their resolution, needs to be continuously analyzed for any signals in trends. The comparison within the same service rank (for example, business domain group at the same complexity level) could give better – holistic view on the particular service of interest.

The method implementation process starts with definition of the service class (S_i). In the paper, S_i artifact is defined and proposal for the service class attributes is given as well. The first step relays on the appropriate grouping of the services in the service catalog. This grouping is based on the objective (the goal) that needs to be accomplished with the comparison procedure.

The second, the third and the fourth steps are performed based on LSP method for comparison at the same hierarchical preferences group level. The proposal for the hierarchical grouping and comparison criteria definition at each hierarchical level for the IT software services is given as well.

The results of estimates for each services in the service class is analyzed in the fifth step, where the average, under, and above the average performance is categorized for further use.

The further analysis is conducted in the last step after which the final result with recommendation for the improvements in the services would be given or/and the next more detail cycle would take a place until the recommendation "instance" for the improvement can be reached.

Through the case study, the method is explained. Because of particular business need in values were grouped, this case study, the services is based on the complexity and business domain attributes'

The detail description for the maintainability and reliability criteria for service operations is explained though illustrations on the received tickets and their resolutions in the specified period of time. These data was used in detail calculation of the preferences, to obtain more precise estimate for the maintainability and reliability criteria.

Further research could be focused on studying requests' curves of one rank of services. If the curve get sudden jump with introduction of the new service, that could be signal of interdependences in the services and might trigger further architectural dependences of these services in the future. The LSP method utilization in services improvement would positively influence IT capability and maturity of the organization.

REFERENCES

Basili V, McGarry F (1998). The Experience Factory: How to Build and Run One, Tutorial TF01, 20th International Conference on Software Engineering (ICSE'98), Kyoto, Japan.

Boyd R (2007). Understanding ITIL key process relationships. Computer Economics. pp. 1-3.

Cater-Steel AP, McBride N (2007). IT Service Management Improvement – an Actor Network Perspective, European Conference on Information Systems, St Gallen, Switzerland.

Choppin J (1995). Total quality management – what isn't it?, Manag. Serv. Qual., 5(1): 47-49.

Culver-Lozo K (1995). Software Process Iteration on Large Projects: Challenges, Strategies and Experiences. Software Process Improv. Pract., 1: 35-45.

Dean JW, Bowen DE (1994). Management theory and total quality: improving research and practice through theory development, Acad. Manage. Rev., 19(3): 392-418.

Deming E (1986). Out of the crisis: quality, productivity and competitive position, Cambridge University Press, Cambridge.

Dion R (1993). Process Improvement and the Corporate Balance Sheet. IEEE Software, July, pp. 28-35.

Dujmović JJ, Bai H (2006). Evaluation and Comparison of Search Engines Using the LSP Method. *ComSIS*, 3(2): 31-56.

Dujmović JJ, Nagashima H (2006). LSP method and its use for evaluation of Java IDEs, Int. J. Approx. Reason., 41(1): 3-22.
Grembergen VW, Haes DS, Moons J (2005). Linking Business Goals to IT Goals and COBIT Processes. Info. Syst. Control J., 4: 18-21.
Hackman JR, Wageman R (1995). Total quality management: Empirical, conceptual, and practical issues. Admin. Sci. Q., 40(2): 309-342.
Harris M, Harrington J (2000). Service quality in the knowledge age – huge opportunities for the twenty-first century. Measuring Bus. Excellence, 4(4): 31-36.
Hindi S (2002). The Blue Screen of Death and Other Deadly Bugs. Comput. Secur., 21(6): 491-496.
Humphrey W (1989). Managing the Software Process, Addison-Wesley, Massachusetts.
Imai M (1997). Gemba Kaizen: A Common Sense, Low-cost Approach to Management McGraw-Hill.
Jones C (1992). Critical Problems in Software Measurement, Burlington, Mass.: Software Productivity Research. pp 72-77.
Juran J (2001). Total Quality Management. McGraw-Hill.
Kan SE (2003). Metrics and Models in Software Quality Engineering. Addison-Wesley, Second Edition.
Karjalainen J, Mäkäräinen M, Komi-Sirviö S, Seppänen V (1996). Practical process improvement for embedded real-time software. Qual. Eng., 8(4): 565-573.
Kuvaja P, Similä J, Krzanik L, Bicego A, Saukkonen S, Koch G (1994). Software Process Assessment and Improvement - The Bootstrap Approach, Blackwell Publishers, Oxford.
Marković V (2005). Building IT Maturity in Fast Chaning Business Environment, First Edition, DP Budućnost.
McFarlan W, Nolan R (1995). How to Manage an IT Outsourcing Strategic Alliance, winter. Sloan Manage. Rev., 36: 9-23.
Menascé DA, Almeida VAF (1998). Capacity Planning for Web Performance: metrics, models, and methods, Prentice Hall.
Nevo S, Wade MR, Cook WD (2007). An examination of the trade-off between internal and external IT capabilities, J. Strategic Info. Syst., 16: 5-23.
Ott ER, Schilling EG, Neubauer DV (2005). Process Quality Control: Troubleshooting and Interpretation of Data, 4th Edition, ASQ.
PMI (2008). Project Management Body of Knowledge (PMBOK Guide), 4th Edition, PMI.
Pollard C, Cater-Steel A (2009). Justifications, Strategies, and Critical Success Factors in Successful ITIL Implementations in U.S. and Australian Companies: An Exploratory Study, Info. Systems Manage., 26(2):164-175.
Pulford K, Kuntzmann-Combelles A, Shirlaw S (1996). *A* quantitative approach to Software Management: The AMI Handbook, Addison-Wesley, Wokingham, England.
Rad PF, Levin G (2002). The Advanced Program Management Office: A Comprehensive Look at Function and Implementation, CRC press.
Ravindra KA, Magnanti TL, Orlin JB (1993). Network Flows: Theory, Algorithms and Applications. Prentice Hall.
Ridley G, Young J, Carroll P (2004). COBIT and Its Utilization: A Framework from the Literature, 37th Annual Hawaii International Conference on System Sciences (HICSS'04).
Sallé M (2004). IT Service Management and IT Governance: review, comparative analysis and their impact on utility computing, Hewlett-Packard Company.
Schwaber K, Beedle M (2002). Agile Software Development with SCRUM, First Edition, Prentice Hall.
Shewhart WA, Deming EW (1986). Statistical Method from the Viewpoint of Quality Control, Dover Publications.
Van Bon J (2007). Foundation of IT Services Management Based on ITIL v3, itSMF International.
Venkatraman N (1997). Beyond outsourcing: managing IT resources as a value center. Sloan Manage. Rev., 38(3): 51-64.
Wysocki RK (2006). Effective Project Management, 4th Edition, Wiley.
Zsidisin GA, Jun M, Adams LL (2000). Relationship between information technology and service quality in the dual - the direction supply chain: a case study approach", Int. J. Serv. Ind. Manage., 11(4): 312-328.