Method for personal capability assessment in an agile teams using personal points

Stipe Čelar[1], *Member, IEEE,* Mili Turić[2] and Linda Vicković[3]

***Abstract* — The problem of effort estimates in software development projects is important issue in small and medium-sized development companies and agile teams. In this paper, we propose a method for personal capability assessment of each individual team member. In assessing the ability of the project team members 18 parameters were defined with appropriate criteria, values and weight factors for assessing personal capability. For individual personal capability a metrics called personal point was introduced. The method was validated in the small agile software development start-up company Venio Indicium Ltd.**

***Keywords* — Personal capability, agile development, personal point, small and medium-sized development company, software estimation, effort estimation.**

I. Introduction

Today's organizations in all areas depend more and more on software. Without software they can not compete, adapt, produce, distribute or sale their products and services. For example, one car manufacturer predicted that by 2010 its cars will have up to 100 million lines of code [1]. Therefore the software projects also become an extremely valuable and important. However, relevant studies have shown that constant cost and effort overruns in software projects is approximately 30 percent [2].

Some of the most important prerequisites for achieving project success are precise ***software size*** and ***project effort*** estimation [3]. In theory and in practice there are many methods of predicting the software (SW) size and project efforts. Besides these two mutually dependent variables there are a number of other parameters/variables that need to be managed throughout the project in order to achieve the desired goal. One of them deserves special attention of small (agile) development teams and small/medium SW companies – ***productivity of developers***. Even with a 100% accurate prediction of the size and effort the project success is not assured if the productivity can not be managed. Productivity can vary by a factor of 5 to 10 from the smallest projects to the largest [4].

[1] Stipe Čelar (*corresponding author*) is with the FESB, University of Split, Ruđera Boškovića 32, Split, Croatia, (phone: +38521305843, e-mail: stipe.celar@fesb.hr).

[2] Mili Turić is with Venio Indicium d.o.o., Doverska 19, Split, Croatia, (phone: +385 21 674287, email: mili.turic@venio.hr).

[3] Linda Vicković is with the FESB, University of Split, Ruđera Boškovića 32, Split, Croatia, (phone: +38521305849, e-mail: linda.vickovic@fesb.hr).

In this paper a novel method for evaluating team members' *personal capability* is proposed. This method is used in Venio Indicium, a small SW development start-up company from Split (Croatia). The rest of the paper paper is organized as follows: Section II gives the overview of size/complexity and effort measurement and estimation methods. Section III introduces relation between team members' capability and project scheduling. Sections IV and V present the method for personal capability estimation in an agile teams while the Section VI gives the evaluation case. Section VII presents the conclusions remarks.

II. Software size and effort measurement and estimation – overview

A. Software size and complexity measures

Software complexity encompasses many characteristics of a software that affect the internal interactions in the software. It describes the complexity of the interactions between a number of software entities. The oldest size/complexity measure is *line of code* (LOC) and can be known only after the project is completed [5], [6].

Halstead is a complexity measure introduced by Maurice Howard Halstead 1977 as part of his thesis on the establishment of empirical science of software development [7], [8].

Function Point (FP) is wide accepted and partially standardised method for measuring software size and complexity. It is based on five defined data components (inputs, outputs, inquiries, files, external interfaces) and 14 weighted environment characteristics (data comm, reusability, performance etc.). The method is introduced by IBM's engineers Albrecht [5], [6], [7]. Today there are more than 20 FP variants [7].

B. Estimation of software development effort

The basis for estimation methods are the measurement methods, project historical data and experience from the filed. Three different categories of effort estimation best practices could be differentiated: model-based (COCOMO, Putnam SLIM), expert-based and methods that combine expert and model-based methods [2], [5], [7], [9].

General-purpose models such as COCOMO and SLIM need to be customised to specific company environment before they could be used effectively [10] [11], [12], [13]. Even with good software size and complexity measure (or estimate) an estimator have a problem to predict the productivity of individuals or teams performing the project tasks. Some tests show

that productivity can differ more than 20 times from person to person [7].

Estimator tend consistently to overestimate the time necessary to complete shorter tasks, but consistently to underestimate longer tasks [14].

III. Team members capability and scheduling

Using size and effort estimation as an input a project manager can produce project schedule (Fig. 1). Once the project execution phase begins a project team should collect project data for the estimation in future projects. Of course, with more data about team mebers and their capabilities manager could schedule more wisely in the scheduling phase.

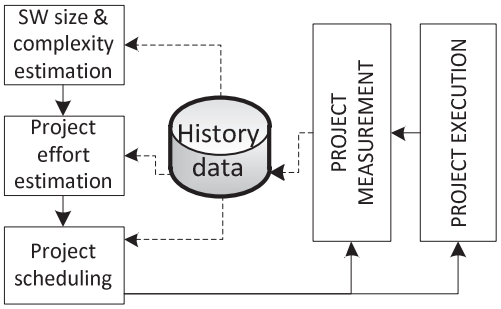

Fig. 1. Estimation and measurement approach.

Productivity generally indicates success in carrying out some work in relation to the resources used. Jones [7] says that there are over 200 factors that can affect the productivity of developers and software quality, and lists the primary factors that have the greatest impact, namely: the skill and experience of staff, the cooperation of users during requirements and design, schedule or resource constraints, methods employed on the project, tools available, appropriate choice of programming language(s), problem complexity, code complexity, data complexity, project organization structures, and the physical environment. Matos et al. [3] include 'team productivity' on the comprehensive list of 90 factors influencing effort estimation. Jørgensen [11] emphasizes importance of 'productivity of the individuals or teams completing the work'.

IV. Method for evaluating team members personal capability

A. Capability parameters

Impact of the team members productivity on software quality and project success is high, particularly in small companies. Observation of software development projects has led to the identification of skills that developer should have needs of developers and 18 parameters for assessing developer's capability (Table I).

The defined parameters are a set of important information about each individual, which greatly affect his/hers level of ability to perform project tasks. They largely reflect the ability of each individual and thus may facilitate the assessment of the time required for the achievement of project objectives.

B. Parameter values (v) and their criteria

For all 18 parameters mentioned earlier a measurement scale with five positive or negative values and their criteria are defined (Table I). The positive values are used for parameters that increase productivity, while negative ones are used for those that decrease it. Work experience (F1) can be precisely measured while other parameters need to be assessed.

C. Weight factors (f)

All parameters do not contribute equally to productivity. A scale with three levels of difficulty factors proved to be sufficient for the quantification capabilities of developers:

- f = 10 – very important parameter with the greatest impact,
- f = 8 – medium important parameter that has a pretty big impact, but not decisive,
- f = 5 – parameter that is not crucial but it would be good to have it.

Number of scale levels and numerical values of their weight factors (5, 8, 10) are based on the author's practical experience in three years capability and productivity monitoring in small agile teams [12], [13]. Level numbers and weight factors values should be adapted to specific needs of the company and its development environment.

V. Personal capability

A. Capability assessment and counting

Once when measures and factors for all parameters are defined, the ability of developers can be assessed in several steps:

1) assess the value (v) of certain parameters (P1 - P18),
2) calculate the personal points (PP) for each parameter Px:

$$Pn=v*f\ [PP], \quad (1)$$

3) sum the total value of the assessment:

$$PC=\sum (Pn)\ [PP]. \quad (2)$$

By using 18 described parameters, their numeric values and weight factors a personal capability (PC) for individual team member can be assessed and numericaly expressed in a value we called ***personal points*** **(PP)**. A higher value means a greater personal ability. Minimum and maximum values for someone's PC are 24 and 600 personal points.

B. Task complexity and personal capability

The resulting value can be used further to assess the task complexity for each individual. Namely, task complexity is not equaly complex for different individuals – it depends on their personal capabilities (PC). Suppose that the task has a complexity Y measured or estimated by any previously mentioned method. The equation (3) gives the complexity of the task (C) for each individual:

$$C = Y / PC \quad (3)$$

Greater value of C means that the task is more complex for the individual team member because PC

includes some factors (e.g. P2 and P5) that influence on task complexity also [3], [7].

If the history database is available (Fig 1.) it would be possible to find more patterns in the database relating to task complexity and individual's capability. In that case it would be easier to find more suitable tasks to the each team member.

TABLE 1: PARAMETERS, VALUES, CRITERIA AND WEIGTING FACTORS.

Px	*Parameter*	*Parameter description*	*Criteria and values (c)*	*Factors*
P1	Work experience	Number of years of experience in similar jobs.	1 – less than 1 year, 2 – 1-2 years, 3 – 2-5 years, 4 – 5-7 years, 5 – more than 7 years	8
P2	Knowledge of technology	Level of knowledge and handling of technology in which the software will be developed.	1 – very poor, 2 – poor, 3 – good 4 – very good 5 – excellent	10
P3	The level of learning and work habits	Adoption of new working procedures as part of the work being performed. It is reflected in self acceptance and performance of the procedures.	1 – very poor, 2 – poor, 3 – good 4 – very good 5 – excellent	10
P4	Ability to work autonomously	Extent to which tasks are performed in accordance with the general or specific guidelines and instructions of his superiors and the scope of control superiors required in performing a particular job.	1 – complete dependence, 2 – mild dependence, 3 – autonomy, 4 – high autonomy, 5 – complete independence	8
P5	Work complexity	Level of complexity of the tasks performed in the workplace and the complexity of the procedures that apply in their resolution, the required level of personal contribution of employees and the scope of the workplace	1 – trivial tasks, 2 – stereotypical affairs, 3 – slightly complex jobs, 4 – complex tasks, 5 – very complex tasks	8
P6	Attitude towards work	Extent to which the employee identifies with the project in which it participates, how relaxed or seriously certain situations, and how it relates to the daily work in terms of their engagement.	1 – very poor, 2 – poor, 3 – good, 4 – very good, 5 – excellent	5
P7	Concentration	Degree of person's presence in the work that is done. Manifested by jumping from one job (or other content) to another or the commitment to carrying out the task.	1 – very poor, 2 – poor, 3 – good, 4 – very good, 5 – excellent	5
P8	Skill of performing of work types	For each task defines the kind of work it belongs, and each person has evaluated the performance of skill for any type of work.	1 – very poor, 2 – poor, 3 – good, 4 – very good, 5 – excellent	10
P9	Knowledge of the project	Knowledge of the project plan and involves participation in the planning of the project from the very beginning.	1 – very poor, 2 – poor, 3 – good, 4 – very good, 5 – excellent	8
P10	Knowledge of the product	Knowledge of the entire system, not just the knowledge of the individual segments.	1 – very poor, 2 – poor, 3 – good, 4 – very good, 5 – excellent	10
P11	Responsebili-ty and influen-ce on decision making	Extent to which the tasks performed in the workplace have an impact on the implementation of the project objectives.	1 – very poor, 2 – poor, 3 – medium, 4 – high, 5 – very high	5
P12	Communi-cation	It reflects the type and frequency of contacts that are achieved when performing tasks, and their importance to the project.	1 – very poor, 2 – poor, 3 – medium, 4 – high, 5 – very high	5
P13	Agility	Agility reflects the degree of dexterity and diligence in the performance of individual tasks.	1 – very poor, 2 – poor, 3 – medium, 4 – high, 5 – very high	8
P14	Producing bugs	It manifests itself as a number of bugs produced during development but also in the time spent on the removal of bugs produced.	-5 – very high, -4 – high, -3 – medium, -2 – low, -1 – very low	8
P15	Knowledge of the area	Extent to which person needs consulting about business area.	1 – very poor, 2 – poor, 3 – good, 4 – very good, 5 – excellent	10
P16	Testing during development	Degree of effort required to test components developed by person, in order to reduce the number of bugs produced.	-2 – very little, -1 – bit 0 – necessarily, 1 – enough, 2 – optimally	8
P17	Experience in similar tasks	A measure that shows how person is experienced in performing similar tasks.	1 – very poor, 2 – poor, 3 – medium, 4 – high, 5 – very high	10
P18	Need for supervision	Extent to which person needs daily mentoring by the project manager and / or head of software product development.	-5 – very high, 4 – high, 3 – medium, -2 – low, -1 – very low	8

VI. CASE STUDY

A method is validated in the SW company Venio Indicium Ltd. (project: PIVIS) by measuring developers capabilities. This could be explained in the case of selecting one of three programmers (Dev 1-3) to create reports. Without usage of presented quantified method they would be characterized as follows:

Programmer 1 (Prog 1): *An engineer with 3 years working experience of with medium technology knowledge; he should improve his agility; he produces very little bugs; he usually does not work on reports.* **RATING: *Medium good programmer***.

Programmer 2 (Prog 2): *An engineer with working experience over 10 years who really knows the technology and very well explains the mathematical problems, but does not know the product to the extent that it should; he produces bugs often due to a lack of testing. A programmer usually does reports.* **RATING: *very good programmer***.

Programmer 3 (Prog 3): *An engineer without experience and product knowledge and with a basic technology knowledge; he is learning capable; he is very independent in task with clear instructions.* **RATING: *trainee programmer who acquires the necessary knowledge and skills***.

Now we make the measurement in accordance with defined parameters, values and factors (Table 2). Resulting personal points quantify the difference between individual programmers.

Measured results clearly quantify and present (expressed in personal points) above described developers' differences.

TABLE 2: PROGRAMMERS' CAPABILITIES.

Px	Prog 1	Prog 2	Prog 3
P1	2	5	1
P2	3	5	2
P3	4	3	4
P4	3	4	2
P5	3	5	2
P6	3	4	5
P7	3	5	5
P8	4	5	3
P9	3	3	2
P10	3	4	1
P11	2	2	1
P12	3	3	3
P13	2	3	4
P14	-2	-3	-2
P15	2	3	1
P16	1	0	1
P17	3	4	1
P18	-4	-4	-4
	309 PP	**414 PP**	**238 PP**

VII. CONCLUSION

This method for quantification of individual personal capabilities differentiate more precise individual developers and gives the planner/scheduler additional valuable information for the fine scheduling. Such information could be usefull especially in small/medium companies and agile teams because the impact of an individual for such teams is higher than for large companies or teams.

REFERENCES

[1] R.N. Charette, "Why Software Fails," *Spectrum*, IEEE Computer Society, vol. 42(9), 2005, pp. 42–49, doi: 10.1109/MSPEC.2005.1502528

[2] T. Halkjelsvik and M. Jørgensen, "From Origami to Software Development: A Review of Studies on Judgment-Based Predictions of Performance Time," *Psychological Bulletin*, vol. 138(2), 2012, pp. 238–271.

[3] O. Matos, T. Conte, E. Mendes, „Is there a Place for Qualitative Studies when Identifying Effort Predictors? A case in Web Effort Estimation," *Proc. of the 18th Int. Conf. EASE '14 40*, May 13 – 14 2014, London, England, ACM, Article No. 40

[4] S. McConnell, *"Software Estimation: Demystifying the Black Art,"* WA, USA: Microsoft Press, 2006.

[5] H. Leung, Z. Fan, *"Software cost estimation,"* Handbook of SW Engineering, Dept. of Computing The Hong Kong Polytechnic University, 2002

[6] S.H. Kan, *"Metrics and Models in Software Quality Engineering",* 2nd ed., Addison-Wesley Longman Publishing Co., Inc. Boston, USA, ISBN: 0201729156, 2002

[7] C. Jones, *"Applied Software Measurement: Global Analysis of Productivity and Quality,"* 3rd ed, McGraw-Hill, Inc. New York, USA, 2008

[8] M.H. Halstead, *"Elements of Software Science,"* Amsterdam: Elsevier North-Holland, Inc. ISBN: 0-444-00205-7, 1977

[9] T. Menzies, Z. Chen, J. Hihn, K. Lum, „Selecting Best Practices for Effort Estimation," *IEEE Transactions on Software Engineering*, vol. 32(11), 2006, pp. 883-895

[10] E. Mendes, M Kalinowski, D. Martins, F. Ferrucci, F, Sarro, "Cross- vs. within-company cost estimation studies revisited: an extended systematic review," *Proc. of the 18th Int. Conf. EASE 2014*: *12*, London, England, United Kingdom, May 13-14, 2014, ACM, Article No. 12

[11] M. Jørgensen, "What We Do and Don't Know about Software Development Effort Estimation," *IEEE Software*, vol 31(2), 2014, pp. 37-40

[12] S. Čelar, Ž. Šeremet, Ž. Marušić, M. Turić, "Using of Web Objects Method in Agile Web Software Projects," *Proceedings of 21st TELFOR*, Beograd, IEEE, 2013, pp. 873-876

[13] S. Čelar, V. Vicković, E. Mudnić, "Evolutionary Measurement-Estimation Method for Micro, Small and Medium-Sized Enterprises Based on Estimation Objects," *APEM*, vol 7(2), 2012, pp. 81-92

[14] L. Hatton, "How Accurately Do Engineers Predict Software Maintenance Tasks?" *IEEE Computer*, vol. 40(2), 2007, pp. 64-69