Real Time Agile Metrics for Measuring Team Performance

Eduard Nicolae BUDACU, Paul POCATILU
The Bucharest University of Economic Studies, Romania
eduard.budacu@csie.ase.ro, ppaul@ase.ro

In order to track the improvements of agile teams, a system of metrics and indicators is very important to be implemented. Agile Software Development (ASD) promotes working software as the primary way of measuring progress. The current set of metrics are more output oriented rather than using lines of code to estimate productivity. This paper presents the results of a background research in order to identify the most important metrics, indicators, measures and tools software development teams use in relation with agile-based methodologies. The paper also presents a case study based on data gathered in a software outsourcing company. The paper proposes an architecture of an automated system used to provide real-time metrics for measuring agile team performance.

Keywords: *Agile Development, Metrics, Indicators, Measurements, Velocity, Lead Time, Cycle Time*

1 Introduction

Software industry has adopted a wide range of Agile Software Development methods to improve productivity [13], [18]. Metrics and measures are required for planning and tracking projects, measuring quality and assessing team performance [10]. This paper aims to review the usage of metrics in industrial software development where agile practices are applied. It also proposes an architecture of a system that provides real-time metrics based on data provided by project and process management software. A case study is conducted in a Romanian software company providing custom software development services using an outsourcing model.

This paper is structured as follows. The first part defines the problem of using metrics in software development. In the second part a background research is performed to identify what kind of metrics and tools software development teams use in relation with agile and lean methodologies. The third part presents the case study, research method, results and presentation of the proposed system. The last part is reserved for discussion, conclusions and future research directions.

Use of metrics in software development has a great importance in both traditional and agile software development methods. The metrics are described in software development standards like [14], [15] and [16]. Software development metrics relay heavily on output oriented measures. Software is analyzed based on lines of code written (LOC), function or class complexity, like cyclomatic complexity [11], documentation coverage, etc. Project management is orientated to evaluate the accomplishment of plans based on what is done on time and within budget so the main way of measuring is tracing project plan completion. Agile promotes working software as the primary measure of progress, but the definition of “working” is vague and can lead to different interpretation based on roles and individual background [1]. In this study we consider the definition of working software as new functionalities that bring value to the business or satisfies a user request. Three key agile metrics are used to assess the ability of delivering working software.

2 Background research

In this section a background research related to the agile software development and software measurements is performed to introduce the key metrics used in the case study. Several articles and books are analyzed to identify what are the best practices in using agile metrics.

A systematic literature review identifies that reasons for using metrics are focused on plan-

DOI: 10.12948/issn14531305/22.4.2018.06

ning, progress tracking, understand and improve quality, fixing software process problems, and motivating people [2]. The metrics are also used in estimating the software project duration [12]. Metrics are categorized in each of the primary studies analyzed. It is shown that the same metric can be used in different contexts. For example, defect count as an external measure of customer satisfaction or as an internal measure of software testing. The study concludes that the most influential metrics in the primary studies are Velocity and Effort estimate in Scrum and Extreme Programming methods and Lead/Cycle time in Kanban method. Key agile metrics are defined in Table 1.

Table 1. Key agile metrics

Metric	Methods	Definition
Velocity	Scrum, XP	The amount of working software delivered in a sprint/iteration.
Lead time	Kanban	The amount of time that passed from a request to fulfilling the request.
Cycle time	Kanban	The amount of time that passed from when work actually started to fulfilling the request.

Graphically, the cycle time and lead-time are presented in Figure 1.

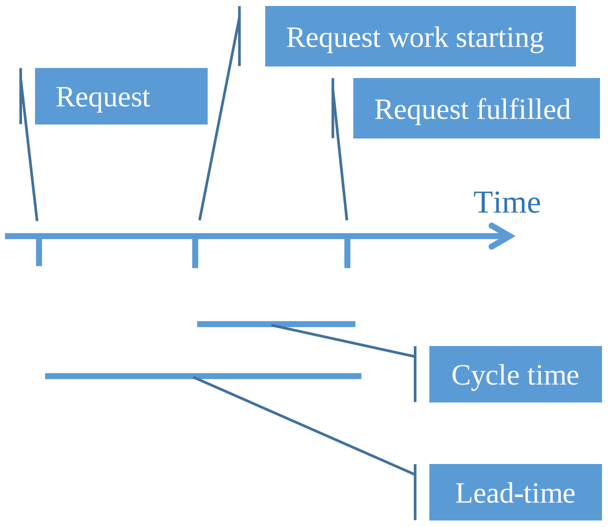

Fig. 1. Lead-time and cycle time

In [3] the author argues that metrics can be dangerous if it's only used a way to set unrealistic targets and manage the work by numbers (e.g. management establishes a target over a period of time that is communicated without a goal so people do everything just to fill a quota). This will eventually lead to unwanted behaviors that derail from the initial intent. In this regard is recommended to always link metrics to goals, use short tracking periods, favor tracking trends rather than precise numbers and stop using a metric when it no longer drives change. In previous research ([4]) the state of agile practices in Romanian software community was evaluated. According to this, Scrum and Kanban are the most used method but only 50% of respondents use Velocity tracking.

Scrum [5] and Extreme Programming (XP) [6], [19], [20], the most used agile methods, have an iterative and incremental approach to delivering software. Work is planned in a time box called sprint or iteration that spreads for two to four weeks. The team aims to go through all development phases in an iteration so that it delivers an improved working version of the product. Velocity is calculated for every iteration based on the count of user scenarios implemented, sum of complexity estimate points or effort estimate points.

Kanban method relies on workflow visualization and limiting work in progress to improve the continuous delivery of software while avoiding overburdening the development team. Rather than batching the work in sprints, items are pulled from the backlog in a continuous flow. Assuming that the items are prioritized by the added value in a que is important to measure the total wait time of an item and the processing time. In practice, Scrum and Kanban methods are used together in what is called an Agile/Lean methodology [7], [17]. Software tools used by agile teams allows accurate activity tracking [8]. This allows issue tracking, direct communication between team members and provides built in reporting. In order to make better use of data gathered most of the tools allow direct access to the databases for custom analysis. Better understanding of team dynamics based on the traces left in information systems are leading to area of research called workforce analytics [9].

3 Research method

Key agile metrics are calculated in relation with a case study in a software company that provides outsourced custom development. Project started in March 2017 when the company took over the development an existing codebase with the goal of going live for two customers. In September 2017 is established a role of Scrum Master with the responsibilities improve the overall collaboration between product management and development team, to facilitate Scrum events, identify blockers, communicate release notes, provide visibility on the process to all parties involved, and organize workshops.

This paper has the following research objectives:

O1: Describe how the team organized in relation with agile practices;

O2: Present the results for the key agile metrics;

O3: Discuss the factors that influence the ability to deliver working software;

O4: Describe a proposed architecture of a system that provides real-time agile metrics.

Observations regarded with team structure, organization and team dynamics are gathered by direct implication in the team for 6 months. In this period were facilitated all Scrum meetings and helped Product Managers to coordinate with the development team. Data used to calculate metrics was exported form JIRA Atlassian software and was processed using spreadsheets together with built in reports in JIRA.

4 Case Study

At the beginning of the engagement an assessment was performed that consisted in one on one interviews with all the team members. The main characteristics of the team and process are:

- Team members were distributed between United Kingdom, Romania and The Republic of Moldova;
- Average sprint length was two weeks (10 working days);
- Total team capacity: 3 x Product Management (Customer Project Manager, Product Owner, Proxy Product Owner), 4 x Quality Assurance (Manual testing, Automation testing), 1 x Technical Lead, 6 x Backend Developer, 4 x Frontend Developer, 1 x DEVOPS, 1 x Scrum Master.

Product Managers worked day by day with the software developers, were available in all the Scrum meetings and available on Skype for task clarifications. The Technical Lead acted as the main contact point for the development team allocating tasks based on his knowledge about individual abilities. Product Owners complained about the lack of visibility regarding the development process. Development team had no awareness about the product roadmap and immediate deadlines were not communicated.

A big pain in the process was not having some clear way of measuring progress. Sprint Goals were introduce as a way to create more focus, shift from task allocation to team commitment and set priorities. In order to create more visibility a product roadmap was created with estimates made by the development team. At the end of each sprint a demo meeting was performed by the development team to customers and product owners. Every three sprints we produced reports to track velocity and review the progress.

Progress is tracked by using a 5 steps workflow on an electronic Kanban board as described in Figure 2.

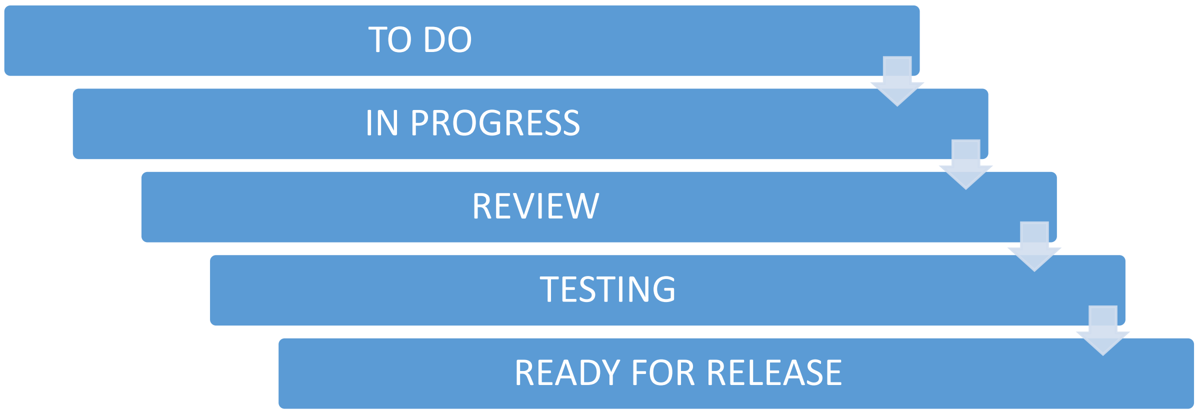

Fig. 2. Development workflow

The first go live was mid-October when the back office component was delivered. A support board was created after the release and customer support representative were allocated to answer user requests. Two swim lanes are used on the support board, one for high priority tickets that are blockers in the client activity and the second for regular tickets. Priority regarding support tickets was decided with the client and were included in the scope of the sprint.

The data set consisted in 1388 tickets distributed as following:

- 800 Bugs (58%);
- 429 User Stories (428);
- 159 Task (11%).

A total of 24 sprints were tracked with an average duration of 15.2 days, min 11 days and maximum 29 days.

The evolution of development capacity is presented in Table 2.

Table 2. Development team capacity

Sprint	Development capacity	Total team members
Sprint 1 – Sprint 2	1 x DEVOPS, 1 x BE, 1 x FE, 1 x QA	4
Sprint 3 – Sprint 4	1 x DEVOPS, 2 x BE, 1 x FE, 2 x QA	6
Sprint 5 – Sprint 7	1 x DEVOPS, 3 x BE, 2 x FE, 2 x QA	8
Sprint 8 – Sprint 10	1 x DEVOPS, 5 x BE, 3 x FE, 4 x QA	13
Sprint 11 – Sprint 24	1 x DEVOPS, 6 x BE, 4 x FE, 4 x QA	15

Velocity chart based on Stories / Bugs completed in a sprint is presented in Figure 3.

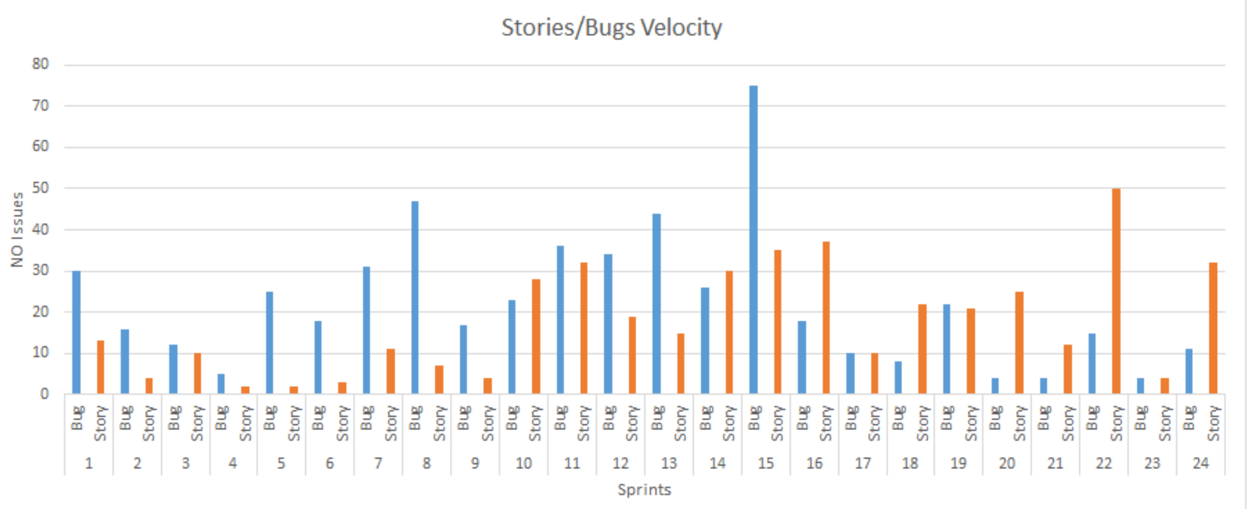

Fig. 3. Sprint velocity

In order to assess the predictability planned vs resolved issues were tracked. Unplanned worked consisted in 447 (24%) tickets that was caused by support tickets (63%) and issues that were completed without proper tracking in a sprint (27%). Out of 941 issues planned in sprints 486 (52%) were completed by the end of the sprint and 455 (48%) were dragged in future sprints. Most of the issues were completed in the immediate next sprint (45%) and only 30% of the issues were dragged more than 3 sprints. It's more likely for a bug than a story to be completed in the sprint that was planned.

Lead-time and Cycle time is calculated in Table 3. Because no information was available of when each specific issue has started the day when the sprint started was considered start time.

Table 3. Lead time and Cycle time

Issue Type	AVG Lead time (days)	AVG Cycle time (days)	AVG Cycle time (working days)
User Stories	44	25	19
Bugs	22	14	12
Task	23	17	14
Support Bugs	12	11	9

Further story point estimates are correlated with average cycle time in Table 4.

Table 4. Story Point Estimate vs Cycle time

Story Point Estimate	User Stories Count	AVG Cycle time (working days)
1	36	12.5
3	43	22.39
5	34	22.32
8	25	28.76
13	8	48
Not estimated	283	17.74

In order to gain more accurate figures each activity could have been tracked with the time management feature. This brings a big overhead of reporting so for the purpose of having a better sense of progress we considered this

approximation enough.
The velocity chart showed highs and lows that are typical in an agile environment considering the complexity of the product, ambiguity regarding system requirement and real life constraints. The main goal of the team was to go live with a complex product that was handed over by a previous company. In order to achieve this four stages are observed that relate with the ability to deliver working software. Each phase showed a short term improvement of velocity and then a drop that led the team in a new stage of development.

- Forming the team (Sprint 1 - Sprint 5);
- Building development capacity (Sprint 6 - Sprint 10);
- Norming the development process (Sprint 11 - Sprint 15);
- Go live and maintenance (Sprint 16 – Sprint 24).

The first important factor to deliver working software is forming a team with strong technical knowledge and good understanding of the business domain. This process the team took several months because it required finding highly skilled individuals. In this phase the focus was on gaining knowledge regarding the system and proving the ability to setup a proper development environment. The second factor that influences the ability to deliver working software is building team capacity. This is not expressed only in terms of allocating specific resources to the project, but creating an environment that fosters collaboration. In this phase the focus was on gaining knowledge regarding the business domain and getting the system to a stable state of functionality. Norming the development process is the third factor that influences the ability to deliver software. In this phase workshops were organized to align team members to the same goals and working practices. A product roadmap was put in place by the product managers and estimated by the development team using t-shirt sizing techniques that had led to an increase in ownership. Introducing Sprint Goals as a way of focusing activities on the team level rather than individual task allocation had a positive impact in collaboration.
Going live in Sprint 16 had a great influence on the team dynamics. Because real life feedback started to come from the users the team had to change its way of working to assure both delivery of new features and maintenance. Support has increased the amount of unplanned work and a system of continuous delivery to release hotfixes. Performance issues prevailed new functionality and feedback from the users became the most important factor that influenced what was delivered. In this phase the most important factor was adaptability.
Because lead-time and cycle time are metrics taken from production systems, they are useful when what is produced remains the same, like a specific product or item. When considering knowledge work, the output is complex and may vary a lot. Data analysis showed a great variance on each type of issue tracked. The types of problems in software development are complex and unique so norming work is a difficult task. While these metrics can create an overview of performance, tracking them should take in consideration that with new features or modules is expected for the team to have a low predictability in the beginning. Kanban metrics tend to be more useful in support and maintenance activities.

5 The proposed architecture

In order to automate the process of data gathering and metric calculation, we propose a system that will collect data from several sources. These sources include project management and issue tracking software. A dedicated module will interrogate the software and will export data in several formats. These files will be processed by the metrics generator and it will generate calculated data in requested formats (XML, JSON etc.). The system also provides a reporting module that presents the benchmarking results. Figure 4 depicts the architecture of the proposed system.

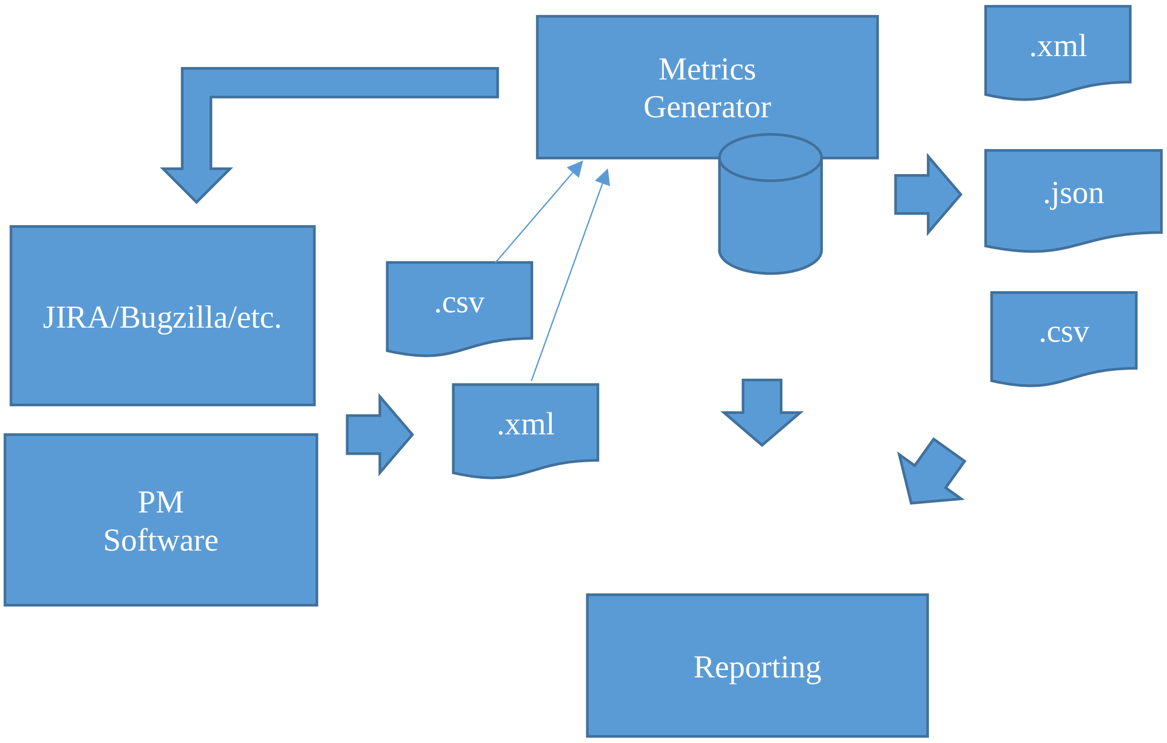

Fig. 4. A real-time agile metric system

The proposed system, depicted in Figure 4, includes the following interconnected components:

- Agile project and process management software and issues tracker software;
- The metrics generator;
- Database and a corresponding access module;
- Import/Export modules;
- Reporting module.

The project and process management software and/or *issues tracker software* represent the essential tools for agile software project management and they provide the required data for metrics calculation. Examples of agile project management tools include JIRA, VersionOne, Trello, Pivotal Tracker, Bugzilla etc. Data provided by the project management software is collected using the *Import/Export modules*. The modules access the project management and issue tracking software and export the required data in the specified format.
The import module will access the project and process management software using several ways:

- through provided API, if available;
- directly accessing the database;
- using a GUI automation sequence to select and export the required data.

After data is imported it is aggregated and normalized.
These modules replace the manual processing of data based on Excel or other spreadsheet software.
The metrics generator is the main component of this system. It includes the logic to calculate the metrics and indicators required for agile project benchmarking. As example of metrics we mention velocity, lead-time and cycle time. The input of this module is represented by files generated from project management software using the export/import modules. This module should allow to design the relations between data in order to define new metrics.
The processed data (metrics, indicators) is exported in the specified format using the import/export modules. The system allows using several formats like XML, JSON and CSV. The metrics generator could be connected with other modules that provide metrics that are not collected by the project and process management software, at a source code level (KLOC, cyclomatic-complexity, Halstead complexity, the number of classes, the number of functions etc.).

All data is stored in a *database* in order to assure its persistence and to allow benchmarking with older projects. A database access module is required to access and manipulate the stored data.

The reporting module helps to display data in a tabular or graphical format (bar charts, pie charts, spider charts etc.), depending on user selection. The data can selected directly from the database, or could be loaded from the exported files.

6 Discussion

Using a system that provides real-time agile metrics for measuring team performance has several advantages like:

- less work required to select, export and process data from agile project and process management software;
- integration with all major agile project and process management software;
- improved benchmarking performance, having historic data;
- team members, project and process managers could better focus on other activities;
- uses a modular architecture that allows to add new feature without changing other components.

The system has to be implemented open enough to allow addition of new modules and updates.

The disadvantages of this type of system are:

- it requires additional work.
- it has to take into account the main project and process management software;
- new versions of project and process management software could require changes in importing modules.

The disadvantages can be avoided through a good design process.

The case study presented is section 4 is based manually processed data. Using an automated system like the one we propose, these reports will be generated in no time.

7 Conclusion and future work

Agile software development has produced a shift in the mindset of measuring progress by focusing on working software. In this paper three key agile metrics were analyzed in order to quantify the performance a development team in an outsourcing company. The ability to deliver working software is influenced by team capacity, having a normed development process and adaptability to changing requirements. Recommendation when using agile metrics is take in consideration the specific phase of team development (forming, storming, norming, performing). Metrics should always be a reason for conversation in seeking improvement. In this specific case metrics provided a basis for discussion and increased transparency. Velocity, lead-time and cycle time are considered to be internal measures of productivity.

Future research will focus on external measures like customer satisfaction, perceived quality and product revenue in order to correlate them with internal metrics.

We also plan to implement the proposed system in order to provide real-time agile metrics and to compare with previous results in a real development environment.

Acknowledgement

Parts of this research have been published in the Proceedings of the 17th International Conference on Informatics in Economy, IE 2018 [21].

References

[1] C. W. H. Davis, *Agile Metrics in Action: Measuring and Enhancing the Performance of Agile Teams*, Manning Publications, ISBN 978-1617292484, July 2015, p 272

[2] E. Kupiainen, M. V. Mäntylä, J. Itkonen, "Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies," Information and Software Technology, vol. 62, pp. 143–163, 2015

[3] P. Kua, "An Appropriate Use of Metrics," Internet: https://martinfowler.com/articles/useOfMetrics.html, February 2013, [Mar. 30, 2018]

[4] E. N. Budacu, Development of Agile Practices in Romanian Software Community, Informatica Economică vol.21, no.2, pp.

92-102,2017
[5] K. Schwaber, J. Sutherland, “The Scrum Guide,” http://scrumguides.org/scrum-guide.html, 2017, [Mar. 30, 2018]
[6] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional, ISBN 978-0201616415, October 1999, p 224
[7] M. Stoica, B. Ghilic-Micu, M. Mircea, C. Uscatu, “Analyzing Agile Development – from Waterfall Style to Scrumban,” Informatica Economică vol. 20, no. 4, pp. 5-14, 2016
[8] A. Mihalache, “Project Management Tools for Agile Teams,” Informatica Economică, vol. 21, no. 4, pp. 85-93, 2017
[9] D. McIvera, M. L. Lengnick-Hallb, C. A. Lengnick-Hallb, *A strategic approach to workforce analytics: Integrating science and agility*, Business Horizons, 2018
[10] I. Ivan and C. Boja, *Metode statistice în analiza software*, Bucharest: ASE Publishing House, 2004.
[11] J. T. McCabe, "A Complexity Measure," *IEEE Transaction on Software Engineering*, Vols. SE-2, no. No. 4, pp. 308-320, 1976.
[12] M. Vetrici, "Software Project Duration Estimation Using Metrix Model," *Informatica Economica* , vol. XII, no. 47, pp. 87-91, 2008.
[13] S. Pressman, *Software Engineering: A Practitioner’s Approach. 8th edition*, New York: McGraw-Hill, 2014.
[14] ISO/IEC/IEEE, ISO/IEC/IEEE 12207:2017 - Systems and software engineering - Software life cycle processes, Geneva, 2017.
[15] IEEE, IEEE 1012:2012, Standard for System and Software Verification and Validation, New York, 2012.
[16] ISO/IEC/IEEE, ISO/IEC/IEEE 29119-3 Software and systems engineering - Software testing, Geneva, 2013.
[17] M. Al-Zewairi, M. Biltawi, W. Etaiwi și A. Shaout, „Agile Software Development Methodologies: Survey of Surveys,” Journal of Computer and Communications, vol. 5, nr. 5, pp. 74-97, 2017
[18] I. Sommerville, *Software Engineering. 9th edition*, Boston: Addison-Wesley, 2011
[19] VersionOne, „The 11th annual STATE of AGILE Report,” 2017.
[20] K. Aguanno, *Managing Agile Projects*, Multi-Media Publications Inc., 2004.
[21] E. N. Budacu, “Agile Metrics in a Software Outsourcing Company,” in Proc. of the 17th International Conference on informatics in Economy (IE 2018), Education, Research & Business Technologies, Iasi, Romania, 17 – 20 May 2018, pp. 393-398.

Eduard Nicolae BUDACU has graduated the Faculty of Cybernetics, Statistics and Economic Informatics from the Bucharest University of Economic Studies in 2010. He has graduated the SIMPRE - ERP oriented Master's Program from the Bucharest Academy of Economic Studies in 2012. He is currently a PHD Student at the Economic Informatics PHD School. His main field of interest is agile software development. He is an agile coach and helps software development teams produce positive change in order to achieve high performance using agile principles and practices. He works with companies to define learning and development strategies for agile transformation.

Paul POCATILU graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 1998. He achieved the PhD in Economics in 2003 with thesis on Software Testing Cost Assessment Models. He has published as author and co-author over 45 articles in journals and over 40 articles on national and international conferences. He is author and co-author of 10 books, (Mobile Devices Programming and Software Testing Costs are two of them). He is professor at the Department of Economic Informatics and Cybernetics within the Bucharest University of Economic Studies, Bucharest. He teaches courses, seminars and laboratories on Mobile Devices Programming, Economic Informatics, Computer Programming and Project Management to graduate and postgraduate students. His current research areas are software testing, software quality, project management, and mobile application development.

DOI: 10.12948/issn14531305/22.4.2018.06