PW-PLAN
A Strategy to Support Iteration-based Software Planning

Deysiane Sande, Arnaldo Sanchez, Renan Montebelo, Sandra Fabbri and Elis Montoro Hernandes
Computing Department, Federal University of São Carlos, São Carlos, Brazil

Keywords: Planning, Planning Tracking, Process Improvement, Agile Method, Small Companies.

Abstract: Background: Although there are many techniques in the literature that support software size estimation, iteration-based software development planning is still based on developers' personal experience in most companies. Particularly for the agile methods, iterations estimation must be as precise as possible, since the success of this kind of development is intrinsically related to this fact. Aim: In order to establish a systematic planning of iterations, this article presents the PW-Plan (Piece of Work Planning) strategy. This strategy is based on four items: the iterative development, the use of a technique to estimate the complexity of the work to be done, the adoption of personal planning practices and the constant evaluation of the Effort Level (EL). Method: PW-Plan evolved from another strategy that was elaborated based on the systematic practice of using Use Case Points, Personal Software Process and constant EL evaluation. Results: PW-Plan was used by two small businesses companies in two case studies and showed that its application is feasible from the practical point of view and that it enhances the development control. Conclusion: The case studies provide insights of the PW-Plan contribution for both the developer's and the manager's processes. Furthermore, the strategy application provides more precise estimations for each iteration.

1 INTRODUCTION

One of the key aspects of the planning and management of projects is the estimation of how long a project will last. According to Pressman (2007), the estimated time and cost are often imprecise. This problem becomes more serious in the context of small business companies which continuously deal with the market pressure to develop high quality systems with restrict deadlines. In such cases, the control of time and costs is vital to continue in the market and the company revenue is directly related to the delivery of each system ordered within the estimations made.

Given this scenario, Sanchez, Montebelo and Fabbri (2007) proposed the UCP|$_{PSP}$ strategy aiming to achieve more precise estimations combining continued planning and control activities in order to keep planning adjusted with current time spent on development. This strategy was built on the lessons learned at Linkway company through the continued use of Use Case Points - UCP (Karner, 1993) and the Personal Software Process - PSP (Humphrey, 1995). Linkway is a small software company that looks to the constant improvement of its software development process. This concern led the company to adopt the use of the PSP. Thus, the development team uses the main practices of PSP 1.1 - related to the project estimation - together with the Process Dashboard tool (Dashboard, 2010), which supports the PSP.

After defining the UCP|$_{PSP}$ strategy, it was observed that its steps were quite naturally related to agile practices proposed in the Scrum framework (Schwaber, 2004). Being agile methods a feasible methodology for small teams and not complex systems (Beck & Andres, 2004), they become appropriate in the context of small businesses.

Thus, the UCP|$_{PSP}$ strategy evolved, in order to become more generic regarding the method used to estimate size, planning and describing iterations not only through use cases, but also by any other unit of work, such as user stories. Moreover, being based on iterations, it becomes adaptable, especially to agile methods. This evolution of the UCP|$_{PSP}$ strategy was given the name PW-Plan, which is presented in this paper.

This article is organized as follows: Section 2 presents the concepts related to agile methods and agile planning. Section 3 details the PW-Plan

Sande D., Sanchez A., Montebelo R., Fabbri S. and Montoro Hernandes E. (2010).
PW-PLAN - A Strategy to Support Iteration-based Software Planning.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Databases and Information Systems Integration*, pages 66-74
DOI: 10.5220/0002909100660074

strategy proposed in this paper. In Section 4 it is presented two case studies in two small business companies, Linkway and NBS, showing the use of the strategy in different situations. In Section 5 it is presented the lessons learned and, finally, in Section 6 it is presented the conclusions and further work.

2 AGILE METHODS AND AGILE PLANNING

The agile approach applied to software project management came into the spot in 2001, date of the publication of the Agile Software Development Manifest (Manifesto, 2001). This manifest highlighted their differences compared to traditional methods, especially by being incremental, cooperative, direct and adaptive. The most prominent agile methodologies are: Extreme Programming - XP (Beck & Andres, 2004), Dynamic Systems Development Method (DSDM, 2010), Feature Driven Development - FDD (Palmer & Felsing, 2002), Adaptive Software Development - ASD (Highsmith, 2002), OpenUP (Openup, 2010), the Crystal Clear and Orange methods of the Crystal methodology (Cockburn, 2002) and Scrum (Schwaber, 2004).

Scrum stands out among the methods due to its emphasis on project management. It is an agile framework that provides a set of best practices to achieve the success of a project, supporting the construction of a software product in iterative steps. It does not define 'what should be done' in all the circumstances. Hence, it may be used in complex projects where it is not possible to predict everything that will occur (Schwaber, 2004).

Scrum projects are carried out in a series of iterations called Sprints. Each Sprint has a certain time in calendar days to be completed. Schwaber (2004) proposes a 30 days Sprint.

Scrum defines three main roles: Product Owner, responsible for the requirements; Team, represented by developers, and Scrum Master, represented by the manager. The two main artifacts of Scrum are the Product Backlog - list of requirements that must be implemented in the system - and the Sprint Backlog - list of tasks to be performed on a Sprint.

The process begins when the product owner has a general description of the product to be developed. From this description, called Vision, the Product Backlog is created. At the beginning of each sprint, the Sprint Planning Meeting takes place, where the Product Owner prioritizes the Product Backlog items. Based on this prioritization, the Team selects the tasks that will be in the Sprint Backlog.

During the Sprint, the Team carries out the Daily Scrum Meeting - which is 15 minutes long – where the work is synchronized and possible issues are discussed. At the end of each Sprint, the Team presents the completed functionality at the Sprint Review Meeting, and the Scrum Master encourages the Team to review the development process to make it more efficient for the next Sprint.

A key point for the proper practice of agile methods is the planning of iterations. This planning must be based on the size estimation of the items that will be developed, and also based on the productivity of the Team members. Although these items may have different representations, the most common one is user story, which is a brief description of the functionality being developed accordingly to the client's project vision (Cohn, 2005).

Among the existing methods to estimate the size of the work to be done, Cohn (2005) highlights:

- Story Points (SP): unit of measure which express the size of a user story, a system characteristic or any piece of work to be developed.
- Ideal days: unit of measure which corresponds to an ideal day of work, which is a day when every resource needed, is available and the work is done without interruptions.

Two scales of magnitude are suggested by Cohn to characterize the complexity (or size) of the work to be done: the Fibonacci sequence, in which the next sequence number is the sum of the two previous numbers (1, 2, 3, 5, 8,...); and a second sequence, in which each number is twice the precedent number (1, 2, 4, 8, 16,...). These scales can be used in conjunction with what Cohn called Planning Poker. In order to estimate the complexity of the task, Team members receive cards with these sequence numbers, and for each Piece of Work, the values are arranged together until a consensus is reached. Haugen (2006) presents results that indicate a good performance of the Team on the accuracy of estimations when this type of technique is used.

In addition to the methods presented to calculate the size of the work to be done, there are other more traditional techniques that can be used to assist in planning. They are:

- Function Points Analysis (FP): proposed by Albrecht (1979) for measuring software projects size. This measure is calculated based on the complexity of the technique five logical components. These points are calculated in two

steps, generating respectively the unadjusted and the adjusted points. In the latter, technical and environmental factors are considered to interfere with the complexity of the development.

- Use Case Points (UCP): proposed by Karner (1993) to estimate software projects based on use cases. This technique was inspired by FP and also calculates the unadjusted and the adjusted points. Unadjusted Use Case Points are based on the complexity of actors and use cases. The Adjusted Use Case Points considers environmental and technical complexity factors, much like FP. Based on the complexity, Karner estimated the development time by multiplying UCP by 20 man-hours. This is a value that should be adjusted to the company size and to the complexity of the software being developed.

The agile estimation techniques previously mentioned are feasible alternatives to achieve the necessary estimations in the strategy proposed in this research, even if agile methods are not being used. In addition, despite the technique used to estimate, it is important to control and monitor the estimation and the software planning. As it will be presented, the strategy proposed herein takes into account this activity.

3 PW-PLAN STRATEGY

The PW-Plan strategy is an evolution of $UCP|_{PSP}$ (Sanchez et al., 2007), which was established by systematically using the Use Case Points and the PSP methods together. This strategy evolution resulted from the observation that the strategies steps would be easily adjusted to Agile Methods, especially Scrum.

The strategy main goal is to support planning and monitoring of the development plan of each iteration, increasing the software development process quality.

While Scrum only determines ‘what should be done’ (Kniberg, 2007), the strategy defines ‘how it should be done’.

In this paper context, it is considered the PSP 1.1 usage with the Process Dashboard tool, which records the total time spent in every tracked activity. However, if it is chosen not to use PSP and Process Dashboard, the strategy can still be used, as long as alternatives ways of time tracking are applied.

The strategy consists of two large blocks which are constantly executed: planning and control. These blocks feed each other with information gathered by the PSP method, which provides constant feedback.

The Control block aim is to assess if the planning elaborated in the Planning block is being correctly followed and, if not, the reasons for this situation. With the control activities feedback, the planning activities are constantly adjusted by the LE (Level of Effort), which represents the relationship between time spent and work done. The work itself is characterized by the complexity of the estimation method that is being used, like UCP, SP, etc.

The Scrum roles are represented in the strategy as follows: the Scrum Master is represented by the manager; the Team is represented by the developers; the Product Owner is the customer representative who is responsible for the return of investment.

The Scrum activities are identified in the strategy as: Sprint Planning Meeting 1 corresponds to Step 1; Sprint Planning Meeting 2 is related to Steps 1, 2 and 3; the Sprint Review Meeting is related to Step 9.

Regarding the work to be done, the correlation between Scrum and the proposed strategy is as follows: the Product Backlog is represented by a system specification that can be described through use cases, stories, etc; the Sprint Backlog corresponds to the Piece of Work (PW), which may be composed by Items of Work (IW) that can be composed of Tasks. For example, a PW may be a set of use cases selected for an iteration; an IW, in this case, means a use case which can be decomposed into tasks.

Figure 1 presents the whole strategy, which is composed by the following steps:

- Step 1 – Planning Meeting: from system specification - which can be represented by use cases, user’s stories and etc - the manager and the developer discuss the complexity of the work to be done. This complexity is characterized by a technique compatible with the representation being used. For example, use cases require the use of the UCP technique; user stories require the use of Story Points, and so on. In the case of Story Points, the complexity will be determined using some technique such as, for example, Planning Poker in conjunction with the Fibonacci sequence.
- Step 2 – PW Detailed Planning: based on the specification of Step 1, it is defined the PW to be developed in the iteration. The IWs that compose this PW are defined based on the development work load of the iteration. If the iteration is the first one, it should be used historical data or the manager and developers’ experience to determine the LE. The subsequent iterations must use the LE calculated in step 8.

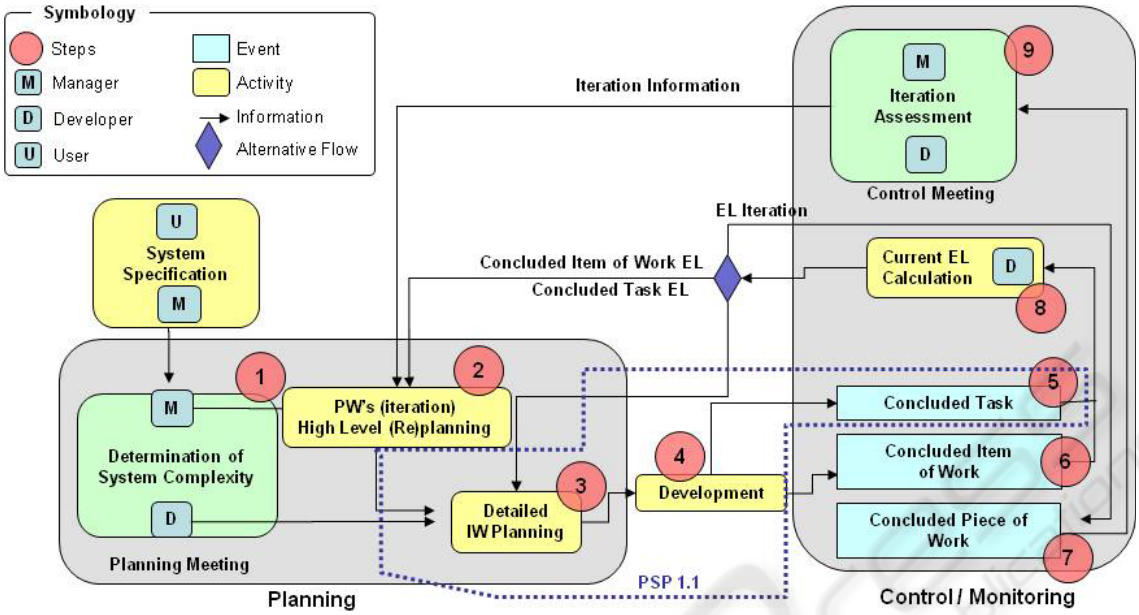

Figure 1: PW-Plan strategy.

- Step 3 – Detailed IW Planning: each developer is responsible for the division of the IWs assigned to them in Tasks, using the method of his or her preference. At the end of each Task the developer must evaluate his LE accordingly to the time recorded by Process Dashboard tool, aiming to obtain a more precisely planning of the next task. If PSP is being used, this auto-evaluation is equivalent to the 'postmortem' activity. In addition, the Process Improvement Proposal report is generated, where errors are reported and estimation improvement activities are proposed.
- Step 4 – Development: based on the detailed planning of the previous step, the developer effectively does coding, testing and defects fixing activities according to his personal process, which can be improved using the PSP guidelines.
- Step 5 – Concluded Task: this event is characterized by the conclusion of a Task, hence triggering Step 8, which calculates a new LE value.
- Step 6 – Concluded Item of Work: this event is related to the conclusion of a IW, triggering Step 8;
- Step 7 – Concluded Piece of Work: end of iteration. This is characterized by a PW conclusion, triggering Step 9 execution;
- Step 8 – Current LE calculation: when a Task or an IW are concluded, the developer's LE must be adjusted to meet the actual relationship between work done and effort in man-hours. The adjustment corresponds to accumulated time spent so far (provided by PSP using the Process Dashboard tool) divided by the number of points that represent the complexity of work already done.
- Step 9 – Control Meeting: at the end of an iteration, a meeting between manager and developers is carried out to discuss lessons learned and project scope. Eventually, the scope may change due to found or eliminated requirements, which may impact the initial planning. Besides, if the LE has a very big variation in one unique iteration, it must be evaluated if any external factor (technical or environmental) may be interfering the developer's performance.

In summary, the PW-Plan Strategy provides a systematic approach for planning and controlling iteration-based software development. It is based on the PSP guidelines for individual software process improvement, particularly on the planning activities. Iterative work is the base of the strategy and then, it is easily adapted to Scrum. Besides, the strategy work unit, named Piece of Work, is a general unit and can also be adapted to the enterprise unit.

In the next section, two case studies will be detailed showing the application of the strategy in different situations.

4 CASE STUDIES

This section presents two application examples of the proposed strategy: the first is the development, from scratch, of a website by the Linkway company;

and the second is an enhancement to a traditional desktop system by the NBS company. Each project used a different estimation procedure: the first used "Use Case Points" (UCP), while the second one used "Story Points" (SP) together with the Fibonacci sequence (Cohn, 2005). This fact shows that the strategy is generic and can be adapted to different habits and needs. In both cases the companies used Process Dashboard (Dashboard, 2010), a free, open-source PSP support tool.

4.1 Linkway Case Study – Use Case Points

This case study was conducted at Linkway company during the development of a web portal to a carpet industry. This portal had the following features: a catalogue of manufactured products, a list of representatives, product news and institutional data. These data were stored in a database and were manipulated by the Web application. The whole portal was built in Java (Sun, 2010) by only one developer, who consumed approximately 216 man-hours distributed over the nine use cases that composed the system. To plan the activities, the strategy presented in this paper was applied using Use Case Points to calculate the estimation.

First, at the Planning Meeting (Step 1), the manager and the developer defined the complexity values of Actors and Use Cases, hence calculating the Unadjusted Use Case Points. Also in this step, the complexity of technical and environmental factors were evaluated. Then, in the PW Detailed Planning (Step 2), it was calculated the Adjusted Use Case Points and the total estimated time to develop the system. This time corresponds to the multiplication of Adjusted Use Cases Points by the LE, which was determined from the company's historical data. These values are presented in Table 1.

Still in the detailed PW planning stage, it was defined the use cases (IWs) that composed an iteration. To define the PW, it is necessary to know the duration of one iteration. At Linkway, an iteration corresponds to a two weeks' period (or around 60 hours). The workload of the iteration is considered this way because a developer has a daily journey of 8 hours of work, but for planning purposes only 6 hours are considered per day. Thus, considering the historical LE (3), each iteration should have a maximum of 20 UCP (hours / LE = 60 / 3 = 20).

Table 1: Initial Web system planning values developed by Linkway.

Description	Value
Unadjusted Use Case Points	101
Technical Complexity Factor	1.10
Environment Factor	0.85
Adjusted Use Case Points	93.45
Initial LE (company's historical data)	3 man-hours
Initial Total Time Estimated	280.40 man-hours

Based on this value, the use cases were distributed considering 20 UCP per iteration. From this point on, the system development was started and both the LE and the total development time were continuously adjusted. This adjustment allowed the re-planning of the current iteration at the end of the use case, and also allowed that the next iteration would be calculated based on actual data instead of historical information.

To perform these adjustments, at the conclusion of each Use Case (Step 7), the following values were updated:

- Accumulated Use Case Points value: the current Use Case value plus the previously finished Use Cases points;
- Accumulated time spent in the Use Cases development: time spent in the current Use Case plus the previously accumulated value;
- Current LE value: Accumulated Time divided by Accumulated UCP (this new LE varies as the developer's performance varies);
- Remaining time to system finalization: new LE multiplied by Adjusted UCP minus Accumulated UCP so far.

Table 2 depicts the application of the strategy, and such table must be elaborated as each Use Case is finished. This systematic monitoring of the development process provides the effective control of the iterations and hence this allows the constant adjustment of the iteration, making the overall planning more feasible and less error-prone.

Thus, after finishing the use case 1, the corresponding row for the use case was updated. The accumulated values were exactly the same as individual values because only this use case was developed so far. The LE is then 1.70 (24.60 / 14.50), not 3.0 as it was initially assigned according to historical data (see Table 1). Due to the decreased LE value, the planning was recalculated and it was possible to predict that the initial planning of 20 points per iteration could be increased to 30.

Table 2: Results of the strategy application at Linkway.

	Step 1	Steps 1 and 2		Step 4		Step 8	Step 2
Use Case	Complexity	Adjusted Use Case Points		Time effectively used for the Use Case conclusion		LE	Time left (in hours)
		Individual	Accumulated	Individual	Accumulated		
Sprint 1							
1	Complex	14,50	14.50	24.60	24.60	1.70	134.22
2	Simple	5.24	19.74	8.36	32.96	1.67	123.10
3	Medium	9.87	29.61	38.20	71.16	2.40	153.22
Sprint 2							
4	Complex	14.50	44.10	53.50	124.66	2.83	139.66
5	Medium	9.87	53.97	10.20	134.86	2.50	98.70
Sprint 3							
6	Simple	5.24	59.22	6.50	141.36	2.39	81.81
7	Medium	9.87	69.09	29.50	170.86	2.47	60.17
8	Medium	9.87	78.96	28.50	199.36	2.52	36.51
Sprint 4							
9	Complex	14.50	93.45	16.85	216.21	2.31	0

It is likely that the experience gained by the developer has caused him to be more productive developing this use case than in previously developed applications, which had gave him an LE equal to 3. Thus, if the developer continued with this new calculated productivity, the application - which initially should consume 280.4 hours - in the current conditions would consume 158.87 hours of work (93.45 UCP * 1.7 LE). Hence, the time estimated to completely finish the application would be 134.22 hours ((93.45-14.50) * 1.70).

When the use case 2 was completed, the same calculations described early were applied. The new LE was, then, 1.67, indicating that to achieve full implementation more 123.10 hours would be needed. This would correspond to an error of 157.30 hours, compared with initial estimations.

Observing the data for use case 9, it is possible to note that the LE has increased to 2.31, and that the actual number of hours spent to develop the system was, actually, 216.21, less than the original estimation of 280.40 man-hours.

It should be highlighted that this constant change in the LE was the result of the application and registration of the planning activities of the PSP. This procedure gives the developer a greater personal planning capacity, as well as a more precise work estimation capacity.

4.2 NBS Case Study – Story Points

The second case study, applied in NBS company, was an update to a desktop system of public accounting, developed in Delphi (Embarcadero, 2010). The existing accounting system was restructured to meet the requirements of electronic auditing by governmental agencies.

Because this is a maintenance activity in a previously existing system, the application of Use Case Points was not appropriate because only some parts of the use cases would be modified. Thus, modifications in the system were described as user stories. As the previous case study, work was performed by only one developer, who consumed approximately 380 man-hours distributed over 40 stories that made up the system.

The Fibonacci sequence - which is one of the methods proposed by studies in the area to characterize the complexity of a user story (Cohn, 2005) - was used to calculate these Story Points.

At the Planning Meeting between manager and developers (Step 1), each user story was given a score, using the Fibonacci sequence. The total Story Points to complete the development of the system was calculated as 308.

Then, in the detailed PW planning activity (Step 2), the total time estimated to update the system was calculated by multiplying the Story Points by the LE, whose value was taken from historical data. From these calculations, a determined story quantity was allocated for each iteration, for the next two weeks. These values are shown in Table 3.

The distribution of the stories per iteration was done, as the previous case study, considering that each iterations lasts for approximately 60 hours. Thus, considering the historical LE (1.3), each sprint should have, approximately, 47 points per story (hours / LE = 60 / 1.3 = 47.2).

Table 3: Initial planning values of the desktop system updated by NBS.

Description	Value
Total Story Points	308
Initial LE (historical data)	1.3 man-hours
Initial total time estimated	400.4 man-hours

The development was then initiated, and the adjustments of the LE and the remaining time for conclusion were done at the end of each story.

When a story was concluded (Step 7), the following values were updated:

- Accumulated Story Points: current story points plus the previously accumulated value;
- Total accumulated time spent developing stories: time spent developing current story plus previously accumulated value;
- Current LE value: total accumulated time spent developing stories divided by accumulated story points;
- Total time to development completion: current LE multiplied by total story points, minus accumulated points so far;

Table 4 shows the developed stories during the iterations and the calculated values when applied the strategy. As an example, the LE calculation for story 3 was the total time (0.48 + 2.20 + 12.42 = 15.10) divided by total story points (1 + 2 + 8 = 11), which is 1.37 (15.10 / 11). The data is only partially shown in table 4 because there were too many stories to be represented in this paper.

When story 1 was finished, the corresponding row was updated. The accumulated values were exactly the same as individual values, because only this story was developed so far. The LE is, then, 0.48 (0.48 / 1), and not 1.30 as initially attributed by historical values. Hence, if the developer continued with this productivity, only 147.36 hours would be remaining to complete the whole development.

However, this LE value produced a very low time estimation. Thus, it was decided to wait for the completion of a more complex story to verify if the productivity would remain so high. When story 3 was concluded, it was possible to note that the LE was then much more near the initial value taken from the developer history. Thus, the manager decision was to keep the iteration development considering the same quantity of stories distributed in the PW Detailed Planning. At the end of iteration 1, the LE was 1.2, nearer the 1.37 of story 3. Based in this new LE, a new Story Points was calculated as the appropriate amount of work for each iteration. This value is (time / LE = 60 / 1.20) 50 points. As this story points values was similar to the initial value (50), no modifications were made to the iterations organization.

For this 308 story points system with average LE of 1.24, the actual time spent developing the whole system was 380 hours, which is less than the originally estimated 400 hours. This difference between estimated time and actual time, yet small one, is a strong evidence of the developer's improvement in his personal planning capacity and work estimation. This is, again, result of the constant application of the PSP methods, which require that the developer plan his work and then become more precise in his estimations.

5 LESSONS LEARNED

The main lessons learned are related to the definition of a personal process, planning and monitoring of a software project.

The definition of a personal process improves the productivity and the decision-making capacity of the developer. In the case studies presented, both companies formally adopted the PSP, which provided a disciplined development environment which productivity could be adjusted constantly. Thus, if the developers do not adopt the PSP, they should always produce estimations of the work to be developed and then track the time spent to effectively develop it.

The iteration-based development facilitates the planning, which must be elaborated according to the productivity of each developer. Hence, chances of success are high, which makes the strategy very feasible in the small businesses context.

The planning monitoring must be constantly done, because it allows estimations to be adjusted at any time in the development. This monitoring should be done by the manager, who will act as a coach of the team, constantly re-estimating the work to be developed and encouraging developers to improve their personal software development processes.

6 CONCLUSIONS

In this paper, the PW-Plan strategy was presented. This strategy supports the planning stage of iteration-based software development. It can be

Table 4: Results of the strategy application in NBS company.

		Steps 1 and 2		Step 5		Step 8	Step 2
Iteration	Story	Story Points (Fibonacci)		Time spent (in hours)		Current LE	Time remaining (hours)
		Individual.	Accumulated.	Individual.	Accumulated.		
1	1	1	1	0.48	0.48	0.48	147.36
	2	2	3	2.20	2.68	0.89	272.47
	3	8	11	12.42	15.1	1.37	421.96
	(…)						
	16	3	47	2.97	56.43	1.20	313.37
7	39	21	300	19.97	371.96	1.24	9.92
	40	8	308	8.67	380.63	1.24	0

used with other development methodologies because the planning phase is essential for every development cycle. Each iteration develops a PW, which can be composed of IWs that are distributed among developers. These, in turn, can decompose an IW into Tasks. Throughout development, the developers must use the planning guidelines of the PSP 1.1, so that the developer has a commitment of planning and tracking the time associated with each development effort. This approach allows the Level of Effort (LE), which reflects the relationship between work and the time spent doing work, to be is constantly updated. This always up to date value allows the monitoring of the project as a whole, as well as each iteration.

According to case studies, it was obtained evidence that the PW-Plan strategy supports the project planning and control, providing an improvement in the activities of both the manager and the developer. For the manager, he will take greater control over the project and will be able to take decisions at the appropriate time if something does not happen as expected. For the developer, he finds out his productivity and, therefore, does more precise estimations. Overall, estimations for the software project are more accurate.

It is noteworthy that in the context of large companies the strategy may have to be adapted, because depending on the size of the development team an individual control is more difficult. For larger teams, the manager must define an approach to "coach" several developers at the same time. Basically, the strategy must be applied by each developer and the manager should control their productivity individually. The two case studies presented had the participation of a single developer, fact that did not allow a preliminary assessment of this issue.

The LE value varied almost 100% from one company to another. This clearly suggests that the LE should be adjusted to the context of each company in order to represent, preferably, the productivity profile of each developer. There was a relative stability of the LE for each project, allowing the constant monitoring of development through the analysis of this variable.

Because PW-Plan is based on iterations and in the performance monitoring of the developer, it is a generic strategy. It can be adapted to agile methods, to the estimation technique used by the company and from small to larger development teams.

As future work, it is intended to include in this strategy other levels of the PSP, apply it to projects with more than one developer and also explore other types of metrics that could further improve planning.

At present, quality practices – especially Verification, Validation and Testing (VV & T) activities - are already being incorporated into the strategy in a way that the same systematic control remains functional. Also, it is intended to perform an analysis of the strategy implementation as a support to the implementation of some processes models, such as Capability Maturity Model Integration (CMMI, 2006) and Process Improvement of Brazilian Software (MPSBR, 2007).

ACKNOWLEDGEMENTS

We would like to thank CNPq and FAPESP for the financial support. Special thanks to Linkway and NBS companies for their cooperation in the work.

REFERENCES

Albrecht, A. J. (1979). Measuring application development productivity. *Proceedings of SHARE/ GUIDE IBM Application Development Symposium*

(pp. 83—92).
Beck, K. & Andres, C. (2004). *Extreme Programming Explained: Embrace Change.* Addison-Wesley Professional.
CMMI - Capability Maturity Model Integration Version 1.2. (2006), *CMMI-SE/SW, V1.2 – Continuous Representation.* (SEI Technical Report CMU/SEI-2006-TR-001).
Cockburn, A. (2002). *Agile software development*. Boston: Addison-Wesley Longman Publishing Co., Inc.
Cohn, M. (2005). *Agile estimating and planning*. New Jersey: Prentice-Hall.
Dashboard - The Software Process Dashboard Initiative. (2010). Retrieved January 5, 2010, from http://processdash.sourceforge.net
DSDM - DSDM Public Version 4.2 Manual. (2010). Retrieved January 10, 2010, from http://www.dsdm.org/version4/2/public/
Embarcadero - Delphi from Embarcadero. (2010). Retrieved January 20, 2010, from http://www.embarcadero.com/products/delphi
Haugen, N.C. (2006). *An empirical study of using planning poker for user story estimation*. Proceedings of Agile 2006 Conference (pp. -34).
Highsmith, J. (2002). *Agile software development ecosystems*. Addison-Wesley.
Humphrey, W. S. (1995). *A discipline for software engineering*. Pittsburgh: Addison-Wesley.
Karner, G., 1993, *Resource Estimation for Objectory Projects.* Objective Systems SF AB (copyright owned by Rational Software).
Kniberg, H. (2007). *Scrum and XP from the Trenches - How we do use Scrum*. Retrieved from http://www.crisp.se/henrik.kniberg/ScrumAndXpFromTheTrenches.pdf
Manifesto - Manifesto for Agile Software Development. (2001). Retrieved November 5, 2008, from http://agilemanifesto.org/
MPSBR. (2007). *Melhoria de Processo do Software Brasileiro – Guia Geral (Versão 1.2)*. Retrieved January 4, 2010, from http://www.softex.br/mpsbr
Openup. (2010). Retrieved January 4, 2010, from http://epf.eclipse.org/wikis/openup/
Palmer, S. R. & Felsing, J. M. (2002) *A Practical Guide to Feature-Driven Development*. New Jersey: Prentice-Hall.
Pressman, R. S. (2007). *Software Engineering: A Practitioner's Approach.* New York: McGraw-Hill, Inc.
Sanchez, A., Montebelo, R. & Fabbri, S. (2007). *PCU|$_{PSP}$: Uma Estratégia para ajustar Pontos por Casos de Uso por meio do PSP em Empresas de Pequeno Porte*. Proceedings of VI Simpósio Brasileiro de Qualidade de Software (pp. 187-202).
Schwaber, K. (2004). *Agile project management with Scrum*. Redmond USA: Microsoft Press.
Sun, Developer Resources for Java Technology. (2010). Retrieved January 5, 2010, from http://java.sun.com