Power to the programmer using measurement to optimise the software process at the individual level

Gerry Coleman and Rory O'Connor

Abstract

With over a decade of Software Process Improvement (SPI) in large organisations, the awareness of its importance has propagated to Small to Medium Enterprises (SMEs). The most well known models for SPI are primarily suited for large- or medium-sized organisations, but with some tailoring they can provide substantial support for SPI in small organisations.

The IPSSI (Improving Professional Software Skills in Europe) project aims to address the problems above and provide a process improvement framework for use by individual software engineers working in European SMEs. This paper describes the architecture of the IPSSI project. In particular it details experiences to date with end user trials of IPSSI SPI training material and the development of the IPSSI tool set which consists of data gathering and data analysis tools, which have been implemented in an web-based environment.

Keywords: Software process improvement, quality, measurement, estimation

1. Introduction

Many user needs in SPI are not fully catered for through existing software process models. In particular, there is an absence of support for process improvement at the individual level, although according to a recent survey of European software organisations 83% of companies believe that SPI is essential for future success and 87% of companies believe that SPI can significantly improve software quality [1]. However time, cost and lack of knowledge are seen as the main barriers to SPI usage. In Europe to date, SPI has focused at the organisational level. There is an absence of support for process improvement at the individual level. As a result a number of issues have been identified in the European software industry [2]. 1) There is no clearly defined framework to enable process improvement at the individual engineer level; software engineers work with an increasing array of tools in a range of development environments; 2) The SME needs a suitable method to improve their software engineers' capability for developing software of higher quality on schedule and to budget and 3) The SMEs need a set of personal process training material that is suitable to the European software industry.

The IPSSI (Improving Professional Software Skills in Europe) project is an ESSI funded project which aims to provide a process improvement framework for use by individual software engineers working in European SMEs. The focus of the project is on improving individual software engineering skills thus generating bottom-up improvement. Companies can experiment with SPI by sending individuals on IPSSI training courses and then monitoring its implementation. By training individuals in this way, costs are reduced.

2. IPSSI Structure

The focus of the IPSSI project is on bottom-up process improvement as illustrated in figure 1 and illustrates the three elements of personal software engineering; defining a personal process, personal project management and personal quality management.

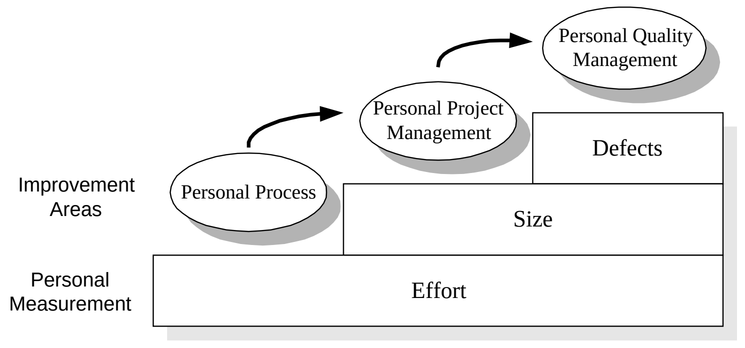

Figure 1 - The IPSSI Structure

The entire model is buttressed and controlled through the use of measurement. By collecting data on their own performance, software engineers learn about how they develop software. The measures help them understand the fundamental relationship between size and effort and, through this understanding, enable them to improve their estimating abilities. Furthermore, by gathering data on their defect rates they witness how employing practices such as personal code reviews and the use of checklists will allow them to produce higher-quality software products. The measures provide information on performance, information can then lead to process improvement and process improvement can lead to the production of better quality software on time. Finally collecting performance data on an ongoing basis moves developers from defining their own development process, through managing it to optimising it.

Through IPSSI training, developers complete programming tasks on which they collect increasing quantities of data. Early exercises capture effort measures. Subsequent exercises gather size data whilst the concluding exercises capture defect and quality measures.

This is hugely empowering for both the programmer and the organisation as a whole. Programmers are now in a position where they can provide the project manager with achievable deadlines and the project manager can develop more accurate and predictable delivery schedules.

The final element of IPSSI is that of personal quality management. As developers complete IPSSI program exercises, they collect data on the defects injected into those programs.

This process illustrates in which development phases they inject and remove defects. Furthermore, the defects are categorised by type thus allowing a causal analysis to be performed which can then lead to defect prevention. IPSSI focuses on proven quality control mechanisms such as design and code reviews which enable developers to remove defects earlier in the development process. This achieves the twin objectives of removing defects at the front end of the development cycle where they are cheaper and easier to fix and, as a corollary, means testing time is more focused as fewer defects are escaping into test.

Feedback from similar training programmes has suggested that the absence of a support tool, to simplify the recording and analysis of the data produced is one of the major barriers to continued usage of the methods [3]. Developers tire of recording data on paper forms and eventually usage of the disciplines peters out. As part of the IPSSI project, a tool set, which consists of data gathering and data analysis tools for use in a web-based environment, has been developed. The IPSSI support tool enables the measures to be collected as a simple complement to the development process. The tool also analyses the data collected to provide the developer with important process feedback.

This process information highlights the developer's strengths and weaknesses and empowers them to make the necessary process improvement adjustments.
We believe the provision of such a tool will ensure the 'buy-in' of training participants and subsequent continued usage of the IPSSI disciplines.

3. IPSSI Delivery

The delivery of the IPSSI training courses is divided into three distinct courses:

- **Introductory course** - aimed at management level within a software development organisation. This one-day event presents a management view of need for and justification of having a defined and established software process. In addition it outlines the IPSSI structure and the IPSSI approach to process improvement at the individual level.
- **Basic course** - is a two-day intensive training course aimed at the software developer. It establishes the need for having a defined software process at the individual level and concentrates on two main areas: personal project management and personal quality management.
- **Advanced course** - provides a natural progression for participants of the basic course who have implemented IPSSI into their software development practices. This three-day course builds on the basic course and covers similar topics at a more detailed level and additional topics such as diversified process models and team related issues at the individual process level.

The main learning objectives of the IPSSI course are:

- presents techniques and methods design to improve planning activity.
- presents techniques and methods design to improve estimation.
- presents techniques and methods design to improve quality.
- understand the need for having a defined and established software process.
- In addition it outlines the IPSSI structure and the IPSSI approach to process improvement at the individual level.

Table 1 details the main objectives of the basic and advanced IPSSI courses under the two main headings of personal project management and personal quality management.

Table 1 - IPSSI course objectives

Course	**Personal Project Management**	**Personal Quality Management**
Basic	• Effort measurement • Size measurement • Size measures • Coding standard • Tracking • Estimation • Size • Effort	• Defects • Measurement • Classification • Density • Detection • Process framework • Process definition • Process improvement
Advanced	• Effort measurement • Size measurement • Counting standard • Productivity • Scheduling/Tracking • Earned value • Estimation • Size and effort • Testing techniques	• Defects measurement • Yield • Density • Defect detection • Defect prevention • Design review • Requirements • Process framework Process improvement

Both the basic and advanced courses contain a series of small programming tasks each of which is used illustrate a particular issue such as measurement, estimation or defects. Each developer gathers data about their programs which are inputted to the IPSSI tool.

4. Results to Date

The section that follows details the results of some preliminary IPSSI trials. These were staged in an academic environment and involved a group of graduate year Computing students. The trials had two objectives. Firstly, to enable the undergraduates to measure their own software process to effect improvement and secondly to provide some quantitative and qualitative feedback about the IPSSI philosophy and the training exercises.

Participation in the trials was voluntary and was therefore extra-curricular. As such students simultaneously had commitments to other subjects on their study programme. The IPSSI classes were held on one afternoon per week over a 5-week period and 1 exercise per week was distributed. 24 students participated at the outset, of which, 7 completed 4 of the 5 IPSSI exercises (a 29% completion rate) and 4 completed all 5 exercises (a 16.6% completion rate). During each of the 5 exercises participants are required to collect successively more detailed data as shown in Table 2.

By following the approach outlined above, participants adhere to the IPSSI model (Figure 1) by commencing with effort measures, then progressing to personal project management by relating effort to task size and finally focusing on quality management through understanding defects. Because of the relatively low level of completion little data emerges from the study about defects. Also because of the limited number of data points no firm conclusions can be drawn about the disciplines and the approaches. However, the results do provide some promising indications which merit further study.

Table 2 - IPSSI Data Gathering Requirement

Exercise	**1**	**2**	**3**	**4**	**5**
Activity	Estimate of time required	Estimate of time required	Estimate of size (LOC)	Estimate of size (LOC)	Estimate of size (LOC)
	Collect time by phase	Collect time by phase	Calculate productivity from programs 1 & 2	Calculate Productivity from Programs 1, 2 and 3	Calculate productivity from programs 1, 2, 3 and 4
		Estimate of size (LOC)	Estimate of time based on size estimate and productivity	Estimate of Time based on size estimate and productivity	Estimate of Time based on size estimate and productivity
		Measure of size on completion (LOC)	Collect time by phase	Collect time by phase	Collect time by phase
			Measure of size on completion (LOC)	Collect defect data by phase	Estimate defects by phase
				Carry out a code review	Collect defect data by phase
				Measure of size on completion (LOC)	Carry out a code review
					Measure of size on completion (LOC)

4.1 Time Distribution

From the first exercise, participants are required to keep track of the time they have taken to complete a particular programming task. A simple process is followed in the early exercises, Design, Code, Compile and Test and the study group recorded their time in each of these areas. Figure 2 illustrates the breakdown, by phase, of time spent by the group whilst completing the programming exercises.

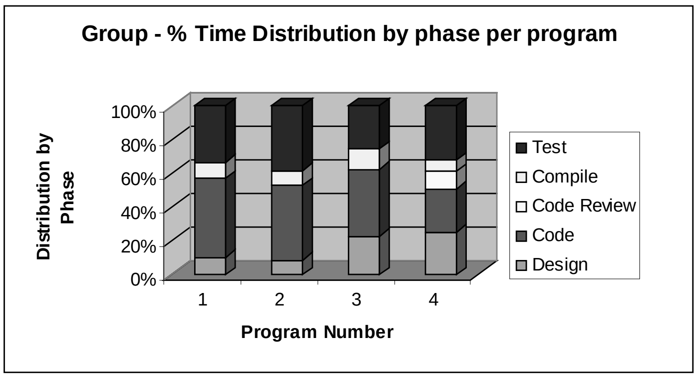

Figure 2 - Group Time Distribution

One of the training objectives was to convince the group of the need to spend more time in the earlier phases of development such as understanding requirements, detailed design and the use of code reviews. Many previous studies have shown how spending a greater proportion of time in the earlier life-cycle phases significantly reduces the amount of time required for testing as the product can be built 'right first time' and less rework and repair is required in test [4, 5, 6]. The indications from the results above are that this lesson was learned by the group, as in programs 1 and 2, 11% and 9% respectively of total time was spent in design whilst in programs 3 and 4, 23% and 25% respectively of total time was spent in design. However, the group figures above don't significantly indicate that the extra effort in design has yet translated into reduced testing effort. Figure 3 shows a student who had some success in reducing test time through increasing effort in the earlier development phases. This student managed to complete all 5 of the IPSSI exercises.

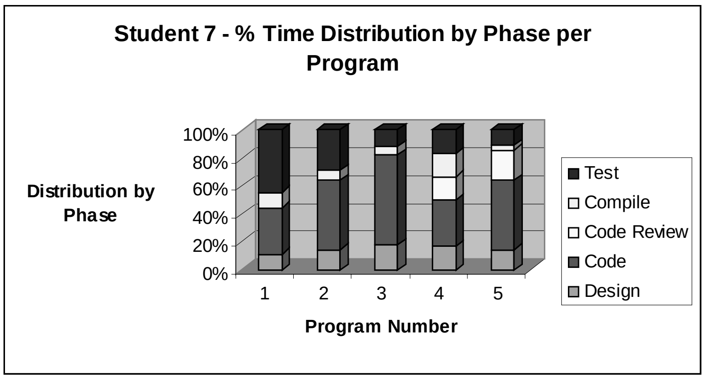

Figure 3 - Student 7 Time Distribution

In programs 1 - 3 this student steadily increased the time he spent in design from 11% in program 1 to 14% in program 2 to 18% in program 3. At the same time the proportion of time spent in test has reduced from 44% in program 1 through 29% in program 2 to 12% in program 3. Whilst program 4 shows the same proportions of time spent in design and test (16% in each) program 5 has the lowest figure for test of all (11%). In program 5 Student 7 spent 14% of his time in design and 21% of his time in code review.

Program 5 also involved the smallest time spent in compile (3.5% of total time). This coupled with the reduced time in test may indicate that the student has been successful at removing defects earlier in the development process.

4.2 Time Estimation

The exercises used for the trials were deliberately short in order to allow for the maximum amount of data to be collected during the training period. Because the exercises were quite small many of the estimating errors are quite large. Humphrey has also found that small tasks can generate significant percentage errors [4]. However, over time when the disciplines have been applied to much larger tasks and sufficient historical data has been gathered then the error percentages can be reduced. Figure 4 shows the time estimation error for the study group for programs 1 to 4.

Whilst the average time estimation error remains around +20% for programs 2 to 4, the range between the minimum and maximum time estimation error fluctuates significantly. This is to be expected with so few data points as individuals, following an overestimate, often overcompensate on subsequent tasks and vice-versa. An example of this can be seen in Figure 5 where Student 7's estimates show such fluctuations.

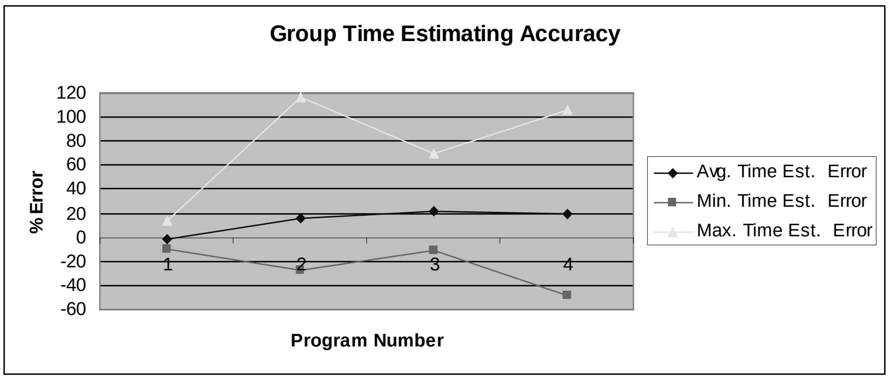

Figure 4 - Group Estimation Accuracy

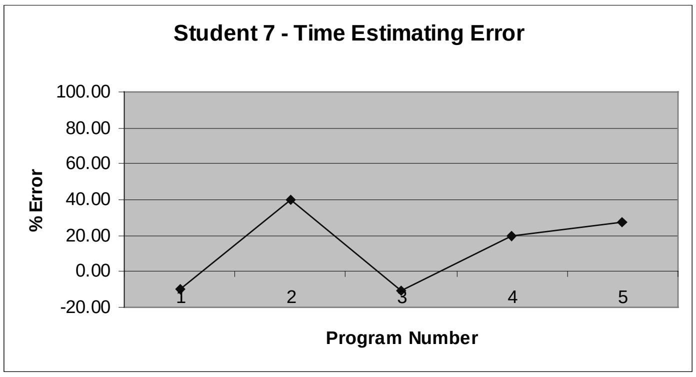

Figure 5 - Student 7 - Time Estimating Accuracy

4.3 Size Estimation

The same pattern of estimation error is also evident from data collected for size. Figure 6 shows the error in size estimates made by student 10. By contrast, Student 7 has very good control over size estimates with a 17% underestimate being the largest discrepancy.

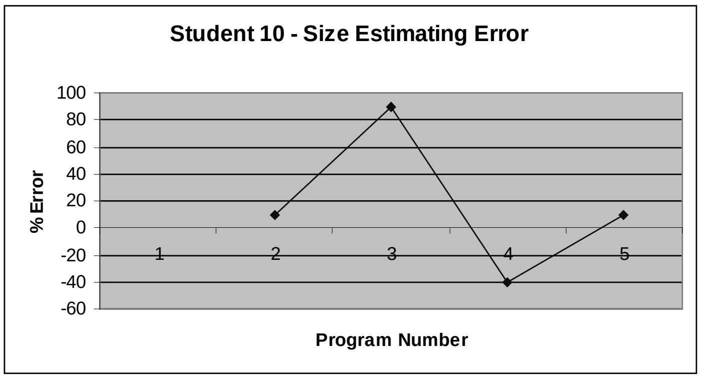

Figure 6 - Student 10 - Size Estimating Accuracy

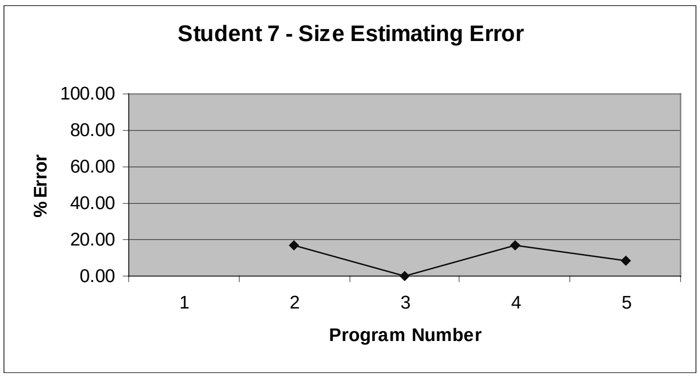

Figure 7 - Student 7 - Size Estimating Accuracy

5. The IPSSI Support Tool

The IPSSI tool was developed in order to support the individual developer using the IPSSI approach and is designed to support two main functions: Data Capture - simply and easy recording of data such as time, size and defects, and Data Analysis - automatic analysis of collected data to generate aggregate project data in textual and graphical form. There were also a number of constraints placed on the development of the tool:

- All data gathered and analysed during a process should be stored in a private database.
- The developer can optionally elect to anonymously submit gathered data to a central database.
- The tool should be platform independent.
- The developer (user) should be able to work in a stand-alone manner, i.e. no network connection required.

The architecture of the tool is shown in figure 8. The user interface, which is implemented in a web browser, provides facilities for data gathering, analysis and submission to a local database. Upon completion of a project, the user may choose to anonymously submit their data to a central (IPSSI project or company specific) database where aggregate data can be further collected and analysed.

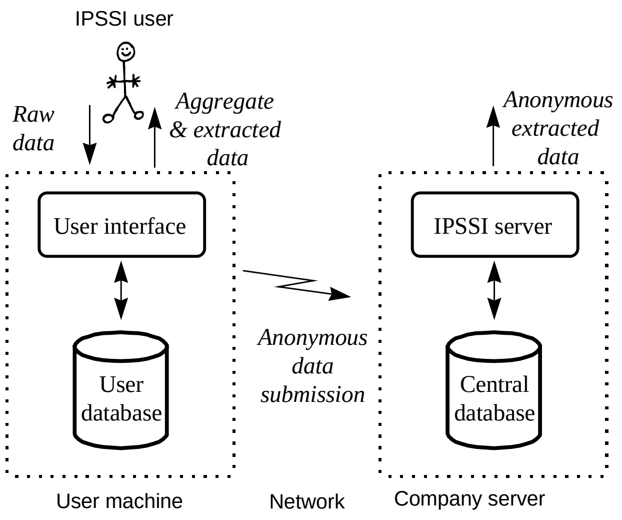

Figure 8 - IPSSI tool architecture

The IPSSI model follows the three elements of personal software engineering; defining a personal process, personal project management and personal quality management. These are represented in the tool by three main levels:

1. Recording basic data such as: size, effort and defects.
2. Advanced review techniques and statistical analysis of derived measures including estimated size, real size and defects found in review.
3. Expert topics such as design methods, earned value tracking, design errors found, schedule and task planning, productivity and design quality.

Currently the initial implementation of the IPSSI tool has been completed and it is being used to gather and analyse data by participants in the pilot IPSSI courses.

6. Future Work

The benefits, to the developer, of using IPSSI are ultimately self-convincing. The results above require supplementary exercises to allow more data to be collected and subsequently a clearer pattern of the developer's process. As IPSSI is aimed at industrial practitioners, these exercises could be real world projects. Ultimately, it is through applying IPSSI disciplines in their own daily work that developers can become convinced of its benefits.

Improving the quality of software products is a very important goal for both companies and developers. The trials show that if all of the exercises as currently designed are not completed then there is very little data on defects and the potential advantages of code reviews. It may be necessary either, to increase the number of exercises or introduce code reviews into earlier exercises. Either way this would provide more defect data and more feedback on the application of the reviews.

At present, within the exercises participants are asked in the later exercises to estimate the time required to complete an exercise based on the their estimate of the size of the program and their historical productivity. Whilst this is useful in highlighting the relationship between size, productivity and effort, no allowance is made for the complexity of the task. The advanced IPSSI training programme will cater for this.

There are two factors, however, which will permit a more effective evaluation of IPSSI. A 'Basic IPSSI' course, which covers the fundamental practices and is staged over 2 consecutive days, is planned. This course is aimed at industrial practitioners and it is expected that motivation and completion rates will be significantly higher than the voluntary trials which have taken place to date.

Secondly, if IPSSI is to succeed, it must be applied in an industrial context. The training and the disciplines are designed to allow individuals and companies achieve meaningful software process improvement. The ultimate benefits should accrue when the skills acquired in training are applied by the practitioners in their workplace and it is this which will truly test the value of IPSSI.

7. Conclusions

As stated previously, the objectives of the IPSSI trials were twofold. Firstly, to teach the undergraduates the benefits of having a defined and measurable software process and secondly to get some feedback on the IPSSI work carried out to date. Both of these objectives have been met. All of the graduates commented favourably on the approaches taken. Comments ranged from "It's something I'd never thought of before", to "Prior to this I had no knowledge of where I was spending my time" to "Up to now I have been a 'hacker' who started coding as early as possible. I have now started to concentrate more of my effort in design and I can already see the benefits". Whilst the results from the study may not readily show this, the study group are now adopting a much more professional approach to developing software and have been convinced of the benefits of following a defined process. The trials also provided the necessary feedback on the IPSSI training material. This will feed into the future work outlined above. The next stage is to trial the IPSSI methods in a formal training course attended by software practitioners. This will be held in the coming months.

Acknowledgements

This research is partially supported by the European Commission as ESSI project 27453. The authors gratefully acknowledge the value gained from discussions regarding this work with their research partners: Dublin City University, European Software Institute, IVF, Politecnico di Torino and Universite Thomson. We would also like to thank the final year ('99-'00) BSc. students from Dundalk Institute of Technology who participated in the trials.

8. References

[1] "The SPIRE Handbook", Centre for Software Engineering, 1998.
[2] Y.Wang, H.Duncan, M.Kartinnen, H.Sjostrom and P.Kokeritz, "IPSSI - A European Methodology on PSP", Proceedings EuroSPI-99, Finland, 1999.
[3] P. O'Beirne & J. Sanders, "Personal Software Process: Does the PSP Deliver its promise?", Proceedings of Inspire '97, Gothenburg, Sweden, 1997, pp. 252-265.
[4] W.Humphrey, "A Discipline for Software Engineering", Addison Wesley, 1995.
[5] R. B. Grady, "Successful Software Process Improvement", Prentice Hall, 1997.
[6] J.D. Blackburn, G. D. Scudder and L.Van Wassenhove, "Improving the Speed and Productivity of Software Development: A Global Survey of Software Developers", IEEE Transactions on Software Engineering, Vol. 22, No. 12, December 1996, pp. 875-885.