AC 2007-1432: TOOL SUPPORT FOR SOFTWARE PROCESS DATA MANAGEMENT IN SOFTWARE ENGINEERING EDUCATION AND INDUSTRY TRAINING

Mark Sebern, Milwaukee School of Engineering

MARK J. SEBERN is a Professor in the Electrical Engineering and Computer Science Department at the Milwaukee School of Engineering (MSOE), and was the founding program director for MSOE's undergraduate software engineering program. He has served as an ABET program evaluator for software engineering and computer engineering.

Mark Hornick, Milwaukee School of Engineering

MARK L. HORNICK has been an Assistant Professor in the Electrical Engineering and Computer Science Department at the Milwaukee School of Engineering (MSOE) since 2004. Prior to that, he worked in private industry for 20 years, implementing and managing the incorporation of standard practices and processes into the development of industrial software products.

Tool Support for Software Process Data Management in Software Engineering Education and Industry Training

Mark J. Sebern, PhD, PE
Milwaukee School of Engineering
sebern@msoe.edu

Mark L. Hornick, PhD
Milwaukee School of Engineering
hornick@msoe.edu

www.msoe.edu/se/

Abstract

Data management tools are necessary for effective support of disciplined software processes that make use of historical data for planning and process improvement. This paper reports describes two such tools and how they have been applied in an undergraduate software engineering program and to support software process improvement initiatives in industry. One of the tools, an open-source development project, has recently added new capabilities that may make it an attractive choice for both educators and practitioners.

Introduction

Software engineering programs generally incorporate courses and other learning experiences that are designed to provide breadth and depth of coverage across the discipline, addressing both practice and process. Software engineering practice deals with what software engineers do, and includes topics such as requirements analysis and specification, architecture and design, verification, and implementation. Software engineering process is concerned with how software engineers work, embracing subjects that include planning, team functioning, quality management, continuous improvement, and integration of development teams, management, and functional groups in an organization.

A variety of capable and cost-effective tools have been available to support software engineering practice, such as integrated development environments, modeling tools, and testing frameworks. Until recently, however, similar high-quality tool support for software engineering process implementation has been lacking. This has been especially true for processes that include a focus on process and product metrics, such as the Personal Software Process (PSP) and the Team Software Process (TSP) developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. Reliable and complete measurement data is critical to effective planning and process improvement, but data gathering and analysis that is burdensome and labor intensive will simply not be done.

The SEI has developed several tools to support the PSP and TSP, but the availability of these tools has often been limited to the SEI's commercial partners. Although some PSP tools are available for academic use[10], the SEI's definition of "academic" generally excludes industry training, use in on-line courses or off-campus programs, or with students not enrolled in a degree program. For this reason, some educators who serve multiple constituencies have been reluctant to adopt these tools.

A number of other educators and researchers have surveyed available process data management support tools, or have proposed alternative methods of gathering and analyzing process data[12,16,22], often with the goal of making data gathering completely invisible to the individual software developer. These surveys generally fail to identify a satisfactory tool, while the automated methods seem to encourage the use of data that is easy to gather.

This paper focuses on the design and application of two data management tools that support software project planning and tracking, based on defined measurement frameworks. One of these tools, named LEIA (Laboratory Engineering Information Archive) was developed by undergraduate software engineering students at the Milwaukee School of Engineering (MSOE), and is used to manage team projects in MSOE's Software Development Laboratory. The second such tool is the open-source Process Dashboard (processdash.sourceforge.net) developed by David Tuma and his colleagues, Recently, this latter tool has been extended to better support development teams and customized process definitions. As a result, the Process Dashboard may well become a de facto standard for software process data management support.

Metrics, Process Design, and Support Tool Functionality

In designing or applying data management tools to a chosen software process, one of the first steps is to choose appropriate product and process measures. Watts Humphrey and his SEI colleagues, while defining the PSP and TSP processes, developed a set of four base measures, from which many derived measures can be calculated (Table 1).

Base Measures	Derived Measures
Product size (e.g., LOC)	Productivity
• Per part or component	Yield
• Plan and actual	Total defect density
Effort (time in minutes or hours)	Test defect density
• Per part, per process phase	Planned value (PV)
• Plan and actual	Earned value (EV)
Quality (number/type of defects)	Test time percentage
• Injected/removed phase	Review rate (e.g., LOC/hour)
• Find-and-fix time	Defect removal rate
• Plan and actual	Total product size
Schedule (task/phase completion date)	Size estimating error
• Plan and actual	(etc.)

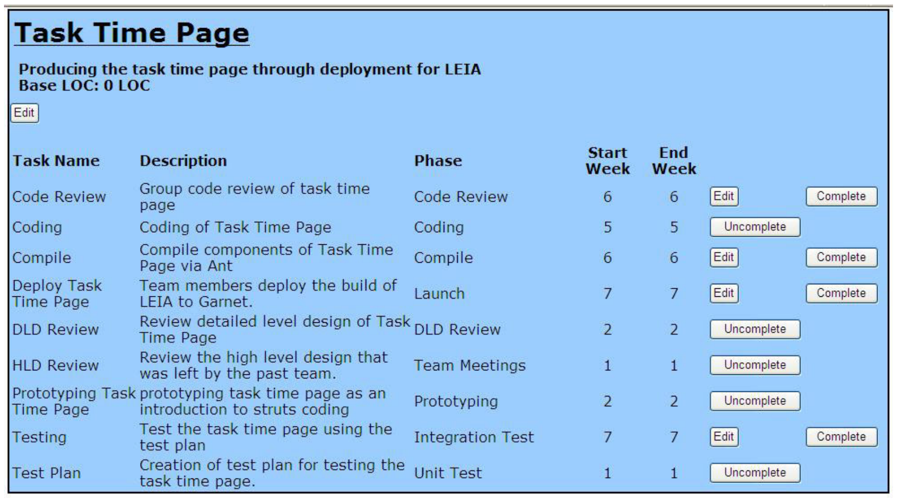
Table 1 PSP/TSP Measurement Framework

While not completely original or unique, this measurement framework neatly describes a small and consistent data set from which a great deal of useful information can be extracted. The data thus gathered is most useful in the context of a defined process[6,10] that identifies the specific steps (e.g., design, design review, coding, peer inspection) that are to be followed in developing a product. With a stable process, historical data can be used to plan and track future projects.

LEIA: Laboratory Engineering Information Archive

The LEIA (Laboratory Engineering Information Archive) process support tool has been developed by a series of student teams in MSOE's Software Development Laboratory.[20] LEIA is a web-based application build on the Apache web server, the Tomcat servlet container, and the Struts application framework. Persistent storage is provided by a PostgreSQL database server. Coding is done in Java, JavaScript, and CSS (Cascading Style Sheets).

While various commercially available products such as Microsoft Project can be used for task and schedule planning and management, such generalized tools typically offer no support for software quality management. By contrast, LEIA provides time and defect logging, task and schedule planning, and reporting of plan and quality metrics. Projects are subdivided into development cycles of configurable length. The work breakdown structure (WBS) model defines a number of components to be delivered in a particular cycle. Each component has a set of tasks that are required to complete it, as shown in Figure 1.



Task Time Page

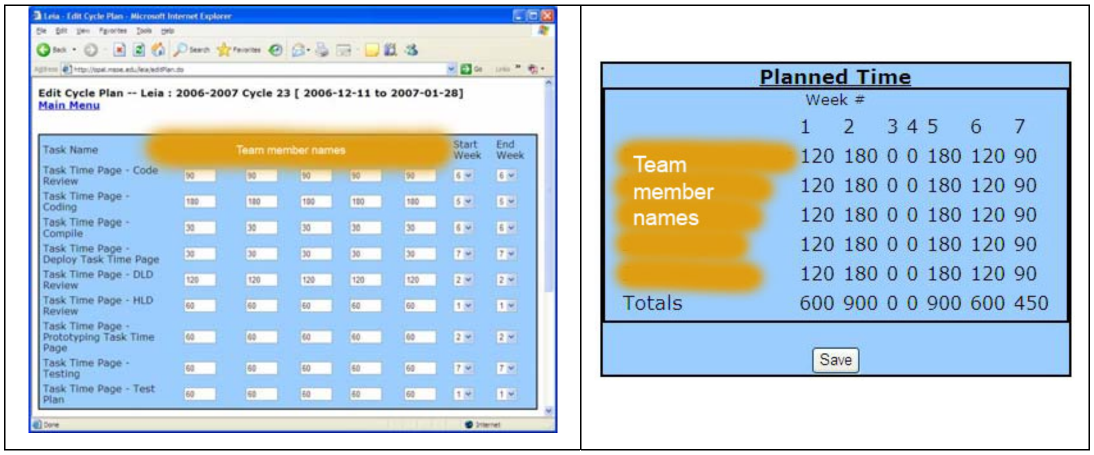
Producing the task time page through deployment for LEIA
Base LOC: 0 LOC

Edit

Task Name	Description	Phase	Start Week	End Week		
Code Review	Group code review of task time page	Code Review	6	6	Edit	Complete
Coding	Coding of Task Time Page	Coding	5	5	Uncomplete	
Compile	Compile components of Task Time Page via Ant	Compile	6	6	Edit	Complete
Deploy Task Time Page	Team members deploy the build of LEIA to Garnet.	Launch	7	7	Edit	Complete
DLD Review	Review detailed level design of Task Time Page	DLD Review	2	2	Uncomplete	
HLD Review	Review the high level design that was left by the past team.	Team Meetings	1	1	Uncomplete	
Prototyping Task Time Page	prototyping task time page as an introduction to struts coding	Prototyping	2	2	Uncomplete	
Testing	Test the task time page using the test plan	Integration Test	7	7	Edit	Complete
Test Plan	Creation of test plan for testing the task time page.	Unit Test	1	1	Uncomplete	

Figure 1. LEIA Task Breakdown for Component "Task Time Page"

Each LEIA task can be assigned to one or more students, with time estimates (in minutes) for each assigned team member, as shown in Figure 2. Each task is then assigned a starting and ending week, relative to the current development cycle; this permits the system to generate a workload summary by week and team member, to facilitate load balancing within the team and across the cycle. The development schedule can take into account external dependencies; in Figure 2, for example, weeks 3 and 4 correspond to a holiday break period when no work is planned (though some may actually be done).


Leia - Edit Cycle Plan - Microsoft Internet Explorer
http://opal.mase.edu/leia/editPlan.do
Done
Internet

Edit Cycle Plan -- Leia : 2006-2007 Cycle 23 [2006-12-11 to 2007-01-28]
Main Menu

Task Name	Team member names					Start Week	End Week
Task Time Page - Code Review	90	90	90	90	90	6	6
Task Time Page - Coding	180	180	180	180	180	5	5
Task Time Page - Compile	30	30	30	30	30	6	6
Task Time Page - Deploy Task Time Page	30	30	30	30	30	7	7
Task Time Page - DLD Review	120	120	120	120	120	2	2
Task Time Page - HLD Review	60	60	60	60	60	1	1
Task Time Page - Prototyping Task Time Page	60	60	60	60	60	2	2
Task Time Page - Testing	60	60	60	60	60	7	7
Task Time Page - Test Plan	60	60	60	60	60	1	1

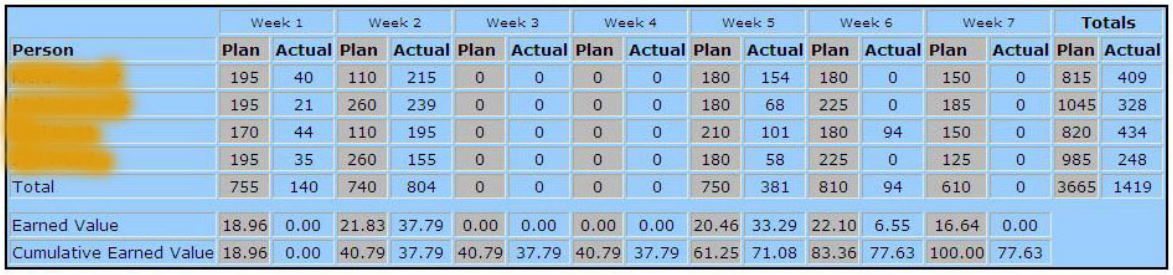
Planned Time

	Week #						
	1	2	3	4	5	6	7
Team member names	120	180	0	0	180	120	90
	120	180	0	0	180	120	90
	120	180	0	0	180	120	90
	120	180	0	0	180	120	90
	120	180	0	0	180	120	90
Totals	600	900	0	0	900	600	450

Save

Figure 2. LEIA Schedule Plan

LEIA supports tracking of team and individual progress, as shown in the effort report of Figure 3. The time values reflect only "task time", not total time spent on the project; new teams (like this one) are often too optimistic in their initial planning. This snapshot was taken in the middle of week 6, so data for that week is not complete.



	Week 1		Week 2		Week 3		Week 4		Week 5		Week 6		Week 7		Totals	
Person	Plan	Actual	Plan	Actual	Plan	Actual	Plan	Actual	Plan	Actual	Plan	Actual	Plan	Actual	Plan	Actual
[redacted]	195	40	110	215	0	0	0	0	180	154	180	0	150	0	815	409
[redacted]	195	21	260	239	0	0	0	0	180	68	225	0	185	0	1045	328
[redacted]	170	44	110	195	0	0	0	0	210	101	180	94	150	0	820	434
[redacted]	195	35	260	155	0	0	0	0	180	58	225	0	125	0	985	248
Total	755	140	740	804	0	0	0	0	750	381	810	94	610	0	3665	1419
Earned Value	18.96	0.00	21.83	37.79	0.00	0.00	0.00	0.00	20.46	33.29	22.10	6.55	16.64	0.00		
Cumulative Earned Value	18.96	0.00	40.79	37.79	40.79	37.79	40.79	37.79	61.25	71.08	83.36	77.63	100.00	77.63		

Another way to visualize team progress is with a LEIA earned-value plot, as shown in Figure 4. As tasks are completed, the team earns their planned value; the plot shows the expected and actual progress toward project completion. When the "earned" line is above the "planned" line, it means that the team is ahead of schedule. The cumulative earned value is updated daily, while the planned value is based on the schedule granularity, which is one week.

Figure 3. LEIA Effort Report (time in minutes)

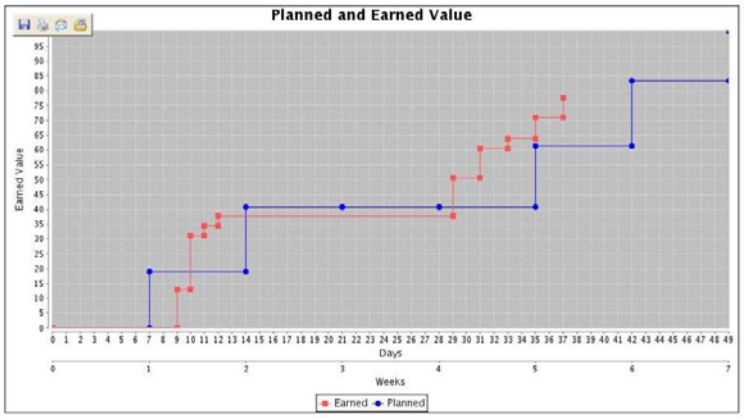


Figure 4. LEIA Earned Value Plot

The availability of up-to-date tracking data helps the team to know when they are in trouble, and to take corrective action such as increasing task time or replanning project work.

<u>Process Dashboard</u>

As an open-source project, the Process Dashboard has the potential to gain widespread use and acceptance. The availability of source code has facilitated its customization, but limited documentation and a difficult learning curve have been a hindrance. Some past intellectual property concerns have also cast a shadow, but recent architectural changes have made it easier to separate out proprietary materials so that more general use and adoption is practical. Perhaps significantly, the Process Dashboard seems to be an ideal candidate for adoption and enhancement by software engineering educators and students.

The architecture of the Process Dashboard is very different from that of the LEIA tool described above. It is a Java application, with basic parts of the user interface implemented in Java. However, it is also a web server, and many forms and reports are rendered in a local browser. Its networking capabilities also support links between individual and team data repositories.

The Process Dashboard's main application window is shown in Figure 1. This small "toolbar" is intended to be unobtrusive on the desktop, and incorporates a running "stopwatch" timer, a link to forms and reports, and a navigation mechanism for the work breakdown structure. The "C" menu provides access to additional tools and dialogs.

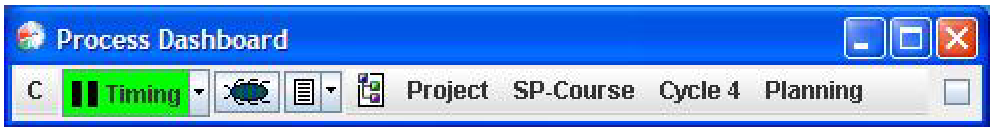
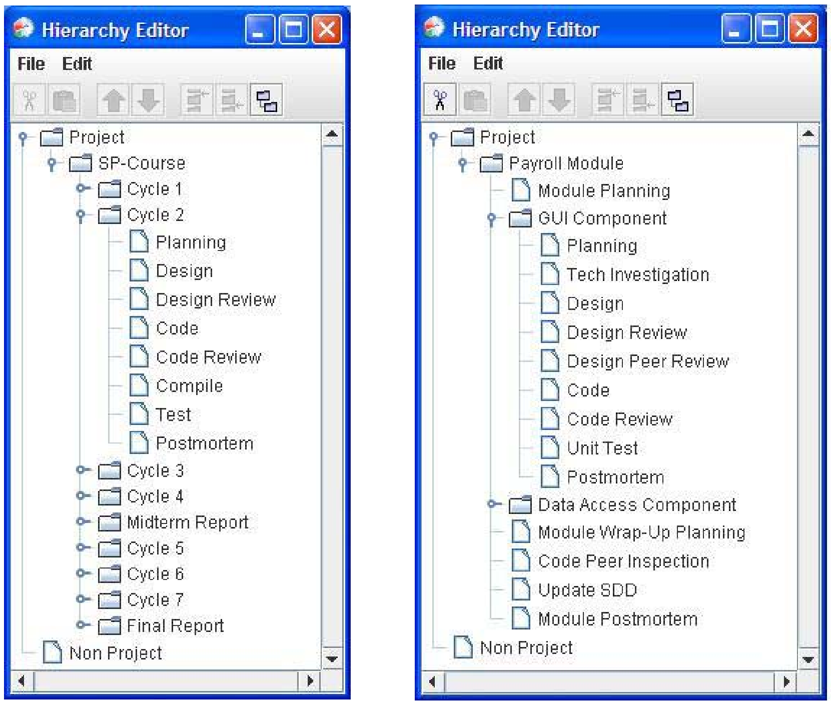


Figure 5. Process Dashboard Main Bar

When used by individual developers, the work breakdown structure is defined in a hierarchy editor, as shown in Figure 6. The hierarchy items can be simple nodes or instances of previously defined processes. The first hierarchy in Figure 6 corresponds to a 10-week academic course based on the PSP, while the second one represents a nested process definition used by one of MSOE's industry partners. In the latter case, the "module" process supports any number of "components"; each component has a set of process phases, while additional phases are part of the enclosing module. Data from the individual components is consolidated at the module level.


```
Project
    SP-Course
        Cycle 1
        Cycle 2
            Planning
            Design
            Design Review
            Code
            Code Review
            Compile
            Test
            Postmortem
        Cycle 3
        Cycle 4
        Midterm Report
        Cycle 5
        Cycle 6
        Cycle 7
        Final Report
Non Project
```

```
Project
    Payroll Module
        Module Planning
        GUI Component
            Planning
            Tech Investigation
            Design
            Design Review
            Design Peer Review
            Code
            Code Review
            Unit Test
            Postmortem
        Data Access Component
        Module Wrap-Up Planning
        Code Peer Inspection
        Update SDD
        Module Postmortem
Non Project
```

Figure 6. Examples of Process Dashboard Work Breakdown Structure

The Process Dashboard defect logging dialog is shown in Figure 7. It incorporates its own timer which can be used to automatically record the "find and fix" time for the current defect. The "fix defect" capability provides correct time accounting when a new defect is injected while fixing an existing one. With properly designed processes, defect injection and removal can be tracked across subprocesses (e.g., from one component in a module to another), though this may not be required in introductory software process courses.

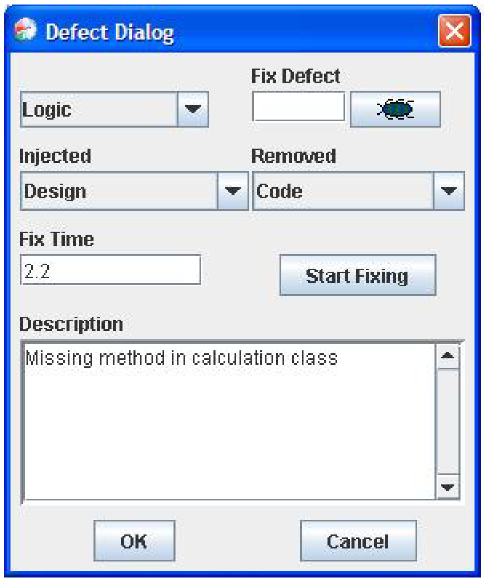
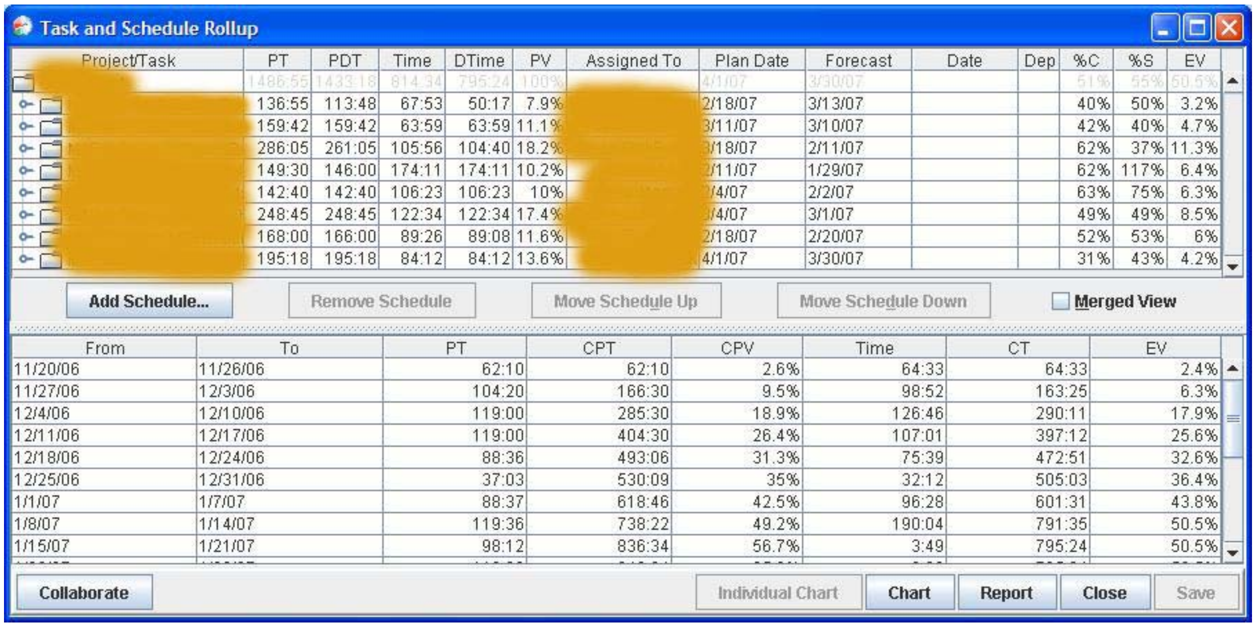


Figure 7. Process Dashboard Defect Log Dialog Box

The Process Dashboard is also capable of schedule planning and tracking, at the individual and team level. Figure 8 shows a schedule "roll up" that consolidates individual schedule data from individual team members. It supports tracking of task completion and earned value, and can generate planned calendar task completion dates from supplied data on the number of available task hours per week.



Project/Task	PT	PDT	Time	DTime	PV	Assigned To	Plan Date	Forecast	Date	Dep	%C	%S	EV
[redacted]	1486:55	1433:18	814:34	795:24	100%		4/1/07	3/30/07			51%	55%	50.5%
[redacted]	136:55	113:48	67:53	50:17	7.9%	[redacted]	2/18/07	3/13/07			40%	50%	3.2%
[redacted]	159:42	159:42	63:59	63:59	11.1%	[redacted]	3/11/07	3/10/07			42%	40%	4.7%
[redacted]	286:05	261:05	105:56	104:40	18.2%	[redacted]	3/18/07	2/11/07			62%	37%	11.3%
[redacted]	149:30	146:00	174:11	174:11	10.2%	[redacted]	2/11/07	1/29/07			62%	117%	6.4%
[redacted]	142:40	142:40	106:23	106:23	10%	[redacted]	/4/07	2/2/07			63%	75%	6.3%
[redacted]	248:45	248:45	122:34	122:34	17.4%	[redacted]	/4/07	3/1/07			49%	49%	8.5%
[redacted]	168:00	166:00	89:26	89:08	11.6%	[redacted]	2/18/07	2/20/07			52%	53%	6%
[redacted]	195:18	195:18	84:12	84:12	13.6%	[redacted]	4/1/07	3/30/07			31%	43%	4.2%

From	To	PT	CPT	CPV	Time	CT	EV
11/20/06	11/26/06	62:10	62:10	2.6%	64:33	64:33	2.4%
11/27/06	12/3/06	104:20	166:30	9.5%	98:52	163:25	6.3%
12/4/06	12/10/06	119:00	285:30	18.9%	126:46	290:11	17.9%
12/11/06	12/17/06	119:00	404:30	26.4%	107:01	397:12	25.6%
12/18/06	12/24/06	88:36	493:06	31.3%	75:39	472:51	32.6%
12/25/06	12/31/06	37:03	530:09	35%	32:12	505:03	36.4%
1/1/07	1/7/07	88:37	618:46	42.5%	96:28	601:31	43.8%
1/8/07	1/14/07	119:36	738:22	49.2%	190:04	791:35	50.5%
1/15/07	1/21/07	98:12	836:34	56.7%	3:49	795:24	50.5%

Figure 8. Process Dashboard Task and Schedule Dialog

The schedule data can also be presented in graphical form, as shown in Figure 9. The "plan" and "actual" lines illustrate current performance. The forecast is generated using Monte Carlo methods, and attempts to define the likely range of future performance based on to-date values of actual task hours and productivity for completed tasks. The optimized forecast is made assuming perfect workload balancing (all team members finish their assigned tasks at the same time).

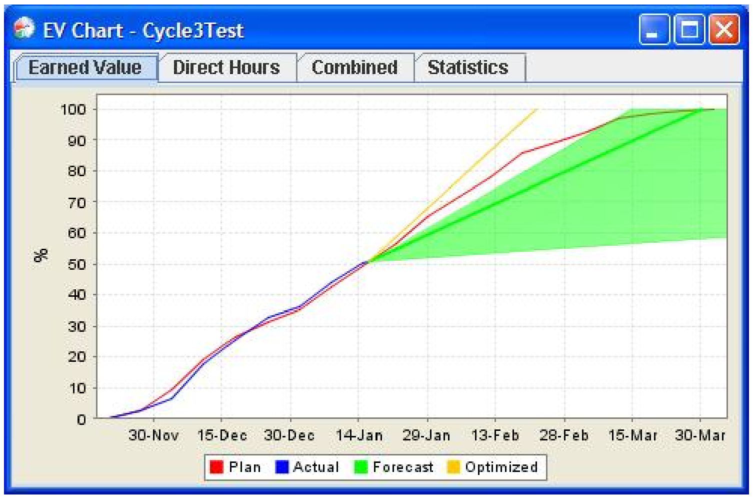
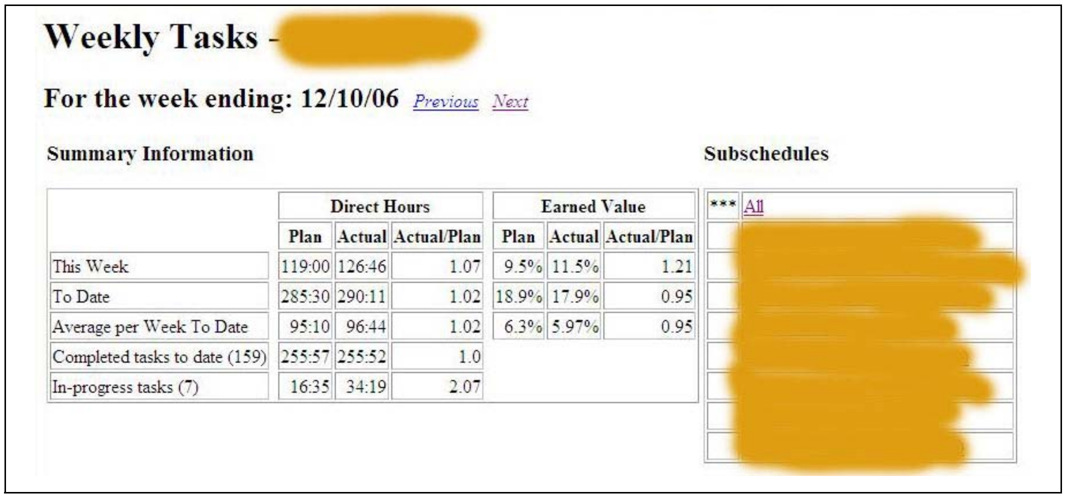


Figure 9. Process Dashboard Earned Value Chart

Another powerful Process Dashboard tool is the weekly report. This report, part of which is shown in Figure 10, provides a capsule summary of individual and team progress to date, or for any past week. Commonly, this data is displayed during team meetings, while individual team members are reporting on their own work. It has proven very useful in detecting obstacles and workload imbalances, so that they can be effectively dealt with by the team. For example, one team adopted a policy that a variance in earned value greater than 10% would be actively addressed before the end of the team meeting.



Weekly Tasks - [redacted]

For the week ending: 12/10/06 Previous Next

Summary Information

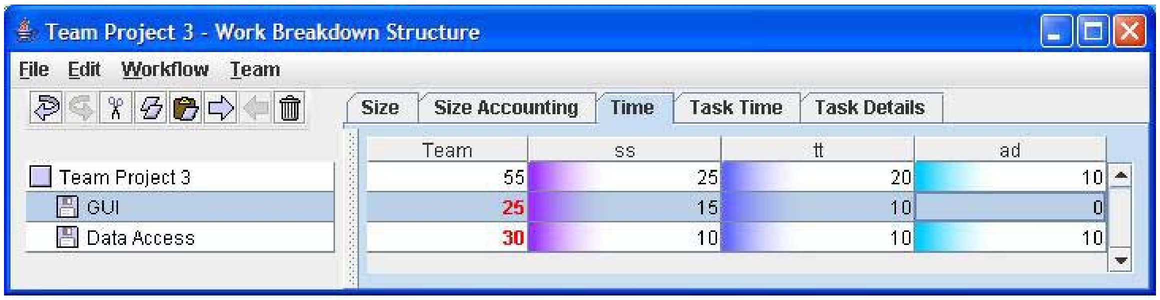
	Direct Hours			Earned Value		
	Plan	Actual	Actual/Plan	Plan	Actual	Actual/Plan
This Week	119:00	126:46	1.07	9.5%	11.5%	1.21
To Date	285:30	290:11	1.02	18.9%	17.9%	0.95
Average per Week To Date	95:10	96:44	1.02	6.3%	5.97%	0.95
Completed tasks to date (159)	255:57	255:52	1.0			
In-progress tasks (7)	16:35	34:19	2.07			

Subschedules

***	All
	[redacted]
	[redacted]
	[redacted]
	[redacted]
	[redacted]
	[redacted]
	[redacted]
	[redacted]

Figure 10. Process Dashboard Weekly Status Summary (partial)

Recently, enhanced team support has been added to the Process Dashboard. This capability supports a team launch process similar to that specified in the TSP, using planning mechanisms like those shown in Figure 11. It provides two-way transfer of work breakdown structure and process data between the team and team member instances of the Process Dashboard program.



	Team	ss	tt	ad
Team Project 3	55	25	20	10
GUI	25	15	10	0
Data Access	30	10	10	10

Figure 11. Process Dashboard Team Work Breakdown Dialog

Tool Application in Software Engineering Education

At MSOE, the LEIA data management tool has been used for several years in the Software Development Laboratory, a year-long experience for all software engineering students. Student teams work on ongoing, large-scale projects for real clients, using a process originally based on the TSP. As LEIA's capabilities have grown, it has become more useful to the student teams. The web interface and central database support collaboration within and across teams; this can be important since an individual student may be a member of both a development team and a staff team (e.g., the software engineering process group). It has also been common for software engineering students to use LEIA for their senior design projects. Since these projects are often multidisciplinary, students from other majors have also learned to use LEIA successfully.

For the last two years, the Process Dashboard has been used in SE-280, a sophomore course that introduces software engineering process. MSOE faculty members have customized the process templates and simplified reporting and assignment submission. Some of these changes have been submitted for integration into the main Process Dashboard code base. While there is a significant learning curve associated with the creation and customization of processes and tool functions, the power and capabilities of the tool have proved to be very impressive. With the advent of new team capabilities, the Process Dashboard may come to play a part in the Software Development Laboratory, integrating into the LEIA framework or perhaps even supplanting some of the latter tool's functions.

Application to Industry Training

MSOE's Business Excellence Consortium (BEC) provides targeted training for industry partners in a variety of areas, including software process improvement. Using concepts and techniques from published books[6,7,9,8,10] on the PSP and TSP, and incorporating other process improvement methods, these efforts assist software development organizations that seek to improve predictability and quality. The Process Dashboard has been very effective in supporting these efforts, particularly in creating customized processes that integrate with existing organizational standards and practices. To date, simple data export and consolidation has been employed to link team member data to the overall team aggregate. Ongoing experimentation with and adaptation of the new team support features is expected to provide a basis for even more effective use of the Process Dashboard with industry teams.

Conclusion

Data gathering and analysis, in a timely manner and with an acceptably low level of effort, is a key success factor in the application of disciplined software processes that rely on correct and complete data. In the past, it was often necessary for individual software engineering educators and practitioners to develop their own tools. With the advent of highly functional and widely available tools like the Process Dashboard, the opportunity may exist to place more emphasis on common efforts and joint development.

Acknowledgment

Major support for the MSOE Software Development Laboratory has been provided by Johnson Controls, Inc. This assistance, as well as continuing support from Johnson Controls management and staff, has made a significant contribution to the success of the software engineering program.

References

1. CMMI Product Team, *Capability Maturity Model Integration (CMMI), Version 1.1: CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SE/IPPD/SS, V1.1), Continuous Representation*, Technical Report CMU/SEI-2002-TR-011, Software Engineering Institute, 2002, http://www.sei.cmu.edu/publications/.
2. N. Davis and J. Mullaney, *The Team Software Process (TSP) in Practice: A Summary of Recent Results*, Technical Report CMU/SEI-2003-TR-014, Software Engineering Institute, 2003, http://www.sei.cmu.edu/publications/.
3. W. Hays and J. Over, *The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers*, Technical Report CMU/SEI-97-TR-001, Software Engineering Institute, 1997, http://www.sei.cmu.edu/publications/.
4. T. Hilburn, "Software engineering - from the beginning", *Proceedings of the 9th Conference on Software Engineering Education* (CSEE'96), April 1996.
5. T. Hilburn, "Teams need a process!", *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE 2000), July 2000.
6. W. Humphrey, *A Discipline for Software Engineering*, Addison-Wesley, 1995.
7. W. Humphrey, *Introduction to the Team Software Process*, Addison-Wesley, 2000.
8. W. Humphrey, *The Team Software Process (TSP)*, Technical Report CMU/SEI-2000-TR-023, Software Engineering Institute, 2000, http://www.sei.cmu.edu/publications/.
9. W. Humphrey, *Winning With Software: An Executive Strategy*, Addison-Wesley, 2002.
10. W. Humphrey, *PSP - A Self-Improvement Process for Software Engineers*, Addison-Wesley, 2005.
11. W. Humphrey, Software Engineering Institute, personal communication.
12. P. Johnson, et al, HackyStat Research Summary, http://csdl.ics.hawaii.edu/Research/Hackystat/.
13. Joint Task Force on Computing Curricula, IEEE Computer Society and Association for Computing Machinery, *Computing Curriculum — Software Engineering*, July 2004; http://sites.computer.org/ccse/.
14. C. Jones, *Software Assessments, Benchmarks, and Best Practices*, Addison-Wesley, 2000.
15. Process Dashboard (software process support tool), http://processdash.sourceforge.net/.
16. M. Postema, et al, "Tool Support for Teaching the Personal Software Process", *Computer Science Education*, Vol. 10, August 2000.
17. L. Pracchia, "The AV-8B team learns synergy of EVM and TSP accelerates software process improvement", *Crosstalk:The Journal of Defense Software Engineering*, January, 2004.
18. P. Runeson, "Experiences from teaching PSP for freshmen", *Proceedings of the 14th Conference on Software Engineering Education and Training* (CSEE&T'01), February 2001.
19. M. Sebern, "Iterative development and commercial tools in an undergraduate software engineering course", *Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education*, ACM Press, 1997.
20. M. Sebern, "The software development laboratory: incorporating industrial practice in an academic environment", *Proceedings of the 15th Conference on Software Engineering Education and Training* (CSEE&T'02), February 2002.
21. M. Sebern and T. Hilburn, "Integrating software engineering process in an undergraduate curriculum", *Proceedings of the 18th Conference on Software Engineering Education and Training* (CSEE&T'05), April 2005.
22. A. Sillitti, et al, "Collecting, integrating and analyzing software metrics and Personal Software Process data", *Proceedings of the 29th EUROMICRO Conference*, IEEE, 2003.

23. Software Engineering Institute, PSP academic materials, http://www.sei.cmu.edu/tsp/psp/download/academic.html.
24. D. Suri and M. Sebern, “Incorporating software process in an undergraduate software engineering curriculum: challenges and rewards”, *Proceedings of the 17th Conference on Software Engineering Education and Training* (CSEE&T'04), March 2004.
25. D.A. Umphress and J.A. Hamilton, Jr., “Software process as a foundation for teaching, learning, and accrediting”, *Proceedings of the 15th Conference on Software Engineering Education and Training* (CSEE&T'02), February 2002.

Biography

MARK J. SEBERN is a Professor in the Electrical Engineering and Computer Science Department at the Milwaukee School of Engineering (MSOE), and was the founding program director for MSOE's undergraduate software engineering program. He has served as an ABET program evaluator for software engineering and computer engineering.

MARK L. HORNICK has been an Assistant Professor in the Electrical Engineering and Computer Science Department at the Milwaukee School of Engineering (MSOE) since 2004. Prior to that, he worked in private industry for 20 years, implementing and managing the incorporation of standard practices and processes into the development of industrial software products.