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.

© American Society for Engineering Education, 2007
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\(^1\), 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\(^\text{12,16,22}\), 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).

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

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.

![Figure 1. LEIA Task Breakdown for Component “Task Time Page”](image1)

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).

![Figure 2. LEIA Schedule Plan](image2)

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.

![Figure 3. LEIA Effort Report](image3)
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)

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.

Process Dashboard

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

<table>
<thead>
<tr>
<th>Person</th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Plan</td>
<td>Actual</td>
<td>Plan</td>
<td>Actual</td>
<td>Plan</td>
<td>Actual</td>
<td>Plan</td>
<td>Actual</td>
</tr>
<tr>
<td></td>
<td>105</td>
<td>40</td>
<td>215</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>195</td>
<td>21</td>
<td>260</td>
<td>239</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>44</td>
<td>110</td>
<td>195</td>
<td>0</td>
<td>0</td>
<td>210</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>105</td>
<td>35</td>
<td>260</td>
<td>188</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>88</td>
</tr>
<tr>
<td>Total</td>
<td>755</td>
<td>100</td>
<td>740</td>
<td>840</td>
<td>0</td>
<td>0</td>
<td>750</td>
<td>381</td>
</tr>
<tr>
<td>Earned</td>
<td>18.95</td>
<td>0.00</td>
<td>21.83</td>
<td>37.79</td>
<td>0.00</td>
<td>0.00</td>
<td>20.46</td>
<td>33.29</td>
</tr>
<tr>
<td>Cumulative Earned</td>
<td>15.95</td>
<td>0.00</td>
<td>40.79</td>
<td>37.79</td>
<td>40.79</td>
<td>37.79</td>
<td>40.79</td>
<td>37.79</td>
</tr>
</tbody>
</table>
to forms and reports, and a navigation mechanism for the work breakdown structure. The “C” menu provides access to additional tools and dialogs.

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.

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.
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.

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).
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.

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.
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\(^6,7,9,10\) 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

11. W. Humphrey, Software Engineering Institute, personal communication.
23. Software Engineering Institute, PSP academic materials,  
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.