SMP: An Integrated Approach to Project Management

Marc Frappier
Département de mathématiques et d'informatique
Université de Sherbrooke
Sherbrooke (Québec) Canada J1K 2R1
+1 819 821 8000x2096
marc.frappier@dmi.usherb.ca

Mario Richard
Merck Frosst Canada Ltd
16711 Rte Transcanadienne,
Kirkland (Québec) Canada
+1 514 428 3459
mario_richard@merck.com

Abstract

SMP is a project management tool specifically designed for process oriented software projects. It integrates six project management functions: process modeling, effort estimation, planning, tracking, control and measurement. Their integration is based on three basic principles : i) a project plan should follow from a process model; ii) effort estimation should be based on project historical data; iii) control should be based on units that can be tracked across several phases and activities. SMP is inspired from Watt's Humphrey *Personal Software Process*, which has been lifted to deal with large projects instead of personal projects.

Keywords

Project management tool, process modeling, effort estimation.

1 Introduction

This paper provides an overview of an integrated approach to project management which is supported by a tool called SMP. SMP was developed by the Université de Sherbrooke in cooperation with Merck Frosst Canada & Co. It is inspired from Watt's Humphrey *Personal Software Process* [2], which has been lifted to deal with large projects instead of personal projects. It also follows from modern software engineering principles (eg, see [3,4]) and the practical experience of several project managers. SMP integrates six project management functions: process modeling, effort estimation, planning, tracking, control and measurement. It was designed for managing software projects, but its principles should also apply to other kinds of engineering projects.

SMP implements three basic principles which, curiously, are hardly ever integrated into a single tool. The first principle is that a project plan should be based on a *process model*. A process model is the skeleton of a project plan; it ensures that activities are not forgotten, and that project tracking and data gathering are conducted in a consistent manner across all projects of an organisation. This in turn enables more reliable effort estimation based on *historical project data*, which is the second basic principle. Finally, a project manager should be able to follow the progression of *units* throughout the course of a project. A unit in the SMP context is a piece of work for which several tasks will be performed. Classical examples of units are use cases, patterns, architectural components. Estimation, planning and tracking are conducted by units.

How does SMP differ from conventional project management tools? SMP *integrates* project planning activities *with* process modeling and effort. It provides the essential functions for project planning, but it does not offer sophisticated features expected from general purpose project planning tools like Microsoft Project, which computes various diagrams (PERT and Gantt), resources loading and critical paths. On the other hand, SMP provides a more sophisticated project plan structure which is designed to efficiently support effort estimation, plan creation, adjustment and tracking, and process modeling. With this structure, SMP provides a better guidance to the project manager, because it can help in shaping its plan and decomposing global project effort estimates into task efforts, ensuring that the project will meet its goals in terms of cost and schedule. SMP is specifically designed for managing projects following a well-defined process, which is typical in engineering projects (like software, electrical, mechanical, civil, construction). SMP takes advantage of this fact by gathering statistics which provide assistance during project management.

SMP has a web interface; it is implemented in Java using JSP, JDBC and the J2EE blueprint; hence it can be easily ported to several platforms and accessed through the internet. It has been used for over two years by Merck Frosst Canada & Co. in Montréal. It is also used by the Silica Consulting Group and the École Polytechnique of Montréal.

2 Overview of SMP Functions

2.1 Process Modeling

SMP allows a project manager to define process models and to plan a project following a selected process model. A process model defines the *phases* and *process components* that should be used for a project. A phase is a time segment of a project with well-defined objectives (eg., requirements analysis, design, implementation). A process component (which is sometimes called an *activity type*) is a type of task (eg., use case definition, programming, testing, inspection). This structure allows the definition of most commonly used process models like the Rational Unified Process [3], the waterfall model, etc. During project planning, the manager instantiate process components to create tasks.

2.2 Estimation of Project Effort

Effort estimation is based on expert judgment and analogy using historical data from completed projects. The project manager first establishes the list of *units* which will be developed during the

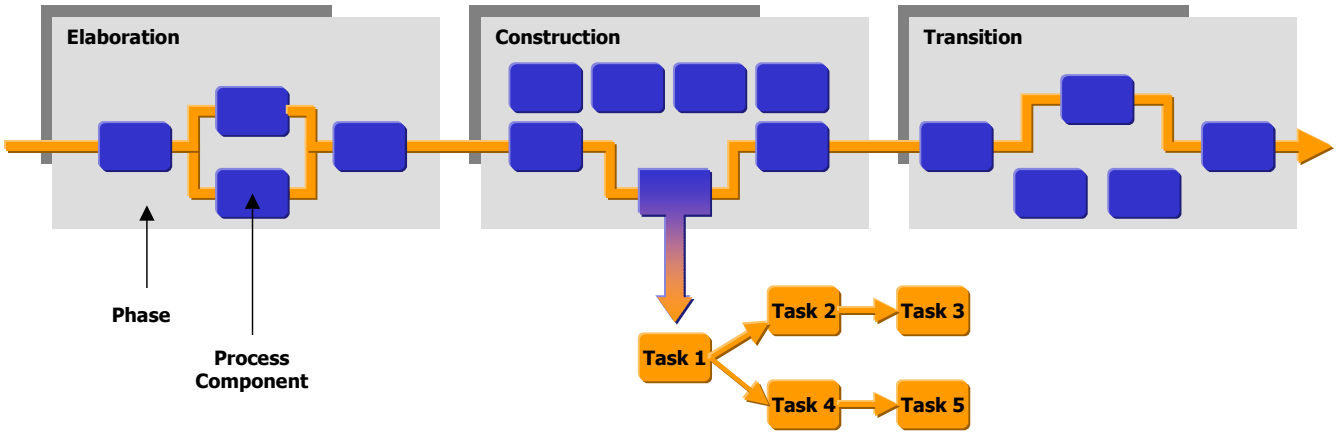

Figure 1. The decomposition of a process component into tasks for a unit

project. A unit can be decomposed into sub-units, recursively. All examples in this paper are based on use case units, but the user defines the unit types he needs.

Once the list of units is derived, one can estimate the effort for the project by estimating each unit separately. For each unit, the project manager must determine the unit *category* and the unit *complexity*. For estimation purposes, this is all that is needed in SMP (and can be usually afforded at this point of the project). Units can be classified into categories to allow a finer effort estimation. For instance, one may want to categorize use cases into database management (add-update-delete use case), complex calculations (eg a statistical analysis use case), algorithmic use case (eg complex manipulation of data structures like graph, tree, matrix), or real-time (real-time control of some hardware component). The categories are determined by the project manager according to its application domain. The complexity of a unit is determined using the knowledge of the project manager about the unit and the project environment like team experience, domain knowledge experience, technology, etc. As in Humphrey's PSP [2], complexity simply denotes the *level of effort* required to carry out the unit. In the example below, we will consider five complexity levels: VS (very simple), S (simple), M (medium), C (complex) and VC (very complex). Table 1 below provides an example of a list of use cases with their category and complexity.

Table 1. List of units for effort estimation

unit	complexity	category	estimated effort
use case 1	M	data mgmt	100
use case 2	S	data mgmt	80
use case 3	S	data mgmt	80
total project effort			260

Table 2. Effort by complexity level for a unit type and category

unit type	use case
category	data mgmt
complexity level	**effort**
VS	60
S	80
M	100
C	120
VC	140

Using tracking data of previously completed projects, an effort is associated to each combination of complexity level, unit type and unit category. Table 1 and Table 2 above provide an example. The effort of a complexity level is determined using the average effort μ and the standard deviation σ for the set of units of a given category and unit type for past projects. The complexity levels are determined as follows :VS = $\mu - 2*\sigma$, S = $\mu - \sigma$, M = μ, C = $\mu + \sigma$, VC = $\mu + 2*\sigma$. Table 2 has been used to compute the estimated effort of the use cases in Table 1.

This approach for unit effort estimation is appropriate for unit types like use case. For other unit types like project management, user training, database loading, and software installation, the statistical distribution is not the same from one project to another. For instance, the effort a unit of type "conduct user training" will depend on the number of users to train, the training class size and the duration of each class. In that case, the manager can always override the system's proposal and manually derive an estimate using a more appropriate formula.

2.3 Project Planning

Project planning consists in defining *iterations* and creating tasks for units according to the components of the process model associated to the project. A phase is decomposed into iterations. An iteration is also a time segment with well-defined objectives; it is like a sub-phase of a phase. It helps to define short-term objectives and to control the project on a regular basis. A task has an estimated effort which is a part of its unit estimated effort. Tasks are assigned to resources. SMP can also compute a project schedule which determines start and end dates for project phases. Figure 1 illustrates the decomposition of a process component, applied to a given unit, and decomposed into five tasks. The unit

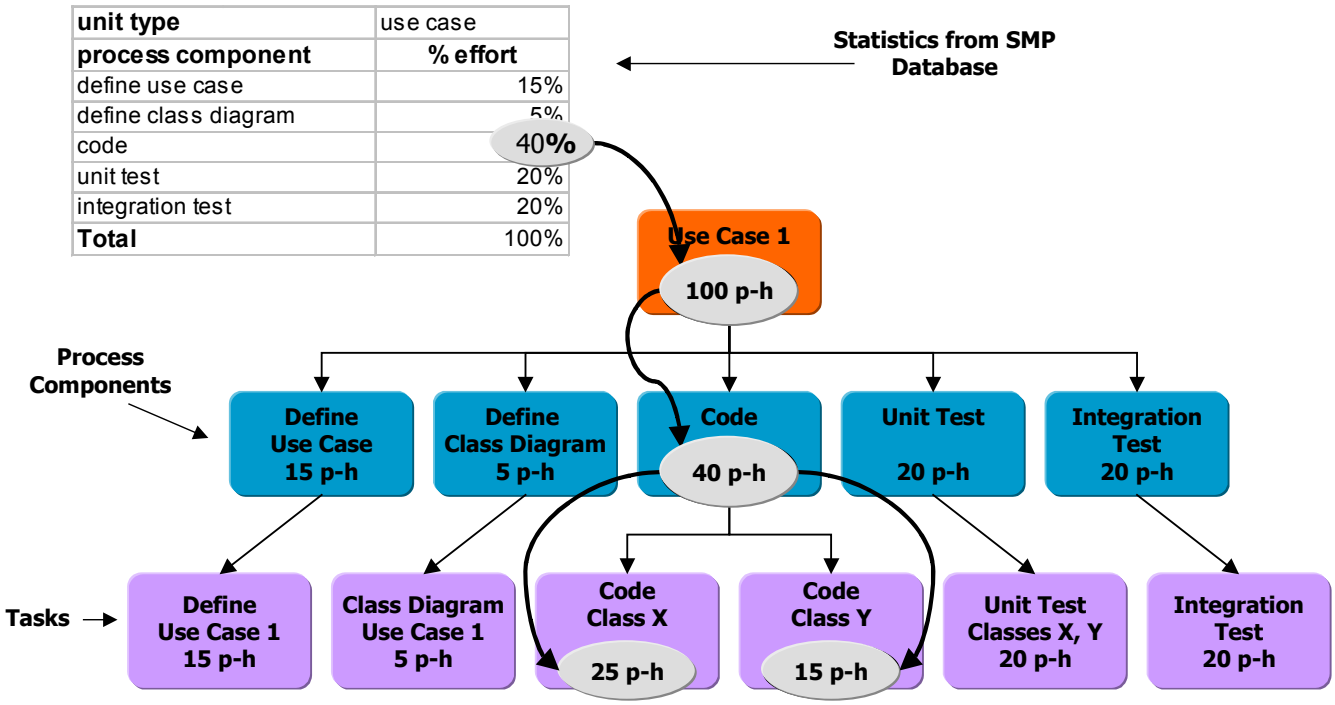

Figure 2. Allocation of effort to tasks

is represented by the (orange) arrow; it crosses the process components needed to realize the unit. Figure 2 illustrates how effort is allocated to each task using the percentage of effort allocated to a process component in past projects. This process is detailed in the next subsections

2.3.1 Iteration Planning

A phase must be decomposed into iterations, and each iteration into a set of tasks according to the process components. Before defining an iteration, it is sometimes necessary to decompose a top level unit into sub-units, in order to have more manageable pieces. Top level units identified in the beginning of a project are usually high level units. They are decomposed during iterations as the requirements analysis is conducted. Table 3 below illustrated how use cases are decomposed. When a use case is decomposed, its estimated effort is allocated to sub-use cases. The total estimated effort of sub-use cases should match the estimated effort of the parent use case, in the ideal situation. However, it could be different (lower or higher), since the understanding of a use case improves as it is decomposed.

Table 3. Unit decomposition

	parent	
unit	unit	effort
use case 1		100
use case 2		80
use case 2.1	use case 2	32
use case 2.2	use case 2	48
use case 3		80

Once units are decomposed at a satisfactory level, an iteration can be planned. A unit is developed according to process components. It may take several iterations to complete all process components for a unit. From historical project tracking data stored in the SMP database, the percentage of the total effort allocated to each process component can be calculated. As an example, consider the table in the upper left corner of Figure **2**.

In an iteration, one must select the units that will be developed, and the process components that will be carried out for each unit. The historical percentage of each process component is used to estimate the effort for each unit. For instance, assume that for iteration 1, one plans to do the tasks listed in Figure 2. The effort for a process component applied to a unit is simply the percentage of that process component multiplied by the estimated effort for the unit. For instance, the total effort for use case 1 was estimated at 100 person-hours in Table 1. The percentage of the process component "Define Use Case" is 15%, according to Figure 2. Hence, the effort for performing the requirements analysis of use case 1 should be 100 * 15% = 15 p-h.

2.3.2 Estimation of Task Effort

A process component of a unit is further decomposed into one or several tasks. For instance, the coding of use case 1 and use case 2 may be decomposed into two tasks, as illustrated in Figure 2. The estimate of each task is derived based on human judgment from the process component estimate and the relative complexity of each task. Again, the total estimated effort for the tasks of a process component should be equal to the process component estimated effort. At this point, task may be assigned to resources for realization. SMP provides basic management of resources and task assignment.

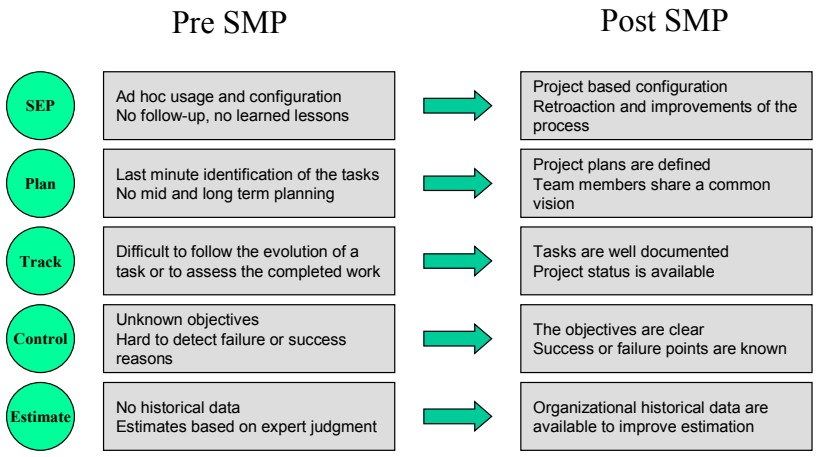

Figure 3. SMP-induced cultural changes

2.4 Project Tracking and Control

Project tracking consists in entering on a regular basis (at least daily) the actual effort spent on each task by the resources. Once a week, resources are required to enter an estimated effort to complete a task. When this estimate is 0, the task is considered completed. Since tasks are associated to project units, project phases and process components, actual effort can be summarized at these different levels: unit, iteration, phase, or project. Project control consists in measuring progression against plan and taking corrective actions when necessary in order to meet project objectives in terms of effort and schedule. Table 4 illustrates tracking by unit. The column estimated effort is the initial estimate derived in Table 1. The actual effort is the sum of the tracking entries from all task associated to the unit over all iterations. The estimate to complete is also a sum of all the estimates to complete entered for each task. When a use case has not been completely planned yet, this column contains the estimated effort for these unplanned parts. The forecast is simply the sum of the actual effort and the estimate to complete. The error is the percentage difference between the forecast and the initial estimated effort. The percentage of completion for the whole project is given by the sum of the actual effort divided by the total of the forecast effort. In the example above, the project is completed at 80 %.

Table 4. Project progress tracking by unit

unit	estimated effort	actual effort	estimate to complete	forecast	error
use case 1	100	90	15	105	5%
use case 2	80	50	40	90	13%
use case 3	80	75	0	75	-6%
	---	---	---	---	---
total project	260	215	55	270	4%
percentage of completion		80%			

Tracking progress using the notion of estimate to complete has sometimes been criticized as ineffective. Boehm [1] and Humphrey [2] suggest the earned value system, where values are earned only when a task is completed. The estimate to complete provides a more precise estimate of the work in progress. It can be misleading when tasks are not decomposed a sufficient level of details; a project may then suffer from the "80 % complete" syndrome, because resources have difficulties to reliably estimate the work remaining to complete a task. However, when tasks are small enough (eg, at most 5 days), this estimate to complete is much easier to determine; progress can be tracked as accurately as with the earned value system.

3 A practical approach to SMP usage

3.1 Organizational Integration

Transiting from the abstract level of a solution to its implementation in a commercial environment often represents an important organizational challenge. In order to reach the implementation objectives for a system like SMP, it is important to present a solution which is tightly coupled with the core activities of the system users. If the overhead is too important, the solution is rapidly discarded. At the opposite, if the solution is properly tailored, the organization is using it as a focus point which creates a synergy affecting positively other areas of the engineering process.

Figure 3 shows the cultural changes observed after the implementation of SMP in the Basic Research Informatics group at Merck Frosst Canada & Co.

3.2 Practical Results

SMP can be used in different ways to manage a project. Figure 4 illustrates a typical project management process supported by SMP which has been used at Merck and Silica. The key in reaching project objectives is to conduct management on a regular basis and to continuously focus on project objectives. SMP streamlines that process, by providing the means to set objectives (units, estimates, tasks, task assignments) and to measure progression towards these objectives (tracking). The overhead associated with management and SMP usage is easily offset by the gain in control, in focus and the ability to detect slippage early in the project. At Merck, SMP has been used on more than 12 projects which where following two different processes. Benefits were reaped right from the first case, where a two-year lagging

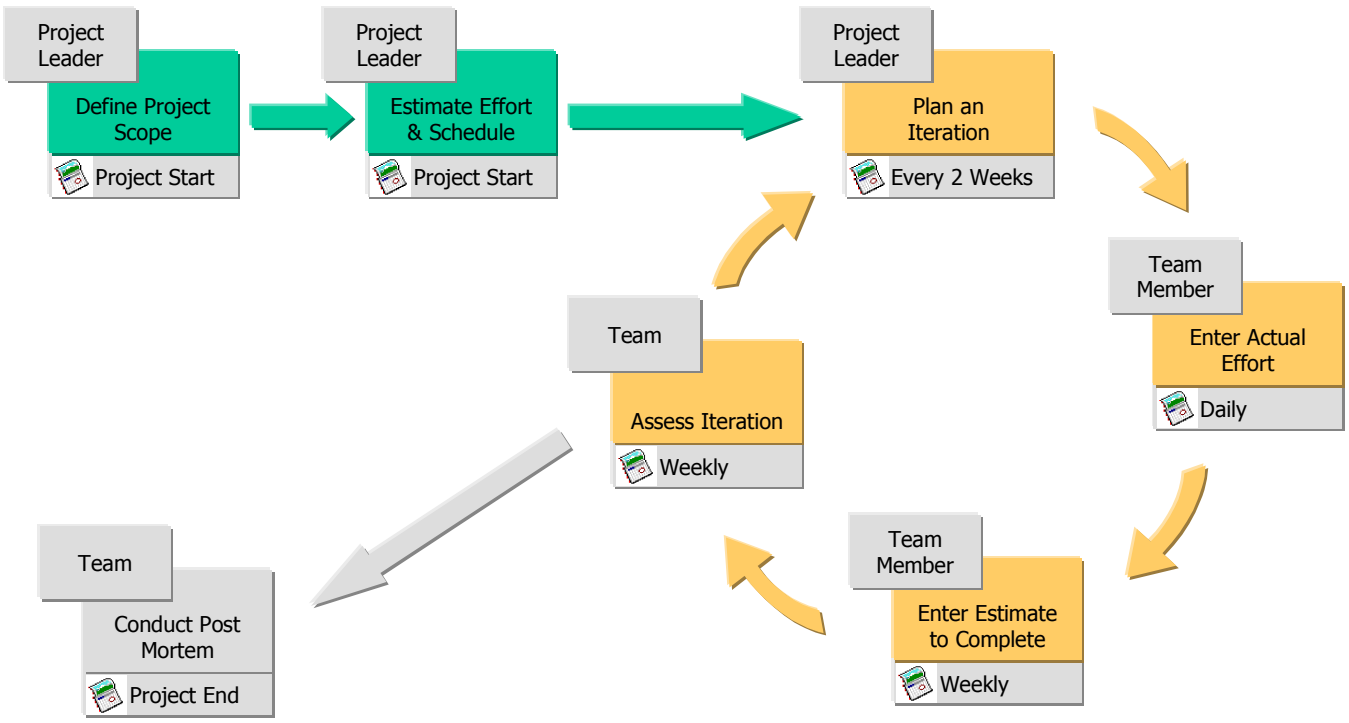

Figure 4. A typical SMP-based project management process

project was harnessed and transformed into a success by reaching the goals which were established at the creation of the first SMP plan. SMP helped the team to maintain productivity level in organizing the development of more then 60 new features for this system while the team size doubled during the 3rd quarter. The new requirements were delivered as scheduled, thereby defying Brooks' Law: "*Adding manpower to a late software project makes it later*".

SMP has been used on teams of up to 9 people. The project leader typically spent 3 hrs to plan an iteration every two weeks. Resources spent a few minutes each day to enter the actual effort spent on their tasks during the day. Once per week, a project meeting was conducted, which lasted roughly between 1 and 2 hrs. Progression was measured, and corrective actions (adjustments to iteration plan) were taken to ensure that objectives could be met.

4 Conclusion: Bringing Order to Chaos

SMP is a lightweight project management tool adapted to process-driven projects. It is configurable, allowing the modeling of multiple processes through a flexible matrix-like structure. It guides the project manager by reusing historical project data to estimate project effort and to decompose this effort into manageable tasks following a process model. SMP provides focus on project objectives, and helps a team to stay on track and to meet commitments, while requiring little overhead.

In the future, we plan to introduce the ability to version the estimates of a project, in order to track the evolution of project scope and to cater for re-estimation at phase ending. We also plan to add more sophisticated status reports (project level and iteration level) and to provide additional support to quickly derive an iteration plan. We recently added several product and process measures to SMP like size (in LOC, function points and full function points), reuse, defects and productivity. We have not yet started to experiment with these measures.

5 References

[1] Boehm, B. : *Software Engineering Economics*, Addison-Wesley, 1983.]

[2] Humphrey, W.S.: *A Discipline for Software Engineering*. Addison-Wesley, 1995.

[3] Rational Software Corp: - Rational Unified Process, 2000.

[4] Sommerville, I.: *Software Engineering*. 6th edition, Addison-Wesley, 2001.