# SMP: A Process-Driven 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 modelling, 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 modelling, effort estimation.

## 1 Introduction

This paper provides an overview of an integrated approach to project management which is supported by a tool called SMP (Software Measurement Program). 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* [5], which has been lifted to deal with large projects instead of personal projects. It also follows from modern software engineering principles (e.g., see [7,8]) and the practical experience of several project managers. SMP integrates six project management functions: process modelling, 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*. In the past decade, the Software Engineering Institute has actively promoted the use of process models in order to increase the maturity level of software organizations [4]. A process model is the skeleton of a project plan; it ensures that all projects of an organisation are structured in a consistent manner, 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. It also facilitates measurement-based process improvement.

Consistency in data gathering is the key to reliable estimates [1,6]. In order to reuse tracking data from past project, one must be sure that these projects used the same definitions for activities and their deliverables. For instance, there is often confusion in software engineering about the meaning of various activities like requirements analysis, software architecture, software design, and so on. There are numerous ways of realising them using different deliverables and formats. A good process definition helps in removing these ambiguities.

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, at various phases of a project are 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 modelling and effort estimation. It provides the essential functions for project planning, but it does not offer sophisticated features expected from

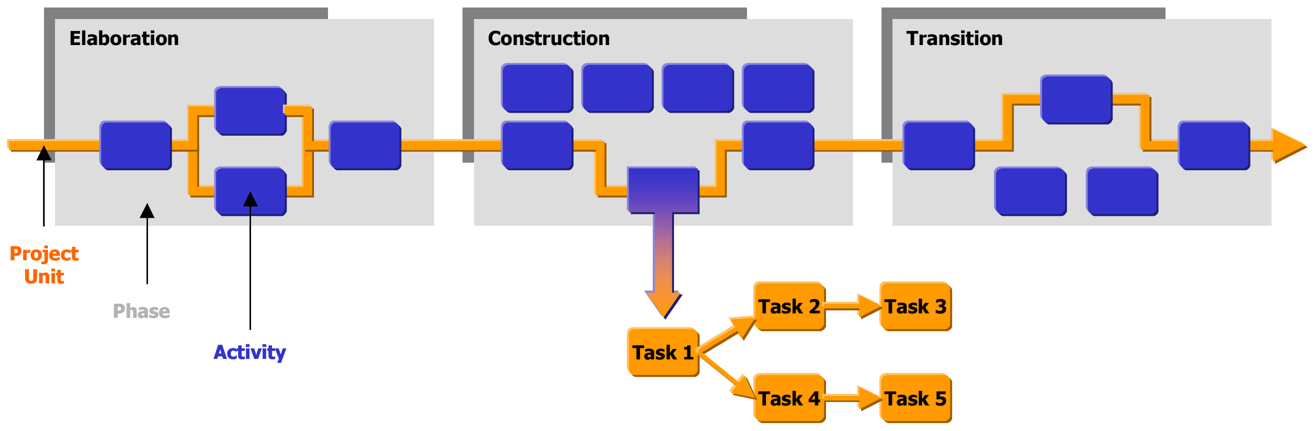

**Figure** 1. The decomposition of an activity into tasks for a unit

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 modelling. 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 by Merck Frosst Canada & Co. in Montréal, the Silica Consulting Group, the Université de Sherbrooke, and the École Polytechnique of Montréal.

## 2 Overview of SMP Functions

### 2.1 Process Modelling

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 *activities* 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). An activity defines a type of task (eg., use case definition, programming, testing, inspection). This structure allows the definition of most commonly used software process models like the Rational Unified Process [3], the waterfall model, etc. During project planning, the manager instantiate activities 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 realised during the project. A unit can be decomposed into subunits, recursively. All examples in this paper are based on use case units, but the user can define 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 $\mu$ and the standard deviation $\sigma$ 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 = $\mu$, C = $\mu$ + $\sigma$, VC = $\mu$ + 2*$\sigma$. Table 2 has been used to compute the estimated effort of the use cases in Table 1. Of course, we thereby assume that unit efforts are normally distributed in the historical data set. This hypothesis may not be satisfied, as sometimes data set are skewed, with most of the units being small. There exists statistical tests to determine if a data set is normally distributed. When it is not, transformations can be applied in order to obtain a normal distribution (e.g., logarithmic transformation). We have not yet invested the effort to implemented these techniques in SMP.

This approach for unit effort estimation is appropriate for unit types like use case, whose effort estimate depend mostly on its functional size and its intellectual complexity. 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, because the actual effort depends on other factors, like duration of the project for project management, or the number of users for user training, the number of database tables for database loading, etc. Each of them has a specific estimation method. For project management, managers will often use a standard percentage of the total project effort; for user training, the effort will depend on the number of users to train, their type (super user, occasional users, etc), the training class size and the duration of each class. For these unit types, the manager can override the system's proposal based on $\mu$ and $\sigma$, and manually derive an estimate using a more appropriate formula.

## 2.3 Project Planning

Project planning is an incremental process, especially in software development, where it is sometimes impossible to create all tasks upfront. Planning consists in defining *iterations* and creating tasks for units according to the activities of the project's process model. A phase is decomposed into iterations. An iteration is a (usually short) time segment with well-defined objectives; it is like a sub-phase of a phase. Iterations help to define short-term objectives and to control a 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 unit into activities, which are further decomposed into tasks. The unit is represented by the (orange) arrow; it crosses the activities needed to realize the unit. Figure 2 illustrates how effort is allocated to each task using the percentage of effort allocated to an activity 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 activities of the phase. Before defining an iteration, it is sometimes necessary to decompose a top level unit into subunits, in order to have more manageable pieces. Top level units are usually high level units identified during project inception. There is not enough time to completely define these units at that point in the course of the project, since the objective of inception is to make a business case for the project and get funding to start it. These high level units are decomposed through requirements analysis during the elaboration phase. Some high level units may be added, others may be deleted, due to changes in the system objectives, the business environments, or the technology chosen.

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

Table 3 illustrates how units 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

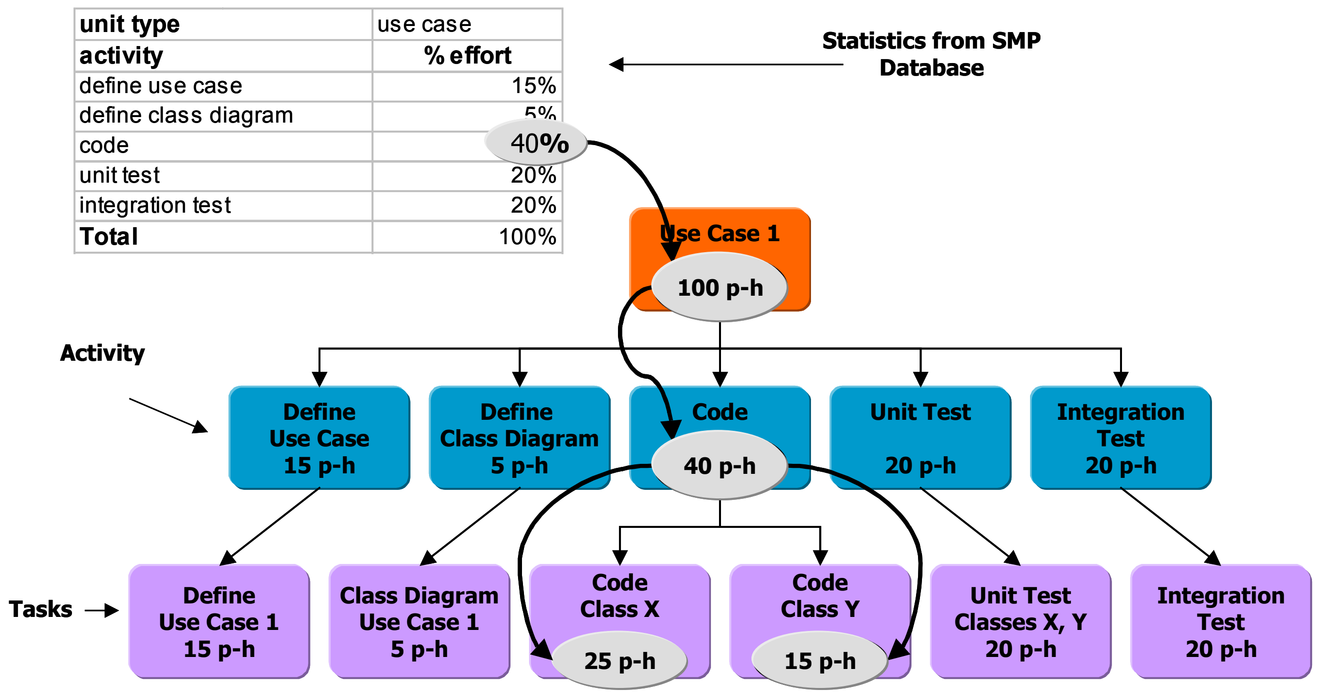

**Figure** 2. Allocation of effort to tasks

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 during requirements analysis.

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

**Table 3. Unit decomposition**

| unit | parent 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 |

In an iteration, one must select the units that will be developed, and the activities that will be carried out for each unit. The historical percentage of each activity is used to estimate the effort of an activity applied to a unit. For instance, assume that for iteration 1, one plans to do the tasks listed in Figure 2. The effort for an activity is obtained by multiplying its percentage with 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 activity "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

An activity for 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 activity estimate and the relative complexity of each task. Again, the total estimated effort for the tasks of an activity should be equal to the activity estimated effort. At this point, task may be assigned to resources for realisation. SMP provides basic management of resources and task assignment.

## 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 (ETC). When the ETC is 0, the task is considered completed. Since tasks are associated to project units, project phases and activities, actual effort can be summarized at these different levels: unit, iteration, phase, or project.

**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% | | | |

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 tasks 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 unit has not been completely planned yet, this column contains the estimated effort for these unplanned parts, which is somewhat complex to derived (see next section). 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 actual effort divided by the forecast effort. In the example above, the project is completed at 80 %.

Tracking progress using the notion of ETC has sometimes been criticized as ineffective. It can be misleading when tasks are not decomposed at a sufficient level of details; a project may then suffer from the "80 % complete" syndrome, which means that week after week, the project reports an 80 % completion level, the ETC never diminishing as more effort is spent on the project. This is often due to improper planning, when tasks have been under estimated, overlooked, or defined with imprecise objectives. However, when tasks are small enough (eg, at most 5 days), the ETC is much easier to determine and it is quite reliable. Boehm [2] and Humphrey [5] suggest the earned value system to track progress; values are earned only when a task is completed. The estimate to complete provides a more precise estimate of the work in progress than the earned value system.

## 2.5 Computing the ETC

The computation of a unit ETC requires to take into account its decomposition. A unit is decomposed at several levels. First, a unit can be decomposed into subunits, recursively. Next, units are realised over several iterations; within each iteration, a unit is decomposed into activities, and these activities are finally decomposed into tasks. A unit ETC cannot simply be the sum of its tasks ETC, because the unit may not be completely planned. Hence, the estimate to complete for a top level unit (i.e., a unit without parent) is derived according to the following procedure.

Let $x$ denote a task, an activity or a unit. Let $ETC(x)$ denote the estimate to complete of $x$; let $E(x)$ denote the estimated effort of $x$; let $subunits(u)$ denote the children of unit $u$; let $activities(u)$ denote the set of activities of unit $u$. First, we define the estimate to complete of an activity $a$, which is denoted by $ETC(a)$. When all tasks of an activity $a$ have been created, we say that $a$ is *completely planned*; the project manager must inform SMP of this fact by marking the activity as completely planned. The ETC of a completely planned activity $a$ is simply the sum of its tasks' ETC.

$$ETC(a) = \sum_{t \in tasks(a)} ETC(t)$$

For a partially planned activity, the unplanned part must be added. It is derived by adding the difference between the activity's estimated effort and the sum of the estimated effort of its tasks.

$$ETC(a) = \sum_{t \in tasks(a)} ETC(t) + E(a) - \sum_{t \in tasks(a)} E(t)$$

This ETC calculation principle of a unit is similar, but more complex, since a unit may have several levels of decomposition and it may have its own activities. A unit $u$ is said to be completely planned when: i) its decomposition into subunits is completed; ii) all its own activities have been created and marked as completely planned. As for activities, the project manager must inform SMP of this fact by marking the unit as completely planned.

A decomposed unit $v$ (i.e., a unit with subunits) can also have its own activities. Hence, a completely planned decomposed unit is computed as follows.

$$ETC(v) = \sum_{u \in \, subunits(v)} ETC(u) + \sum_{a \in \, activities(u)} ETC(a)$$

A a partially planned decomposed unit $v$ is computed as follows:

$$ETC(v) = \begin{array}{l} \displaystyle\sum_{u \in \, subunits(v)} ETC(u) + \sum_{a \in \, activities(u)} ETC(a) \\ \displaystyle + E(u) - \sum_{u \in \, subunits(v)} E(u) \\ \displaystyle - \sum_{a \in \, activities(u)} E(a) \end{array}$$

This formula can be explained as follows: the ETC of a decomposed unit is the sum of two quantities: its planned part and its unplanned part. The ETC of the planned part is the ETC of its subunits and its proper activities. The ETC of the unplanned part can be estimated by subtracting

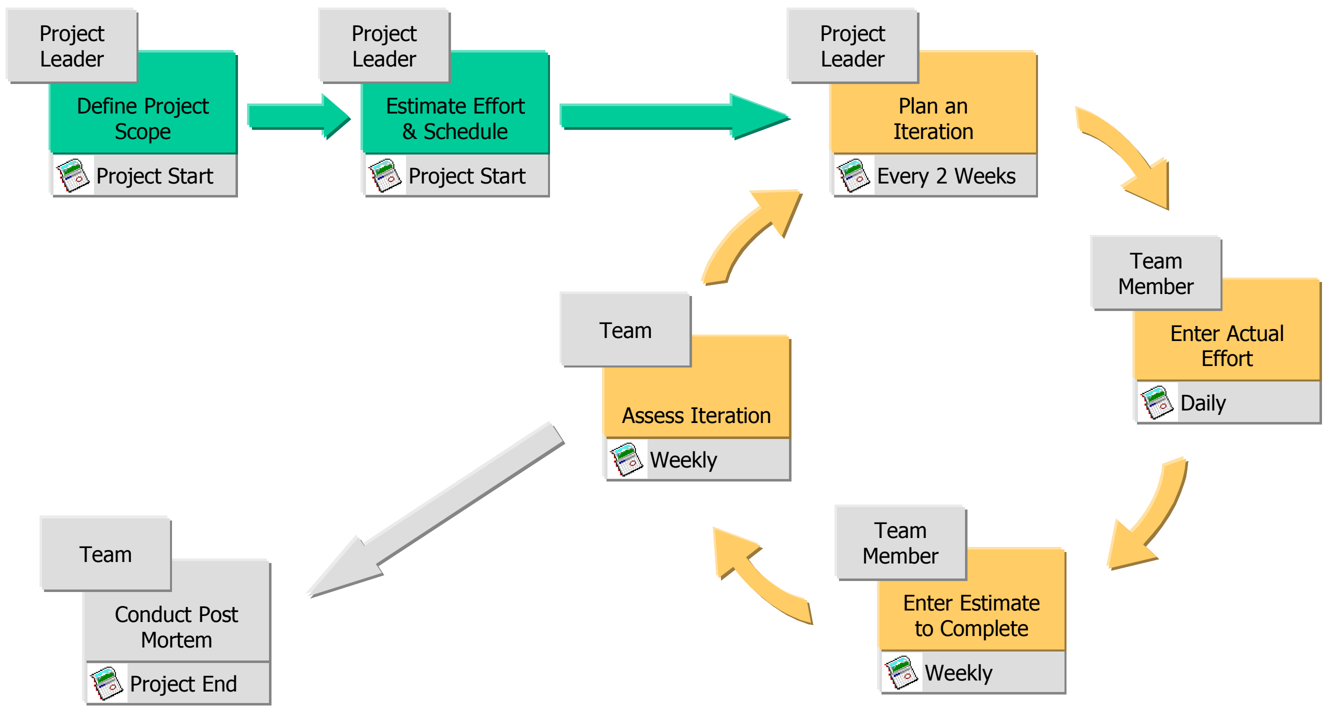

**Figure** 3. A typical SMP project management process

from the estimated effort of the unit the estimate effort of its planned parts, which consisting of its subunits and its proper activities.

### 2.6 An SMP Project Management Process

SMP can be used in different ways to manage a project. Figure 3 illustrates the target project management process supported by SMP. The first step is to delineate the project scope by defining the list of units to be constructed during the project. This list is preliminary, and it will certainly evolve during the course of the project. Next, the effort for these units must be estimated, and a project schedule, which determines start and end dates for project phases, must be derived. This information is usually sufficient for project inception, where the feasibility of the project is evaluated.

After these steps, project iterations are performed to realise the project. Iterations can be planned ahead of time, but they are usually created at the end of the current iteration. As an iteration progress, resources enter their actual effort on each task on a daily basis. At the end of the week, the ETC of each task is determined by its assigned resources. At the beginning of a week, the current iteration is assessed during a project team meeting. The project manager analyses project progress reports and discusses with his team the status of the current iteration, the correctives actions to take, and the plan of the next iteration when the current iteration is completed. Tasks which are not completed at the end of an iteration can be transferred by SMP to the next iteration.

Phase end reviews are not presented in Figure 3, because they are quite similar to iteration assessments. SMP contains the appropriate information to produce phase end reports, but they haven't been implemented yet.

## 3 An Assessment of SMP

In this section, we describe how SMP has been used in the Basic Research Informatics (BRI) group of Merck Frosst in Montreal, and also provide some feedback from the École Polytechnique de Montréal, with more emphasis on the first one because it is the most typical industrial environment, and assess its suitability in these contexts.

### 3.1 Pre-SMP Environment at Merck

The Basic Research Informatics (BRI) group of Merck in Montreal triggered the development of SMP in 1998. The main motivation was to improve the ability to meet delivery dates while providing reliable software. An initial study was conducted to define the requirements of SMP; it was completed in 1999.

BRI employed 7 employees at that time; its size has constantly increase to reach 15 employee in 2003. BRI provides sophisticated scientific information systems to support pharmaceutical researchers. The group is committed to use leading edge technology, which leads them to frequently experiments with new technology.

Before 1998, there was no formal software engineering process and no formal project management in place. Projects were conducted and managed in an ad-hoc manner. The emphasis was on delivering as much high quality systems as possible with the resources available, with minimal management overhead.

Projects were initiated by discussing needs with the users on a regular basis. Users expressed their desired features and delivery dates. Developers assessed the difficulty of developing these features and estimated to the best of their knowledge when the features could be delivered. Needs were prioritised, and delivery dates were agreed upon. Objectives were then assigned to resources, without preparing a detailed project plan or effort estimates.

Project were driven by deadlines and features to be delivered. Actual effort was not tracked. During a project, the project manager evaluated progress by ad-hoc discussions with the resources. Delivery dates were estimated using expert judgement.

This measurement of progression was not very precise, was quite subjective, and often lead to late discovery of schedule slippage, making it hard to meet deadlines set with users. It was also difficult to assess the impact of changes to user requirements and resource allocation.

The difficulty to make realistic commitments was an important issue for the BRI group. It was this issue that lead BRI to try a more systematic approach to conduct software projects.

## 3.2 Implementation of SMP at Merck

The first version of SMP was delivered in January 2001. A second, web-based version of SMP was delivered in January 2002. Transiting from the abstract level of a solution to its implementation in an industrial environment often represents an important organizational challenge. Moreover, the BRI group could not afford to devote several internal resources to tackle the issue of process and project management. Hence, SMP was developed as a side activity conducted on top of the day-to-day software development activities, which explains the time span between the inception of the idea and its first implementation.

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 positively affecting other areas of the engineering process.

SMP introduces significant management overhead, compared to the existing process used in BRI in 1998. It was decided that SMP would be used by one team of BRI as a pilot project in order to evaluate its applicability and suitability. The idea was to incrementally integrate SMP in BRI according to cultural changes and resource availability. The size of this team grew from 5 people to 9 people during the course of the pilot project.

When SMP is first implemented, there is no historical data available to feed the estimation process. Reusing historical data from other organisations is usually not a good idea, because environmental characteristics vary too much from one organisation to another (i.e., process definitions, data gathering techniques, technology, resource profile, and application domain). An objective of this pilot was to start building a good historical data set. Hence, estimates have been derived using expert judgement.

## 3.3 Assessment of SMP from Usage at Merck

Figure 4 shows the cultural changes observed after the implementation of SMP in this team of BRI. The cultural changes were important. By recording what was done and how much effort was spent, the team members could now assess was they were doing from an objective perspective. For the first time, they could quantify activities of their software process, which lead them to modify some of them.

Team members were invited to participate in the estimation of the tasks they were assigned. These estimates were discussed during a project meeting held on a weakly basis. It created a positive feeling of public commitment towards the rest of the team, which in turn stimulated the project team to focus on the project objectives. Adherence to the plan became a cultural value. Moreover, discussing the detailed project plan on a regular basis improved the communication between team members. They better understood what their objectives were, and how they related to the objectives of the other team members. It helped to discover loopholes in the system design and to avoid building similar software components that were used in different features. Tracking and reporting progress was also easier, which enabled early detection of schedule slippage. It was also easier to evaluate the effect of changes to the scope of the project or to resource allocation.

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. The weekly project meeting lasted roughly between 1 and 2 hrs.

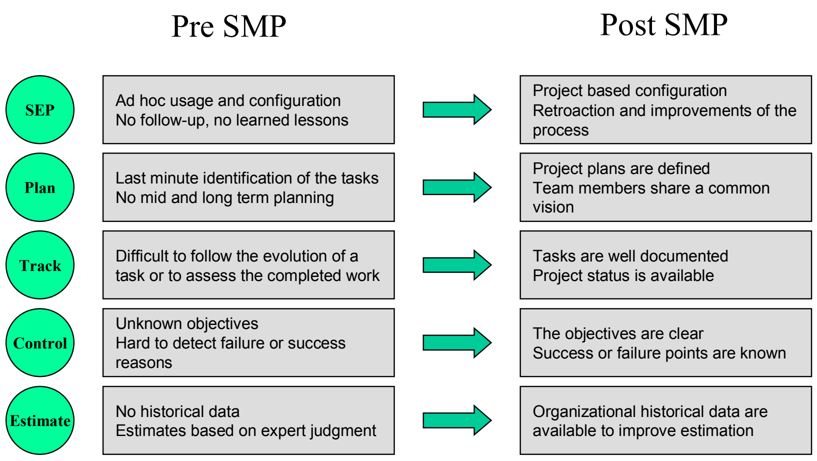
Pre SMP
Post SMP
SEP
Ad hoc usage and configuration
No follow-up, no learned lessons
Project based configuration
Retroaction and improvements of the process
Plan
Last minute identification of the tasks
No mid and long term planning
Project plans are defined
Team members share a common vision
Track
Difficult to follow the evolution of a task or to assess the completed work
Tasks are well documented
Project status is available
Control
Unknown objectives
Hard to detect failure or success reasons
The objectives are clear
Success or failure points are known
Estimate
No historical data
Estimates based on expert judgment
Organizational historical data are available to improve estimation

**Figure** 4. SMP-induced cultural changes

SMP has been used on more than 12 projects which where following two different processes. Major benefits were observed right from the first case, where a two-year lagging project was set under control and transformed into a success by reaching the goals which were established at the creation of the first SMP plan. SMP also helped to maintain productivity level by organizing the development of more then 60 new features for this system while doubling the team size 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*" [3].

It is difficult to quantitatively assess if the cost of using SMP was offset by the improved control over the software projects, because there is no reliable quantitative data available on non-SMP projects to compare with. From a qualitative view-point, SMP definitely had a positive impact.

SMP has also been used in a final year software engineering project at École Polytechnique de Montréal. Students were divided into teams of 8 people and used SMP to plan and track their project. They were introduced to SMP in a lecture of three hours, which enabled them to start planning their projects. They still required support from the instructor to handle some of the more sophisticated aspects of the plan structure. The students felt that the user interface was not easy to use; the navigation among the structure of a plan is somewhat cumbersome.

At both Merck and École Polytechnique, the activity decomposition level which is used during the creation of an iteration plan (i.e., the middle layer of the hierarchical diagram in Figure 2) was deemed unnecessary for small units, because there is often only one task for each activity. Indeed, this level was introduced to help the decomposition of large units into several tasks for a given activity. A similar difficulty occurs, for instance, when SMP is used to track maintenance projects or small student projects. In that case, each unit can denote a change to one function of the software. Most of these changes are usually very small in size (a few days or hours). There are several solutions to this issue. First, a proper process must be defined, with coarser activities. Second, unit aggregation must be properly used, so that activities are realised at the right level of decomposition in a unit. For instance, in a software maintenance project, related changes which will be integration tested together and moved into production as one group should be defined as one unit, with one subunit for each individual change. Activities like testing and installation are planned at the parent unit level, while change analysis, coding, and unit testing are planned at the subunit level. This approach to unit structuring is not obvious for first-time users of SMP. In the worse case, SMP can be used exactly like Microsoft Project, by creating one unit and one activity for each task, thereby losing all information on the structure of the work carried out and weakening the estimation process.

## 4 The Future of SMP

In the future, we plan to introduce the ability to baseline units and to version the estimates of units, in order to track project scope evolution and to cater for re-estimation at phase ending. As more information is gathered during each phase, the understanding of the project scope, software functionalities and technical difficulties increases significantly. Accordingly, the estimates of the various units can be adjusted to better match reality. Project scope can also be increased or reduced, due to budget restrictions, changes in the business environment, and so on. The ability to track this evolution would improve the quality of the historical database. For instance, units that are dropped in the course of a project should not be taken into account in the computation of the statistical distribution. Project scope variations can also be used to explain deviation from initial project estimates, and to justify project performance to upper management. Scopes changes are expensive in software engineering, because software architecture depends on the set of units to implement. Adding new units may cause severe changes in the software architecture, which may in turn require to redo several activities of existing units.

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 Conclusion: Leveraging Structure to Bring Order

SMP is a lightweight management tool adapted to process-driven engineering projects. It is configurable, allowing the modelling of multiple processes through a flexible matrix-like structure. It guides the project manager by reusing historical process data to estimate project effort and to decompose this effort into manageable tasks following a process model. SMP provides focus on project objectives, by structuring a plan based on units, by allowing for a close follow-up of these units throughout the project, by supporting incremental planning with the concepts of iteration and unit decomposition. These features help a team to stay on track and to meet commitments, while requiring little management overhead. When an organisation moves from an unstructured project management approach to an SMP-based approach, significant cultural changes occur.

SMP differs significantly from general purpose project management tools like Microsoft Project, by providing more structure to the plan, and better guidance for the project manager, but SMP lacks the classical features (Gantt, PERT, critical path analysis, resource loading analysis, calendar management) found in these tools. A possible solution would be to integrated SMP with them, in order to reuse these features.

## 6 References

[1] Basili, V.R., Weiss, D.M. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, 10(6), Nov 1984, pp. 728-738.

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

[3] Brooks, F.P.: *The mythical man-month : essays on software engineering*, Addison-Wesley, 1975.

[4] Chrissis, M.B., Konrad, M., Shrum, S. : *CMMI®: Guidelines for Process Integration and Product Improvement*. Addison-Wesley, 2003.

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

[6] Lott, C.M., Rombach, H.D.:Measurement-based guidance of software projects using explicit project plans. *Information and Software Technology*, 35(6/7):407–419, June/July 1993.

[7] Kruchten. P. : *Rational Unified Process - An Introduction*. Addison-Wesley, 2000.

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