Behavioral Monitoring of Software Projects

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul Calculatoare şi Tehnologia Informaţiei
de către

Ing. Ciprian-Leontin Stanciu

Conducător ştiinţific:	prof.univ.dr.ing. Vladimir-Ioan Creţu
Referenţi ştiinţifici:	prof.univ.dr. ing. Mircea Petrescu
	prof.univ.dr.ing. Dumitru Burdescu
	conf.univ.dr.ing. Ioan Jurca

Ziua susţinerii tezei: 25 Noiembrie 2011

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica” din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii mele în cadrul Departamentului de Calculatoare al Universităţii „Politehnica” din Timişoara. Mulţumiri deosebite se cuvin conducătorului de doctorat, prof.dr.ing. Vladimir-Ioan Creţu, şi domnului dr. ing. Dacian Tudor.

Timişoara, Ciprian-Leontin Stanciu
25 Noiembrie 2011

This PhD thesis was partially supported by the strategic grant POSDRU/6/1.5/S/13, ID6998, co-financed by the European Social Fund "Investing in People", within the Human Resource Sectorial Development Program 2007-2013.

Stanciu, Ciprian-Leontin

Behavioral Monitoring of Software Projects

Teze de doctorat ale UPT, Seria 14, Nr. 4, Editura Politehnica, 2011, 134 pagini, 38 figuri, 4 tabele.

ISSN: 2069-8216
ISSN-L: 2069-8216
ISBN: 978-606-554-375-1

Cuvinte cheie: Software projects, behavioral monitoring, progress forecasting

Rezumat,
Software projects are known for their high overruns in terms of execution time and budget. The particularities of software projects and also the amount of data produced by the development of the most problematic software projects (the large-scale ones), information regarding work progress mainly, make the monitoring and, consequently, the control of such projects extremely difficult. In this context, the main contribution of this thesis is the modeling of the monitoring process concretized in an integrated monitoring approach based on a monitoring framework for software projects, named the Behavioral Monitoring Framework. This framework enables the automation of the monitoring process, providing also support for the control of such projects. The proposed monitoring framework is centered on the modeling of the behavior towards work of the human resource involved in the project and its main feature refers to the dynamic perspective that it provides over the monitored project through progress forecasting.

Contents

List of figures

List of tables

List of acronyms

ALM – Application Lifecycle Management
BMF – Behavioral Monitoring Framework
Dim – Task Dimensions
DV – Diversification
EB – Estimation Behavior
EL – Elapsed Effort
ERP – Enterprise Resource Planning
ES – Estimated Effort
EV – Evolution
EVM - Earned Value Management
GSD - Global Software Development
IM – Implementation Moment
LEB – List of Estimation Behaviors
LEBC – List of Estimation Behaviors for Complexity
LEBS – List of Estimation Behaviors for Size
LEBT - List of Estimation Behaviors for Technology
LWB – List of Work Behaviors
LWBC – List of Work Behaviors for Complexity
LWBS – List of Work Behaviors for Size
LWBT - List of Work Behaviors for Technology
MAD – Mean Absolute Deviation
MAPE – Mean Absolute Percentage Error
MFE – Mean Forecasting Error
PES – Project Early Start
ST – Stability/Stagnation
TR – Trend
VL – Velocity
VP – Virtual Present
VTP – Velocity Trend Prediction
WB – Work Behavior
WBP – Work Behavior Prediction
WBS – Work Breakdown Structure
WES – Worker Early Start
WMAPE – Weighted Mean Absolute Percentage Error

Abstract

Software projects are known for their high overruns in terms of execution time and budget. The particularities of software projects and also the amount of data produced by the development of the most problematic software projects (the large-scale ones), information regarding work progress mainly, make the monitoring and, consequently, the control of such projects extremely difficult. In this context, the main contribution of this thesis is the modeling of the monitoring process concretized in an integrated monitoring approach based on a monitoring framework for software projects, named the Behavioral Monitoring Framework. This framework enables the automation of the monitoring process, providing also support for the control of such projects. The proposed monitoring framework is centered on the modeling of the behavior towards work of the human resource involved in the project and its main feature refers to the dynamic perspective that it provides over the monitored project through progress forecasting.

The first part of this thesis presents the current state of knowledge in the realm of projects monitoring, discussing the main techniques and methodologies employed today for project assessment. The second part of the thesis defines in details the proposed monitoring approach, presenting the Behavioral Monitoring Framework with its concepts, methodologies and modes. The third part is concerned with the primary validation of the proposed Behavioral Monitoring Framework on data that comes from the development of several real-world commercial software projects. The last part of this thesis focuses on the applications that can benefit from the implementation of the proposed monitoring framework.

1. Introduction

In this first chapter, the problems and challenges of software projects management are presented as a motivation for this thesis' objectives which are described afterwards along with the proposed solution and thesis structure.

1.1. Problems and challenges in software project management

Software projects differ from other types of projects in important ways, requiring a different approach from the working team, involving innovation and creativity, as shown in [26]. Consequently, there is a tendency to develop software in an artisan manner, which makes it mandatory to employ a well established project execution monitoring strategy in order to reduce the risk of resource waste, which otherwise is very high according to [39].

In the next subsections, we consider that a project is divided into tasks, the project's tasks being planned to be executed in a specific order, every task having an initial estimation of the effort required for completion. Moreover, the order and the effort estimations for the project's tasks are specified in the project execution plan.

1.1.1. Ideal scenarios

If all the tasks of the project are included in the initial plan, an ideal scenario is the one in which the initial task estimations and the order of project tasks don't change in time. There is no need for control in this ideal scenario, because every worker completes their tasks in the order and with the effort estimated in the initial plan. The project is completed as planned in this scenario.

A more relaxed ideal scenario is the one in which there is no change in the order of project tasks from the beginning to the end of the project and there are only changes in task estimations, while the provided task estimations are convergent. The required control is reduced to minimum, since workers finish their tasks in a predictable time. In such a scenario, even if the project does not complete as estimated in the initial plan, it completes in a predictable moment in time.

In ideal scenarios, the initial planning of the project is a lot more important than any other action taken during project development by the project management. If the initial plan fits the needs of the developed project's stakeholders, then only few and minor changes are expected for the specifications of the outputted software product. Such a philosophy was used for developing the resource, budget and execution time estimation methodologies that today are especially popular with software projects management.

However, such scenarios are hardly ever found in the real-world. The common scenarios are presented next.

1.1.2. Real-world scenarios

In real-world scenarios, many transformations occur in the project plan and these changes are difficult to foresee and even to monitor. In real-world scenarios, there are potentially many changes in the structure and costs of the project.

Changes can be grouped into two main categories, by considering the modification source factor: internal and external changes [91]. Fig. 1 shows the categories of changes. Next, we explain the meaning of category of changes.

Internal changes include: project requirement changes (for example, due date changes), and execution performance changes. Project requirement changes are made by the project manager and may be suggested by clients and suppliers. The project management can change the project structure (or graph) at any time. The project manager can add a new task, assign a task to another user, split a task, and modify the precedence of tasks. These changes have a great impact on the project work, changing the tasks completion time, and, consequently, the project execution time. In large-scale projects, a small change made to a task from the beginning of the project execution propagates in the immense project graph to the end of the project execution.

Changes regarding the execution performance refer to changes made in the project plan by workers, and include: estimation changes, task order changes, task ownership changes caused by worker's unavailability, and other modifications associated to the human factor. Workers can make changes regarding their assigned tasks. Workers can choose the order in which they execute their assigned tasks. Other workers' tasks may depend on the completion of some tasks assigned to some other worker that is unavailable for a particular time-span, so that these workers might have to wait unexpectedly, and to stop working on some of their assigned tasks. Workers can make changes on their task estimations, as well. These

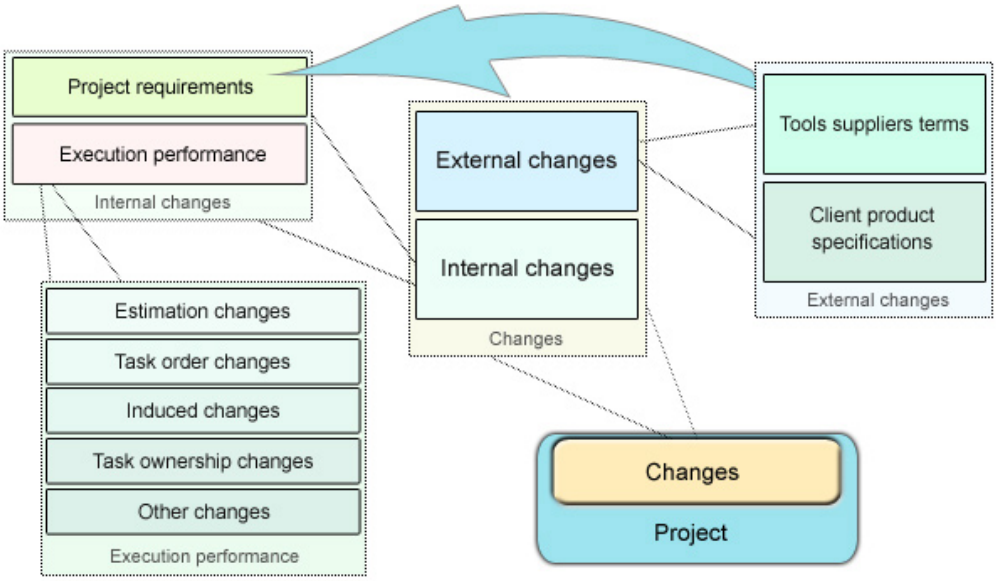

Figure 1. Project changes

changes might not converge, and have their own effect on other tasks and, accordingly, on the project graph. Furthermore, workers may influence each other in the re-estimation that they make. If a worker becomes unavailable, the project manager has to assign the tasks of this worker to other workers. These workers may not have the expertise to complete the tasks in the same manner as the unavailable worker. The execution performance sub-category of changes contains: the task estimation changes, modifications in the task order of execution, the task ownership changes, induced changes (changes resulting passively from other modifications), and other changes regarding the execution performance.

The external changes refer to changes suggested or triggered by clients and third party suppliers. A client may change final product's requirements. In such a scenario, the project manager might decide to make profound changes in the structure and costs of the project. Moreover, the client's indecision may put on hold some tasks, or even the project, as a whole. A supplier that doesn't provide the requested software tools needed in the developing of the project may induce delays in the project execution, and, consequently, changes in the project plan.

All these changes make the project monitoring activity extremely important especially for large-scale software projects that, due to their size, are prone to a great deal of changes.

1.2. Thesis objectives

The main objectives of this thesis are:

O1. Defining a methodology for processing the available information from progress reports in order for it to be more easily taken into consideration in decision-making: this is also important for using such information in forecasting.

O2. Defining a model for extracting the key information for elaborating a more accurate project status: this model must consider all the decisions taken during project development by project management as well as by the employed human resources.

O3. Defining a progress forecasting methodology: providing a dynamic perspective over project development progress is especially important for the automated analysis of a project status quo.

O4. Defining a model for analyzing the project status, providing both warnings and recommendations to the involved human resources: the automation of status analysis is especially important for large-scale software projects, where such an analysis can hardly be done by the project manager due to the large amount of data.

O5. Defining a monitoring framework for software projects, validating it on data from real-world projects, and analyzing the application domain of this framework: this framework is regarded as a collection of specialized models that work together to enable the automation of project monitoring.

O6. Defining the specifications of the software implementations of the proposed monitoring framework and implementing a software prototype of the framework to be used for validation purposes.

1.3. Proposed approach in a nutshell

The proposed approach is centered on a monitoring framework named the Behavioral Monitoring Framework. This framework has three component models that work together for rendering the image of the project's status quo. These models are: Project Status Model, Work Behavior Prediction Model, and Project Status Analysis Model [91] [92].

Fig. 2 shows how the Behavioral Monitoring Framework integrates in a project implementation environment. This environment has two spaces. One is the operational space, which refers to the activities involved by actually developing the project. The other is the managerial space which concerns the decision-making process within the project.

The Project Status Model gathers the progress information from the reports provided or available in the operational space, the main concern of this model being to collect the data using a pattern that accurately characterizes the development activity of software projects. The Project Status Analysis Model interprets the information available in the framework, offering support in the managerial space. A key feature of the Behavioral Monitoring Framework is forecasting, which offers a dynamic perspective over the project progress. In this context, the core of the proposed framework is represented by the Work Behavior Prediction Model which makes project progress forecasts based on the observed behavior towards work of the project team members.

1.4. Thesis structure

This thesis is structured on six chapters as follows:

Chapter 1. Introduction, presents the motivation, the objectives and the structure of this thesis. It starts by describing the idealistic project execution

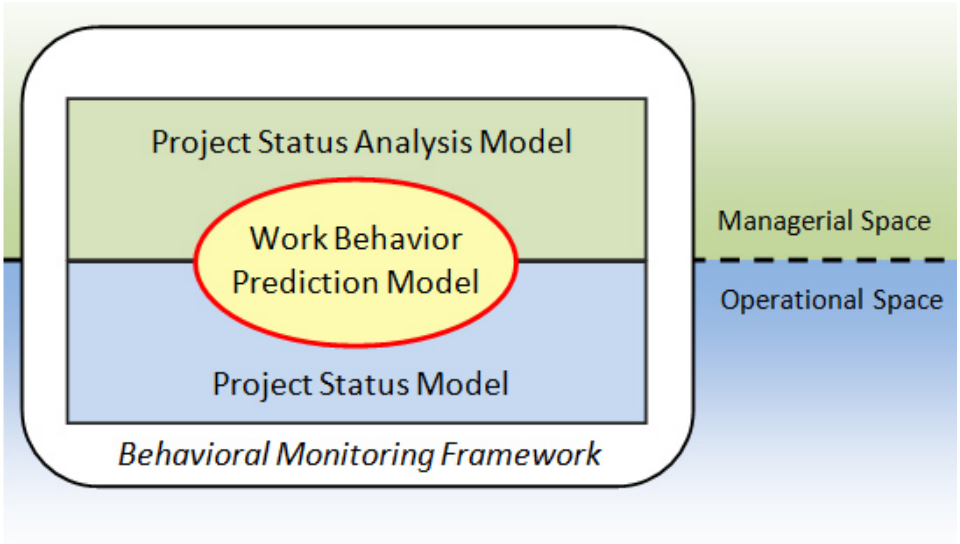

Figure 2. Behavioral Monitoring Framework in project implementation environment

scenarios in contrast to the real-world challenges. Afterward, the objectives of this thesis are presented, followed by the brief description of the proposed solution for the challenges of the monitoring process in software projects. Finally, the structure of the thesis is presented.

Chapter 2. Project Monitoring in Software Projects, presents the state-of-the-art in the realm of project management. At first, it presents an overview of the monitoring process with focus on the methodologies developed for software projects. At second, it describes the most popular project assessment techniques, including here the initial estimation methods for budget, execution time and required resources as well as the very few available methods for project assessment during project development. Finally, this chapter presents existing results regarding the estimation of project success.

Chapter 3. The Behavioral Monitoring Approach, describes in details the proposed approach for project monitoring. Actually, this approach is represented by a monitoring framework, which is a collection of interconnected models, each of these models being specialized on a particular process from the monitoring process group.

Chapter 4. Behavioral Monitoring Software Prototyping, discusses the main requirements for an application that implements the proposed Behavioral Monitoring Framework to fully benefit from its capabilities.

Chapter 5. Experiments on Real-World Data, presents the first experiments made on real-world data from commercial software projects development along with their results. The core of the proposed monitoring framework, which is the forecasting model (elaborated as part of this thesis) is compared to the forecasting model proposed by a very popular project management framework (Scrum) by using an evaluation methodology that is presented. The results are discussed at the end of this chapter.

Chapter 6. Behavioral Framework Applicability, describes the applications that can benefit from implementing the proposed monitoring framework. The application domain is not restricted to software projects as shown in this chapter. Versatility is an important trait of the proposed monitoring framework.

Chapter 7. Conclusions, presents the conclusions, focusing on the contributions of this thesis to the project management domain.

2. Project Monitoring in Software Projects

This chapter presents the state-of-the-art in the project monitoring of software projects. It starts by presenting why monitoring is such an important process especially for the problematic large-scale software projects, insisting on the fact that, nowadays, the project manager's reasoning is not added by well established monitoring and control methodologies. Even though many initial estimation techniques were developed, such methods can only be employed for the initial planning at project start, and in the context of project monitoring they have only the role of providing a baseline to this process. This chapter also presents several methods for project assessment during project development and a research regarding the estimation of project success. Finally, this chapter discusses the white spots in the current state of knowledge, introducing the main principles of the proposed approach to monitoring.

2.1. Overview

Projects have a limited life. They are initiated, work is done for their implementation and, finally, they are completed and closed. Projects are generally based on contracts and have a planned time for completion and a planned budget. Monitoring and control is a project management process group that has the role of assessment of project progress. The main objective of the monitoring process group is to create the context for the project management for taking the best decisions for the managed project, with respect to the defined time and budget constraints.

Software projects are a particular type of projects that, according to [18] and [8], are characterized by:

- abstract objectives and assets that can correctly be evaluated only by experienced project managers and leaders
- specialized and, thus, very expensive human resource involved
- usage of new technologies, that are not always well documented
- usage of software tools for the different processes that take place within the project

Having abstract objectives, software projects are difficult to evaluate concerning work performance. Employing expensive human resources, software projects must use such resources wisely and rework should be avoided as much as possible. In this context, monitoring in software projects is extremely important and difficult in the same time.

Large-scale software projects are even more difficult to monitor since monitoring requires the analysis of the large amounts of information produced during project development and available through the work progress reports provided by project team members [39].

One of the most important providers of case information on real-life software project failures and environments is The Standish Group, a management consulting company [Tsg2010]. Regarding one of the latest studies conducted by this company, "Chaos Summary 2009", Jim Crear, The Standish Group CIO, said

that these results revealed the highest failure rate in over a decade [98]. This suggests that software projects are more and more difficult to monitor and control as they grow larger.

Software tools for progress tracking were developed and are used within software projects of various sizes. However, such tools are generally more concerned with easing the reporting activity within the project and less with easing the understanding of the reported information for the project management.

2.1.1. The monitoring process

According to [73], the project execution must be actively, continually, and consistently managed in order to improve resources' efficiency and final product's quality.

Monitoring software projects is a very important since high failure rates are constantly reported for such project. In 1994, The Standish Group made a study that revealed interesting facts [97]. Although the validity of its results is questioned, this study was cited by several governmental reports on software development and was used as benchmark for several projects' estimation performances [48]. To continue with the results, according to this study: 52.7% of projects will cost 189% of their original estimates, the average overrun is 222% of the original time estimate, only 61%, in average, from the initially agreed features are delivered, only 16.2% for software projects that are completed on time and on budget, and 31.1% of projects will be cancelled before they ever get completed. After 1994, more studies were made on the subject of software projects. The results showed a fluctuant evolution to a better situation. Thus, two studies of the same The Standish Group, from 2000 and 2003, showed an increase of time overruns to 82%, in 2003, from a lower 63% in 2000. In addition, in 2003, only 52% of required features and functions reached the released product. This compares to 67% in 2000 [103]. We illustrate in fig.3 how time overrun (relative to the initial estimation) and the

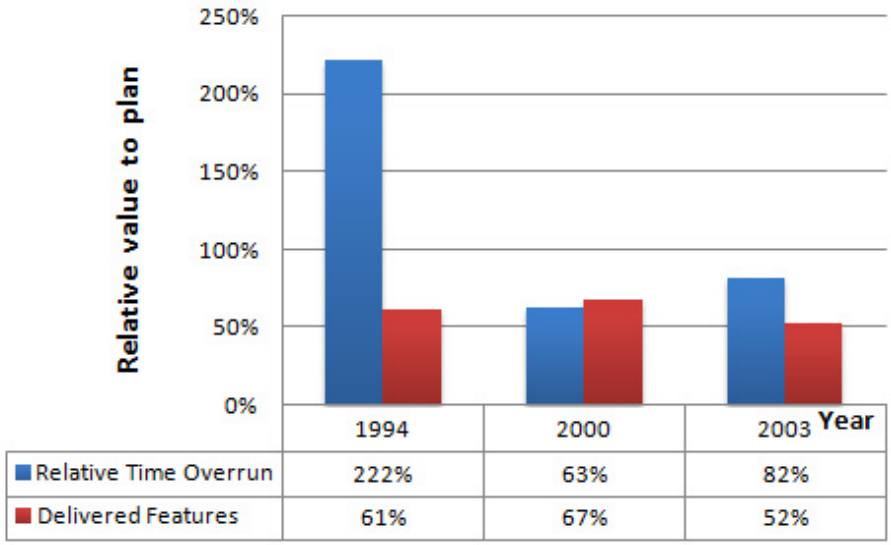

Figure 3. Statistics regarding software projects

percentage of features that are delivered from those established in the initial plan evolve in the years of the presented studies. Moreover, fig.3 suggests that there is an inverse correlation between time overrun and delivered features, as relative values to the values established in the initial plan, in that when the time overrun is high, the percentage of delivered features from those that were initially convened is low. The last study published in 2009 under the name "Chaos Summary 2009" showed the highest failure rate in over a decade, 24%, compared to an average of 18% for the last ten years [98]. A project is considered failed if it was cancelled before being completed or if it was delivered and never used, so that it is critical to find ways to minimize this high failure rate.

These studies revealed the main factors that make software projects vulnerable. Those factors are: the lack of users' feedback, incomplete requirements & specifications, changing requirements & specifications, lack of executive support, the lack of experience with a specific technology, the lack of resources, unrealistic expectations, unclear objectives and unrealistic time frames. By using a well established monitoring process, the impact of these factors can be diminished as follows. For example, to reduce the negative impact of the lack of users' feedback, monitoring might suggest the assignation of more testers to the project so that the potential bugs will be identified and fixed as early as possible.

Monitoring may be time consuming because of the data that must be gathered from the working team. The overhead introduced by the monitoring process must be as little as possible, so that expressions like "a barely sufficient process", or "a little bit less than just enough" are used in literature with regards to monitoring, as shown in [52]. Consequently, the monitoring process must be adapted to the complexity of the software project to which it applies. However, in our opinion, understanding just what's sufficient for any given project is a challenge.

For short-sized, simple and stable projects, the Waterfall Model (fig.4) [81] is a very effective project life-cycle according to [38]. In such a project, the events affecting the project are predictable, the tools and activities are well known and understood, and each completed phase is considered closed. For these software projects, the monitoring activity is simplified by the fact that the events that may affect the project execution are known from the start, so that it is clear what data should be gathered for the input of the monitoring process. Moreover, the management knows exactly what to look for in the monitoring activity. However, we believe that projects rarely follow the sequential flow.

Unlike the Waterfall model with its rigid structure and sequential flow, Agile methodologies are known for their effectiveness for software projects [30]. This effectiveness stands in the characteristics of the Agile approach that will be briefly presented next in relation to the monitoring process.

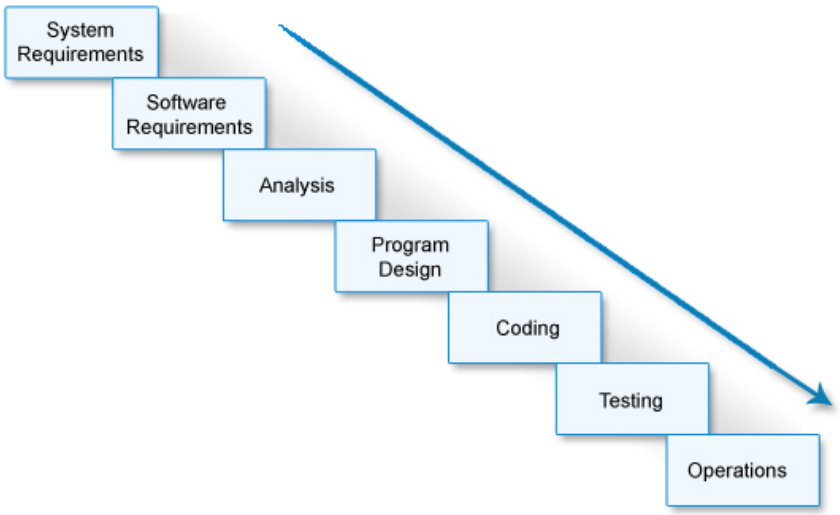

Figure 4. The original Waterfall model as presented in [Roy1970]

Agile represent a group of software development methodologies based on incremental development, that consider changes as a natural evolution of requirements and solutions and that put individuals in front of tools [30]. The main principles of the Agile approach are: rapid feedback, simplicity, dynamic perspective, and incremental change [14]. In the monitoring context, the most important is the principle that regards the rapid feedback which encourages the communication between project manager and project team members. Provided an enhanced communication within the project, the monitoring becomes more natural, better integrating in the general workflow. According to [76], the Agile approach encourages detailed monitoring and controlling within software projects. Moreover, the general attitude towards change, introduced by the Agile approach, positions monitoring as a key activity for the project management.

A survey presented in [78], showed that the projects that use the Agile approach have 15 to 23 % average gain in resource productivity, 5 to 7 % average reduction in cost and 25 to 50 % less time compared to the previous projects within the survey participating companies. Considering these results, and the fact that the monitoring process is the central part of the Agile approach, we can imply that a well established monitoring methodology is the key for minimizing the chaos that characterize software projects.

2.1.2. Traditional approaches and their deficiencies

Traditional project monitoring is made manually and implies regular adjustments and risk assessment, according to an established process protocol, as stated in [40].

In traditional approaches to monitoring, the data regarding the project execution is gathered manually, the managers and workers having to write specific reports from time to time, in which they must describe, for example, the progress

made and the challenges they ran into. However, data gathering is expensive being time-consuming, and it affects the human resources that have the smallest amount of time for being involved in such a process as shown in [39].

The data gathering is a very important task in the monitoring process, especially in the case of traditional approaches where the manager judgment is critical. In traditional monitoring, as marked in [39], there is no way to learn how to gather and analyze data without gathering and analyzing data.

There are four main principles for gathering data, according to [39]:

- the data is always gathered in accordance with specific objectives and a plan
- the selection of the data to be gathered is based on a model
- the impact on the entire organization must be considered
- the data gathering plan must have management support

Project management can benefit from forecasting especially in projects with high risks of time and budget overruns, such as the large-scale software projects. For this purpose, depending on the creativity of each project manager, economics specific forecasting methods can be used. Economics is a domain that employs forecasting on a wide-scale because of the need for making future more predictable. Forecasting methods that can be considered in project management can be the time-series extrapolation, and the technical analysis methods.

Time-series extrapolation relies on quantitative methods to analyze data for the variable of interest, according to [4]. Such methods are reliable and easily automated, so that they are widely used, especially for inventory, production forecasts and population forecasting according to the same [4]. In the case of software projects, the time-series extrapolation can be applied on series of project tasks effort estimations.

Several time-series extrapolation methods may be of interest: Exponential Moving Average (EMA), Autoregressive Moving Average Model (ARMA), Generalized Autoregressive Conditional Heteroskedasticity (GARCH), and Autoregressive Tree (ART).

The EMA method, referred in [21], uses weights for the historical data. The weighting decreases exponentially, giving more importance to recent observations. The ARMA method, mentioned in [57], uses weights for the historical data as well, but these weights follow a pattern which is dependent on the historical data. We consider in fig.5 an example for the prediction of a given effort estimation series using three prediction models: EMA, ARMA(1,1) and ARMA(2,2). For computing the forecasts presented in fig.5, we employed an open source prediction tool [27]. Fig.5 shows that using time-series extrapolation methods for making effort estimation predictions is inappropriate when no supplementary information exists to help the selection of the most suitable extrapolation method for such a purpose. As shown in fig.5, each method provides slightly different results. Moreover, the same model (ARMA), but with two different sets of parameters give significantly different results. Therefore, it is still the job of the project manager to identify the method that best fits to the context of the managed project. Considering that traditional monitoring approaches don't use well established methodologies, the project manager can use available data regarding the project or similar projects for the selection of a suitable prediction method and for identifying the corrections to be applied to the prediction results.

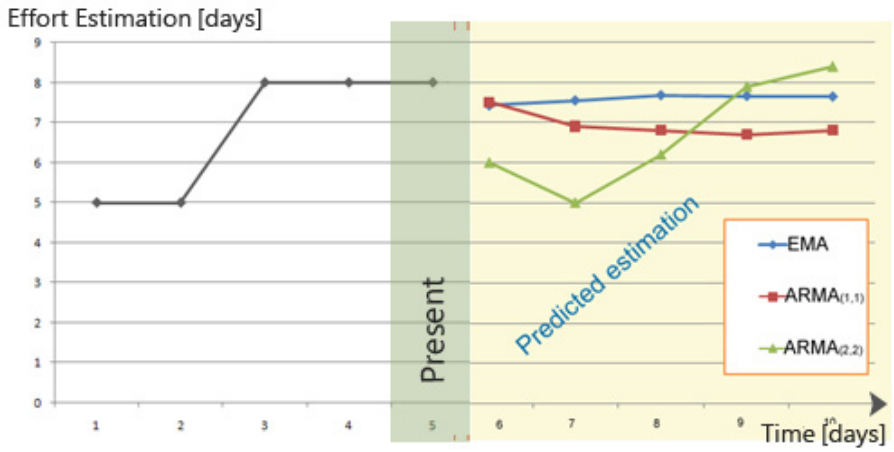

Figure 5. EMA versus ARMA in effort estimation prediction

GARCH is very similar in results to ARMA except when the time-series contains a seasonal component [34]. We tested this affirmation and the results are shown in fig.6. For computing the forecasts (optimistic, pessimistic and average), we used the same open source tool as for the previous example, [27]. As fig.6 suggests, GARCH performs worse than ARMA when the time series contain a seasonal component. In the case of project progress forecasting this is generally not a problem because seasonality is hardly ever a component of any effort estimation series, but when it is, ARMA is expected to make better predictions in terms of accuracy than GARCH [34].

Important candidates for the forecasting of effort estimation within a project are the ART models, which are a generalization of the standard autoregressive models. The tests performed on over 2,000 time-series data sets from the International Institute of Forecasters, demonstrate that ART models provide superior predictive accuracy than standard autoregressive models, as shown in [58]. Microsoft SQL Server 2005 introduced the ARTxp algorithm, which is based on ART, but is applied to multiple, unknown prior states, according to [61].

The other group of methods that can be useful in computing important indicators for project management is the technical analysis. Technical analysis is used in financial markets to forecast the future trend of prices through the study of past market data.

The technical analysis is based on the fact that all of the internal and external factors that affect a market at any given point in time are already factored into that market's price [59]. Regarding effort estimations, we assume that they incorporate all the external and internal factors that affect the project and the workers.

Using the analogies price-effort estimation for task completion, and market data-project data, technical analysis can be applied to available data to predict the effort estimation trend for a task, or group of tasks. A technical analysis method that can be used is the Bollinger Bands [16]. The purpose of the Bollinger Bands is to provide relative definitions for "high" and "low".

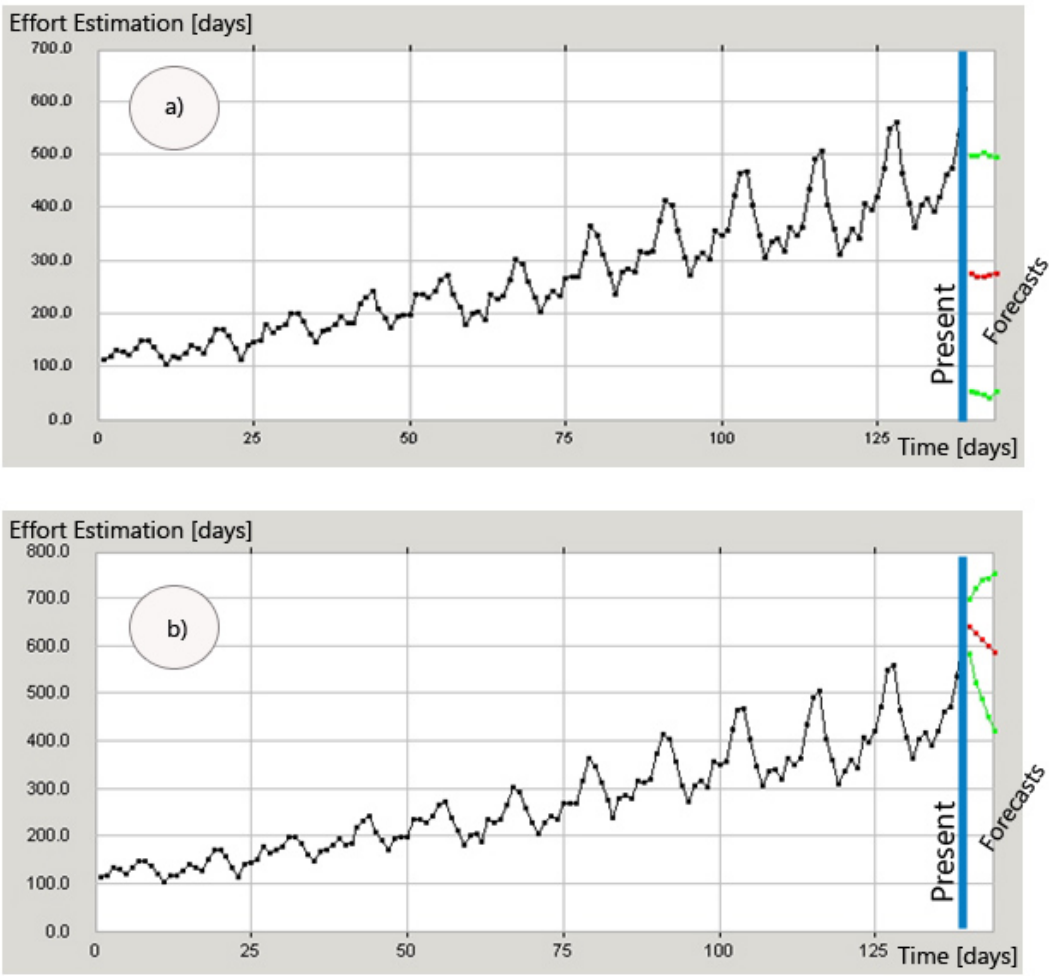

Figure 6. GARCH versus ARMA in effort estimation prediction; a) GARCH(1,1); b) ARMA(1,1)

As stated in [16], prices near the upper band are high, and prices near the lower band are low. Using the above analogies, when an effort estimation is near the upper band of the Bollinger Bands, this estimation is high, and when the effort estimation approaches the lower band, this estimation is low. We illustrated this in fig.7. The interpretation is that, when the estimation is high, a change to a lower value is expected for the estimated effort for task completion, and when the estimation is low, a change to a higher value is expected for the effort estimation. The result of using technical analysis methods is a predicted trend of effort estimations for a task or group of tasks.

Figure 7. Bollinger Bands for effort estimation trend prediction

Traditional approaches to project monitoring present a number of shortcomings especially in the context of large-scale projects. A drawback of traditional approaches refers to the great amount of information that project managers have to process when no well established monitoring methodologies are used. Another shortcoming of traditional approaches is that there are many methods to choose from when it comes to forecasting (adaptations of economics prediction methods) for example. However, experience is still needed on the part of the project manager since a selection of the methods that are actually suitable for the managed project is required.

Due to these important shortcomings, the traditional approaches to project monitoring are not suitable for large-scale software projects, where the activity of maintaining consistency among requirements, design, and implementation is a job for a superhuman according to [39]. As a result, new approaches to monitoring were developed as we will show in the next section.

2.1.3. In-time, automated, and adaptive monitoring

Considering the deficiencies of the traditional project monitoring process, and the statistical data regarding the evolution of software development projects offered in [97] and [98], new approaches to monitoring had to be developed. The modern approaches to monitoring software projects are adaptive, in-time, and automated processes.

The modern monitoring process has one or more of the above characteristics. For example, the monitoring process may adapt to the project to which it applies, it may make available the information in the first moment this information is available, and this process may be automated in that it is continuous and it provides critical information to the decision factors without this to be requested explicitly.

To apply a modern monitoring approach to a project execution, there must be an infrastructure for data gathering, so that project execution specific data is available to the monitoring process as soon as possible. Because of the high interdependency between data gathering and monitoring processes, and considering

the fact that the monitoring process is of higher complexity than the data gathering process, the modern approaches to project monitoring may integrate the data gathering process.

The modern monitoring process may include the methods used, for example, for forecasting in traditional monitoring approaches. The difference is that, in modern approaches, the outputs analysis may be automated, in-time and adapted to the specific of the project on which the monitoring applies.

The in-time notification is a characteristic of modern approaches to monitoring. The in-time notification is especially important in global software development (GSD). Typical characteristics of GSD projects (for example, geographical distribution and cultural differences of team members) bring challenges regarding communication, collaboration interdependencies, and knowledge management, according to [100].

An in-time notification system stands for enhanced communication enabling enforcing an Agile approach to monitoring, as described previously in this chapter. Large-scale distributed projects are required to be supported by integrated monitoring software to describe the project in order to assess its status and to elaborate early warnings in order to enable the consideration of the necessary actions in specific conditions. Moreover, we believe the project status and activities must be visualized in a unified way or in an aggregated form of report by the project manager, being able to retrieve detailed information only when they need them. According to [100], to successfully conduct a GSD project, the collaboration of all team members is necessary along with an in-time notification system. Therefore, the current focus of monitoring software, which is on the project manager, should be extended to all project team members [100].

Automated monitoring refers to a process that runs without the need of human handling. Modern monitoring processes are required to be automated in that they must continuously compute the values of some predefined indicators by using the available project data, being also able to send warning notifications to the involved workers and project managers.

The adaptive monitoring approaches refer to monitoring processes that are adaptable to the monitored project facts. The project facts may refer to the static project characteristics like the project type, the project size, the tools and technologies employed, the working team cohesion, size and expertise, the management experience of the project manager, and the importance of the project for the working team and client. Also, the project facts may refer to dynamic project characteristics such as the to-date progress compared to the planed progress, and the current estimations compared to the initial estimations of the effort needed for task, WBS (Work Breakdown Structure), or project completion.

We present next an adaptive project monitoring process together with its context (fig.8) [41]. In such a monitoring approach, based on experience and knowledge, the execution plan is developed and updated. The plan of execution contains the latest effort estimations for tasks completion. The information present in the plan represents a baseline for the monitoring process. The monitoring process triggers, for example, when estimation changes occur, an update of the execution plan, and adds new data to the knowledge base for future project planning. This process is continuous and it is interesting to analyze the evolution of the three blocks shown in fig.8: Measure & Analyze, Estimation & Plan, and Monitor & Control. The content of the Measure & Analyze block is quasi-constant in time. Generally, this block contains methods and algorithms, for measuring and analyzing the received or existent data, which do not change. However, improvements may be

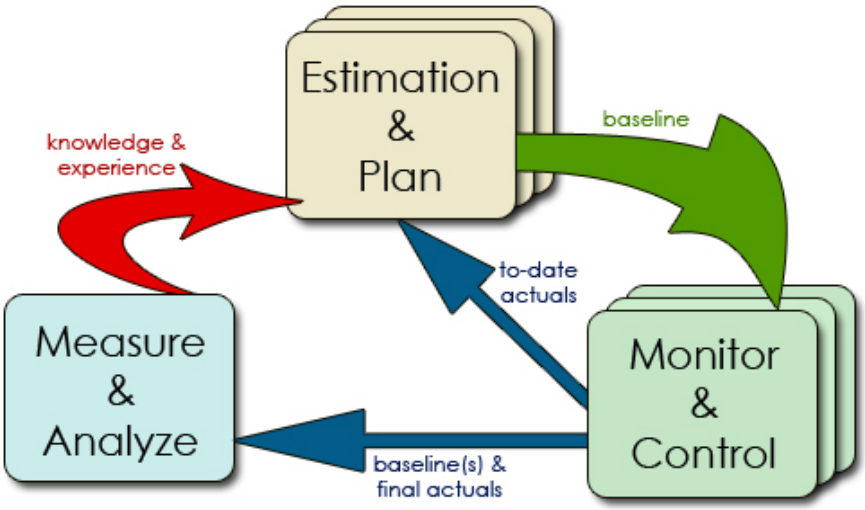

Figure 8. An adaptive project monitoring process [Hun2007]

taken into consideration even for these methods and algorithms. The execution plan may change in time taking into consideration the to-date progress and effort estimation changes, so that the Estimation & Plan block has several instances from the start to the end of the project execution.

Regarding the Monitor & Control block in fig.8, the dark-blue arrows tagged as "to-date actuals" and "baseline(s) and final actuals" create a feedback loop. This feedback loop adjusts the monitoring process using static and dynamic project characteristics as they were described earlier, so that the monitoring process presented in Fig. 7, is an adaptable process.

A best practice for monitoring and controlling the progress of software projects is considered to be the Earned Value Management (EVM) with its performance measurement approach [29]. Actually, EVM is technique for objectively measuring the project progress. In this context, there is a wide-spread idea that the monitoring process can be significantly improved if well established estimation methodologies and algorithms are integrated into this process, according to [41].

2.1.4. Project development data gathering

Data gathering must be done in order to monitor the execution of a project. Based on the gathered data, the project manager may decide upon the best directions for the project in order to minimize the overruns in terms of budget and execution time.

2.1.4.1. The importance of data gathering

Traditionally, project monitoring is made manually and implies regular adjustments and risk assessment, based on a given process protocol [40]. The data

gathering is a very important task in the monitoring process because it enables project managers to take specific corrective actions when certain situations occur.

In traditional monitoring, there is no way to learn how to gather and analyze data without gathering and analyzing data, so that it is very difficult to train project managers in other conditions than the real ones involving real-world projects, with their high budgets and risks, as marked in [39].

2.1.4.2. The data to be gathered for monitoring purposes

Generally, the data regarding the project execution is gathered manually, the managers and workers having to write specific reports regarding their work.

Data gathering is expensive and time-consuming, and it affects the busiest people, being even viewed as personally threatening as suggested in [39]. Consequently, the data gathering must be done so that the workers are as little disturbed as possible from their tasks while there is enough data for performing an effective project progress monitoring and assessment.

There are two possible approaches to data gathering, considering the meaning of data to be gathered: the code-centered approach and the worker-centered approach. In a code-centered approach to data gathering, the information used in monitoring is acquired from the code written by developers and refer to the evolution of code bugs, number of lines, code writing rate and other information of interest [37]. The main disadvantage of such an approach when used for project monitoring is the fact that not all the workers involved in the project write code. There are also testers, software architects, code designers and other members of the project implementation team. This is why, for monitoring purposes, a worker-centered approach to data gathering is more appropriate. In such an approach, the workers report specific information concerning project execution. Although the worker-centered approach is more suitable for project monitoring, it has its own disadvantage: the gathered data is subjective and depends on social and psychological factors that may affect workers' reports. Furthermore, in order to not affect project implementation, the specific information that workers are requested to report must be thoroughly selected in order to be very simple in type and little in amount. On the other hand, the gathered application must be suggestive enough for the project manager to be able to understand the real status of the managed project.

2.1.5. The importance of understanding the real status of a project

Software development companies generally develop several projects in the same time. Furthermore, it is not unusual for the workers to be involved in more than one project at a time. This is especially common when large-scale projects are developed, which require many workers involved in the implementation process. In this context, a worker might be requested to implement several tasks, maybe from different projects, in the same period of time. Consequently, the worker has to assign their own priorities to their tasks and to implement them accordingly. This is why the project status has to consider the decisions of workers regarding the execution of their own tasks beside the decisions of the project manager regarding managed project structure and costs (remaining effort for completion).

A certain importance in the understanding of the real status of a project has the initial plan against which the current state is compared. We believe that a realistic initial plan is the first step for an effective monitoring, being a baseline of all the changes that occur during project development. This is why, the initial estimation techniques are also important for monitoring. The most important such techniques are presented next.

2.2. Initial estimation techniques

In the process of project monitoring and control, a continuous progress assessment must be done for an effective project management [73]. To meet project deadlines and cost limits, using the assessment of the progress and the plan of the project, the project manager or supervisor may re-estimate several tasks. Good quality estimations are a must to keep the project on its tracks.

In 2007, a study was conducted regarding the way software engineers make their effort (and cost) estimations when little information is available [99]. According to the results of this study, when it is a lack of information, humans make assumptions that help them develop software effort and cost estimations [99]. Moreover, even though these assumptions are not always justified, they have a great impact on software effort and cost estimations.

Two experiments are described next [99]. Three different populations participated at these experiments: psychology students, engineering students, and engineering practitioners. The participants were provided a set of historical data and an estimation model, having to choose the way they make the estimations: by using their own judgment or by using the given model. These experiments were intended to test how accurate software engineers are at estimating future values given limited information (the first experiment), and how much engineers rely on historical data versus a cost model to perform cost estimates (the second experiment). The results from the first experiment showed that all three populations predicted values of every day events with relatively equal accuracy. The results from the second experiment showed that the responses from engineering students have a higher variance compared to the responses from practitioners. The results from the second experiment showed a tendency for overestimation from the engineering students compared to practitioners. Both populations tended to use their own judgment for making the estimations, generally ignoring the provided estimation model.

There are two main implications of the result of the presented experiments, according to [99]. At first, it was shown that students almost equal practitioners in estimation quality, although students tend to overestimate perhaps because of their lake of experience in working on real-world projects. Second, both populations were influenced more by historical information than by the answer provided by the provided estimation model.

The need for effort estimation models for open source software is argued next [6]. In open source software projects, most of the time is spent on fixing bugs, so that the existing effort estimation models are inadequate for such projects, according to [6]. An important challenge in developing an estimation methodology for open source software projects is how to find development data that can be used in the validation of a new such methodology.

In most engineering systems, historical information is used for effort and cost estimation for future projects, but, in most cases, especially for software

products, reliable data are difficult to find. According to [99], the research regarding estimation models should continue even though practitioners will not depend entirely on the answer provided by these models.

2.2.1. Human judgment in effort estimation

Human (expert) judgment is a wide-spread technique of effort estimation for software projects is presented in [46]. Regarding this, several studies showed that the estimators do not have much understanding of the cause-effect complexity of the software project environment, but expert estimations become more accurate when they include risk analysis in the estimation process [46]. However, there seems to be a close relationship between software tasks characteristics (size and type of modules subject to changes, for example) and expert estimation accuracy.

Studies from other domains than software showed that experts are performing better than models in a highly predictable environment, but worse in a less predictable environment [46]. The same studies suggested that experts perform better than models in short-term forecasting, but worse in long-term forecasting. Other results of these studies are about the opportunity of task decomposition for estimation purposes, which is not recommended because it will generate more information and this will only bring more complexity in the estimation process.

According to [46], the combination of expert judgment and effort estimation models may lead to more accurate effort estimations. The reason for that is the fact that humans and models have different strengths and weaknesses: the models have less bias towards too optimistic effort estimations, while the experts can identify new variables that might be relevant for the context of a particular project.

Better knowledge about human judgment strategies, known as heuristics, can be used to improve estimation and prediction in software processes, according to [47]. It is shown that the main condition for human judgment heuristics to perform well is that there is a fit between the heuristics and their environment. Moreover, it is argued that to select a proper estimation strategy, the information about the estimation uncertainty is essential [47].

According to [103], learning from experience is an important process in human judgment. The same paper illustrates the usefulness of experience, claiming that much of the gained experience has no value for future work. This is the case for software projects, since there are a lot of variables related to the software processes that can change from one project to another. The difficulty in working with historical data is that a change in conditions makes history invalid [103].

According to [46], the expert judgment estimation technique should be further improved through the cooperation of software estimation researchers with psychologists that can contribute with their own experience regarding the existing knowledge on human judgment.

2.2.2. Effort estimation by analogy

Effort estimation by analogy is an established method for software effort estimation according to [87] and it is mainly a data-driven method. As shown in [45], this method compares a target project with similar historical projects by using their common attributes for estimating the effort for the target project as a function

of the known efforts of the considered similar historical projects. Effort estimation by analogy can be used also for the effort estimation of project tasks, WBS or other elements at different levels of the project, feature, or requirement.

There are three basic steps required by the effort estimation by analogy method to estimate the effort for a given element (for example, project, work package, task) under estimation [45]:

Step1. find the analog elements from the historical data set (the similar projects) for the given element by using a set of common attributes and measures

Step2. determine the closest analogs to the given element

Step3. forecast the effort of the given element by using a function of the known effort required for completing the closest analogs

The effort estimation by analogy can be regarded as a meta-method, according to [45]. This is especially useful after a progress assessment for re-estimating the elements that show overruns at different levels of the target project (tasks, for example). A decision-centric process model of the effort estimation by analogy method from a decision making point of view is presented in [45].

Similarity is defined as Euclidean distance in n-dimensional space where n is the number of project features. Each dimension is standardized so all dimensions have the same weight [87]. This way, the known effort values of the nearest neighbors to the new project can be used as forecasting basis for this new project.

The results of an exhaustive search conducted to determine the consistency within and between the results in empirical studies of software engineering cost estimation, with focus on regression and analogy techniques are presented in [55]. The findings were that about 25% of studies were internally inconclusive, their conclusions being decisively influenced by the considered context. A conclusion regarding effort estimation by analogy was that there is approximately equal evidence in favor of, and against this type of methods.

2.2.3. Effort estimation using fuzzy logic

Algorithmic effort prediction models are unable to cope with uncertainties and imprecision characterizing the early cycles of the software projects life, as stated in [82]. Fuzzy logic based models are possible solutions to the limitations of the algorithmic effort prediction models, according to [82], [67] and [63].

Two estimation models based on fuzzy logic are presented in [63]. The two models differ only by the fact that the second model considers the also the methodology of the project in the estimation process.

The models proposed in [63] are evaluated on several NASA software projects considering four error metrics: VAF (Variance Accounted For), MAPE (Mean Absolute Percentage Error), VARE (Variance Absolute Relative Error), and Pred [50].

According to [63], the second model, which considers the project methodology (which is available in the public data sets of the NASA projects) is the best for the software projects taken into consideration, on the basis of VAF, MAPE, VARE, and Pred(25).

Fuzzy logic is powerful representation of imprecision in inputs and outputs that can be used for extending other techniques for software cost estimation, like analogy, neural networks approach and case based reasoning, as suggested in [82].

2.2.4. Effort estimation using artificial neural networks

Unlike regression models, the neural networks are not based on mathematical formulas, being able to take many shapes with learning, as stated in [10].

Even though extensive research regarding prediction models has been conducted at least in the last decade, the management of software projects still cannot be advised as to use one forecasting method or other, due to the contradictory results obtained in evaluations according to [10].

Artificial neural networks are computational models of nervous systems as shown in [68]. The neuron computes a weighted sum of its inputs and generates an output if the sum exceeds a certain threshold. This output then becomes an excitatory (positive) or inhibitory (negative) input to other neurons in the network. The process continues until one or more outputs are generated [10] [11].

The learning methodology is an important part of using artificial neural networks. There are many different learning algorithms. Feed-forward Multilayer Perceptrons are the most commonly used form of artificial neural network, although many more sophisticated artificial neural networks have been proposed [10].

According to [10], the studies regarding the use of artificial neural networks for software development effort prediction have focused mostly on the accuracy comparison of the models rather than on the suitability of the proposed approach for building software tools for effort prediction.

A comparison among several effort estimation techniques, including the artificial neural networks approach was presented in [11]. Two error metrics were used: MAPE and R^2. The results indicated that the considered neural network estimation model performs remarkably well, in terms of MAPE values, compared to the considered regression models.

Although the artificial neural network approach has demonstrated some advantages in certain circumstances, it cannot replace regression approaches, which are more practical. The neural network approach should be regarded as a powerful tool for the calibration of software effort estimation models, according to [78].

2.2.5. The neuro-fuzzy approach to effort estimation

The neuro-fuzzy approach is a very popular combination of soft computing methods, as stated in [82]. Soft computing can be regarded as the fusion of methodologies designed to model real-world problems that are very difficult to model mathematically. These systems are the ones that model the real-world and are of very interesting to the modern science, according to [64].

Neural network techniques are based on the principle of learning from historical data, while fuzzy logic is a method used to make decisions in an uncertain environment. Neuro-fuzzy systems combine the advantages of both techniques [56].

There are two ways in which the neural networks can be combined with fuzzy logic: fuzzy-neural networks (FNN) and neuro-fuzzy systems (NFS). FNN is a neural network that is capable of handling fuzzy information, while NFS is a fuzzy system enhanced with learning capabilities by incorporating neural networks [56]. There are two basic types of neuro-fuzzy systems: Mamdani Neuro-Fuzzy System and Tagaki-Sugeno, both presented in [3].

According to [71], Tagaki-Sugeno has some advantages over Mamdani like, for example, requiring a smaller number of fuzzy rules.

A neuro-fuzzy approach (using the Tagaki-Sugeno) is compared to artificial neural network only and fuzzy only approaches to effort estimation in software projects [56]. The results were compared by using the MAPE error metric and showed that the neuro-fuzzy system performs much better than the two other mentioned methods.

An experiment in which a neuro-fuzzy approach that uses Tagaki-Sugeno is compared to algorithmic models that use only LOC (lines of code) as input parameter is also presented in [71]. The comparison of the results was made by using two error metrics: MAPE and Pred and revealed that the neuro-fuzzy system has the lowest MAPE from the effort estimation models taken into consideration in this study.

The main benefit of the neuro-fuzzy approach refers to its good interpretability, due to the fuzzy rules. Another important advantage is that it can combine expert knowledge with fuzzy rules having the learning ability of neural networks into one general model, potentially with wide applicability range for software projects estimation [56].

2.2.6. Case-based reasoning

The case-based reasoning is a machine learning technique. The basic approach is that each completed project is considered as a separate case and added to a case base. Each case is characterized by a number of features which might be continuous, discrete or categorical. Example features might include the number of interfaces, the level of code reuse and the design method employed. A restriction is that these features must be known or estimated at the time of prediction [49].

The paper [1] presents a clarification of the case-based reasoning methods, which are: exemplar-based reasoning (where solving a problem is a classification task), instance-based reasoning (which uses concept learning), memory-based reasoning (where reasoning is regarded as a process of accessing and searching in "the memory" of cases), case-based reasoning (which is the typical case-based reasoning), and analogy-based reasoning (where the major focus of the studies concerning this approach has been on the reuse of a past case).

According to [1], the typical case-based reasoning contains the following processes (that can be regarded as steps):

1. Retrieve the most similar case or cases
2. Reuse the information in the most similar case to solve the problem
3. Revise the proposed solution
4. Retain the parts of this experience for further reference

An evaluation of a developed case-based reasoning model named ESTOR is presented in [10]. The results of this evaluation show that ESTOR performs very similar to a human specialist and significantly better than COCOMO and Functional Points on restricted samples of problems.

A challenging problem in case-based reasoning is the feature subset selection. Regarding this, several strategies for feature subset selection are examined [49]: random feature subset selection, multi-start steepest ascent hill climbing and forward sequential selection. As the authors explained, they restricted their choice mainly because other groups have had some success with these algorithms for finding good feature subsets.

As a search problem there are two additional issues according to [49] and [23]: representation of solutions and measurement of fitness. In the case of feature subset selection problems, the set of candidate features can simply be represented as a bit string, 1 for selected and 0 for excluded.

Fuzzy logic is especially useful for case-based reasoning because this approach is use analogical reasoning, which can operate with linguistic expressions. An example of combining case-based reasoning with fuzzy logic is described in [66].

2.2.7. COCOMO suite

The COCOMO suite of models is described in [15] and shown in fig.9. In the late 1970s and the early 1980s, the need for software estimation methods started to take shape. Back then, besides a number of proprietary estimation models, the Open Constructive Cost Model (COCOMO), one of the most frequently quoted algorithmic approaches was developed [74]. During the years, many other models were developed based on the COCOMO suite of models. Important ideas and improvements regarding these models are presented in this section.

To distinguish between the capabilities of different estimation methodologies is a difficult task. According to [101], such methodologies are learned from very small data sets, involving a notable risk of inadequacy. A model that tries to overcome this is the COSEEKMO model, presented in [53], which learns for small datasets by using a set of model generation techniques like local calibration, linear regression and model trees, each of which selected for the mitigation of the accuracy variability risk. COSEEKMO uses rejection rules, which have an important role in the selection criteria of the model parameters.

According to [53] and [43], cost models generally have many parameters, so that they tend to be too specified. According to the same papers, simpler models would probably provide clearer and more reproducible results.

In the original COCOMO model developed in 1981, a software project is divided into "components" that are estimated individually. The overall project size, that is the sum of the size of the components, is used to compute the overall productivity. The productivity is used as nominal productivity to estimate the effort for the individual components. However, the gain in productivity when working with small components is not explicitly present in these models. Considering this, a model for estimating incremental development effort is presented in [13].

Realistic cost models must use as inputs the quality of the product and the time-span for which the product will be on the market [5]. Another important input for a realistic cost model is represented by the cost of technical tradeoffs that must be done by the system's architect while designing the system for maximizing system's benefits. Consequently, an architectural approach is the key for realistic cost models, the software architect being able to understand the real risks in the system along with the cost of their minimization [5].

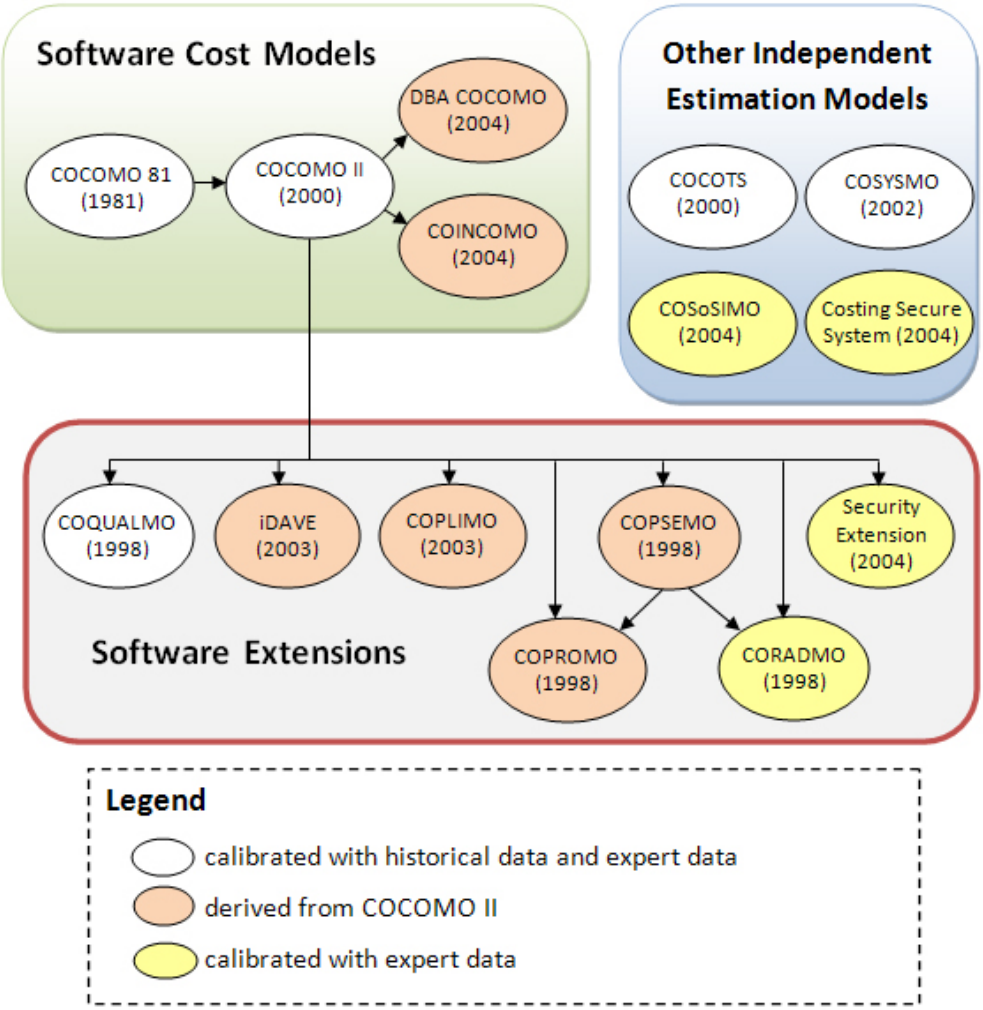

Figure 9. The COCOMO suite of models [Boe2005]

One of the major challenges of the estimation methodologies is how to combine results in an effective and meaningful manner, this being a challenge due to the diversity of the existing estimation methods [89]. Moreover, the ad hoc manner of data sets selection adds to the difficulty of this task, as shown in [51].

The COCOMO suite of models allows users to understand cost and schedule implications of their development and investments decisions. Enhancements for the COCOMO model are proposed in [88]. At first, genetic algorithms are used for providing a new estimation of the COCOMO model parameters. Based on NASA software projects and using genetic algorithms for estimating the parameters, two effort estimation models, only one based on ME, where ME represents the methodology (NASA projects database). As shown in [88], the model containing ME performed better that the one than didn't consider ME.

In 2000, a new version of COCOMO was released, that is COCOMO II [20]. The release of the COCOMO II created the need of a process for understanding COCOMO analysis in the context of the new COCOMO II estimation model. For satisfying this need, the Rosetta Stone was developed, which can be regarded as a process and a tool that addresses this subject, according to [77]. A calibration approach of the COCOMO II model is proposed in [22].

An interesting use of the COCOMO model is presented in [36]. This paper presents the Constructive SCORM Cost Model (COSCOMO). The algorithm behind this applies the concepts of COCOMO to SCORM development projects. SCORM is the acronym for Sharable Content Object Reference Model, which is a collection of standards and specifications for e-learning.

COCOMO measures the size of the project is lines of code. When the size of the project is measured in function points, COCOMO uses functional points to lines of code conversion methodology. However, "functional points" is the better metric for project size, according to [19]. The same paper presents a new model which uses function points as a direct input into the model, which is the f^2 COCOMO.

By experimentation, it was showed that software projects data can be analyzed on a programming language basis [19]. The different programming languages are reflected in the constants the employed model. This paper suggests that f^2 COCOMO is feasible.

An original idea was presented in [80]: COCOMO II combined with functional size measurement. It is argued that using an incorrect "lines of code" per "functional points" ratio as an input to the COCOMO II model can produce notable errors in the estimation process. The experiments revealed a considerable variation in the number of lines of code generated per functional point [80]. Consequently, a vulnerable part of the COCOMO II model (and of course of the first COCOMO model) is the function points to lines of code converter [80]. However, the so far proposed alternatives did not prove to enhance the produced estimations, according to [80].

2.3. Assessment during development methods

Considering the existing project management prediction methods, we can state there are many methods destined to forecast the resources required by a project, also known as estimation methods, which are used at the beginning of the project, and which resulted from documented research. In the same time, there are just a few prediction methods that can be used during project development to support decision making. One is the Velocity Trend prediction which is a part of the popular Scrum Agile framework [31], and which is offered in most ALM tools, such as CollabNet Team Forge [25] and IBM Rational Team Concert [42]. In Scrum, velocity means how much effort a team or a developer can handle in a defined amount of time [83]. Knowing the velocity trend and considering the estimated effort required for a task, the completion date of that task can be forecasted. This is, in short, the Velocity Trend prediction methodology, which is a generic remaining effort forecasting methodology that can be used during the implementation of any type of project. Being used during project development, such a method is regarded as a dynamic forecasting method.

The development of dynamic forecasting methods is difficult due to the data required in the validation process: data from project development progress reports. We believe both the amount and the confidential nature of such data made it very

difficult for the researchers to come with new and reliable forecasting methodologies.

A distinct approach to dynamic assessment is the scenario-based analysis, which is centered on the system dynamics representation of the project development process, described in [84]. System dynamics enables the building of project execution scenarios using the gathered data, this being a wide spread representation in the field of software project management, as shown in [54]. These scenarios can further be simulated for understanding and forecasting future project evolution.

In addition to the scenario-based analysis, a number of models were developed for the monitoring process of the project management. A generic monitoring model for dynamic systems is presented in [33]. This model does not refer directly to software projects, but can be regarded as a generic monitoring model that can apply to software projects as well, since the project development can be seen as a dynamic system.

A more particular monitoring model is presented in [79], as a part of an integrated project management model. The monitoring subsystem is represented using system dynamics. An original approach to software monitoring and control is presented by in [9]. This approach includes two types of models: the project model and several scenario models. The scenario models, which mainly describe the occurrence of particular events that may affect the execution of the project, are applied to the project model, so that the resulted structure would describe the project status and dynamics when those particular events occur.

A model for knowledge acquiring, which is an important part of the project monitoring process, is presented in [12]. Several models using system dynamics are described in [69], such as the basic stocks and flows of software development, the positive and negative impact of overwork, and the negative effects of errors and rework in software development projects.

All this models can be considered during project development for different parts of the monitoring methodology in a scenario-based data analysis.

2.4. Estimation of project success

The estimation of project success is especially important for the stakeholders. Both internal (owners, employees, managers) and external stakeholders (customers mainly), must know at least at key moments of the project execution if the project will be completed as described in the project plan, within the established time and budget.

The probability of project success is discussed in [28], in the context of Enterprise Resource Planning (ERP) projects. In this paper, a solution for maximizing the probability of project success is proposed. This solution combines: the COCOMO II reference model, a Monte Carlo simulation for cost parameters uncertainty, and a concept of probability-based projects portfolio management.

The results presented in [28] showed that, when managing projects as portfolio, the probability of success was almost 100% under effort constraints and almost 90% under time constraints. The same paper proposes two portfolios: the first having "very high" cost parameter values, while the second had "very low" such values. The conclusions were that most of the COCOMO II parameters (cost drivers) can be adjusted in a way that maximizes the probability of success and that the

probabilities of success for projects that have high failure risks are greater when these projects are managed as a portfolio.

However, the solution described in [28] just proposes a way for making a better initial estimation of the project effort for completion and it is not applicable during the project execution. For estimating the project success even during the project execution, a relevant set of metrics is required. Besides software metrics, the Bayesian classifier can also be used in the estimation of project success.

Next, software projects metrics and the Bayesian classifier method for estimating project success will be discussed.

2.4.1. Software metrics

A realistic image over a software project can only be rendered by using a validated selection of software metrics, according to [44]. Quantitative expressions of the data resulting from projects development add clarity and simplicity to the assessment of project status and goals, metrics having the benefit of helping organizations and individuals in the process of self-discovery, as shown in [75].

The most important benefit of using metrics is the decision-making support they provide to project management. According to [75], the information support systems, which had focused on informational management in the past years, have transformed to management information systems lately.

A taxonomy of software metrics is defined in [102]. The metrics in this taxonomy are organized in groups for client satisfaction, product, process, organization and drivers or psychological parameters of involved personnel. The same paper defines metric relationship rules, the so called Metrel rules. There are two interesting such rules, as illustrated in [102]:

- the time derivative of a valid product metric is a valid process metric
- the time derivative of a valid process metric is a valid organization metric

According to [102], Metrel has the main benefit of providing a methodology for offering all the information required by management, development staff and customers in a single view. In our opinion, the problematic large-scale software projects require greater visibility for their inner activities in order for the project management to see much clearer the first signs of project deviation from plan. The Metrel rules presented in [102] represent an important step forward in this direction.

In [65], another set of software metrics, YEEM, is proposed. The YEEM metrics set is structured on product, resource, risk, technology, environment, and prediction. The main aim of this set of metrics is to provide better and more useful results from software development prediction studies and models, according to [65]. According to the [65], when the YEEM set of metrics is used the results of the experiments are consistent and reproducible.

2.4.2. Bayesian classifier

The software projects are considered to be successful if their cost and duration are within the estimated ones and the quality of the resulted product is satisfactory. The estimation of the final status, which is successful or unsuccessful, of projects by applying Bayesian classifier to software development metrics values is described in [2].

The naive Bayesian classifier is one of the most common approaches to classify categorical data into several classes. The variables are risk factors or metrics and a class denotes the status of a project [2]. The status of a project can take one of two values: successful and unsuccessful. There are three viewpoints when evaluating a project success, according to [2]: the quality of product, the cost of development and the duration of the analyzed project.

In order to obtain high estimation accuracy in what regards project success, the selection of the metrics to be used in evaluation is a critical point. Two selection methods are considered [2]: the first is the selection of metrics made by experts and the second is the selection made by statistical tests. Moreover, an experiment was conducted, using several software projects and metrics data in an organization of a certain company. The result showed that the statistical tests are better metric selectors than experts and that the Bayesian classifier is suitable for project success estimation [2].

2.5. Concluding remarks

This chapter presents the current state of knowledge in the monitoring of software projects. In summary, there are many methods developed and used for the initial estimation of the effort and cost required by a software project. Moreover, many improvements were proposed for these existing methods. However, there are only few methodologies that can be used for understanding where the project is heading during development. These methodologies are not only few in number, but also very different in approach and address particular activities of the monitoring process. Combining these methodologies for an effective project monitoring is very difficult, no attempt being recorded in this direction.

Most of the assessment methods presented in this chapter, like estimation by analogy, the neural networks approach, case-based reasoning, fuzzy and neuro-fuzzy approaches, the COCOMO suite of models, were not developed for being used for the assessment of the project progress during project development. However, we believe that a continuous assessment of the project progress has to be done during project development for the project manager to be able to manage the limited life of a project in an effective way. On the other hand, building a methodology that can be used for dynamic project assessment, meaning during project progress, is a difficult task. In such a methodology, progress data (like successive remaining effort estimates for project tasks) has to be interpreted in a way that humans (e.g., project managers) are able to understand and use for making conclusions regarding trends (e.g., how fast tasks are completed). Meanwhile, progress data generally refer to a large variety of tasks, tasks from different projects, tasks that differ from each other from their size to the main technology involved. This means that progress data cannot be regarded as a uniform set of data that can be used directly for understanding trends in project progress. We believe that such an understanding is very important for taking early corrective actions and, consequently, the key for an effective monitoring and control in the most challenging types of software projects, which are the large-scale software projects.

Another important aspect regarding an effective monitoring and control refers to data gathering. The assessment methodologies presented in this chapter fail, in our opinion, on one important aspect. They do not consider that within a project, the project manager is not the only person that makes decisions. We

believe that all project team members make decisions, even though, at different levels within the project. For example, a developer has several tasks from different projects assigned and the last progress report sent to the project manager by this project team member shows a remaining effort for each assigned task that comes from the project managed by that particular project manager. Even if this is institutionally correct (a project manager requests reports regarding only the tasks of their managed project), the project manager doesn't have a clear understanding over the true status of the managed project. This is where the decisions of the project team members come into the scene. Each worker involved in the projects of an organization prioritizes his or her work, and some tasks come before others in their sequence of work. By using the current methodologies that were presented in this chapter, the project manager is not aware of those decisions and skips them when deciding upon the required corrective actions and this might concretize in erroneous decisions. In the problematic large-scale software projects, this shortage of information reflected in wrong and ineffective corrective actions can cause projects to fail, situation that, as shown at the beginning of this chapter, is more and more frequent.

Finally, another very important aspect especially for the monitoring and control of the most problematic software projects, the large-scale ones, which produce enormous amounts of data (e.g., there are many human resources involved and many tasks for which progress is reported regularly), is the possibility for automation. However, the current methodologies, presented in this chapter, don't allow for such automation, leaving all understanding and reasoning demands to the project manager.

In this context, the present thesis proposes the development of an integrated monitoring methodology destined to be used during the development of the problematic software projects. This methodology allows for process automation being formally defined as a collection of models that work together for solving the critical problems of the existing project monitoring and assessment methodologies described above, which are data gathering, progress trends understanding and the automation of the whole monitoring activity. This methodology along with its primary validation and application domains will be described in the next part of the thesis (Chapters 3, 4 and 5).

3. The Behavioral Monitoring Approach

This chapter presents our approach to an effective project monitoring. This approach is centered on the concept of work behavior which is also described in this chapter. The Behavioral Monitoring approach is concretized in a framework that contains models based on algorithms and equations, allowing for the automation of the whole monitoring process.

3.1. The Behavioral Monitoring Framework

The behavioral monitoring approach to project monitoring is implemented by a framework, named the Behavioral Monitoring Framework, which is a collection of three interconnected models, each of which specialized on a particular action performed in the monitoring process: data gathering, forecasting and analysis. In this first paragraph, we present in the form of an overview, the main motivations and ideas on which the approach to monitoring relies. Also, we describe the structure of the Behavioral Monitoring Framework to create the context for defining, further in this chapter, the component models of this framework.

3.1.1. Overview

How to keep projects on track is a major concern for project management which is the main responsible for both project success and failure according to [72].

In the process of project monitoring and control, a continuous progress assessment must be done for an effective project management, as argued in [73]. Basically, project management should monitor what every project team member does throughout project development.

In Scrum, which is a very popular project management framework, there are short daily meetings in which every project team member answers three questions [31]: what they have worked since the last meeting, what they intend to do before the next meeting, and what is keeping them from reaching their full efficiency [83]. We believe this is done for two main reasons, both regarding project management responsibilities. The first is to help project management understand how project team members work and their attitude towards work. The second is to help project management to influence in some way the observed work behavior of project team members in order to maximize the chances for project success. During project development, project management takes multiple corrective actions to maintain project on track. Most of the time, the target of those corrective actions is the human resource involved [32]. Moreover, one of the most important outputs of the monitoring process may be the information to be learned about the project team members working behavior, considering the individualities and the team, according to the same [32]. For all the above reasons, work behavior is a central concept in the project monitoring and controlling processes, and, consequently, very important for project management.

Project management is not the only discipline for which the understanding of human behavior is important. In economics it was developed a branch, behavioral economics, which combines psychology with economic analysis in order to improve decision making, generating theoretical insights, making better predictions of specific phenomena, and suggesting better strategies, according to [24].

Monitoring and control are critical process groups in project management [72]. According to [41], including a well defined estimation methodologies and algorithms as part of the monitoring and control process may lead to significant process improvements.

For enhancing project management efficiency, we developed a new approach to monitoring that involves the utilization of the monitoring framework that we propose, destined to the monitoring of software projects. Because it is based on the modeling of the behavior towards work of the project team members, we refer to the proposed monitoring approach as the behavioral approach to monitoring. Moreover, we named the underlying framework of this approach as the Behavioral Monitoring Framework. The utilization of this Behavioral Monitoring Framework in project monitoring represents the behavioral monitoring approach.

Because The Behavioral Monitoring Framework is fairly complex, before defining in detail each of the component models, we present next the structure of the proposed framework along with the informational flows that exist between the component models represented as black-boxes, for creating a general view over each model's position within the proposed framework and approach.

3.1.2. The structure of the framework

The proposed monitoring framework is presented in fig.10 [92]. It contains three models: the Project Status Model, the Work Behavior Prediction Model, and the Project Status Analysis Model. These models are described in detail in the further sections of this chapter. At this point, we disclose only the interconnections among the component models, which are regarded here as black-boxes, and the connections between these models and the project management environment in terms of informational flows. Next, we describe how each model integrates in the structure of the Behavioral Monitoring Framework, presenting for each its inputs and outputs.

As shown in fig.10, the Project Status Model has several inputs that come from outside the framework: project structure and remaining effort for each project task (meaning the relations between project tasks and their current remaining effort), the assignation of the work (meaning the tasks assigned to each worker involved in the monitored project) and the current time (meaning that the project status knows about the present time). The Project Status Model has also an input that comes from inside the framework, which is the predicted evolution of the remaining effort for each task that is considered in the project status. The Project Status Model outputs the current and a predicted project status, which further be used as inputs by the Project Status Analysis Model, and the micro and macro-universes (which refers to how tasks are assigned to workers and how tasks are structured in the managed project) which keeps the Work Behavior Prediction Model updated with respect to this information. The Work Behavior Prediction Model has two inputs from outside the framework: the effort estimation history and the elapsed effort history which relate to the internal input that comes from the Project Status Model (micro and macro-universes, as shown in fig.10). Finally, the Project

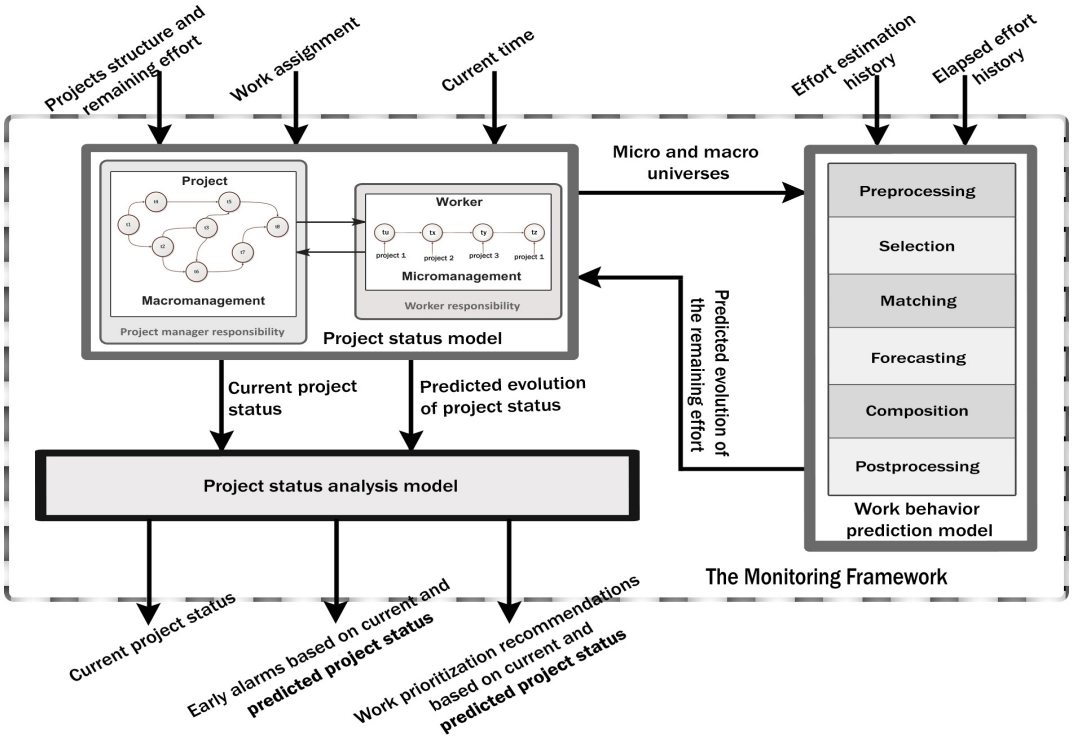

Figure 10. The proposed monitoring framework

Status Analysis Model uses the inputs provided by the Project Status Model, with the help of the Work Behavior Prediction Model, providing the current status of the monitored project in a human readable format, early warnings regarding existing or predicted project execution problems, as well as recommendations regarding work prioritization individually for the human resource involved in the project.

Before proceeding with the presentation of the component models, we introduce very important concepts and ideas for our approach to monitoring, like Work Behavior, which is the key concept of the Behavioral Monitoring Framework, and a classification of the project status accuracy levels discussing the suitability of each accuracy level for different types of projects, focusing on the problematic large-scale software projects.

3.2. Modeling work behavior

During development, large-scale projects produce a large amount of information that project management should process in order for it to take the best decisions for the project. At least in software projects, such information is collected and stored by the tracking tools that are generally used during project development. However, processing such enormous amount of information is extremely difficult. In this context, we propose a more concise representation of the information collected during project development that we named Work Behavior, illustrating also its benefits for project management.

3.2.1. Identifying work behavior

While understanding simple information requires only the observation of the data behind it, understanding complex information, consisting in large amounts of data of various types, requires a-priori processing and analysis, according to [39]. This is the case of the information regarding project team members' behavior towards work, which is very important information for project management in order for it to make aware decisions during project development.

3.2.1.1. Where to look for work behavior

In large-scale projects, ALM software tools are used for implementing tasks' lifecycle. Such tools are CollabNet Team Forge [25], IBM Rational Team Concert [42], and JIRA [7]. These tools enable project team members to efficiently communicate and correlate their actions towards fulfilling project's objectives, as stated in [85]. As expected, these software tools have large databases containing a wide range of information.

We believe that the large amount of information available for the ALM tools, which are widely used in software development organizations, hides something very valuable for the project management's decision-making process: project team members' behavior towards work.

In large-scale software projects, tasks have a certain lifecycle which is established by project management or by a higher managerial entity (e.g., from the organizational level) at project initialization. During project development, tasks are created, assigned, opened for work, stopped, resumed, completed, re-opened, re-assigned, and closed.

At project start and during project development, project management decides upon the creation of tasks. After a task is created, the task will be provided with an assignee, established by project management. From when the task is started, the assignee's behavior towards work, concerning that particular task, can be observed through periodical inquiries over his or her work progress.

During task implementation, in large scale-software projects, it is very likely for the assignee to stop working on a task and to start or resume working on another one, which has, for example, a higher priority. Actually, this will happen with high probability for several times before the assignee completes the implementation of a task. Finally, when assignees consider their work done for a task, they set the task to "completed". As long as a task remains in this state, no observations are available anymore regarding assignee's behavior towards work in relation to this task.

If the evaluators (e.g., testers, project manager) of the completed task consider the task is incomplete or that the resolved task has other flaws, the task is re-opened. If the initial assignee is considered apt for solving the identified problems, the task is given back to this assignee. In this case, observations regarding this worker's behavior towards work in relation to this task are further available again, until they set the task back to "completed". Otherwise, if the task is re-assigned to other worker, this task for its new assignee can be viewed as a new task on which the next reports on work progress will further provide information regarding the behavior towards work of its new assignee.

3.2.1.2. Work progress inquiries

Work progress inquiries refer to the reports regarding progress that project team members are requested to provide in a form and with a frequency established by the project management.

A very important decision of project management is the one that concerns the information that project team members are required to provide for identifying own work progress. This is a difficult decision since it has to conciliate two opposite requirements: one is the high informational needs of project management for aware decision making and the other is the need of using most of the working time of the project team members for actually developing the project. Consequently, it is very important for the project management to understand what amount of data is enough for a satisfactory understanding of where the project is heading.

Today, ALM tools are widely spread especially in organizations that develop large-scale software projects. These tools enable project team members to provide information regarding work progress more easily (e.g., by electronically filling a simple form in a web based application). In this context, the project management can ask for more progress information without the fear of disabling project team members from their assigned work for too long.

Generally, progress information is requested on a task-basis and refers to the estimation of the total effort needed for a task completion, the effort spend working on a task, or the remaining effort for completing a task.

For observing work behavior, project team members should provide one of the above information regarding work progress that is the remaining effort for completing a task, with a defined frequency (for example: at the end of each day, workers should provide the progress information for their assigned in-work tasks). This is not difficult to do when the reporting process uses an automated tool that provides all the context information (e.g., assigned in-progress task names and descriptions) and requests workers to provide only the critical progress information.

3.2.1.3. Observing work behavior

When progress information is available for project tasks on a regular-basis, charts like the ones presented in fig.11 can be built.

For example, the charts in fig.11 refer to a task that has an initial effort estimation of 7 days. The information regarding work progress is available on a daily-basis, at the end of each day. The task is in-work for 14 days, in the 14-th day being completed.

Fig.11.a shows that, for the given task, in day 1 (when the task is started), the remaining effort is 7 days. In day 2, the same remaining effort is observed. This suggests that the task assignee didn't work on the task in day 2. At the end of day three the task has the same remaining effort as in the previous day. At the end of day 4, the remaining effort is 6 days, showing that 1 day of work was spent on this task. At the end of day 14, the task is completed, a remaining effort of 0 work days being reported.

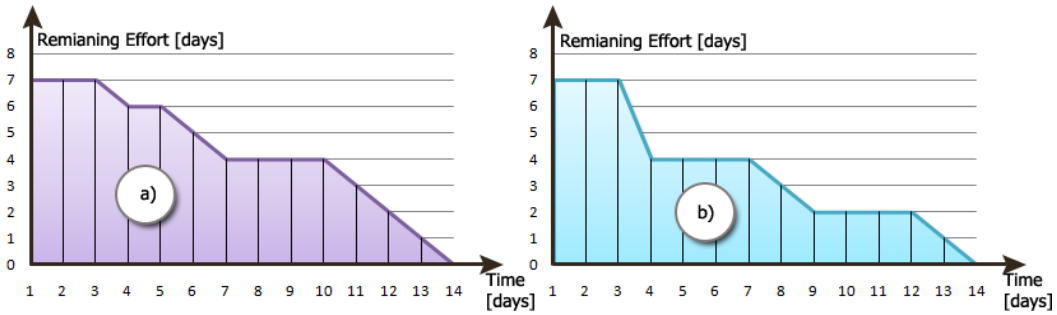

Figure 11. Work progress for a task: a) without explicit effort re-estimation and b) with explicit re-estimation

Generally, to have a clear understanding of the work progress, task assignees should provide two types of information: effort re-estimation (if applicable) and the elapsed effort. The effort re-estimation represents the amount of effort considered necessary, at a given moment in time, for completing a task. On the other hand, the elapsed effort is the amount of effort spent so far working on a task. Unlike the effort re-estimation, the elapsed effort is a fact. To reduce the reporting requirements for project team members, the effort re-estimation and the elapsed effort for the project tasks can be combined into only one observation: the remaining effort.

Coming back to fig.11.a, although we may understand that no work was spent on the task in the second days, there is another possibility: there was work spent on the task, but it covered some effort re-estimation that the assignee considered for the task. For example, during day 3, the assignee decides to re-estimate the effort required for task completion from 7 days to 8 days; at the end of day 3 however, because the assignee spent 1 day of work for the re-estimated task, the reported remaining effort is 7 days again. This alternative explanation exists because of the fact that the remaining effort for a task might contain re-estimations of required effort for completion.

Fig.11.a doesn't show any explicit effort re-estimation, because at the end of each day, the difference between the new remaining effort and the one of the end of the previous day does not exceed 1 day of work.

Unlike fig.11.a, fig.11.b shows explicit effort re-estimation. At the end of day 4, the remaining effort is 4 days, while at the end of day 3 it was 7 days. Because one cannot spend 3 days of work during only 1 day, this suggests that during day 4 the assignee re-estimated the effort required for task completion.

We believe that the behavior of project team members towards work can be observed in such data as the one used in rendering the charts in fig.11.

In terms of work behavior, the work progress in fig.11.a can be described as follows. Task development stagnates in days 2, 3, 5, 8, 9, and 10. In days 4, 6, and 11, the assignee resumes task work (after a stagnation). In days 5 and 8 the assignee stops task work (without completing it). Regarding the work progress shown in fig.11.b, task development stagnates in days 2, 3, 5, 6, 7, 10, 11, and 12. The assignee stops task work in days 5 and 10 (without completing it), and resumes task work in days 4, 8, and 13.

Although this is valuable information overall, it is very difficult to understand the meaning of these numbers (values on day indexes in our case) without proper modeling and analysis methodologies.

3.2.2. Modeling work behavior in real-world

In a project, there are complex tasks as well as simple tasks, tasks that require more time to be completed and tasks that require less time for completion. The differences among tasks and the large amount of data make the job of project management very difficult. We needed to find a way to characterize, in a normalized form, the progress observed for project tasks, so that we developed for this a set of metrics, named Behavioral set of metrics, based on which we define the concept of Work Behavior.

In the next section, we will define Work Behavior along with all the underlying concepts and metrics in the Behavioral set involved in this definition.

3.2.2.1. Definitions

In this section, we define, at first, the basic concepts with which the Behavioral set of metrics operates. At second, we define the metrics of the Behavioral set. Finally, we define de concept of Work Behavior, which uses the so introduced Behavioral set of metrics.

3.2.2.1.1. Basic concepts

The basic concepts that will be further used in the definitions of the metrics in the Behavioral set as well as in the definition of Work Behavior are the remaining effort and the history of remaining efforts for a task.

Definition 1 (Remaining Effort). Remaining Effort (RE) for a task Θ is an amount of work considered necessary to be spent for completing task Θ.

Definition 2 (History). History (H) for a task Θ is a chronologically ordered set $\{RE_i: RE_i$ is the Remaining Effort in day i for task Θ, i $\in [d_s, d_c]\}$, were d_s is the start date of task Θ and d_c is a defined date after d_s, being defined on this set a chronologically order relation C, where $(RE_i, RE_j) \in C$ if day i chronologically precedes day j.

As Definition 1 and Definition 2 suggest, the concepts of Remaining Effort and History have only sense in relation with a task. Consequently a History, as defined in Definition 2, contains remaining efforts provided for only one task (the task on which it is defined), ordered chronologically by the date when they were provided.

3.2.2.1.2. The Behavioral set of metrics

The Behavioral set of metrics contains three metrics: Stagnation, Diversification and Velocity that are all computed on a History provided for a task. Consequently these metrics are actually computed for that task. The metrics in the Behavioral set will be presented next.

Stagnation (ST)

Stagnation computed for a task can be regarded as the fraction of the total time passed from first starting a task to a defined moment in time, but not later than task completion, in which the assignee doesn't spend effort on the task. Stagnation is defined next.

Definition 3 (Stagnation). Stagnation (ST) computed for a task Θ is the probability that, given the History H for the task Θ, two consecutive History H elements show the same Remaining Effort.

Equation (1) shows the Stagnation (ST) computed on the History H (for task Θ).

$$ST = P_{H(i)=H(i+1)} \quad (1)$$

As suggested by (1), Stagnation takes values in the interval [0, 1]. Considering that historical information is available on a daily-basis, a Stagnation of value 0 means that from the start to the end of the observation period, the assignee worked on the task every day. Meanwhile, a Stagnation of value 1 suggests that no progress was logged for the task in the observation period.

Diversification (DV)

When a project team member stops working on a task (without completing it), maybe to move working on another, that team member diversifies his or her work. With this meaning, we define next the Diversification metric for a task.

Definition 4 (Diversification). Diversification (DV) computed for a task Θ is the probability that, given the History H for the task Θ, exactly two of three consecutive History H elements show the same Remaining Effort.

Equation (2) shows the Diversification (DV) computed on the History H provided for task Θ.

$$DV = P_{H(i)=H(i+1)\neq H(i+2)} + P_{H(i)\neq H(i+1)=H(i+2)} \quad (2)$$

Simplifying, Diversification for a task represents the fraction of the total time passed from first starting the task to a defined moment in time, but not later than task completion, in which the assignee stops or resumes the work on that task.

As suggested by (2), Diversification takes values in the interval [0, 1]. A Diversification of value 0 means a low fragmentation of the task work from the start to the end of the observation period. Meanwhile, a Diversification of value 1 suggests a high fragmentation of task work. Considering that historical information is available on a daily-basis, a Diversification of 1 means that the assignee resumed or paused its work on the task every day in the observation period.

Velocity (VL)

Velocity for a task can be regarded as the speed with which the task progresses to completion from when the assignee starts the work on the task, to a defined moment in time not later than task completion.

Definition 5 (Velocity). Velocity (VL) computed for a task Θ with its History H is the mean difference between the consecutive elements of the History H.

Equation (3) shows the Velocity (VL) computed on the History H (for task Θ).

$$VL = \overline{H(i) - H(i+1)} \quad (3)$$

Unlike Stagnation and Velocity, Diversification theoretically can take values in the interval (-∞, +∞), since there is no limit in the differences that may be between consecutive history elements' remaining effort. Also, please note that the remaining effort might increase from one history element to another when an important effort re-estimation to upward occurs.

A positive Velocity value near 0 suggests that the task's speed to completion is very low. A Velocity value near 1 means that, the speed to completion is very high when no effort re-estimations were made. Of course, Velocity values of over 1 are possible when re-estimations are made to downward by the assignee. In the meantime, negative Velocity values are possible when upward effort re-estimations are made by the assignee.

As suggested by Definitions 3, 4 and 5, the metrics in the Behavioral set (Stagnation, Diversification and Velocity) are computed for a task considering the History, as defined in Definition 2, for that task. Having defined these metrics, we will proceed to the definition of Work Behavior, which is also computed for a task.

3.2.2.1.3. Work Behavior

The definition of the Behavioral set of metrics has an important role for introducing the concept of Work Behavior. The definition of Work Behavior is presented next.

Definition 6 (Work Behavior). Work Behavior (WB) computed for a task Θ with its History H is a triplet (ST, DV, VL) composed of the Behavioral metrics values computed for that History H.

As suggested in Definition 6, the Work Behavior is related to a task and is computed starting from that task's History (as defined in Definition 2) using the Behavioral set of metrics defined in the previous subsection.

To illustrate the utilization of Definitions 1, 2, 3, 4, 5, and 6, in the following section we will show examples on how the Work Behavior for a task is computed on project development data.

3.2.2.2. Using Work Behavior on project development data

Next, we will show how Work Behavior is computed for three tasks: task A, task B, and task C, which describe three real-world inspired situations. Work progress, as reported for those tasks, is illustrated in fig.12.

For task A (fig.12.a), there are 17 history elements (for 17 days). There are 16 pairs of consecutive history elements, and 15 triplets of consecutive history elements. The remaining effort is the same for the following pairs of consecutive history elements: (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,9), (9,10), (10,11), (11,12), (12,13), (13,14), and (14,15), which are 13 of the total of 16 pairs. Consequently, using (1), ST = 13/16 ≈ 0.81. To continue, the consecutive history element triplets (1,2,3) and (14,15,16) show a work diversification attempt (as

described in the previous subsection). Consequently, according to (2), DV = 2/15 ≈ 0.13 (15 is the total number of consecutive history element triplets). Finally, using (3), VL = (3 – 0) / 16 ≈ 0.19. The high value of ST suggests that task A was

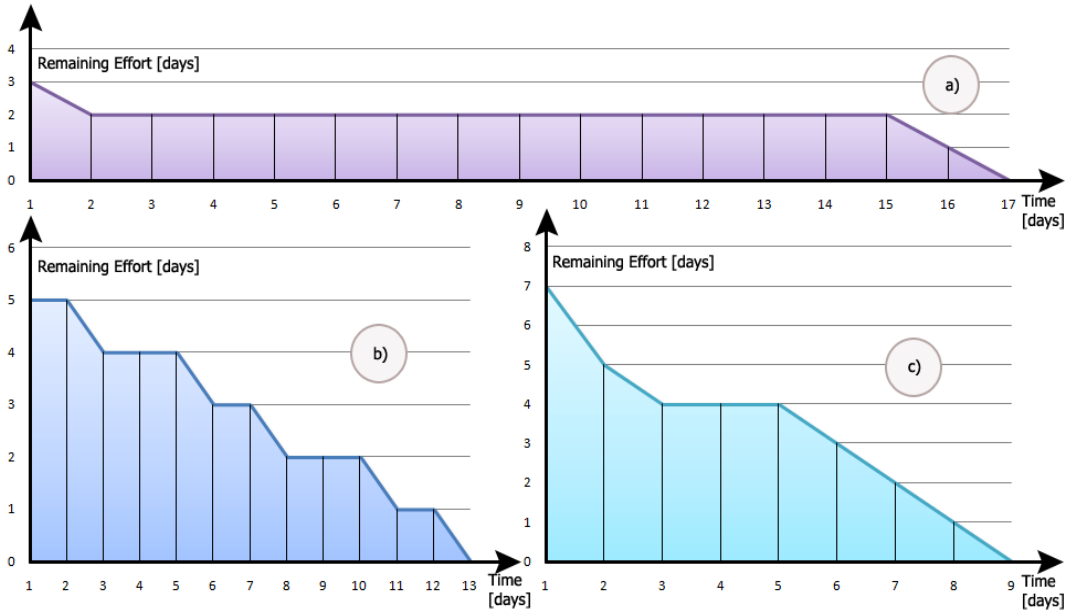

Figure 12. Real-world work progress for 3 tasks: a) task A, b) task B, and c) task C

considered a low priority task. Considering also the low value of DV, this meaning a low work diversification, we conclude that the assignee spent most of the observation period working on other tasks. This conclusion is confirmed by the low completion speed shown by the low value of VL.

For task B (fig.12.b), there are 13 history elements, for 13 days. There are 12 pairs of consecutive history elements, and 11 triplets of consecutive history elements. The remaining effort is the same for the following pairs of consecutive history elements: (1,2), (3,4), (4,5), (6,7), (8,9), (9,10), and (11,12), which are 7 of the total of 12 pairs. Consequently, using (1), ST = 7/12 ≈ 0.58. The consecutive history element triplets that show work diversification are: (1,2,3), (2,3,4), (4,5,6), (5,6,7), (6,7,8), (7,8,9), (9,10,11), (10,11,12), and (11,12,13), 9 of the total of 11 triplets. Consequently, according to (2), DV = 9/11 ≈ 0.82. Finally, using (3), VL = (5 – 0) / 12 ≈ 0.42. The high value of DV and the value of ST (which is greater than 0.5) suggest that task B was frequently paused and resumed by its assignee. Continuing with the interpretation, such Work Behavior component values suggest that the assignee found the task uninteresting, requiring such a high diversification. The value of the last metric, VL, which is near 0.5 comes to support our previous conclusion: the assignee had no problem in completing this task without explicit upward effort re-estimations ("explicit" having the meaning presented in the previous section).

For task C (fig.12.c), there are 9 history elements (for 9 days). There are 8 pairs of consecutive history elements, and 7 triplets of consecutive history elements. The remaining effort is the same for the following pairs of consecutive history

elements: (3,4) and (4,5), which are 2 of the total of 8 pairs. Consequently, using (1), ST = 2/8 = 0.25. The consecutive history element triplets (2,3,4) and (4,5,6) show a work diversification attempt. Consequently, according to (2), DV = 2/7 ≈ 0.29 (7 is the total number of consecutive history element triplets). Finally, using (3), VL = (7 – 0) / 8 ≈ 0.88. The low values of ST and DV suggest that task C was considered a high priority task. As shown in fig.11.c, an explicit effort re-estimation is done in day 2, in which the remaining effort decreases from the previous day with more than 1 working day. Overall, the high completion speed shown by the high value of VL supports the conclusion that task C was treated as a high priority task by its assignee.

3.2.2.3. Reconstructing work progress history from Work Behavior

As shown in the previous subsection, historical information regarding work progress can be translated into the significantly more concise representation which is Work Progress. In this subsection, we present the reverse action: how Work Behavior can be translated back into work progress history. A very important application for this reverse operation is forecasting.

Work Behavior representation is a normalized representation of work progress histories, since it doesn't consider any of the characteristics that make tasks different, like estimated complexity or size. As shown in the previous subsections, Work Behavior concerns only the way project team members spend effort on their assigned tasks. This means that work progress histories of very different tasks assigned to the same worker, can be translated into Work Behavior elements that can further be compared among each other.

Let's consider, for example, an in-progress task that has a remaining effort of 8 working days. Let's also consider that, by using a forecasting methodology based on Work Behavior, the resulted predicted Work Behavior for the given in-progress task was (0.5, 0.66, 0.5), meaning ST = 0.5, DV = 0.66, VL = 0.5. Project management wants to see the predicted work progress evolution for the next 4 days, this being the prediction time-span T. Because the time-span T considered for the prediction (reconstruction) is 4 days, and given (1), (2), and (3), project management should expect ST × T = 0.5 × 4 = 2 days in which the assignee doesn't spend any work for target task, and DV × (T-1) = 0.66 × 3 ≈ 2 days in which diversification will be observed (a day with no work progress after a day with recorded work progress or vice-versa). Also, the project manager should expect an elapsed effort for target task during the prediction time-span of VL × T = 0.5 × 4 = 2 work days.

Fig.13 illustrates the reconstruction of work progress history from the Work Behavior given as example: (0.5, 0.66, 0.5). At first, step I (fig.13.a), we know that the current remaining effort for target task is 8 days (day 1 in fig.13). Also, we know the time-span for reconstruction is 4 days (the last day of this interval being day 5 in fig.13). At second, step II (fig.13.b), we draw the Velocity Trend Line based on VL (0.5) considering that it contains the point with coordinates (1, 8) in fig.13.b. We find that on day 5 (which is the last day of the reconstruction interval) the remaining effort for target task is 6 days. We'll not stop here with the forecasting process because we believe that of interest is not only the destination, which is the remaining effort in day 5, but also how the worker gets there. Fig.13.c shows the reconstructed shape of work progress history for target task. It contains 2 days without work spent, and 2 days in which diversification is observed. Even though

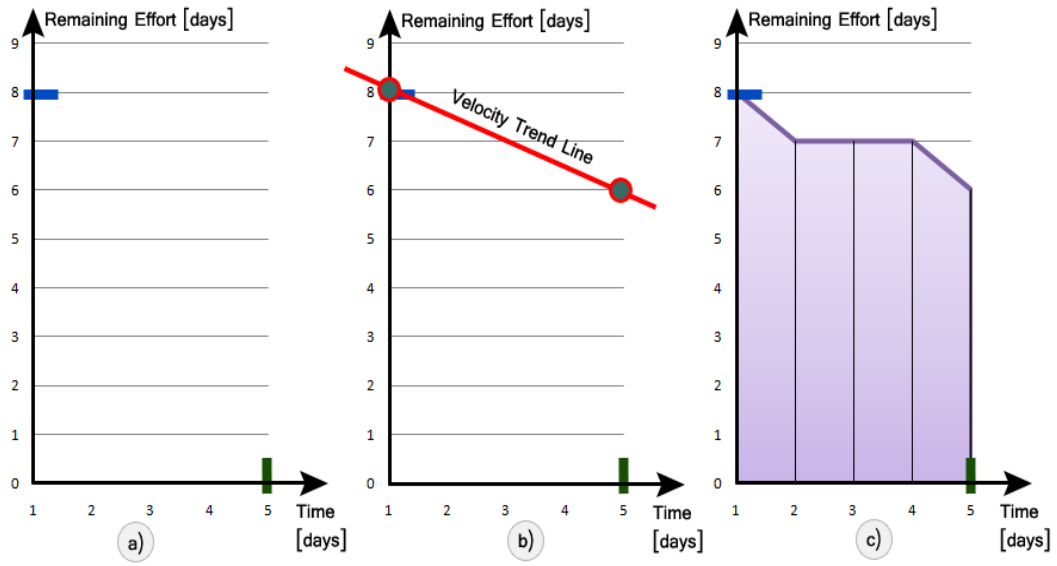

Figure 13. Reconstructing work progress history from Work Behavior : a) step I, b) step II, and c) step III

this representation (fig.13.c) isn't unique, it must be seen as a more descriptive representation of Work Behavior. For example, fig.13.c suggests that, sooner or later, in the given time-span, task's assignee would spend 2 days without working on that task. Knowing this, if the target task has to be completed as soon as possible, such work stagnation being not allowed, the project manager is able to early take the necessary actions to avoid such a situation.

3.2.3. Benefits of modeling work behavior

Understanding the behavior towards work of project team members is one important task of project management for several reasons: project management assigns the tasks to project team members, project management decides upon incentives, and also takes the corrective actions needed to get the managed project on track.

The application of our proposed Work Behavior representation of work progress histories on projects development data helps project management on three main directions: project status analysis, work progress forecasting, and human resources evaluation. Each of those three directions is described in the next subsections.

3.2.3.1. Project status analysis

Project status is identified and analyzed with an established frequency during project development. Generally, project management uses only the newest information available regarding project tasks estimates and facts to produce the

status of the managed project. Unfortunately, such information shows only a static picture of the project status, without any explanations regarding how the project came to such a status.

To understand project status evolution, project management would have to analyze work progress histories for each task of interest, which would be very difficult considering the large amount of data.

The Work Behavior values for a task can be regarded as a concise representation of that task's work progress history. This representation makes work progress more transparent to project management. At a project level, considering the large number of tasks, such a representation provides project management with valuable information regarding what happened between consecutive project status meetings, without being required to analyze lots of historical information.

An example of how project management can benefit from using Work Behavior is the following. Let's consider task A in fig.12.a and that the project status is built in day 10 (see the time axis). At that particular moment, only the first 9 history elements (remaining effort reports) regarding the progress of task A are available. Using (1), (2), and (3), ST = 7/8 ≈ 0.88, DV = 1/7 ≈ 0.14, and VL = (3-2)/8 ≈ 0.13. Knowing these values, project management understands that the work on task A might be far from completion. Of course, such a conclusion cannot be derived from the information that, in day 10 (the day in which the project status is built) the remaining effort for task A is only 2 work days.

3.2.3.2. Work progress forecasting

Forecasting work progress enables project management to take early corrective actions based on past experience. However, the past experience is not easy to understand when it is represented as work progress histories for tasks with different sizes, complexities, or main technologies used for their implementation.

We developed the Work Behavior representation of work progress histories for making past experience more accessible to project management. Tasks can be compared in terms of Work Behavior regardless of their type, complexity, and size, this being very useful for forecasting, as we will further show when we define the Work Behavior Prediction Model, later in this chapter.

3.2.3.3. Human resource evaluation

Another benefit of using Work Behavior is that it helps project management to evaluate project team members.

Work Behavior doesn't have good or bad values taken individually, but good or bad values taken together, for a defined target of evaluation. It is project management's role to define the evaluation target and to interpret the Work Behavior in order to assess each team member part in the project's road to success.

An example of how to use Work Behavior in human resources evaluation is the following. Let's consider that the behavior towards high priority tasks represent the target of evaluation. Also, let's consider two workers, A and B. Worker A is the assignee of a high priority task, task A, and worker B is the assignee of other high priority task, task B. Tasks A and B are completed and Work Behavior is computed for those two tasks. Let's consider that, for task A, ST = 0.8, DV = 0.2, and VL = 0.2, while for task B, ST = 0.2, DV = 0.3, and VL = 0.9. Because a high priority task

must be completed as fast as possible, a good Work Behavior is one with low ST (meaning little work progress stagnation), low DV (meaning little work diversification and more work spent on that task taking also into consideration the low ST) and high VL (meaning high speed to completion). Consequently, for such an evaluation target, worker B is better than worker A. In the mean time, for the same Work Behavior values for task A and task B, but for an evaluation target as: behavior towards very low priority tasks, when there are many other tasks with higher priorities, where a good Work Behavior is one with high ST, low DV, and low VL, worker A is better than worker B.

3.2.4. Conclusions

We have developed the Work Behavior as a significantly more concise representation for work progress histories for enabling project managers to understand more easily such information. We have presented where work progress histories can be found in software projects and the difficulties that project management faces when trying to understand the large amounts of information produced in the project development process. We have shown how Work Behavior maintains all the relevant meanings of the work progress history on which it is computed, which enables project management to easily understand the dynamics of the work spent for the project without analyzing large amounts of information. Also, we illustrated with examples the benefits of using the representation that we propose for Work Behavior regarding project status analysis, work progress forecasting, and human resources evaluation. Such benefits distinguish Work Behavior as a central representation of how a project progresses to completion. We further use this concept and representation, which is Work Behavior, as base for the Behavioral Monitoring Framework that we propose.

3.3. Project status accuracy levels

Elaborating the project status for large-scale software projects is a very important task of the project manager due to the high need of control of such projects. Depending on the strategy used for elaborating the project status, the project manager has more or less information that they can further use in their decision-making process [94].

3.3.1. Project workflow and project status evaluation

A project's workflow follows the process groups of project management as defined in [72]: initiation, planning, execution, monitoring and control, and closing. Moreover, every process group has its own workflow. Next, we briefly present the process groups and their workflow in relation to project status in the context of large-scale software projects, the most problematic software projects. The process groups and their implications to project status are illustrated in fig.14.

3.3.1.1. Initiation

This is the process group that takes place generally only once, at the beginning of the project. The main workflow here concerns the project manager and

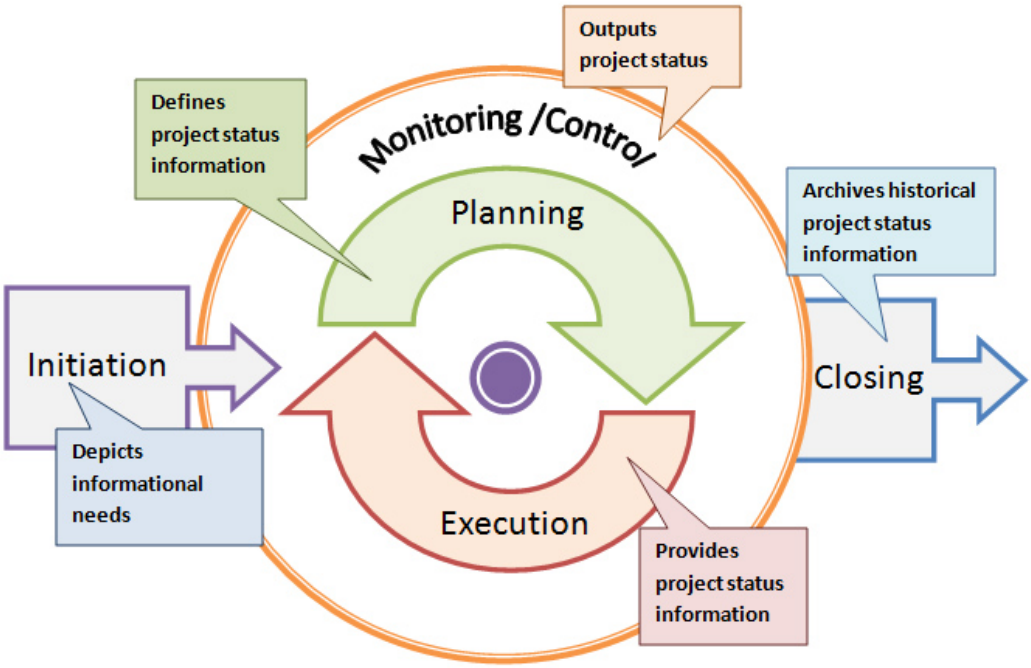

Figure 14. Project workflow and project status

refers to the identification of project stakeholders. Those are all the persons and institutions that have an interest of any type regarding the project.

Identifying project stakeholders is an important task since project management has to understand and meet all the expectations that concern the project. Especially for large-scale projects, there are many stakeholders, with divergent expectations.

Although this process group doesn't output any decision regarding how project status will be reported, it has a very important role in the grasp of the informational needs concerning the project.

3.3.1.2. Planning

Planning is a recurrent process group within a project. The workflow here refers to collecting project requirements, defining project scope, creating project WBS (Work Breakdown Structure), defining activities and constraints, defining the need of resources (including human resources), assigning defined activities (tasks),

estimating activities duration and budget, and many other activities concerning project planning.

In this process group, the project manager decides, maybe by discussing with the most important project stakeholders, upon the information to be presented in project status reports and upon the frequency of these reports by creating the so called communications plan. Establishing the structure and the frequency of project status reports has an important impact on how and how often project team members report their own work progress. Software projects and especially those of large-scale dimensions already contain lots of other activities than those that regard directly the development of the project like: change requests that follow established protocols, business trips, trainings or documentation on new technologies, team-building activities.

Considering this, it is advisable for project management to reduce as much as possible the number and the content of the work progress reports that are requested from project team members in order to let them focus as much as possible to actually developing the project. Of course, project stakeholders (like sponsors, for example), generally want as much information as possible regarding the project progress. It is the project manager's job to find the best compromise between those divergent requirements.

3.3.1.3. Execution

The workflow in this process group refers to the development of project components. Project team members are the main actors here. In typical large-scale software projects, this workflow goes as presented next.

A project team member starts a task that was assigned to that team member in the planning process group. There are several constraints that apply to the moment when a task can be started. One is that a task cannot be started until all the tasks on which that task depends are completed. Another constraint is that a worker cannot start a task before completing the tasks that have higher priorities than that task. Please note that activity prioritization is always present during project development. After all the conditions are met for starting a task, its assignee is able to begin the work on that task. This is the moment from when task's assignee is responsible for providing task work progress reports to the project management, as established in the planning process group. Progress reports are provided until the worker completes the task. The task completion is then tested by qualified testers (which exist distinctively in large-scale projects) or other reviewers that can also be subject of reports regarding the tested tasks. If the task is considered incomplete by the testers, that task is re-opened and waits to be re-assigned by the project manager.

This process group provides project management the required progress information needed in project status identification. Consequently, this is the main data source of project status.

3.3.1.4. Monitoring and Control

This is the process group in which all information for building project status is centralized by the project management.

The workflow here refers to this data centralization and to the corrective measures that project management tasks in order to get the project on track. The project manager conducts various meetings as part of this process group in order to decide, take, and assure the implementation of the corrective measures required by the project. The main document that is discussed during such meetings is the project status.

This process group is repeated as established in the planning process group. In [73] it is argued that project monitoring has to be a continuous endeavor in order to correct as early as possible the project deviations from plan.

3.3.1.5. Closing

This is the last process group in the project development workflow and it refers, among others, to the actions taken for improving organization's knowledge bases, such as lessons learned and historical information.

Although such information is hardly ever used when computing and discussing project status, we believe this is valuable information that might improve the efficiency of project management.

3.3.2. Project status accuracy levels

Depending on the information used by project management in the construction of project status, considering also the organizational environment established for the project, we identified four accuracy levels for project status

We further present each project status accuracy level that we identified, along with its purpose, the data acquisition methodology that it uses, and the tools that it employs.

3.3.2.1. Level 0 accuracy project status

A project status of Level 0 accuracy is a status that is computed from time to time, manually by the project manager, using only the most recent estimates for each project task. Fig.15 illustrates how the communication takes place within a project for which a Level 0 accuracy project status is elaborated, this being explained in detail next.

The main purpose of this approach is for the project manager to create a new project plan (rather than a project plan update) based on the latest estimates of the effort required for project tasks completion. The project manager is not interested in understanding the work progress to the point when the project status is computed. Gathering all the needed information for such an analysis is somewhat difficult since no automation is employed for this purpose.

As shown in fig.15, the reports regarding work progress from project team members are gathered by the project manager after a-priori report request from project manager's part to project team members. Actually, in practice, this data gathering is done through discussions between the project manager and each team member.

Because such an information gathering process requires so much effort from project manager's part, this occurs seldom, only when it is strictly required.

The reports contain as little information as possible and, generally, every project team member is asked just for an estimation of the remaining effort for their assigned tasks. A Level 0 accuracy project status can provide only little marginal information regarding what happened in the past with respect to the project or other projects developed within the organization. Moreover, the project manager

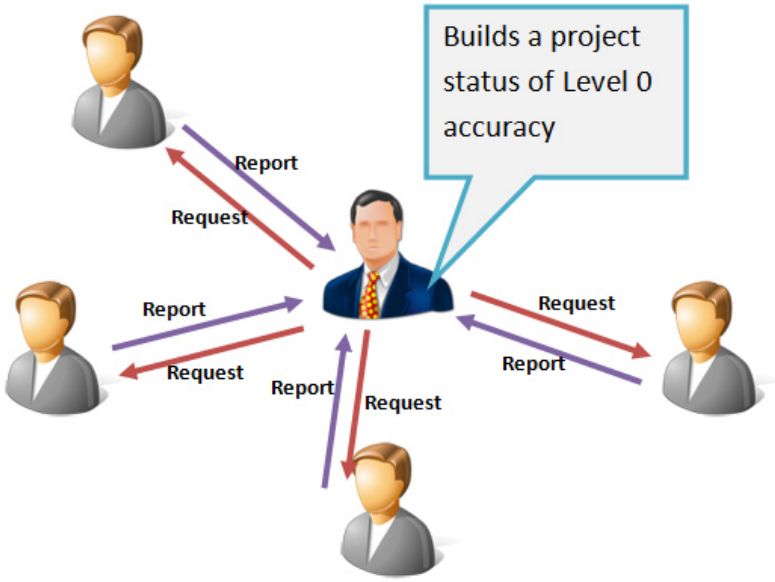

Figure 15. Level 0 project status accuracy

has no dynamic view over the managed project progress. Furthermore, there is a lag between data gathering and project status calculation that makes project status outdated before its computation being complete.

In order to elaborate a Level 0 accuracy project status, the project manager will use tools like Microsoft Project [62] and OpenProj [86]. These two are dedicated tools for project management, but besides such tools, any document editor can be employed for determining a Level 0 accuracy project status.

3.3.2.2. Level 1 accuracy project status

A project status of Level 1 accuracy is a status that is continuously updated based on the latest available estimates for each project task. Fig.16 shows the interactions that take place within a project for the elaboration of a project status of Level 1 accuracy.

There are at least two main differences between Level 0 and Level 1 accuracy project statuses. One is that the focus in Level 1 accuracy is on the employed software tools rather than on the personnel involved in the project. Of

course, humans are behind those tools, but the interactions among project team members and between each team member and the project manager are assured by those software tools. The second difference is that in Level 1 project status accuracy, the reports on work progress are not consequences of a-priori requests. The reports are asynchronous and they follow the actions performed by the project team members.

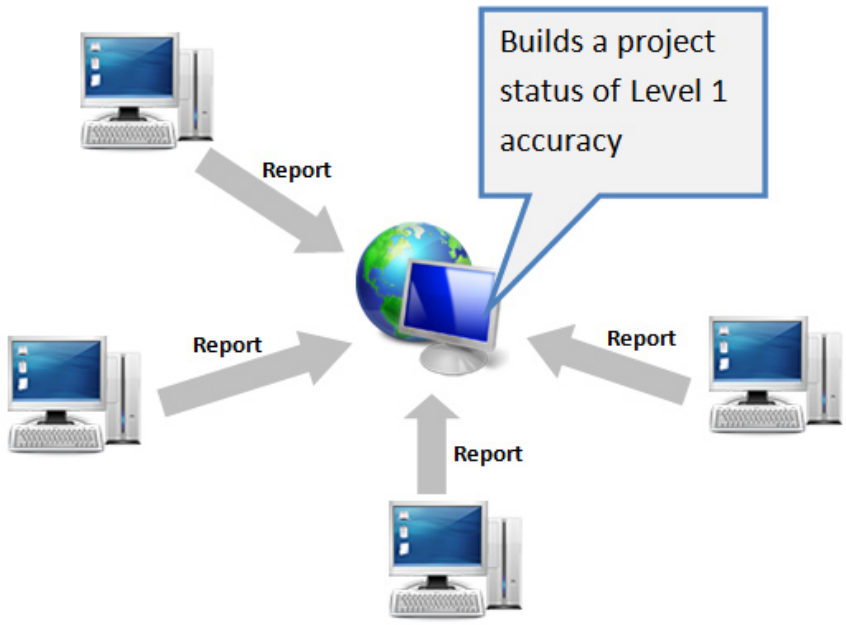

Figure 16. Level 1 project status accuracy

Consequently, a Level 1 accuracy project status offers the project manager a continuous image of how project progresses to completion. Moreover, the problematic lag between data gathering and project status computation present in Level 0 accuracy is inexistent in Level 1 accuracy project statuses, so that the project manager has, all the time, up-to-date information regarding project progress and status.

However, a Level 1 accuracy project status does not offer the project manager an understanding of the decisions that workers make every day regarding their own tasks (for example, a project manager has no clear idea of each project team member's task prioritization). This might be a problem for making the best decisions in critical situations.

For elaborating a Level 1 accuracy project status, the project manager needs to employs in the project an ALM (Application Lifecycle Management) tool like JIRA [7]. This ALM tool will be used by all project team members for all the communication that takes place within the project.

3.3.2.3. Level 2 accuracy project status

A project status of Level 2 accuracy is a status that is computed taking into account the latest decisions that each project team member considers for their own tasks. Fig.17 shows the interactions that take place within a project for the elaboration of a project status of Level 2 accuracy.

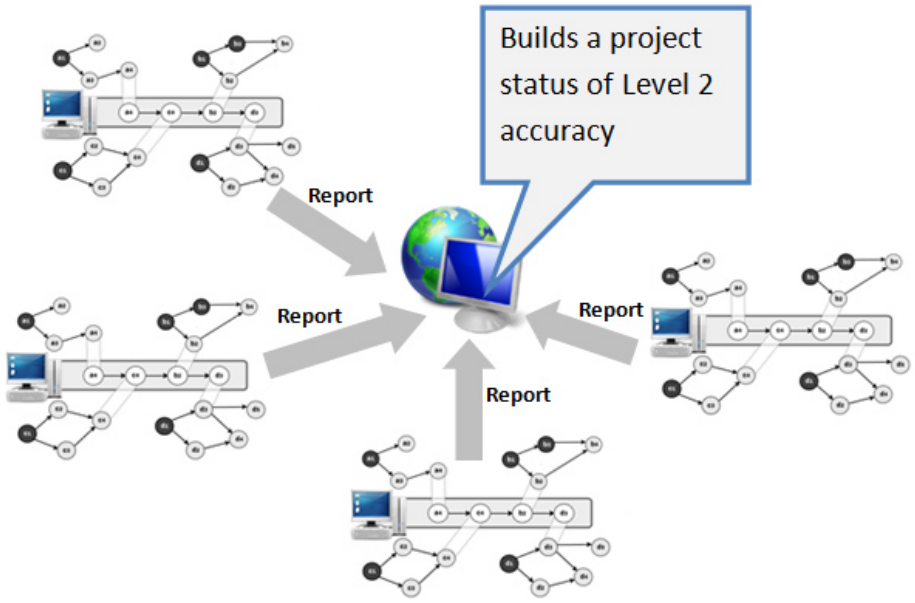

Figure 17. Level 2 project status accuracy

Software tools are also a focus here, just like for Level 1 accuracy project statuses. However, more important in the case of Level 2 accuracy project statuses are the models and algorithms implemented in the employed software tools.

Many times during a project's development, project team members are assigned several tasks with overlapping implementation time intervals. Generally, the project manager doesn't establish a particular prioritization for such tasks, so that each project team member decides upon their assigned tasks' order of implementation.

Choosing a particular prioritization for the development of different sets of tasks, most of the time has an impact on the progress of the project: some tasks are dependent to others, their assignees having to wait for those depending tasks to be completed; some workers have a higher efficiency if they implement simple tasks first, but they cannot do this because these tasks can only be started after certain other tasks are completed. Consequently, for a more accurate project status, the project manager should be aware of such things that have so important impact to project progress to completion.

The project team members prioritization of own tasks can be modeled and this model can be implemented in an ALM tool to be used during project development by all personnel involved. Such a model is just an example. For obtaining project statuses of Level 2 accuracy, any type of models and algorithms that enable project management to understand the work status at a worker level can be used.

A Level 2 accuracy project status gives the project manager an insight on what are the latest tendencies and concerns of project team members in terms of project work. Such information helps the project manager to maximize management efficiency by adjusting such tendencies. However, Level 2 accuracy project statuses offer project manager only a snapshot of the situation available at project status computation, without any references to similar situations in the past.

For elaborating a Level 2 accuracy project status, the project manager should employ in the project an ALM tool like IBM Rational Team Concert [42]. This tool will be used by all project team members for all the communication activities within the project, including work progress reporting and work prioritization reporting.

3.3.2.4. Level 3 accuracy project status

A project status of Level 3 accuracy is a status that is computed taking into account the historical changes in the decisions that each project team member considers for their own tasks. Fig.18 shows the interactions that take place within a project for the elaboration of a project status of Level 3 accuracy.

Level 3 accuracy project statuses require the employment during project development of software tools that implement complex models that use project

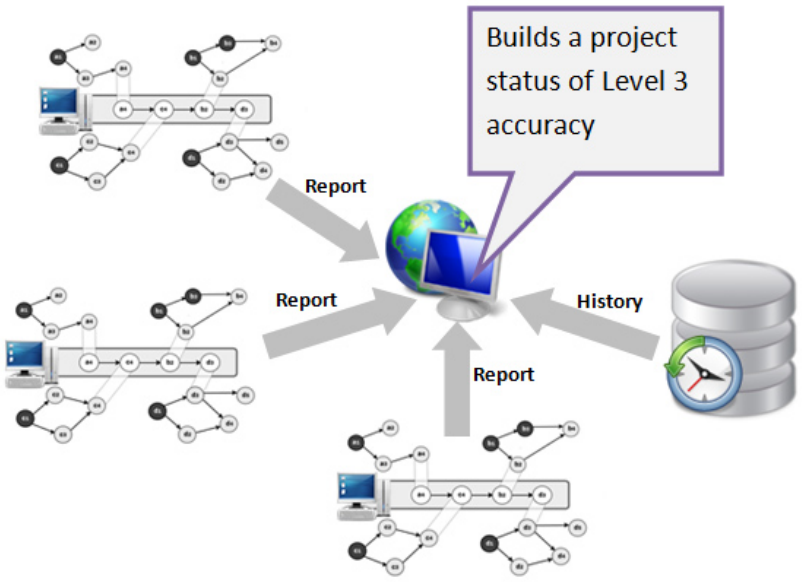

Figure 18. Level 3 project status accuracy

status historical information. For fully benefiting from the utilization of the historical information, these models include suitable forecasting methodologies that are able to interpret the past of the project, or of other projects developed within the organization, in the context of the managed project's present.

The historical information employed in the construction of a Level 3 accuracy project status and the models that reflect the work status at a project team member level, enables the project manager to clearly understand the current context of the project having also the support of past information regarding progress evolution (e.g., lessons learned or similar information). An established forecasting methodology implemented in the ALM tool employed in the project helps the project manager to foresee the future project evolution.

A Level 3 project status accuracy offers the project manager a clear view of the present situation of project development, showing also how most likely the project will progress in a defined future.

The development of the theoretical background required by the software tools used in computing a Level 3 accuracy project status is a difficult task for several reasons. At first, the historical information that is a central factor in computing a Level 3 accuracy project status is generally incomplete and contains un-normalized information regarding very different situations and tasks, according to [51]. A second important reason is that project development information that can be used as historical information in the research activities on this subject is very hard to get. Generally, such information is confidential in the case of commercial software projects. On the other hand, the open source projects usually are much more flexible than commercial projects when it comes to terms of execution, so that historical information is hardly ever found in such projects.

3.3.3. Discussion

Project status is required for two main reasons. One is to communicate to stakeholders the project progress and the second is to help the project manager understand what to do and what decisions to make in order to keep the project on track.

Depending on the informational needs of the main project stakeholders, the project manager should employ the corresponding tools in order to compute the project status with the required accuracy level.

For example, let's consider a project of which sponsors make available a virtually unlimited budget, having no pressure, and feeling no risks. In such a situation, the project manager should definitely choose Level 0 accuracy project statuses. The investment in an ALM tool and in the infrastructure that it requires makes no sense in such a context. On the other hand, if the organization depends on the development and commercialization of a particular project, than the informational requirements regarding project status are high in order to minimize the involved risks. In such a situation, project statuses of higher accuracy levels are required.

The complexity, size, and main technologies used in the project, the available human resources, and the established budget and execution time constraints, weight in the project manager's decision regarding the accuracy level used in computing project statuses. For example, a project with a tight deadline requires an efficient project management, which needs the computation of project statuses with higher accuracy levels.

The first 3 accuracy levels of project status presented in the previous section are used today in project development, being available by using from simple to more complex software tools. The last accuracy level, Level 3, can be reached by employing our monitoring framework, the Behavioral Monitoring Framework, in the project monitoring and control process.

We believe that, as software projects become larger and more complex, employing new technologies and using costly human resources, the capability of computing Level 3 accuracy project statuses is a must for the ALM tools of the future.

Choosing an accuracy level for computing project statuses is, as shown, a matter of expectations and informational requirements that concern the project. There are no generally good or bad accuracy levels, but suitable or inappropriate accuracy levels with respect to a given project. Consequently, considering its particular context, each project has its own best choice concerning project status accuracy level.

The behavioral approach, with its Behavioral Monitoring Framework, was developed for the monitoring software projects with certain benefits for the most problematic large-scale ones, as described previously, and it is the only approach that is able to provide a project status of Level 3 accuracy, as defined within this paragraph.

Having defined the Work Behavior, as a central concept of our approach, and having presented the project status accuracy levels classification, revealing the requirements for the elaboration of a project status that can provide major support to project monitoring, we further define and describe, one by one, the component models of the proposed framework.

3.4. The Project Status Model

The proposed Project Status Model is able to provide the status of the monitored project. The status of the project can be described at any time by using this model. Consequently, the provided status can be the current status or a probable status at a given time in the future. Next, we define the concepts with which this model operates along with its equations. Furthermore, we will discuss how it identifies the status project, providing also a case study for this purpose.

3.4.1. Definitions

The following definitions introduce the underlying concepts of the Project Status Model which are: the project macro-universe, the worker micro-universe, the task evolution, the snapshot, and the project status. The following definitions assume the existence of a set of projects, P, a set of tasks, Θ, and a set of workers, W.

Definition 7 (Project Macro-Universe). The Macro-Universe M_i for a project $p_i \in P$ is a quadruplet $(\Theta_i, W_i, dep_i, \Phi)$, where Θ_i is a subset of Θ, W_i is a subset of W, dep_i is a binary relation defined on Θ_i, and Φ represents time, so that, if we assume an arbitrary macro-universe M_k of a project $p_k \in P$, with $k \neq i$, and $M_k = (\Theta_k, W_k, dep_k, \Phi)$, then $\Theta_k \square \Theta_i = \emptyset$ ($W_i \square W_k$ might not be an empty set), no matter the time $\varphi \in \Phi$.

Definition 8 (Worker Micro-Universe). The Micro-Universe μ_i for a worker $w_i \in W$ is a triplet (Θ_i, ord_i, Φ), where Θ_i is a subset of Θ, ord_i is a binary relation defined on Θ_i, and Φ represents time, so that, if we assume an arbitrary micro-universe μ_k of a worker $w_k \in W$, with $k \neq i$, and $\mu_k = (\Theta_k, ord_k, \Phi)$, then $\Theta_k \square \Theta_i = \emptyset$, no matter the time $\varphi \in \Phi$.

Please note that the fact that Definition 8 implies that a task can be assigned to only one worker at a time is not a restriction: for example, a task assigned to two workers can be regarded as two tasks with the same position in the project macro-universe as the original task, each resulted task being assigned to only one worker.

Definition 9 (Task Evolution). The Task Evolution ε_i of a given task $t_i \in \Theta$ is a quintet $(M_i, \mu_i, D_i, \zeta_i, \Phi)$, where M_i represents the macro-universe of the project to which t_i belongs, μ_i represents the micro-universe of the worker to which t_i is assigned, D_i is the due date for task t_i established at task creation, ζ_i is a function of time, named Status Function, that outputs a quadruplet $(ES_i, EL_i, PES_i, WES_i)$, and Φ represents time, where ES_i is the estimated effort for task t_i; EL_i is the elapsed effort for task t_i, meaning the total time spent actually working on task t_i; PES_i is the earliest date when task t_i can be started considering only the macro-universe M_i (PES is the acronym for Project Early Start and it is associated with a task); WES_i is the earliest date when task t_i can be started considering the macro-universe M_i and the micro-universe μ_i (WES is the acronym for Worker Early Start and it is associated with a task).

It is important to be aware of the difference between the parameter D introduced in Definition 9, which refer to the due date of a task established at task creation, and the actual due date of the task, which has a time-dependent value.

Definition 10 (Snapshot). The Snapshot at a given moment in time $\varphi \in \Phi$, Φ representing time, is a set $\{\varepsilon_i(\varphi)$, where $\varepsilon_i(\varphi)$ is the task evolution for task $t_i \in \Theta$ at the given time $\varphi \in \Phi$, for any $t_i \in \Theta\}$.

The introduction of snapshots is important because, as shown in the previous definitions, the project macro-universe, the worker micro-universe and the task evolution are variable in time, while for finding the status of a project of interest is the situation at a particular moment in time.

Definition 11 (Project Status). The Project Status for a project $p_i \in P$, at a given moment in time $\varphi \in \Phi$, Φ representing time, is a set $\{\zeta_i(\varphi)=(ES_i, EL_i, PES_i, WES_i)$, where $\zeta_i(\varphi)$ is the status function of the task evolution ε_i of task $t_i \in \Theta_i$ at the given time $\varphi \in \Phi$, for any $t_i \in \Theta_i$, Θ_i being the set of tasks of the project $p_i \in P\}$.

Regarding Definition 11, please note that ζ_i as the status function of a task evolution ε_i was introduced in Definition 9. Definition 11 suggests that, for finding the status of a project, EL, ES, PES and WES (introduced in Definition 9) must be determined for all the project tasks that exist at the moment when this status is computed.

Next, we present and discuss the underlying equations of the Project Status Model.

3.4.2. Project Status Model equations

This section presents the equations of the Project Status Model. These equations regard the computation of PES and WES, for every task of the project for which the status is required. Because project status is defined as a set of quadruplets (ES, EL, PES, WES), one such quadruplet for each project task

(according to Definition 11) and because ES and EL for each project task are known directly from progress reports (it is nowadays common for the workers involved in a project to report the estimated effort, ES, and the elapsed effort, EL, for all their assigned tasks regularly), computing the status of a project as defined in Definition 11 requires only the identification of PES and WES for all project tasks. This is why the equations of the Project Status Model regard only the computation of PES and WES that were introduced in Definition 7.

We present next, the equations that are used for the finding of PES. Considering Definition 7, the dep binary relation is asymmetric and not transitive, and it is defined on the set of tasks of a project, so that given two tasks t_a and t_b, (t_a, t_b) Є dep means that t_b is a task on which t_a depends directly (t_a cannot start before the completion of t_b). The ord relation introduced in worker's definition is asymmetric and not transitive, and it is defined on the set of tasks assigned to a worker, so that given two tasks t_c and t_d, (t_c, t_d) Є ord means that t_d is the successor of t_c in the local order. Although the worker to which t_c and t_d are assigned may change the order of their tasks at any time, at the given moment in time when (t_c, t_d) Є ord, we consider that t_d cannot be started or continued before the completion of t_c. The value of PES for a task t_k and a moment in time φ Є Φ, is computed using (4) if there is at least one task on which t_k depends (a task t_x exists so that (t_k, t_x) Є dep) and that task is not completed ($ES_{tx} \neq EL_{tx}$) at time T. Basically, PES is the date when the depending task can be started to which is added a number of time units representing the remaining working time regarding the respective depending task.

$$PES_{t_k} = \max_{\substack{(t_k,t_x)\in dep \\ ES_{t_x}\neq EL_{t_x}}} \{WES_{t_x}^{w_x} + ES_{t_x} - EL_{t_x}\} \tag{4}$$

Because there are cases when a task depends on more than one task, equation (4) uses a max operator which returns the maximum value for PES from the values computed using the depending tasks individually.

In the case when no task t_x exists so that (t_k, t_x) Є dep and $ES_{tx} \neq EL_{tx}$, PES is given by (5) and its value is T (the current time).

$$PES_{t_k} = T \tag{5}$$

Having presented the equations for finding PES, we introduce next the equations required for the computation of WES.

The value of WES for a task t_k (task t_k is assigned to the worker w_k) and a time T is computed using (6) if there is a task to which t_k is the direct successor in the local order of worker w_k (a task t_y exists so that (t_y, t_k) Є ord) and that task is not completed ($ES_{ty} \neq EL_{ty}$) at time T. The meaning of this equation is that a task t_k can be started or continued only when the following two conditions are simultaneously met (max operator):

1) the preceding tasks in the project macro-universe to which t_k belongs are completed (PES_{tk});

2) the task to which t_k is the direct successor in the local order of worker w_k is completed (WES of the predecessor task, t_y, in the local order, a task that might be assigned to other worker, w_y; to this WES value is further added a number of time units representing the remaining working time regarding the respective predecessor task).

$$WES_{t_k}^{w_k} = \max \{ PES_{t_k} ; WES_{t_y}^{w_y} + ES_{t_y} - EL_{t_y} \}, \quad (6)$$

$(t_y, t_k) \in ord$ and $ES_{ty} \neq EL_{ty}$

In the case where no task t_y exists so that $(t_y, t_k) \in ord$ and $ES_{ty} \neq EL_{ty}$, WES of task t_k is given by (7) and its value is the same as PES for task t_k.

$$WES_{t_k}^{w_k} = PES_{t_k} \quad (7)$$

Using the equations (4), (5), (6), and (7), WES and PES can be computed for every project task. Knowing EL and ES directly from the common reports on progress provided regularly by the human resource for each assigned task (wide spread best practice according to [72]), and knowing PES and WES for each task (by using the Project Status Model equations provided in this section), the status of the project, as defined in Definition 11, is known.

3.4.3. Status identification methodology

After introducing Project Status Model's definitions and equations, we present in this section how the defined concepts and equations are used in the identification of the status of a project with the Project Status Model. For this, we start from fig.19, which shows a detail upon the Project Status Model as presented in fig.10, illustrating how the Project Status Model integrates in the Behavioral Monitoring Framework as well as the inputs and outputs of this model.

As shown in fig.19, the Project Status Model has three inputs from outside the framework: the project structure and remaining efforts for project tasks, the work assignation and the current time. The Project Status Model requires to know about the project structure and remaining efforts for project tasks (e.g., the project tasks, the involved workers, the dependencies among project tasks, and also the EL and ES for each project task) in order to find the project macro-universe as defined in Definition 7 and the snapshot of the current time, as defined in Definition 10. The second input of the model, work assignation (e.g., which tasks are assigned to whom, task prioritization by worker), is required in order for the Project Status Model to identify the micro-universe of each worker, as defined in Definition 8, involved in the project. Finally, the current time is required by the Project Status Model in order for it to know about the present time. Having these inputs, each with its meanings described above, the Project Status Model is able to find, using equations (4), (5), (6), and (7) the PES and WES for all project tasks for the current time. Knowing also EL and ES (besides PES and WES computed using the model's equations) for each project task from the first input (project structure and

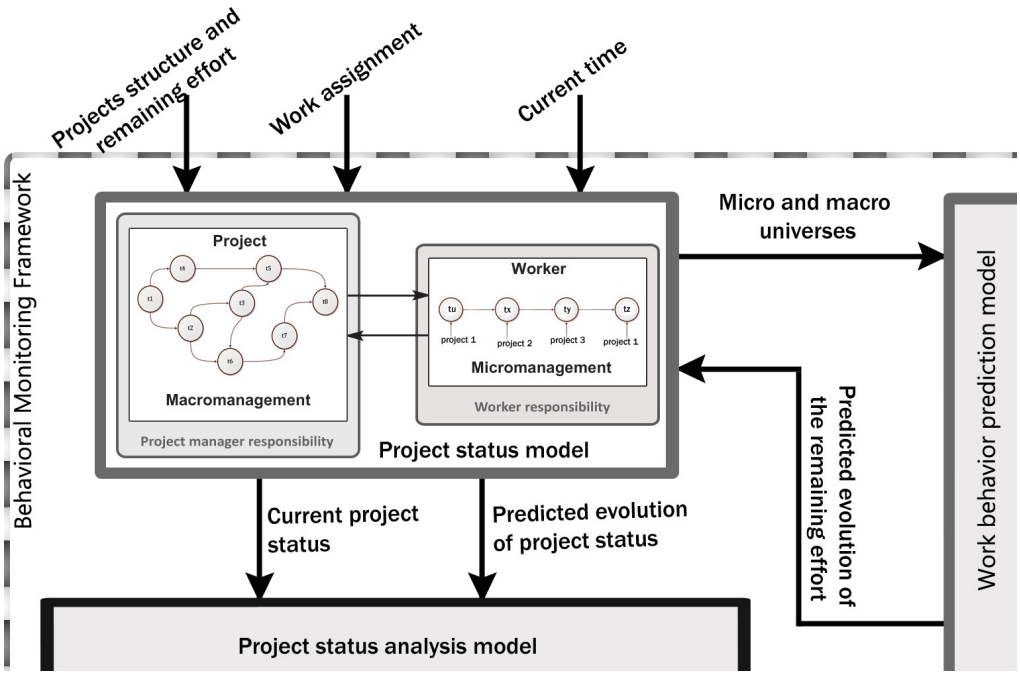

Figure 19. The Project Status Model

remaining efforts for project tasks) the Project Status Model outputs the project status as defined in Definition 11 for the current time given as input.

Besides these three inputs from outside the Behavioral Monitoring framework, fig.19 illustrates the existence of another input, the predicted evolution of the remaining effort that comes from the Work Behavior Prediction Model and refers to predicted EL and ES for the tasks of the project. Please note that this input is not available to the Project Status Model until the Work Behavior Prediction Model provides it to this model. In order to do so, as shown in fig.19, the Work Behavior Prediction Model must receive from the Project Status Model the project macro-universe and the workers micro-universes (which are found based on the Project Status Model inputs from outside the framework as shown above). Using the predicted evolution of the remaining effort (e.g., predicted EL and ES) for each of the project tasks and the project structure and work assignation (the outer inputs presented earlier), the Project Status Model is able to compute, using the same equations (4), (5), (6), and (7) but applied on predicted ES and EL for each task, the predicted PES and WES. Knowing the predicted ES, EL, PES, and WES for each project task, the Project Status Model outputs the predicted status of the project (shown in fig.19).

Having presented the project status identification methodology employed by the Project Status Model starting from fig.19 with references to the model's concepts (definitions) and equations, we will next describe from a more practical perspective the most important concepts used by this model: project macro-universe and worker micro-universe, which have a very important role in the computation of PES and WES, and finally, in the elaboration of the project status as introduced in Definition 11.

In this context, we use a modified PERT for representing a project macro-universe: a directed acyclic graph, as in [91]. This graph's vertices are the tasks of the project and the arcs suggest that the pointed task is dependent to the source task. If a task is dependent to another task, the dependent task cannot start before the completion of the task on which it depends. As part of the project macro-universe according to Definition 7, the workers involved in the project are not represented in fig.20 for the sake of simplicity. In fig.20, a project macro-universe is represented at different moments in time, suggesting the possible changes in the project structure that can take place during project development. However, for establishing the status of the project at a moment in time, only the snapshot (as defined in Definition 10) describing the project at that moment in time is needed. In fig.20, a snapshot of the macro-universe is marked (at $time_x$).

In an organization, there are as many project macro-universes as projects being developed and currently in work. In this context, the available workers may be assigned with many tasks, from different projects being currently in work in the organization. Generally, in such a context, the workers might decide the rejection of several tasks, the order in which they execute their assigned and accepted tasks, the re-estimation of the effort required for the completion of their tasks and so on. This way, the workers may be seen as the managers of their own tasks. Consequently, another perspective of the project development must be taken into consideration in monitoring. We refer to this perspective as the micro-universe of the worker, which is defined in Definition 8. Fig.21 illustrates such a perspective: a worker micro-universe at a given time (the black nodes/tasks in fig.21 are completed tasks).

Fig.21 shows the tasks assigned to and accepted by a worker, ordered as desired by the worker at a particular time. The order of these tasks is established and can be changed at any time by the assignee, who can also re-estimate the required effort for the completion of their tasks (ES values for the respective tasks). Moreover, the worker reports the elapsed effort for their tasks (EL values, meaning the time spent actually working on the respective tasks) which is nowadays a common activity for the human resource involved in a project. These ordered tasks are further referred to as local sequence of tasks or local order. A local order is associated with a micro-universe of a worker at a given time.

As suggested in fig.21 and by Definition 8, a worker micro-universe is not related to a project, so that the tasks in a micro-universe do not necessary belong to the same project. Every task in the local sequence of tasks has an associated PES

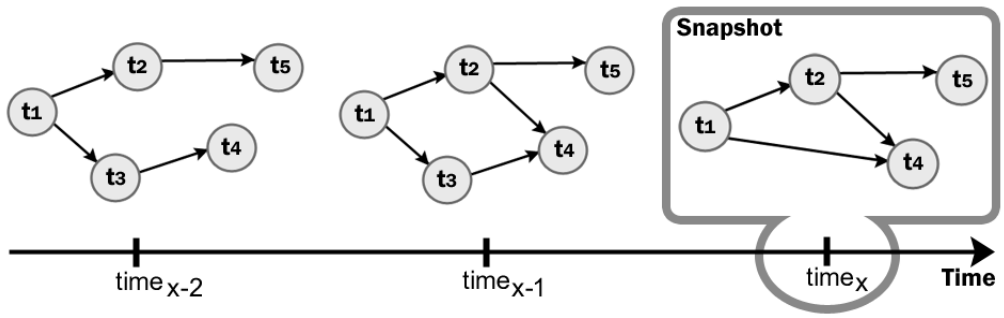

Figure 20. A project macro-universe: evolution and a snapshot used in determining the status of the project at a moment in time (time x)

and an associated WES. To explain the Project Status Model's equations, the value of PES associated to a task t_i represents the date on which every task, that belong to the same project as t_i and on which t_i depends, is completed; the value of WES associated to a task t_i represents the latest date between PES associated with t_i and the date when all previous tasks in the local sequence where t_i belongs (at the time when WES is computed) are also completed. For example, considering fig.21, PES for c_4 determined at $time_x$ is the latest date between the completion date of c_2 and the completion date of c_3. PES for b_2 at $time_x$ is $time_x$ since b_1 is already completed at $time_x$ (b1 is black). Meanwhile, WES for c_4 determined at $time_x$ is the latest date

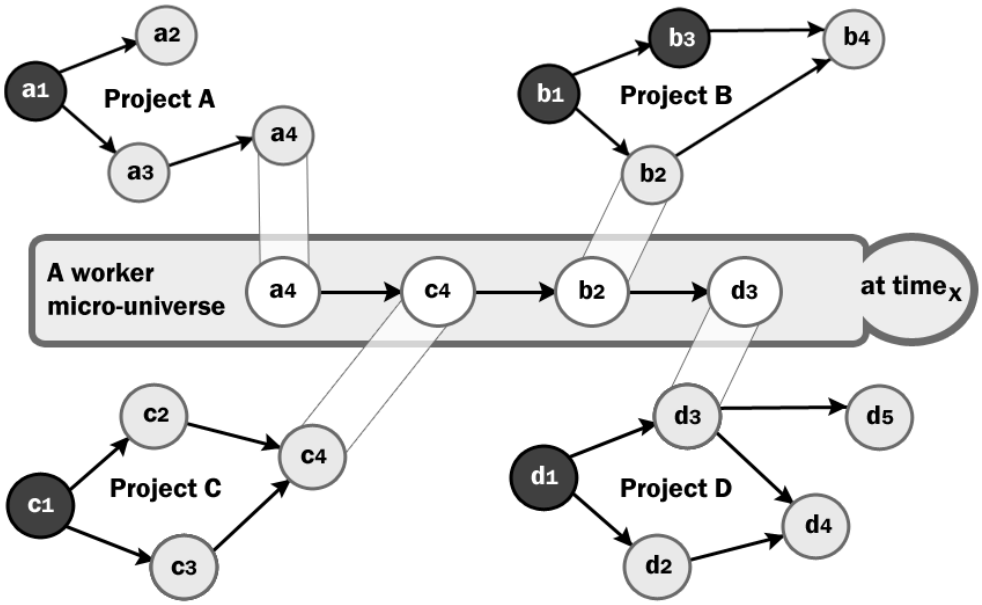

Figure 21. A worker micro-universe: a snapshot used in determining the status of the project at a moment in time (time x)

between PES for c_4 determined at $time_x$ and the completion date of a_4. WES for b_2 at $time_x$ is the latest date between $time_x$ (which is the PES for b_2 computed at $time_x$) and the completion date of c_4. Since c_4 is not completed at $time_x$, WES for b_2 at $time_x$ is the completion date of c_4.

The underlying concepts of the Project Status Model of project macro-universe and worker micro-universe, as well as the PES and WES and how the project status is defined (Definition 11) imply that the Project Status Model computes the status of a project not based only on the project-level decisions taken by the project manager (e.g., the decisions regarding the project structure of tasks and the relation among project tasks), but also on worker-level decisions (e.g., decisions regarding the prioritization of the assigned tasks) which introduces a second level of decision-making unutilized until now. Including in the elaboration of the project status a second level of decision-making besides the project-level, that is the worker decision-making level, enables the so computed project status to be more detailed in terms of the information that it is able to provide. Such detailed project status can offer support to the project managers in the managing of the

problematic large-scale software projects which require a tight control over everything that can take them out of their planned track.

Having clarified both the project status identification methodology employed by the Project Status Model and the meaning of the very important concepts of project macro-universe and worker micro-universe, we proceed with the presentation of a case study that shows how the Project Status Model concepts (definitions) and equations apply to a real-world inspired situation.

3.4.4. Case study

The aim of this case study is to exemplify the computation of the project status using the proposed Project Status Model. In this example, the current project status is computed. However, in the same way, the Project Status Model is able to retrieve a future probable project status by using predicted ES and EL values as described in the previous section.

Consider an organization that is currently developing two software projects, P_1 and P_2. These two projects contain a number of tasks: Θ_0, Θ_1, Θ_2, Θ_3, Θ_4, Θ_5, Θ_6, Θ_7, Θ_8, Θ_9, Θ_{11}, and Θ_{12}. In the implementation of these tasks, a number of workers are involved: W_1, W_2, W_3, W_6, W_{10}, W_{20}, W_{21}, and W_{22}.

The links among tasks, projects and workers are provided in Table 1. Moreover, for every task, its status and its ES and EL current values are also provided in Table 1. Fig.22 shows the micro and macro-universes in the organization as defined earlier.

Consider that the project manager of P_2 needs the managed project status in order to take the best decisions regarding project execution. The methodology of determining P_2 project status by using the proposed Project Status Model is described next.

The project status of a project at a given time, T, is considered determined when all existing and not completed tasks of that project have an associated WES and PES, along with their ES and EL values for the given T (as shown in Definition 11). Consequently, the WES and PES values must be computed for Θ_4, Θ_6, Θ_8, and Θ_9 (considering Table 1 and fig.22).

Regarding Θ_4, since Θ_5 is a completed task (its ES and EL values are equal), PES_4 is computed using (5), so that $PES_4 = T$. Meanwhile, WES_4 is computed using (6). Because Θ_4 follows Θ_2 in the micro-universe of W_{10}, as shown in fig.22.c, WES_2 must be determined in order to find WES_4. In the project macro-universe of P_1, Θ_2 is preceded only by a completed task, Θ_1, so that $PES_2 = T$, using (5). Moreover, in the micro-universe of W_{10}, W_{10} being the owner of Θ_2, Θ2 is not preceded by other not completed tasks so that, using (7), $WES_2 = PES_2 = T$. Consequently, using (6), WES_4 is $WES_2 + ES_2 - EL_2 = T + 5 - 3 = T + 2$. Considering that EL and ES values refer to days, Θ_4 can be started not earlier than two days after the given moment T.

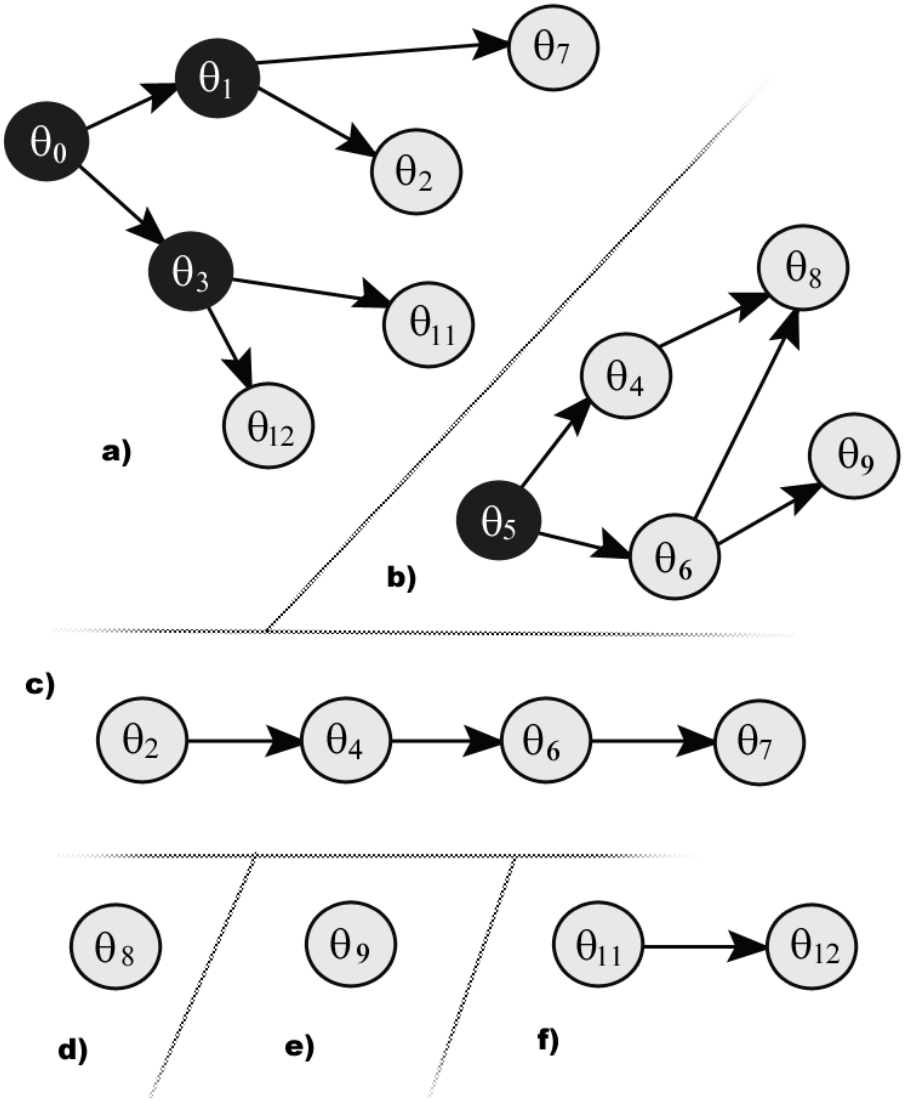

Figure 22. Organization macro and micro-universes: a) P_1 macro-universe; b) P_2 macro-universe; c) W_{10} micro-universe; d) W_{20} micro-universe; e) W_{21} micro-universe; f) W_{22} micro-universe

Regarding Θ_6, since Θ_5 is a completed task (its ES and EL values are equal), PES_6 is computed using (5), so that $PES_6 = T$. WES_6 is computed using (6). Because Θ_6 follows Θ_4 in the micro-universe of W_{10}, as shown in fig.22c, WES_4 is used for finding WES_6. Consequently, using (6), $WES_6 = WES_4 + ES_4 - EL_4 = T + 2 + 3 - 1 = T + 4$.

Table 1. Task details for Project Status Model case study

Task	Owner	Parent Project	Current ES	Current EL	Status
Θ_0	W_1	P_1	5	5	completed
Θ_1	W_2	P_1	3	3	completed
Θ_2	W_{10}	P_1	5	3	in-work
Θ_3	W_3	P_1	2	2	completed
Θ_4	W_{10}	P_2	3	1	in-work
Θ_5	W_6	P_2	7	7	completed
Θ_6	W_{10}	P_2	3	2	in-work
Θ_7	W_{10}	P_1	8	2	in-work
Θ_8	W_{20}	P_2	3	0	not started
Θ_9	W_{21}	P_2	6	0	not started
Θ_{11}	W_{22}	P_1	6	1	in-work
Θ_{12}	W_{22}	P_1	3	1	in-work

Regarding Θ_8, it depends on Θ_4 and Θ_6. Using (4), $PES_8 = \max \{WES_4 + ES_4 - ES_4; WES_6 + ES_6 - EL_6\} = \max \{T + 2 + 3 - 1; T + 4 + 3 - 2\} = \max \{T + 4; T + 5\} = T + 5$. Because the micro-universe of W_{20} contains only one task, Θ_8, as shown in fig.22.d, using (7), $WES_8 = PES_8 = T + 5$.

Regarding Θ_9, it depends on Θ_6. Using (4), $PES_9 = \max \{WES_6 + ES_6 - EL_6\} = \max \{T + 4 + 3 - 2\} = T + 5$. Because the micro-universe of W_{21} contains only one task, Θ_9, as shown in fig.22.e, using (7), $WES_9 = PES_9 = T + 5$.

To summarize, considering Definition 11, the project status of P_2 at time T is:

1. Task Θ_4: $ES_4 = 3$, $EL_4 = 1$, $PES_4 = T$, $WES_4 = T + 2$
2. Task Θ_6: $ES_6 = 3$, $EL_6 = 2$, $PES_6 = T$, $WES_6 = T + 4$
3. Task Θ_8: $ES_8 = 3$, $EL_8 = 0$, $PES_8 = T + 5$, $WES_8 = T + 5$
4. Task Θ_9: $ES_9 = 6$, $EL_9 = 0$, $PES_9 = T + 5$, $WES_9 = T + 5$

3.4.5. Conclusions

In this section, we define the first component model of the proposed Behavioral Monitoring Framework, which is the Project Status Model. This model is

able to compute the current status of a project or its status at a given moment in the future, using for this information provided by the Work Behavior Prediction Model.

A distinct characteristic of the proposed Project Status Model and an innovation factor is that this model takes into consideration two perspectives over the monitored project: the macro-universe of the project and the micro-universe of the worker. As presented earlier, these perspectives refer to two decision-making levels: the project level, which regards the decisions of the project manager (e.g., decisions regarding project structure of tasks, dependencies among tasks etc.), commonly used when elaborating a project status, as well as the worker level, which regards the decisions taken by the workers involved in the project (e.g., own tasks prioritization), which was not considered until now in the elaboration of the project status. This means that the status of the monitored project is built not just upon the big picture of the project, but also upon the individual working decisions of the workers involved in the project. Moreover, the Project Status Model considers the common situation when a worker is assigned several tasks from different projects at a time by defining the worker micro-universe to a set of tasks (the tasks assigned to a worker at a time, no matter the project) and not to a project.

As shown in this section, the Project Status Model works with data provided by the Work Behavior Prediction Model to compute a predicted project status. In this context, we define next the Work Behavior Prediction Model.

3.5. The Work Behavior Prediction Model

We propose a model for Work Behavior Prediction, which can be seen as the core of the proposed project monitoring framework [95]. The Work Behavior Prediction Model provides the dynamic trait to the proposed approach that enables the proposed Behavioral Monitoring Framework to elaborate a project status of Level 3 accuracy as defined earlier in this chapter.

3.5.1. Definitions and equations

This section presents a generalization of the Work Behavior representation, in that the remaining effort used for computing the component metrics is replaced by a more general notion, which is work measurement.

Definition 12 (Work Measurement). Work Measurement (M) for a task t is an estimated effort ES or an elapsed effort EL reported for the task t.

As suggested by Definition 12, work measurement refers either to effort estimation (ES) or to reported elapsed effort (EL) concerning a task. ES and EL where introduced in Definition 9, for the Project Status Model. Considering the workflow within a project, the first ES value for a target task is given by a worker which not necessary is the owner of the task. The following ES values for the target task are given by the owner of that task. The first EL value for a task is 0, meaning that no work was spent on that task at the beginning.

Definition 13 (Sampling Time). Sampling time is a moment in time $\varphi \in \Phi$, Φ representing time, when a work measurement M exists.

If we consider the time span between sampling times, as defined in Definition 13, constant and of one day, at the end of every working day, for every

owned in-work task, the worker either reports a new ES, or confirms the ES available at the end of the previous working day. In the same time, the worker either reports a new EL, or confirms the EL available in the previous working day, for every owned task. Consequently, EL values are at most 1 (one day of work if the time span between sampling times is one day), because in a working day, a worker cannot spend more than one day at working on a task.

Definition 14 (Stability). Stability (ST) is the propriety of a work measurement (M) to maintain its value from a sampling time to another, considering its history (H).

Please note that Stability was introduced in [37]. However, we use it in a different context, applying it to different measurements with different connotations. Equation (8) illustrates the Stability ST, computed at a sampling time i, for a measurement M and its history H.

$$ST_i(M,H) = \begin{cases} 1, & M_{i-1} = M_i \\ 0, & M_{i-1} \neq M_i \end{cases} \quad (8)$$

Equation (9) is used for computing ST for a time span, for a work measurement M and its history H. The stability of M for a time span is the average of the ST values computed at every sampling time in the given time span.

$$ST(M,H) = \frac{\sum_{i=1}^{n-1} ST_i(M,H)}{n-1} \quad (9)$$

Definition 15 (Evolution). Evolution (EV) is the property of a work measurement (M) to change its dynamic state among three consecutive sampling times, considering its history (H).

A dynamic state is established between two consecutive sampling times and can be either a static state (M has the same values at both sampling times) or a changing state (M has different values at the two sampling times).

Equation (10) illustrates the Evolution EV computed at a sampling time i, for a measurement M and its history H.

$$EV_i(M,H) = \begin{cases} 1, & M_{i-2} \neq M_{i-1} \text{ and } M_{i-1} = M_i \\ 1, & M_{i-2} = M_{i-1} \text{ and } M_{i-1} \neq M_i \\ 0, & M_{i-2} = M_{i-1} \text{ and } M_{i-1} = M_i \\ 0, & M_{i-2} \neq M_{i-1} \text{ and } M_{i-1} \neq M_i \end{cases} \quad (10)$$

Equation (11) is used for computing EV for a time span, for a work measurement M and its history H. H contains n values for M. The Evolution of M for a time span is the average of the EV values computed at every sampling time in the given time span.

$$EV(M,H) = \frac{\sum_{i=2}^{n-1} EV_i(M,H)}{n-2} \quad (11)$$

Definition 16 (Trend). Trend (TR) is the difference between the values of a work measurement (M) at consecutive sampling times, considering its history (H).

Equation (12) illustrates the Trend TR, computed at a sampling time i, for a measurement M and its history H. If TR at sampling time i is greater than 1 and the first value of M (M_0) is not 0, a normalized TR is computed as in (13). As a general rule, (12) is used when M refers to EL and (13) is used when M denotes ES.

$$TR_i(M,H) = M_i - M_{i-1} \tag{12}$$

$$TR_i(M,H) = \frac{M_i - M_{i-1}}{M_0} \tag{13}$$

Equation (14) is used for computing TR for a time span, for a work measurement M and its history H. H contains n values for M. The trend of M for a time span is the average of the TR values computed at every sampling time in the given time span.

$$TR(M,H) = \frac{\sum_{i=1}^{n-1} TR_i(M,H)}{n-1} \tag{14}$$

Definition 17 (Generalized Work Behavior). Work Behavior (WB) is a triplet (ST, EV, TR), where ST, EV and TR represent Stability, Evolution and Trend respectively, computed for a given work measurement (M) considering the history of this measurement (H).

Equation (15) illustrates the meaning of the generalized Work Behavior concept.

$$WB(M,H) = (ST(M,H), EV(M,H), TR(M,H)) \tag{15}$$

Definition 18 (Estimation Correction). Estimation Correction (EC) is the relative error of the initial ES, which is ES_0 and which is given by an estimator, considering the first estimation of the task owner, which is ES_1.

Equation (16) is used for computing EC for a task j. In (16), H is the history of the ES measurement for task j containing ES_0 and ES_1.

$$EC^j(H) = \frac{ES_1 - ES_0}{ES_0} \tag{16}$$

Definition 19 (Estimation Behavior). Estimation Behavior (EB) is the mean EC computed for the tasks initially estimated by an estimator.

Equation (17) is used for computing EB for m tasks initially estimated by the same estimator. In (17), H is the history of the ES measurement for those m tasks.

$$EB(H) = \frac{\sum_{j=1}^{m-1} EC^j(H)}{m-1} \quad (17)$$

Definition 20 (Task Dimensions). A triplet (T, C, S), where T stands for main technology measured on nominal scale, C refers to complexity measured on ordinal scale and S stands for size measured on ordinal scale, is referred to as Task Dimensions (Dim).

Equation (18) illustrates the meaning of the task dimensions concept.

$$Dim = (T, C, S) \quad (18)$$

Definition 21 (Implementation Moment). Implementation Moment (IM) is the number of sampling times from the moment when a task was started to present divided by the first estimation (in number of sampling times) provided by that task owner.

Equation (19) is used for computing IM for an in-work task started d days ago (ES_1 is the first effort estimation provided by task owner).

$$IM = \frac{d}{ES_1} \quad (19)$$

Besides the concepts defined above, the Work Behavior Prediction Model uses also concepts like: target task, target worker and target estimator. The target task is the task for which the prediction is made. Target worker refers to the target task owner. Finally, target estimator refers to the worker that first estimates the effort required for the completion of the target task.

Before presenting the structure of the Work Behavior Prediction Model, we further describe the data required by the model in order to operate.

3.5.2. Required information

The role of the Work Behavior Prediction Model is to forecast ES and EL values for a target task for a time span in the future. The requirements of this model refer to the information that needs to be available for the model to work. At first, the Work Behavior Prediction Model requires that all project tasks have an associated ES before being in-work. At second, this model requires that all project tasks have an associated Dim before being in-work. Finally, the Work Behavior Prediction Model requires that, at an organization level established sampling time, all the project workers report new ES or EL values for their assigned tasks or confirm the existing ES or EL values according to their work.

The requirements of the Work Behavior Prediction Model are not difficult to implement. For example, at task creation, the project manager might provide the first ES and might estimate Dim for the created task. Regarding ES and EL regular

reports from task owners, these reports will be made by using a software tool (that we are currently developing) meant to significantly simplify the reporting process.

Having presented this model's information requirements in order for it to operate, we proceed with the presentation of the model structure, describing in detail its inputs and outputs, followed by the illustration of its underlying prediction methodology.

3.5.3. The structure of the Work Behavior Prediction Model

This section presents the structure of the Work Behavior Prediction Model, focusing on its inputs, outputs and internal structure. The structure of the Work Behavior Prediction Model is illustrated at a general level, in fig.10, which shows how this model integrates in the Behavioral Monitoring Framework presenting its main inputs and outputs. For a more detailed look over the model structure, inputs and outputs, we propose fig.23. Fig.23 shows all the inputs and outputs of the Work Behavior Prediction Model along with the stages of prediction and the informational flows involved by the forecasting process. Next, we show how the representation of the Work Behavior Prediction Model in fig.10 relates to its detailed representation in fig.23.

Before proceeding, a key aspect in understanding how the Work Behavior Prediction Model operates is the fact that it makes predictions for only one task at a time. For making forecasts on a whole project, the Work Behavior Prediction Model makes forecasts on each task that belongs to that project. This decision is justified by the way this model is interconnected with the Project Status Model, illustrated in fig.10 and commented below.

As shown in fig.10, the Work Behavior Prediction Model has two inputs from outside the proposed framework, the estimated effort history and the elapsed effort history. This high-level inputs presented in fig.10 are detailed in fig.23 as I1, I2 and I3. In fig.23, I1 refers to information regarding target task: ES and EL history (if it exists) concerning target task and Dim (Dim target). I2 refers to information regarding the completed tasks of target worker: ES and EL history and Dim for every such task. I3 refers to information regarding all the tasks initially estimated by target estimator: ES provided by the estimator (ES_0) and the first ES provided by task owner (ES_1), and Dim for every such task. The last input, I3, is required only for tasks that are not yet started. Besides I1, I2 and I3, fig.23 shows another input, TS, which is the time span for which the prediction is made (e.g., a number of days). Please note that, as we will show in the next section, the Work Behavior Prediction makes the forecast for a prediction time span, this forecast being a chronologically ordered set of values rather than one value for each ES and EL work measurements and for the prediction time span provided as input. In fig.10 this input (TS in fig.23) is not present for the sake of maintaining the simplicity required by the high level presentation of the framework.

As stated earlier, the Work Behavior Prediction makes forecasts for one project task at a time. In this context, all the inputs and outputs in fig.23 refer to the forecasting of one task (even though information regarding other tasks is required, as suggested by the description of the inputs presented in fig.23, earlier in this section). However, in order to compute a predicted project status, the Project Status Model requires forecasts for all the tasks of the project, as shown in section 3.4.3, when we describe the status identification methodology of the Project Status Model. For being able to make the forecasts for all the project tasks, the Work

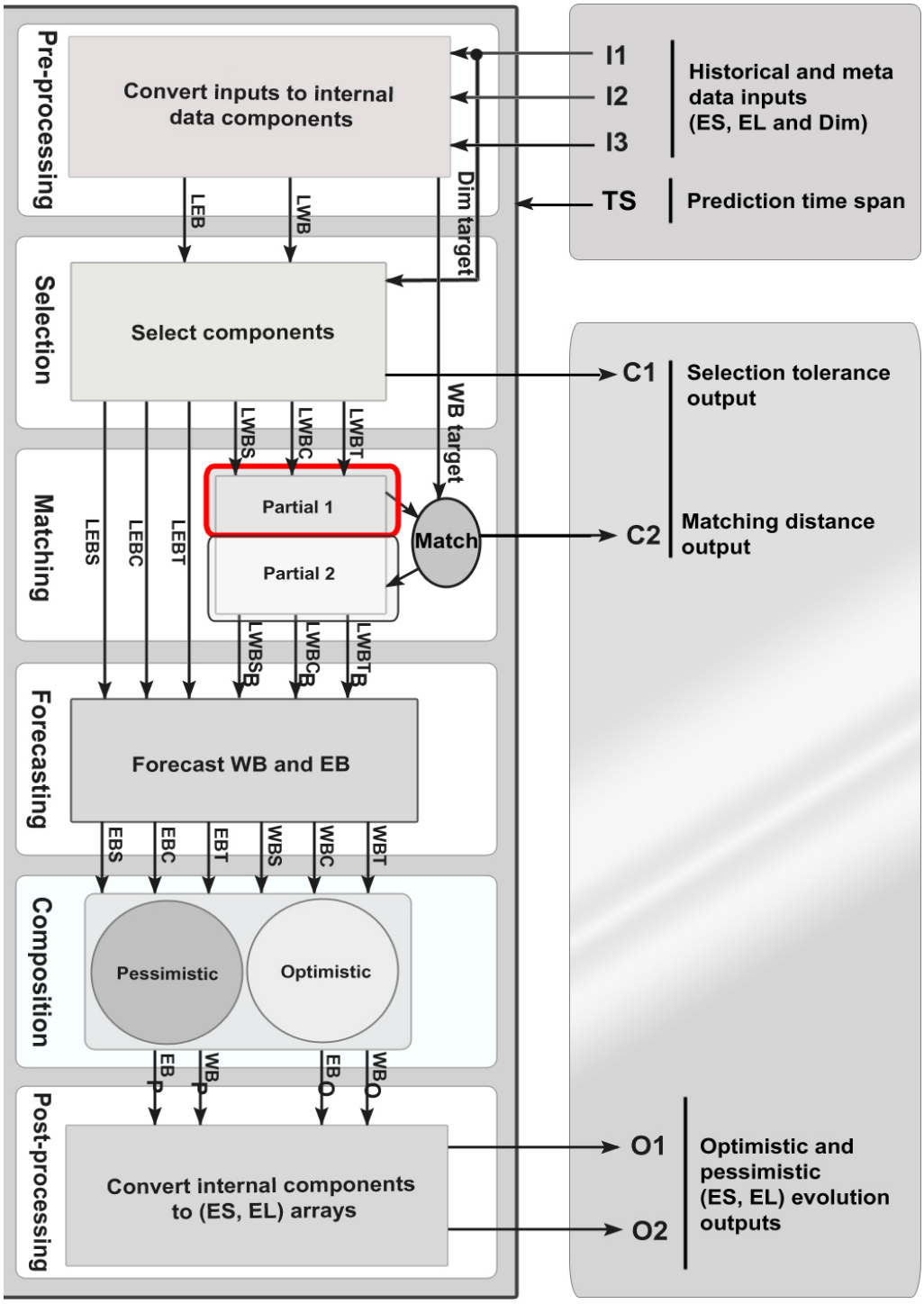

Figure 23. The Work Behavior Prediction Model

Behavior Prediction Model must know eventually about what tasks the project contains and to whom these tasks are assigned. This is why fig.10 shows another input for the Work Behavior Prediction Model, input that refers to the macro and micro universes (introduced for the Project Status Model), the prediction model processing the project tasks one at a time. However, focusing on already complex informational flows, fig.23 considers the macro and micro universes input from fig.10 to implicitly providing the required information for making the forecast on a task.

In the high level representation of the Work Behavior Prediction Model in fig.10, this model has one output that is the forecasted evolution of the remaining efforts for project tasks. As shown earlier when discussing the model's inputs, the prediction is made for one task at a time by the Work Behavior Prediction Model. The entire process of making forecasts for all the project tasks produce the model's output shown in fig.10. In the more detailed view which regards the prediction process for just one task presented in fig.23, the model has two prediction outputs: an optimistic prediction (O1) and a pessimistic prediction (O2) of ES and EL evolution for the given prediction time span. The outputs O1 and O2 retrieved by the forecasting process for all the tasks of the project, makes for the output presented in fig.10. As presented in the description of the status identification methodology of the Project Status Model (section 3.4.3), forecasts are provided for each task in terms of ES and EL (their values in the prediction time-span), which are further used in the computation of the predicted evolution of the project status. The model has also two outputs that describe the conditions in which the prediction outputs are obtained: C1 and C2 (explanations regarding the meaning of C1 and C2 are presented later in this section), which for the sake of simplicity were omitted from the high-level fig.10.

The Work Behavior Prediction Model contains six stages: Pre-processing, Selection, Matching, Forecasting, Composition, and Post-processing. Please note that the WB components that appear in these stages refer to both ES and EL histories. The model stages are described next defining the prediction methodology, which uses Definitions 12 to 21 along with equations (8) to (19).

3.5.4. Prediction methodology

The Work Behavior Prediction Model defines a prediction methodology that is described next. This methodology follows the six stages of the model which will be presented in details in this section.

Before proceeding with the description of the prediction stages, it is important to note that there are three cases for the target task:

Case I. the target task is not yet started, so that it has no available historical information regarding progress reports

Case II. the target task have just been started so that the WB_{target} (work behavior concerning target task) cannot be computed yet because at least three reports are required for this (condition imposed by the definitions regarding this model).

Case III. the target task is in-work and WB_{target} can be computed.

Each of the following prediction stages consists in several actions. The actions performed in a prediction stage are conditioned by the above cases in which the target task is situated in that some actions are performed and some actions are not depending on those cases. Consequently, for each particular action of a stage,

the cases for which it applies are provided. As the cases descriptions show, a target task can only be in one case at the time when the prediction is made.

3.5.4.1. Pre-processing

This is the first stage of the model. In this stage, the input I1 is converted into Work Behavior resulting WB_{target} (shown in fig.23). This action is performed only for a Case III target task (description provided above). I2 is converted into a list of work behaviors (LWB). This action is performed in all three cases. I3 is used for computing a list of estimation behaviors (LEB). This action is performed only for a Case I target task. Please note that LWB and LEB contain elements for every task referred in I2 and I3 respectively. The elements of LWB and LEB contain also information regarding Dim, besides WB and EB elements for the tasks referred in I2 and I3.

In this stage, for a Case III target task, the implementation moment IM is computed for the target task. Based on this IM, the information in input I2 is spitted so that every element of LWB will not contain just a WB for a target task in Case III (as for Cases I and II will), but two WB components, one computed from start to IM adjusted for the particular list element, WB_{before}, and one from the adjusted IM to completion moment, WB_{after} (every element of LWB refers to a task; the adjusted IM for that task is obtained by multiplying IM of target task with ES_1 of the task).

3.5.4.2. Selection

In this stage, the elements of LWB and LEB, if available, are filtered considering Dim target. The selected LWB and LEB elements are not necessary those with identical Dim as Dim target. In case such elements do not exist, elements with similar Dim to Dim target considering an adaptive tolerance are selected. The tolerance used in selection is one of the outputs of the model (C1 in fig.23). The selected elements are categorized based on Dim components resulting WB lists for technology (LWBT), complexity (LWBC) and size (LWBS). In Case I, there are resulting EB lists also (LEBT, LEBC and LEBS).

3.5.4.3. Matching

This stage is used only in Case III. The LWBT, LWBC and LWBS are filtered based on an adaptive Euclidean distance which is one of the outputs of the model (C2 in fig.23). Only for a Case III target task, every list element (according to the first stage description) contains a WB_{before} (Partial 1 in fig.23) and a WB_{after} (Partial 2 in fig.23).

The filtering is made by computing the Euclidean distance between WB_{before} of every list element and WB_{target} and comparing the resulting distance to the current value of the adaptive Euclidean distance used by the model. The filtering results are $LWBT_B$, $LWBC_B$ and $LWBS_B$, the base lists each component of which containing only one WB element, WB_{after}. Only for the Cases I and II, $LWBT_B$, $LWBC_B$ and $LWBS_B$ are the selected lists of WB elements in the selection stage.

3.5.4.3. Forecasting

This is the internal prediction stage. All the previous stages were meant to transform the existing information regarding work into a uniform representation that may be used for understanding the particularities of the human factor regarding work, which is Work Behavior. We currently use a weighted arithmetic average as prediction algorithm. The lower Euclidian distance computed in the previous stage, the higher the weight of the respective work behavior.

The results are WB and EB components for the three elements of Dim: technology (WBT, EBT), complexity (WBC, EBC) and size (WBS, EBS). The WB components are obtained no matter the case in which target task situates. The EB components are computed only if target task is in Case I.

3.5.4.4. Composition

In this stage, the components of WBT, WBC and WBS are combined to form just two WB elements: optimistic (WB_O) and pessimistic (WB_P). This action is performed no matter the case in which the target task is situated (Case I, II, or III).

As stated earlier, WB is computed for both ES and EL. Considering this, the composition of the pessimistic and optimistic solutions for ES and EL is performed by following the next two rules.

- The optimistic WB components are obtained by selecting: ST_{max}, EV_{min}, TR_{min} for ES and by selecting ST_{min}, EV_{min}, TR_{max} for EL.
- The pessimistic WB components are obtained by selecting ST_{min}, EV_{min}, TR_{max} for ES and by selecting ST_{max}, EV_{min}, TR_{min} for EL.

The explanation of these choices comes from the definitions of this model. An interesting aspect is that for optimistic and also for pessimistic solution selection, the minimum EV must be selected. This comes from the definition of the EV metric presented at the beginning of this model's description.

Regarding EB selection (Case I only), for the pessimistic solution component (EB_P) EB_{max} is selected and EB_{min} is selected for the optimistic solution (EB_O).

3.5.4.5. Post-processing

In this stage, the internal components (WB and EB) are converted, using the TS model input, into optimistic and pessimistic predictions of pair values of ES and EL. In Case I, the starting ES (ES_1) is computed using the pessimistic and optimistic EB components. In Cases II and III, the starting ES is considered the current ES.

Afterward, in Cases I, II and III, the future pessimistic and optimistic evolution of ES and EL are computed for the given time span (TS) based on the optimistic and pessimistic WB components.

3.5.5. The identification of the future status of a project

This section focuses on presenting how the future status of a project is computed by the Project Status Model based on the forecasts provided by the Work Behavior Prediction Model. For this, we consider fig.10, focusing on the

interconnection between the Project Status Model and the Work Behavior Prediction Model.

As shown in fig.10, the Project Status Model provides the Work Behavior Prediction Model with the macro and micro universes of a given project. The Work Behavior Prediction Model is applied for all tasks in the macro-universe of the project, for one task at a time, as shown in section 3.5.3. The prediction time span input (TS) of the Work Behavior Prediction Model illustrated by fig.23 and described in section 3.5.3 determines the time period for which ES and EL are predicted for the tasks in the project macro-universe. Finally, the Project Status Model is provided as illustrated in fig.10 from the Work Behavior Prediction Model with the predicted evolution of the remaining efforts for the project tasks. For computing a forecasted project status for a particular moment in time in the prediction time-span, the predicted ES and EL predicted for that moment in the future for all the tasks in the project macro-universe (for which prediction was maid individually, as insisted in section 3.5.3) are used by the Project Status Model in its underlying status identification methodology (described in section 3.4.3). This way the Project Status Model computes the evolution of project status (Project Status Model output in fig.10), which actually is a collection of predicted project statuses for the prediction interval.

Because the Work Behavior Prediction provides to the Project Status Model the predicted evolution of remaining efforts for the project tasks (the tasks in the project macro-universe as this concept was introduced in Definition 7 at Project Status Model definition), the Work Behavior Prediction Model can be regarded as the model of the proposed framework that provides a dynamic perspective over project monitoring, showing how project progresses in terms of how each task of the project progresses to completion.

3.5.6. Adaptations for scarce datasets

Real-world is characterized by scarce datasets. Generally, only the basic information is provided by the project team members regarding their own work progress. Such information might refer to the remaining effort logging related to their tasks. Most of the time, no additional information regarding task nature (technology, size and complexity) is provided. Moreover, the logging might be done from time to time, at non-equal time spans.

This section presents the adaptations of the Work Behavior Prediction Model for scarce data sets. Before proceeding, this adapted methodology uses the Work Behavior concept defined in section 3.2 (Definition 6). Moreover, it operates only with remaining effort as work measurement. Ideally, the histories used in the forecasting process should contain elements for equally distanced moments in time. If this is not the case, an extrapolation method on the existing data is used beforehand.

Definition 22 (Implementation Moment for scarce datasets). Given a History for an in-work task, the Implementation Moment (IM) is the number of History elements divided by the first History element's Remaining Effort.

Equation (20) shows the Implementation Moment for scarce datasets computed on a History H. Please note that a first History element, H(0), of value 0 (meaning an initial Remaining Effort of 0 effort units) makes no sense.

$$IM = \frac{n}{H(0)}, n - size\ of\ H. \tag{20}$$

Definition 23 (Virtual Present). Given a History for a completed task and an Implementation Moment of an in-work task, Virtual Present (VP) is the first History element's Remaining Effort multiplied by the given Implementation Moment.

In other words, Virtual Present is the position of a given in-work task's present in the History of a completed task. Equation (21) shows a Virtual Present computed on a History H of a completed task and for a given Implementation Moment IM of an in-work task.

$$VP = H(0) \times IM. \tag{21}$$

The Work Behavior Prediction methodology for scarce datasets is presented in fig.24 and described next. The prediction process starts with the selection of a project task to be the subject of prediction. This is the target task in fig.24. The tasks are represented as histories of remaining efforts. This is why a time axis is shown for each task in fig.24.

The target task has a History, named Known history in fig.24. Based on this History, the target Work Behavior (WB_{target}) is computed. As shown in fig.24, completed tasks are used in the prediction process. These tasks actually represent a selection of completed tasks that have their assignee in common with the target

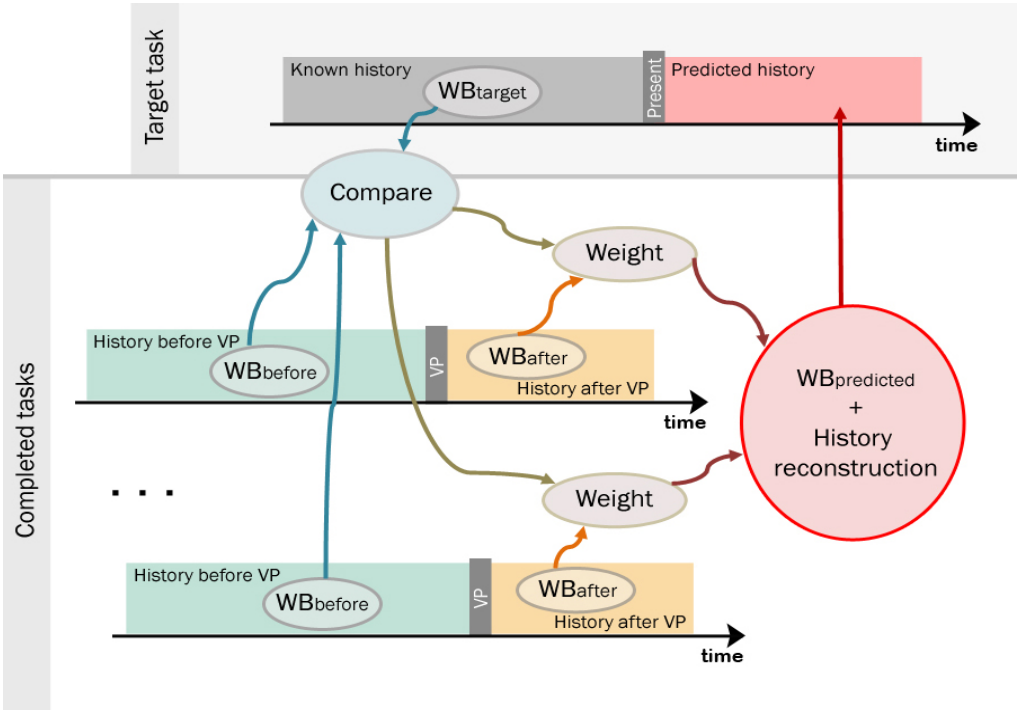

Figure 24. Work Behavior Prediction methodology for scarce datasets

task. Their histories characterize the behavior towards work of their assignee, this being a good reason for using their histories in the forecasting process.

The Virtual Present (VP) shown in fig.24 for the completed tasks is computed using Definition 23 and equation (21) based on the Implementation Moment (IM) computed for the target task using Definition 22 and equation (20).

For the target task, the Implementation Moment IM is computed. By using IM, the Virtual Present is computed for all the completed tasks selected for prediction (a VP is computed for each completed task in fig.24). This way, the Histories of the completed tasks are split into two parts, so that the History for a task contains a History before the Virtual Present of that task, and a History after this Virtual Present. In case the History after the Virtual Present for a task contains no element (this is a possibility), that task is ignored in the prediction process.

For each History before VP in fig.24, a Work Behavior is computed resulting a WB_{before}. In the same time, for each History after VP in fig.24, a Work Behavior is computed resulting a W_{Bafter}.

The WB_{before} elements are than compared with WB_{target} producing a weight for each WB_{after} element, which will be further used in the prediction process. The closest WB_{before} to WB_{target} produces the biggest weight for its twin, WB_{after}.

Next, the WB_{after} elements are weighted and combined for computing the predicted Work Behavior ($WB_{predicted}$ in fig.24). A weighted mean is used in this process. The Known history in fig.24 is used along with $WB_{predicted}$ to build a History structure that corresponds to the predicted progress for the target task (Predicted history in fig.24).

The forecasts upon project progress evolution, at task level, that are outputted by the Work Behavior Prediction Model are used, along with information regarding project's current status, within the Project Status Analysis Model.

3.5.7. Conclusions

The Work Behavior Prediction Model is a component model of the Behavioral Monitoring Framework that enables the forecasting of the effort estimation, elapsed effort or of the remaining effort (as shown in the model's adaptation to scarce datasets) of a task for a defined prediction time-span.

Regarding its integration to the Behavioral Monitoring Framework, the Work Behavior Prediction Model provides the predicted evolution of remaining efforts for project tasks to the Project Status Model, enabling this last model to compute predicted project statuses. Moreover this predicted evolution outputted by the utilization of the Work Behavior Prediction Model for each of the project tasks (one at a time, as shown), provides a dynamic perspective over how the project and its tasks progress to completion. In this context, due to this dynamic perspective offered to monitoring, we consider that the Work Behavior Prediction Model is the core of the proposed Behavioral Monitoring Framework.

The key concept on which the Work Behavior Prediction Model is based is Work Behavior. We believe that without using such a concept, no prediction methodology could have been developed to work with the heterogeneous data that characterize the work progress reports (consider reported estimated efforts and reported elapsed effort for tasks of different sizes and complexities) in the way the underlying forecasting methodology of the Work Behavior Prediction Model does. As shown when we described the prediction methodology of the Work Behavior Model, for predicting the future evolution of a task, know evolutions of other tasks are

employed. This prediction methodology that uses fact from other tasks to compute a forecasted evolution of a target task is the innovation that the Work Behavior Prediction introduces.

Next, we present the Project Status Analysis Model which is the last component model of the Behavioral Framework and which has the role of analyzing the current project status as well as a predicted status of the monitored project.

3.6. The Project Status Analysis Model

This model is concerned with providing: the current status of the monitored project in a format that facilitates the representation of the status in a more human-readable manner, recommendations for the workers in order to maximize tasks completion rate, and automated notifications regarding detected project execution problems [93].

The Project Status Analysis Model is able to analyze the current status of a project as well as future probable project statuses, in order to provide valuable recommendations to workers regarding work prioritization and early notifications regarding project execution problems. Next, we present the underlying definitions and equations of the Project Status Analysis Model.

3.6.1. Definitions and equations

This model operates with Definitions 7-11 introduced earlier in this chapter for the Project Status Model. Besides these definitions, the Project Status Model is described by two groups of equations: equations used in provide individual recommendations on work prioritization for the human resources involved in the project, and equations that are used in finding project execution problems that project manager must be aware of. In the following sections, we present these equations grouped by their role.

3.6.1.1. Equations regarding recommendations

In this subsection, the focus is on the Project Status Analysis Model's equations that describe the recommendations concerning the order of task execution for the workers.

The recommendations provided by the Project Status Analysis Model refer to local task sequences, and more specifically to the task order of execution that can be chosen by the workers for their tasks. A possible solution for this issue is provided by the scheduling methods used in operating systems for ordering the execution of processes. A good candidate for establishing the recommended local task order of execution is the shortest remaining time scheduling method as illustrated in [90] in the context of operating systems. According to this scheduling method, the task with the smallest remaining execution time to completion is executed first. An advantage of this scheduling method refers to the fact that the short tasks are handled very quickly. This is especially important in the context in which, shorter tasks generally have earlier deadlines established in the project execution plan, so that a worker is better to finish the short tasks first than to pause the short tasks while trying to finish large tasks. Another advantage of the shortest

remaining time scheduling method is that it requires little overhead because the worker starts a new task when the current task is completed or a new task, with lower remaining time, is ready to be stared. The overhead in the context of project execution refers to the effort required by the worker transition from working on a task to working on another task. The amount of these transitions ought to be as little as possible during the execution of a project. Consequently, the shortest remaining time scheduling method is a good candidate for the recommendation strategy of the proposed monitoring model. The following equations, (22) and (23) together, define the criteria for ordering the local task sequence of a worker, (22) being the first sub-criteria and (23) being the second.

The recommended order is obtained through ordering the tasks assigned to a worker so that (22) is true for all worker's tasks, and if several tasks have the same values for PES, further ordering of these tasks so that (23) is true for all the tasks assigned to the respective worker.

$$PES_{t_i} \leq PES_{t_{j'}} (t_i, t_j) \in ord \tag{22}$$

$$ES_{t_i} - EL_{t_i} \leq ES_{t_j} - EL_{t_{j'}} (t_i, t_j) \in ord \tag{23}$$

The meaning of (22) is that the tasks in a local sequence are ordered by their PES value in an ascending manner. The meaning of (23) is that the local tasks with the equal PES values are ordered by their remaining execution time.

3.6.1.2. Equations regarding project execution warnings

For an effective management, a status analysis model must be able to identify problems in project execution and to notify these problems through alarms. The generated alarms may concern a worker, the manager or both.

Based on project status, several project execution problems can be identified. We believe there are three main alarm categories based on project status: alarms regarding work assignation, alarms regarding work progress, and alarms regarding effort estimation changes.

3.6.1.2.1. Alarms regarding work assignation

An alarm of this type may be generated when deviations from the execution plan might occur because of the manner in which the work is assigned. This alarm concerns the worker and its aim is to make the worker decide upon the rejection of their new assigned task.

Another alarm of this type may be generated during task execution. This alarm concerns the project manager and its aim is to make the project manager decide upon the re-assignation of one of the involved tasks. This type of alarms is generated when (25) is true for at least two tasks (t_i and t_k) assigned to a worker, considering (24). In equation (24), D_{ti} and D_{tk} are those introduced in task's definition (the definitions from Project Status Model) and refer to the due dates established at task creation.

$$D = \max\{D_{t_i}, D_{t_k}\} \quad (24)$$

$$(ES_{t_i} - EL_{t_i}) + (ES_{t_k} - EL_{t_k}) > D - T \quad (25)$$

Consider the assignation of a new task to a worker. In (24) and (25), t_i and t_k are two tasks assigned to the respective worker, so that one of these tasks is the new assigned task and the other is a task that was earlier assigned to the same worker. In this scenario, the meaning of (24) and (25) is that, if there is an earlier assigned task so that the sum of the remaining effort for this task and the remaining effort for the new assigned task are greater than the remaining time to the latest due date of the two tasks, then the worker must decide upon the rejection of the new assigned task.

Alarms of this type may be generated during task execution as well, when there are two tasks assigned to the same worker so that the sum of their remaining effort are grater then the remaining time to the latest due date of the two tasks. In such a situation, the project manager must decide upon the re-assignation of one of these tasks.

3.6.1.2.2. Alarms regarding work progress

Alarms of this type are generated when the work progress endangers the completion of a particular task at the established due date. This type of alarms concerns both the worker to which the problematic task is assigned and the project manager. When (26) is true, an alarm is generated.

$$ES_{t_n} - EL_{t_n} > D_{t_n} - T \quad (26)$$

The meaning of (26) is that, considering the effort estimation and the elapsed effort for a task (introduced in the definitions form the Project Status Model), if the remaining time to the established due date is not enough to complete the task, the project manager and the involved worker must find an appropriate solution for this situation.

3.6.1.2.3. Alarms regarding effort estimation changes

These alarms are generated when a worker re-estimates the effort required to complete an owned task, t_n. In (27), δ is the hierarchical dependency relation defined on the set of tasks of the same project, so that $(t_n, t_i) \in \delta$ means that t_i depends not necessary directly on t_n.

$$ES_{t_i} > D_{t_i} - WES_{t_i}^{WT_i},\ (t_n, t_i) \in \delta \quad (27)$$

An alarm is generated when (27) is true for at least one task that depends on the task that the worker re-estimates. This alarm concerns mainly the worker who makes the re-estimation. The meaning of (27) is that the re-estimation of an owned task influences the starting time of a future task in such a way that the future task will not be able to be completed at its established due date. The aim of this alarm is to make the worker reconsider their new estimation.

Having introduced the underlying definitions and equations of the Project Status Analysis Model, we proceed with describing how this model integrates in the Behavioral Monitoring Framework. In the next section, we present the underlying methodology of the Project Status Analysis Model for status interpretation, recommendations elaboration and project execution problems identification starting from the integration in the proposed framework.

3.6.2. Status interpretation, recommendations and project execution warnings

As shown in fig.10, the Project Status Analysis Model has two inputs that both are provided by the Project Status Model: the current project status and the predicted evolution of the project status (which is a collection of predicted project statuses). These inputs, the involved actions of the model and the corresponding outputs are presented next. We further use fig.25, which shows a detail upon the Project Status Model as presented in fig.10, to discuss model's inputs and outputs.

The current project status provided by the Project Status Model (as shown in fig.25) is presented in the form of a collection of quadruplets (ES, EL, PES, WES)

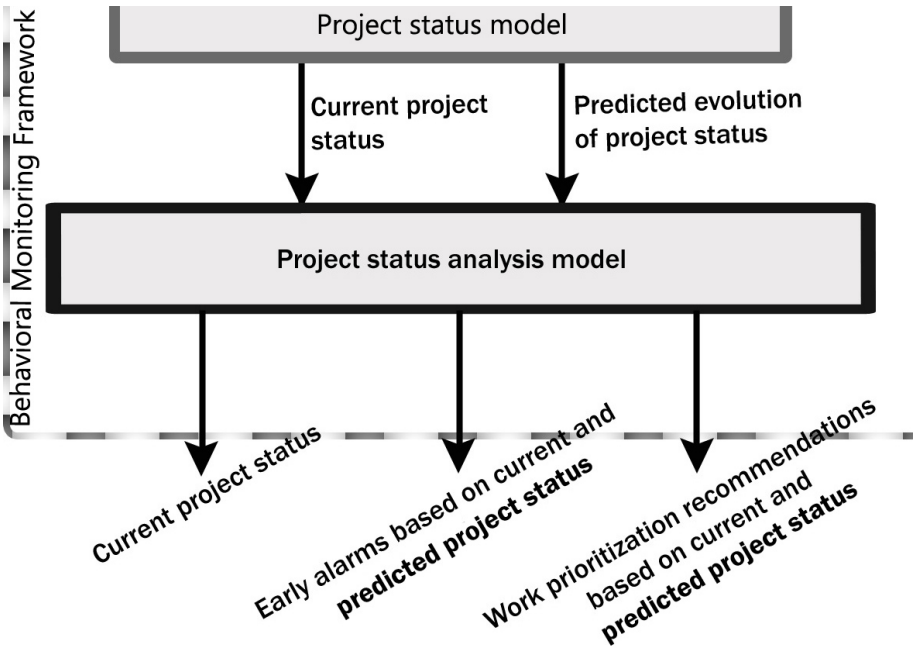

Figure 25. The Project Status Analysis Model

for all project tasks, according to Definition 11. One role of the Project Status Analysis Model is to interpret the status provided by the Project Status Model and to output it, eventually, in a more readable form. The Project Status Analysis Model does not define a formal approach for the transformation of the current project status provided as input into something more suitable for humans to read to be provided as output of the Project Status Analysis Model. The reason for this is that the project status as introduced in Definition 11 can be processed to provide any derived information that might be of interest. For example, if the project status is required to be represented as a Gantt chart, the start date of a task (this is the early start) is that task's WES, while the expected due date considering the respective project status provided by the Project Status Model, is WES to which is added the remaining effort (ES - EL). The task completion percent, that is EL/ES, may also be represented in the Gantt chart. Please note that EL, ES and WES are computed for the present (current project status) or for a given moment in the future (project probable status in the future).

Besides status interpretation, the Project Status Analysis Model is able to elaborate recommendations to project team members concerning work prioritization using the current project status or a predicted project status from the predicted evolution of the project status input (which is basically a collection of forecasted project statuses) employing equation (22) and (23). So, another output of this model, as shown in fig.25 refers to these recommendations. Depending on the input (current or a predicted project statuses), recommendations can be computed for the present time or for a moment in the future. Regarding the recommendations upon task prioritization, depending on the software implementation, this model is able to use three types of constraints for building a full schedule for each resource involved in the project: constraints at project level, constraints at worker level, and constraints at task level. The constraints at project level refer to predecessor tasks that must complete before a task can start and to successor tasks that depend to the completion of a given task within a project. The constrains at worker level are similar to the first type of constraints except that this second type of restrictions don't refer to tasks that belong to a project, but to tasks that are assigned to a single given worker. Finally, the constraints at task level refer to restraints regarding start date or due date. In Microsoft Project, tasks can be assigned restrictions like "Must Start On" or "Finish No Later Than" (among others) [60]. The utilization of this last type of constraints is dependent to the application with which the software prototype of the proposed framework integrates. If the software implementation uses as input data Microsoft Project Plan files, then these constraints are available and can be considered. On the other hand, if such constraints are not available as inputs, the recommendations consider only the first two types of constraints. Fig.26 shows an example of recommended prioritization of work for a worker that has two assigned tasks, task A and task B, provided the fact that task A has a "Must Start On" restriction for day 0, while task B has a "Finish No Later Than" for day 4. fig.26.a shows the current prioritization of work, which is Task A followed by Task B. Considering the given constraints the Project Status Analysis Model is able to generate the recommended project plan individually for the worker involved as shown in fig.26.b: the worker should start task A in day 0 with respect to its constraint. Even though task A is not yet completed, in day 1 the worker should start the work on task B in order to complete it according to its constraint in day 4.

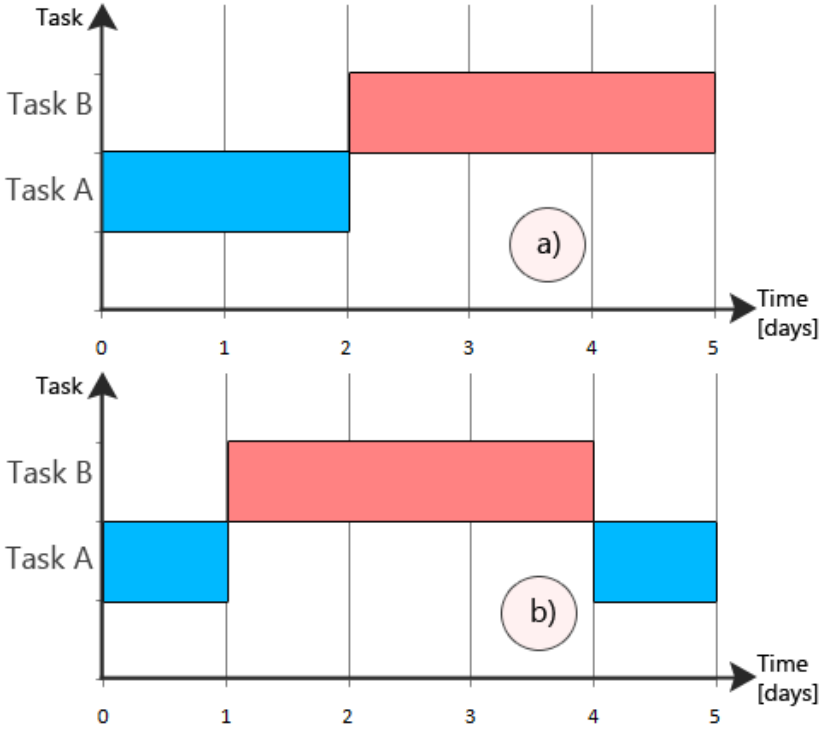

Figure 26. Own tasks prioritization: a) current task order; b) recommended order

There are situations when it is impossible for a worker to meet all the constraints for all the assigned tasks in a given period of time. For example, if two tasks must be completed by tomorrow, but each task requires 8 hours of work for completion, than it is physically impossible for the assignee to cope with those deadlines. Such situations are caused by bad work performance from assignee's part or by faulty task assignation decisions from project manager's part. Whatever the source, the Project Status Analysis Model is able to early identify these and many other similar situations and to offer a solution for each. Fig.25 shows that another output of the Project Status Analysis Model refers to these alarms which just like the recommendations, can be identified on the current or on a predicted project predicted project status depending on the input used. For identifying current or predicted project execution warning (or alarms), the underlying equation (24) to (27) of the Project Status Analysis Model are employed.

3.6.3. Case study

For exemplifying the utilization of the Project Status Analysis Model, it is used the status resulted in section 3.4.4 by applying the Project Status Model for the information available in Table 1 and fig.22.

To be able to use the Project Status Analysis Model, every project task must be assigned a due date, D. Table 2 shows the due date, PES and WES for all tasks involved in the current project status analysis. In Table 2, T is the current time.

At first, the information required for a Gantt chart representation can be computed. Thus, the start date for every task is its WES. The actual due date for every task is WES + (ES - EL). The completion rate for every task is EL/ES. Consequently, considering the Gantt chart representation of P_2, the project status is as follows:

1. Task Θ_4: start date = T + 2, actual due date = T + 4, completion percent = 33%
2. Task Θ_6: start date = T + 4, actual due date = T + 5, completion percent = 66%

Table 2. Task information for Project Status Analysis Model case study

Task	D	PES	WES	Parent Project
Θ_2	T + 2	T	T	P_1
Θ_4	T + 3	T	T + 2	P_2
Θ_6	T + 7	T	T + 4	P_2
Θ_7	T + 20	T	T + 5	P_1
Θ_8	T + 10	T + 5	T + 5	P_2
Θ_9	T + 15	T + 5	T + 5	P_2

3. Task Θ_8: start date = T + 5, actual due date = T + 8, completion percent = 0%
4. Task Θ_9: start date = T + 5, actual due date = T + 11, completion percent = 0%

Regarding the recommendations to workers, by applying equations (22) and (23) to the micro-universes of the involved workers in the development of P_2, the worker W_{10} receives the recommendation to swap task Θ_4 with task Θ_6 in their micro-universe.

The model identifies an alarms regarding work assignation by applying (24) and (25) to the micro-universes of the workers involved in the development of P_2. This alarm concerns tasks Θ_2 and Θ_4. Through this alarm, a software prototype of the framework will inform the project manager about the situation and is asked to re-assign one of the tasks. The worker W_{10} is also notified about the situation in order to make sure they understand why the project manager must re-assign a task they own.

Alarms regarding work progress or alarms regarding effort estimation changes are not identified by the model in the current example.

3.6.4. Concluding remarks

The Project Status Analysis Model is used in conjunction with the Project Status Model and the Work Behavior Prediction Model for a more efficient project monitoring and control. Its aim is to interpret the information provided by the Project Status Model and translate it into a format that is suitable for humans to read, that is the project status document.

In addition to project status presentation role, the Project Status Analysis Model has the capability to provide recommendations to workers in order to maximize their task completion ratio. A very important aspect regarding this refers to the fact that the recommendations are not made on a project level, but on an organization level, meaning that these recommendations are not limited to the tasks of a single project. The recommendations are consider all the tasks from all the projects that are developed in the same time-span within an organization. This feature is especially important for organizations that develop software projects, where it is common for a human resource to be involved in more than one project at a time.

One of the most important capabilities of the Project Status Analysis Model refers to the early warnings that it can provide to project managers, enabling them to be aware of the existing or potential project execution problems and to take early corrective actions. The Project Status Analysis Model is able to identify project execution problems based on the current or a predicted project status, which makes it very responsive in determining the events that can get the project out of its track.

3.7. Conclusions

This chapter presents the behavioral monitoring approach which is implemented by the proposed Behavioral Monitoring Framework for project monitoring.

This chapter starts by introducing the Behavioral Monitoring Framework as the behavioral approach to monitoring that we propose. This framework is a collection of three models: the Project Status Model, the Work Behavior Prediction Model and the Project Status Analysis Model, that work together in a synergy for a more efficient project monitoring. This first part of the chapter presents the structure of the proposed monitoring framework, with its component models represented as black-boxes and the informational flows that exist within the framework and between the framework and the environment in which operates. Before defining the component models in detail, the very important concept of Work Behavior was presented. This concept and the way Work Behavior can be used in the modeling of progress histories are key aspects of our proposed monitoring approach as shown in this chapter. Moreover, for justifying our design decisions regarding the Behavioral Monitoring Framework (e.g., its structure and features), we propose a classification and an analysis of the accuracy levels of the project statuses as project management documents in correlation to the types of managed projects (e.g., small sized, large-scale), concluding that the highest accuracy level is required for an efficient monitoring and control of the problematic large-scale software projects. As shown in this chapter, such an accuracy level can only be obtained by employing an integrated approach to monitoring that is able to offer a dynamic perspective over project progress, providing a forecasting feature.

After defining Work Behavior and after presenting and discussing the project status accuracy classification, we introduce the component models of the proposed framework. We start by defining the Project Status Model with its concepts of project macro-universe and worker micro-universe. These concepts are very important in the data gathering methodology involved by the Project Status Model. The Project Status Model, through its concepts and equations, actually defines the information that is needed for building the project status. In this context, not only the relations among a project tasks that are established by the project manager in the project planning phase are important, but also how each human resource involved in the project (e.g., developers, testers, designers) prioritize their tasks during project implementation. Moreover, many times the human resources within an organization are assigned tasks from more than one project in the same time span. Consequently, knowing the real project status of a project is a difficult task. The project manager needs to be aware of the state of virtually all the tasks that are implemented within virtually all the projects developed in an organization. Due to the concepts and equations with which operates, the Project Status Model is able to provide the project status that takes into consideration all the connections among the project tasks within an organization, even when these connections are on assigned human resources (e.g., a human resource is assigned tasks from several different projects in a time-span). Due to its inputs which come from the operational environment and that represent project progress information but also work prioritization or precedence decisions, the Project Status Model can be regarded also as a data gathering methodology.

After presenting in details the Project Status Model, this chapter introduces the next component model, the Work Behavior Prediction Model. Due to the fact that the most important aspect of the Behavioral Monitoring Framework is that it provides a dynamic perspective over project progress allowing it to provide a higher accuracy project status (as shown earlier in this chapter), the Work Behavior Prediction Model can be regarded as the core of our proposed framework. The prediction model uses a forecasting methodology based on the concept of Work Behavior enabling the utilization of past experience for foreseeing future progress trends.

The last model of the Behavioral Monitoring Framework is the Project Status Analysis Model. This model is able to analyze the current status of a project, as provided by the Project Status Model, or a predicted status which is obtained by forecasting the project status by using the Work Behavior Prediction Model. This last possibility is especially important because the analysis provided by the Project Status Analysis Model is able to indicate that a particular problem is very likely to occur in a defined time-span, enabling project manager to take early corrective actions. The Project Status Analysis Model is able to provide work prioritization recommendations to project team members considering the organization perspective (e.g., project team members are generally assigned tasks from more than one project at a time), these recommendations being destined to accelerate the task completion ration for project team members. A very important feature of the Project Status Analysis Model is that it is able to identify current or probable issues of project execution regarding time overruns. The model can find several types of problems signaling their presence not only to the project manager, but also to the involved project team members, for early, effective, and transparent corrective actions.

As shown throughout this chapter, the behavioral monitoring approach, which consists in the employment of the proposed Behavioral Monitoring

Framework, is suitable for the tracking and control of software projects with certain benefits for the most problematic software projects that are the large-scale ones, being an integrated approach to monitoring and offering the possibility for process automation.

4. Behavioral Framework Software Prototyping

As shown in the previous chapter, the Behavioral Monitoring Framework is defined as a collection of models each specialized to perform a particular activity within the monitoring process. As a consequence, the proposed framework can be implemented by a software tool, enabling the automation of the monitoring process, which is very useful especially when it comes to the problematic and difficult to monitor and control large-scale software projects.

Implementing the underlying methodologies of the Behavioral Monitoring Framework in a software prototype is critical in what regards the prototype's design for an efficient utilization of the proposed framework's features and benefits as presented in the previous chapter.

In this context, this chapter proposes, at first, a set of design decisions that must concern any software implementation of the Behavioral Monitoring Framework, discussing the specifications, architecture and main features of such a prototype. At the end of this chapter, we present the software prototype of the Behavioral Monitoring Framework that we implemented for framework validation purposes.

4.1. Requirements for software prototypes

This section presents the specifications, architecture and main features that we recommend for any software implementation for the proposed Behavioral Monitoring Framework, defining the main requirements for these software prototypes.

Any software prototype of the proposed framework should be based on three design principles that are presented and explain next:

1) *Ease in progress reporting*: due to the fact that project team members use the implementation of the framework to report progress on a regular basis, the reporting process must be as simple as possible, disturbing as little as possible project team members from their assigned work.
2) *Transparent decision-making*: the warnings that signal existing or predicted project execution problems should not only be available to the project manager, but also to the involved human resource; this way, any corrective actions considered by the project manager can benefit from the support of the project team members that have a responsibility in the existing problems.
3) *Enhanced communication*: the users of the software implementation of the Behavioral Monitoring Framework should be offered a simplified way for accessing all the information related to the projects in which they participate (e.g., by employing a plug-in for the email client they use every day).

In fig.27, that illustrates the architecture that we recommend for any software implementation of the proposed framework, which as shown in the following sections cope with each of the above three principles of design. In fig.27,

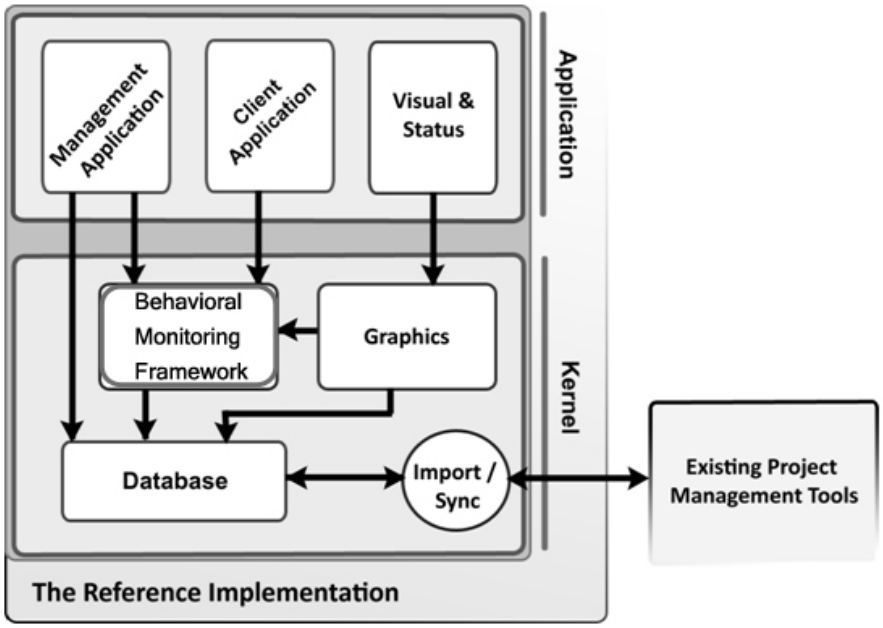

Figure 27. The architecture of a software prototype

the arrows suggest that there are interfaces through which the blocks pointed by the arrows are accessed. The implementation architecture should contain, as shown in fig.27, two levels: the kernel level and the application level. These levels and their components are presented next.

4.1.1. Kernel level

The kernel contains: Database, Algorithms, Graphics, and Import/Sync blocks. The kernel is the part of the monitoring prototype that implements the proposed monitoring framework.

The database holds all data used by the software prototype and contains three types of tables. The first type of tables refers to those that hold the settings which are set by the organization and by its project managers. These settings may refer to the amount of time between two consecutive work progress reports, the default prediction time span, the maximum selection tolerance and the maximum Euclidean distance used by the Work Behavior Prediction Model etc. The second type contains the tables that hold the information regarding tasks, projects, and resources. The third type of tables is of great importance considering the monitoring process. These are the log tables and they hold the project structure and work assignment changes, the reported work progress concerning project tasks, as well as the relevant events occurring during the project implementation, such as alarms and recommendations.

The algorithms module is the residence of the proposed monitoring framework. The database provides the data required by the algorithms module, which implements the proposed monitoring framework. In our current

implementation of the software prototype which is written in .NET C# the classes in this module access the database (which is held by a Microsoft SQL Server) through .Net LINQ [17]. Concerning our current implementation particular technologies, there are two .Net specific technologies to access the database. One is LINQ and the other is the so called Datasets. Before choosing LINQ, we studied the comparative behavior of the two .Net specific technologies, and the conclusions were that Datasets requires more time for connecting to the database than LINQ and the memory required by Datasets is greater than that required by LINQ. These conclusions determined us to choose LINQ as the technology to access the database.

Another important module in this architecture, the graphics module offers the methods for fetching and processing data from database in order to provide the required information for the application level of the software prototype. The data provided by this module is used in visualizations allowed by the dynamic perspective over project progress offered by the Behavioral Monitoring Framework, as described in the previous chapter.

The Import/sync module provides the possibility to import and synchronize prototype data from and with existing project management tools. Because the existing project management tools don't have a uniform API, methods for importing and synchronizing data differ from one tool to another. However, our current software prototype works with Microsoft Project Plan files.

4.1.2. Application level

The application level of the reference implementation contains: the Management application, the Client application, and the Visual and Status module. The components of the application level are presented next.

The management application offers the features for adding, editing, and removing data in the database. An administrator named by the organization that uses the software prototype use this application for adding, editing, and removing resources, groups of resources, tasks, and projects. Also, the organization project managers use this application to edit work assignments, project structure and tasks. Furthermore, the project managers use this application to access the work progress information concerning managed project and triggered alarms related to project execution problems. Fig.28 shoes a screenshot of the management application as provided by our currently developed software prototype.

The client application is used by the workers in order to facilitate the work progress reporting and the communication between system and workers. The client application provides a list of active tasks that are assigned to the worker. To report work progress, the respective worker has only to click a task in the list and to provide the effort estimation for that task. Moreover, the alarms concerning workers are available as information in this application so that the controlling of own work is enhanced. The client application is very important for the effectiveness of the proposed monitoring framework. Consequently, the application must provide the essential information regarding own work for its users and the possibility for very quick and easy work progress reports.

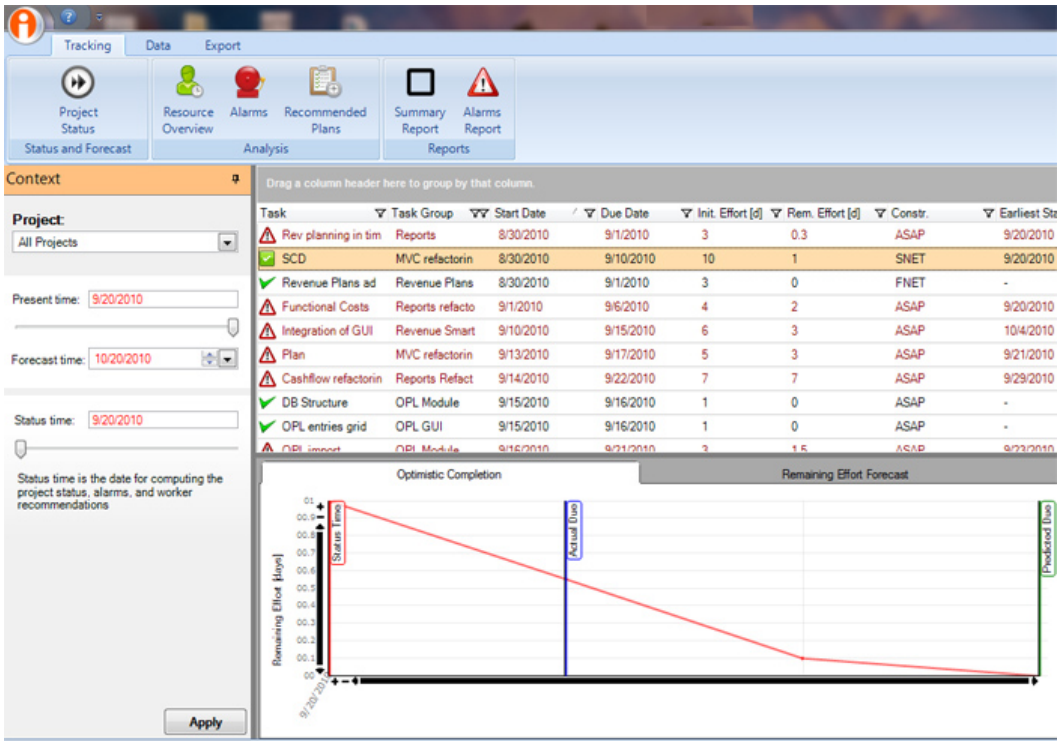

Figure 28. A screenshot of the management application (current implementation)

Finally, the visualization and status module uses the graphics module outputs and it is able to draw charts based on the provided data.

4.1.3. Integration with existing software management tools

The software prototype should be able to integrate with existing software management tools. Our current software application that implements the framework is able to import data from Microsoft Project. However, for the software implementation of the proposed monitoring framework more complex integration with other existing project management tools might be considered. For example, a more tightly integration will require synchronization as a feature of the software implementation of the proposed framework. Moreover, synchronization might be offered in two flavors: light and tight synchronization. Light synchronization is used when changes made to the integration project management tool file must be visible to the prototype, but changes made to data used by the prototype are not required to be visible to the corresponding project management tool file. On the other hand, tight synchronization is used when data changes must be visible to both framework prototype and project management tool no matter the where the changes are made.

Having presented the main specifications, design decisions and features expected from any software application that is implements our framework for an effective utilization of its benefits we describe next the software prototype that we developed for validation purpose.

4.2. Software prototype for validation purposes

For validating the framework, we developed a distinct software prototype of our Behavioral Project Monitoring Framework, which follows broadly the architecture described earlier in fig.27. The aim of this prototype is to be used as a tool in the validation of the core of our Behavioral Monitoring Framework, which is the Work Behavior Prediction Model, the model that adds the dynamic perspective over the monitored project progress and which is the main responsible for the fact that the Behavioral Monitoring Framework can be used in the elaboration of a Level 3 accuracy project status as described earlier in this chapter.

The software prototype for validation purposes works with Microsoft Project Plan files, providing the possibility for importing such files and to manage the projects for which imports were made. Because only Microsoft Project Plan files are used as data inputs (no additional information regarding project tasks size, complexity and technology being available), this software prototype implements the Work Behavior Prediction Model for scarce datasets. In addition to this model, this software prototype for validation purposes implements an auxiliary forecasting model, which is the competing prediction method considered for validation. Currently the auxiliary module implements the Velocity Trend Prediction, a method that will be described in the next chapter of the thesis that presents the primary validation of the Behavioral Monitoring Framework.

If project progress reports (Microsoft Project Plan files updates) are not available on a daily basis (which is very common when using Microsoft Project as data provider), a custom extrapolation method that we developed is used by the software prototype for validation purposes. We named this custom extrapolation method the Edge Work. In Edge Work, if one has 20 days to an assigned task's due date and 10 days as remaining effort for the same task, at the start of the 20-day interval one will work 5 days for the respective task, after which one leaves the task for 10 days (maybe working on other tasks), and then comes back to the respective task to complete it (the last 5 days). In the first 5 days, the work is expected to be concerned with the major requirements of the task. This is the most stressful part of a task work and, in Edge Work, is performed as early as possible. The last 5 days are concerned more with the fine-tuning, and is expected to be less stressful. In Edge Work, this last work part is performed as late as possible, with respect to the existing constraints.

For comparing the effort forecasts made with the Work Behavior Prediction Model and with the competing (auxiliary) prediction method to the real effort values from the existing reports, this software prototype is able to use a given set of error metrics. Currently the error metrics available in the implementation are: MFE (Mean Forecasting Error), MAD (Mean Absolute Deviation), MAPE (Mean Absolute Percentage Error), and WMAPE (Weighted Mean Absolute Percentage Error). More details regarding these error metrics and why we decided to select them will be provided in the next chapter that presents the primary validation of the proposed framework.

The software prototype offers the possibility to choose a date for the present, so that all the available project data (from Microsoft Project Plan files imports) to this date are used in the prediction process. A forecast is computed for a selected "future date" (considering the selected present date and not the actual present) which is also a date at which we have project data from Microsoft Project Plan files imports. The obtained forecasts by using Work Behavior Prediction and

competing (auxiliary) prediction method and the actual data are used for computing the implemented set of error metrics. Consequently, as desired, remaining effort forecasts computed for a "future date" are compared to remaining effort values from actual project reports (from Microsoft Project Plan files) data that is available for the same date.

Fig.29 shows a chart outputted by the software prototype for validation purposes that we developed. This chart presents the actual evolution of the remaining effort for a task versus the forecasts made with Velocity Trend Prediction (VTP), which is the competing method currently implemented by the prototype, and with our Work Behavior Prediction (WBP). The forecasts are computed considering a selected date for present, marked by the "Now" flag in fig.29, and for a selected "future date", marked as "Selected Future" in fig.29.

4.3. Conclusions

This chapter discusses the software prototyping decisions involved by the implementation of the Behavioral Monitoring Framework in a monitoring software application.

This chapter contains two parts. The first part discusses the basic specifications, architecture and features that we recommend for any software application that implement the Behavioral Monitoring Framework in order to fully benefit from its concepts, methodologies and capabilities.

The second part of the chapter presents the software prototype that we developed for the validation of the proposed Behavioral Monitoring Framework. This software implementation is able to compare forecasts to real values from existing reports for the same moment in time by using several error metrics, providing the comparison results in a document that contains error metrics values.

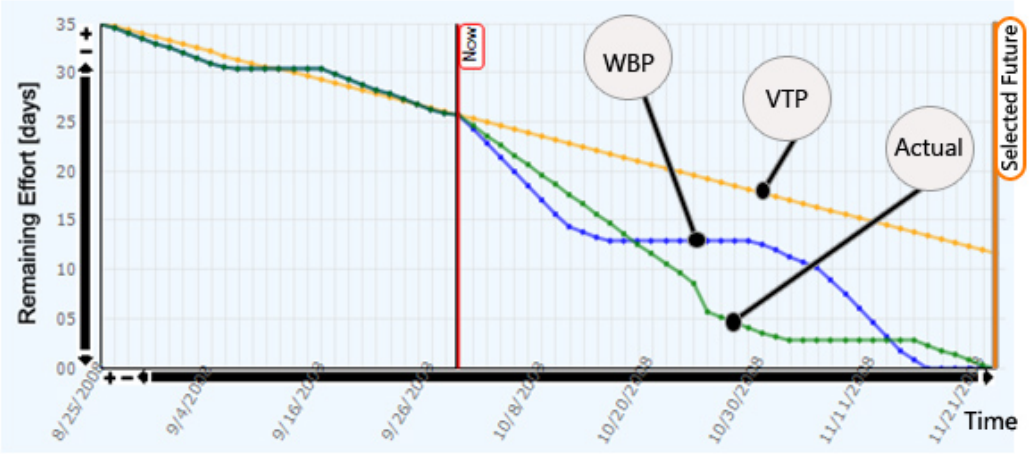

Figure 29. A screenshot of a chart showing forecasts and actuals

5. Experiments on Real-World Data

In this chapter we present the experiments used for the primary validation of the core of our Behavioral Monitoring Framework, which is represented by the Work Behavior Prediction Model. This model provides the dynamic perspective over the monitored project progress to the proposed framework and assures the possibility of computing a project status of Level 3 accuracy, as described in the previous chapter. The experiments that we performed will be described in detail along with the obtained results in this chapter.

5.1. Data used in experiments

In the experiments that we conducted for the primary validation of the core of the Behavioral Monitoring Framework came from two real-world commercial software projects developed by two European companies (one from Germany and the other from Romania).

The data provided by those companies consist in project progress reports that include names of human resources, names of project tasks and other sensitive information related to commercial projects. Consequently, due to the legal aspects that concern to this collaboration with these two companies for using such data, we will not disclose in this thesis the real names of the projects, development companies, tasks and human resources.

We will further refer to the first project which was developed by the German company as project X, and to the second project, developed by the Romanian company, as project Y. Project X was an automotive project, developed by a project team of 23 members of different nationalities, its implementation starting in 2008 and ending 2 years later. Meanwhile, project Y was a pure software project, developed by a team of 6 members of the same nationalities, its implementation starting in the summer of 2010 and ending 2 months later.

The data for those two projects were provided in the form of Microsoft Project Plan files. A Microsoft Project Plan file has a table-like structure, containing on each row details regarding a project task. From those details, we use in our experiments several values regarding task's parameters: remaining effort, established due date, established start date, task constraints, task successors and task predecessors. All these are provided by each Microsoft Project Plan file for each task that exists in the structure of the project at the time when the Microsoft Project file was elaborated.

The data that we use in experiments come from successive elaborations of Microsoft Project Plan files from those two projects. For the larger project X we had available for experiments 12 Microsoft Project Plan files, elaborated between 2008 and 2009, one file per month. For the smaller project Y, we had available only 6 Microsoft Project Plan files elaborated between August and September 2010, one per week.

In the experiments that we present here, we used only the information from the available Microsoft Project Plan files and no other supplementary information regarding the tasks or the human resources involved in these two projects.

5.2. Velocity Trend Prediction

Velocity Trend Prediction is a very popular forecasting methodology used during project development. Velocity Trend prediction uses concepts of its parent framework, Scrum, like Sprint, Backlog, and Burndown Chart [83].

A Sprint is an iteration of work. The Backlog defines the work for a Sprint. The Burndown Chart depicts the total effort remaining per Sprint.

In Scrum, considering also our context, Velocity is how much backlog effort a team member can handle in one Sprint. This can be estimated by viewing previous Sprints, assuming the Sprint duration is kept constant.

Regarding the evolution of Backlog and the representation in fig. 30, the change in terms of Backlog among Sprints can have two causes: the work spent and the work added or removed from one Sprint to another.

Fig.30 shows a Velocity Trend forecast, on a Burndown Chart. The methodology of the Velocity Trend Prediction is described next, considering fig.30 as a starting point. In fig.30, Sprint 4 is the current Sprint and corresponds to the present. Work is expected to be completed in Sprint 7, which is the intersection of the Velocity Trend line (the red line in fig.30), with the Burndown Chart's abscise. The Velocity Trend line, which is the red line in fig.30, is defined by two points: one is the Backlog value in Sprint 1 (which is equivalent to the initial remaining effort for a task, considering our context) and the other is the Backlog value in Sprint 4

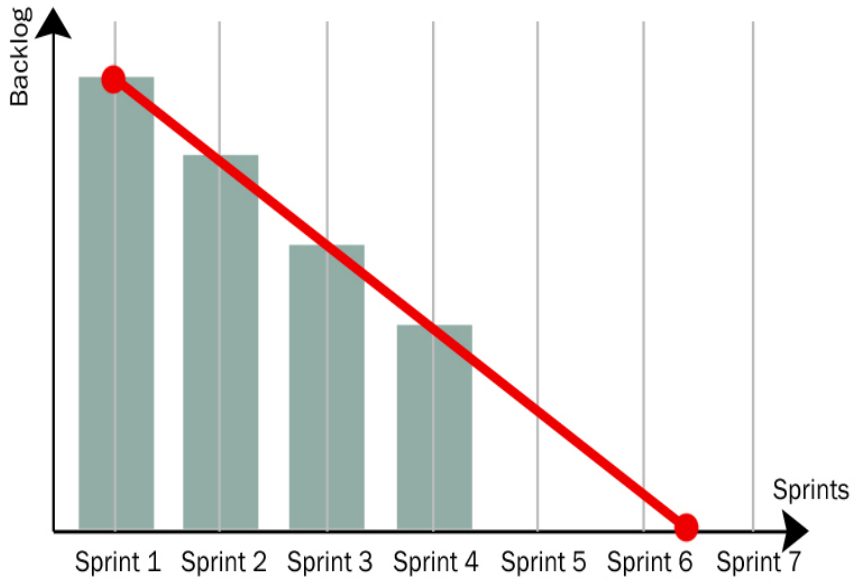

Figure 30. Velocity Trend Prediction

(which is equivalent to the remaining effort for a task at present date).

Because of its simplicity and proved effectiveness, Scrum's Velocity Trend Prediction is implemented by most ALM tools that offer a forecasting capability. Consequently, we use Velocity Trend Prediction in our experiments as a competing method for our Work Behavior Prediction, as we show next, in the presentation of the experimentation methodology.

5.3. Experimentation methodology

In the experiments that we perform in order to primarily validate or Behavior Monitoring Framework, we employ the software prototype for validation purpose of which implementation was described at the end of the previous chapter.

The software prototype used in experimentation implements Velocity Trend Prediction as the competing forecasting method for our Work Behavior Prediction. As shown earlier in this chapter, Velocity Trend Prediction is a simple and effective forecasting methodology that is implemented by most ALM tools that provide forecasting as a feature. Moreover, the Velocity Trend Prediction is able to work with the same type of data as our Work Behavior Prediction that is data regarding project progress from Microsoft Project Plan files. This means that it can be used on the data that we use in these experiments that was described in paragraph 5.1.

We use several error metrics to assess the prediction quality. These metrics, along with their strengths and weaknesses are presented next and described in [104]. These are the metrics that are implemented by the software prototype that we developed for validation purposes. In the following equations, D represents an observation, F is a forecast, and n is the number of (D, F) pairs.

The simplest metric is MFE (Mean Forecasting Error). Equation (28) shows how this metric is computed. A value of 0 doesn't mean that the accuracy is 100%, but that the prediction is on target (the negative and positive deviations cancel out). This metric is recommended to be used in conjunction with other metrics.

$$MFE = \frac{\sum_i (Dt - Ft)}{n} \tag{28}$$

Another metric used in this evaluation is MAD (Mean Absolute Deviation). Equation (29) shows how this metric is computed. A lower MAD means a lower prediction error. Unlike MFE, positive and negative deviations cannot cancel out here. The weakness of this metric is that its value is a number, so that it cannot be interpreted as large or small just in relation to the data it applies.

$$MAD = \frac{\sum_i |Dt - Ft|}{n} \tag{29}$$

The third metric used in this evaluation is MAPE (Mean Absolute Percentage Error). Equation (30) shows how this metric is computed. It measures absolute deviation of forecast from observation as a percentage of observation and indicates the persistent absolute error in forecast. A lower MAPE value means a lower forecasting error. Although MAPE, also known as MMRE, is the most common measurement of forecast accuracy, it has an important weakness, as demonstrated in [35]: MAPE will always be lower for models that provide an estimate below the

mean than for models that predict the mean. Moreover, observations with low amplitudes can produce large distortion to this metric's value.

$$MAPE = \frac{\sum_t \left|\frac{Dt - Ft}{Dt}\right|}{n} \times 100 \quad (30)$$

The last metric used in this evaluation is WMAPE (Weighted Mean Absolute Percentage Error). Equation (31) shows how WMAPE is computed. A lower WMAPE value means a lower prediction error. Because this is a weighted measure, it does not have the same problems as MAPE such as over-skewing due to low amplitude observations. However, this metric has its own weakness: observations with large amplitudes can bias the metric value in their favor.

$$WMAPE = \frac{\sum_t \left|\frac{Dt - Ft}{Dt}\right| \times Dt}{\sum_t Dt} \quad (31)$$

These four metrics, as presented above, have weaknesses and strengths as shown previously. This is why we don't use one, but all these metrics in this evaluation.

The software prototype automatically computes the four metrics for all the project tasks for which data is available, so that the index i of D and F from equations (28), (29), (30), and (31) refer to one task.

A prediction method is considered better than the other for a prediction case if at least three of the available metric values are lower for the first method (considering, of course, the metrics that are used for this evaluation for which lower means better).

5.4. Results and discussion

The forecasts evaluation results are presented in Table 3, for project X, and Table 4, for project Y. Table 3and Table 4 show the prediction time span, which is measured in months, in the case of project X, and weeks in the case of the smaller project Y. The main reason for making predictions on such time spans was that project development data is available on a monthly-basis, in the case of project X, and on a weekly-basis, in the case of project Y. Consequently, forecasts at the end of the prediction time span can be compared to existing information regarding project progress.

The four metrics used in evaluation that were presented in the previous section, are computed for Velocity Trend prediction (VPT in Table 3 and Table 4) and for our prediction method, Work Behavior Prediction (WBP in Table 3 and Table 4).

In Table 3 and Table 4, the cases in which our prediction method (WBP) is better than Velocity Trend prediction (VTP) are shaded.

Table 3. Evaluation results for project X

Prediction time span	Case no.	WMAPE		MAPE		MAD [days]		MFE [days]	
		VTP	WBP	VTP	WBP	VTP	WBP	VTP	WBP
1 month	1	0.579	1.149	3.003	25.000	8.530	16.936	7.579	-6.302
	2	0.673	0.583	123.15	44.741	8.480	7.352	7.887	-3.229
	3	0.665	0.276	44.117	40.149	5.425	2.249	1.312	-0.796
	4	0.458	0.769	5.228	14.662	3.043	5.108	1.736	0.990
	5	0.577	0.616	20.224	21.754	13.170	14.073	4.409	-5.245
	6	0.683	0.692	27.281	25.658	11.560	11.711	8.635	-3.166
	7	0.501	0.305	40.469	7.843	10.199	6.212	5.614	0.040
	8	1.094	0.919	40.014	29.534	16.322	13.709	13.647	0.777
2 months	9	0.822	1.402	21.579	30.357	8.315	14.180	7.897	-1.670
	10	3.350	1.146	94.713	97.422	12.462	4.264	12.026	2.087
	11	1.669	1.026	10.881	7.491	6.964	4.282	3.180	1.079
	12	1.512	1.180	10.281	6.799	6.864	5.358	2.064	2.242
	13	0.752	1.047	25.561	26.972	13.892	19.357	7.562	-7.988
	14	1.481	1.079	65.402	17.527	16.919	12.246	15.022	-1.330
	15	2.345	1.563	7.584	9.833	19.276	12.845	18.667	7.138
	16	0.673	0.873	42.902	56.699	24.956	32.393	-8.456	-30.89
	17	10.932	2.928	521.88	170.30	34.393	9.211	34.393	7.843
3 months	18	9.714	5.027	84.660	25.755	12.993	6.723	12.993	6.323
	19	-	-	-	-	9.995	4.533	9.995	4.533
	20	2.122	1.358	13.808	9.539	8.169	5.229	2.229	0.008
	21	1.527	1.366	11.404	4.649	5.561	4.973	2.594	2.885
	22	1.014	1.103	66.473	30.956	16.589	18.048	10.192	-7.120
	23	4.971	2.231	6.492	5.106	23.575	10.583	23.575	3.268
	24	0.903	0.800	14.429	15.622	17.103	15.152	7.837	-7.315
4 months	25	-	-	-	-	10.442	7.661	10.442	7.661
	26	-	-	-	-	7.886	3.914	7.886	3.914
	27	2.062	1.193	24.726	9.584	7.609	4.401	2.329	-0.025
	28	1.952	1.654	23.990	6.767	6.022	5.102	3.140	2.700
	29	2.455	1.329	21.497	10.196	19.826	10.733	19.826	-1.883
	30	2.577	1.222	7.264	7.638	27.233	12.914	19.433	-1.086
	31	42.685	4.021	11.111	11.111	32.725	3.083	31.192	1.550
5 months	32	-	-	-	-	8.154	7.661	8.154	7.661
	33	-	-	-	-	7.126	3.610	7.126	3.610
	34	3.233	1.573	52.009	14.440	8.704	4.236	3.709	0.231
	35	9.631	8.856	7.905	2.483	5.911	5.435	5.991	5.005
	36	6.069	2.216	2.511	4.762	28.900	10.551	28.900	1.027
	37	62.068	10.250	6.865	3.940	26.767	4.420	26.767	4.420
6 months	38	-	-	-	-	7.777	7.661	7.777	7.661
	39	-	-	-	-	7.275	3.535	7.275	3.535
	40	6.743	2.664	19.446	3.220	6.001	2.371	6.001	1.747
	41	2.971	1.561	9.210	7.692	22.857	12.005	22.857	-3.380
	42	75.979	13.015	6.667	22.693	34.950	5.987	34.030	5.987

Analyzing the results presented in Table 3 and considering all the available 42 presented cases, our prediction method (WBP) proves to be systematically better than Scrum's Velocity Trend prediction (VTP). The 1 month prediction time span shows the lowest differences between the two prediction methods. Even so, in 7 of the 8 cases our prediction method has a lower MFE, meaning that is more "on target" than the competing Velocity Trend method. The 2 month prediction time span shows better results for our prediction method in 6 of the 9 cases. For 3 month time span prediction, according to the metrics values, our prediction method is better in 6 of the 7 cases. Further analyzing Table 1, for 4, 5, and 6 month

prediction time span, our method is better than the Scrum's Velocity Trend prediction in all the cases.

The results presented in Table 3suggest that, for long term prediction, considering the available information, our method is more appropriate to be used for decision support than the popular Velocity Trend prediction. For example, for case 17 (Table 3), using Work Behavior Prediction, the project manager knows two months ahead of time where project tasks will be in terms of work progress with an average absolute prediction error per task of only 10 working days (see MAD for case 17 in Table 3) meaning 2 calendar weeks. Applying Velocity Trend Prediction on the same data and for the same time span, the average absolute error per task is 35 working days, meaning one calendar month and a half, which almost equals the prediction time span.

Analyzing the results shown in Table 4 and considering all the available 10 cases, we conclude than our prediction method is better than Velocity Trend prediction for project Y also. For 1 week prediction time span, our method shows better results in 3 of the 4 cases. For the other prediction time spans (2, 3, and 4 weeks), our prediction method is better in all the cases.

Just like for project X, the results for project Y, which are presented in Table 4, suggest that, for long term prediction, our method is more appropriate to be used for decision support than the popular Velocity Trend prediction. For example, for case 5 (Table 4), using Work Behavior Prediction, the project manager knows two weeks ahead of time where project tasks will be in terms of work progress with an average absolute prediction error per task of only 0.2 working days (see MAD for case 5 in Table 4) meaning 2 working hours, considering that a full working day consists in 8 working hours. Applying Velocity Trend Prediction on the same data and for the same time span, the average absolute error per task is 1.2 working days, meaning 10 working hours.

Table 4. Evaluation results for project Y

Prediction time span	Case no.	WMAPE		MAPE		MAD [days]		MFE [days]	
		VTP	WBP	VTP	WBP	VTP	WBP	VTP	WBP
1 week	1	0.333	0.083	33.333	12.500	0.750	0.188	-0.250	-0.188
	2	0.250	0.000	25.000	0.000	0.750	0.000	-0.750	0.000
	3	0.657	0.791	98.886	221.694	1.557	1.876	-1.107	-0.676
	4	0.318	0.070	72.727	27.895	0.382	0.084	-0.382	-0.084
2 weeks	5	0.500	0.083	41.667	12.500	1.125	0.188	-0.875	-0.188
	6	0.375	0.000	37.500	0.000	1.125	0.000	-1.125	0.000
	7	0.393	0.382	63.750	127.323	0.412	0.401	-0.337	0.326
3 weeks	8	1.159	0.250	262.500	137.500	1.912	0.413	-0.587	0.413
	9	0.625	0.000	62.500	0.000	1.875	0.000	-1.875	0.000
4 weeks	10	1.235	0.407	229.167	146.139	2.038	0.672	-0.962	0.153

Given these results, we consider helpful to visualize the trends of the error metrics values for the considered cases (those presented in Table 3 and Table 4). These trends are illustrated in fig. 31, 32, 33, 34, 35, 36, 37, and 38 by showing the linear regression for the error metrics values for the two considered projects. In

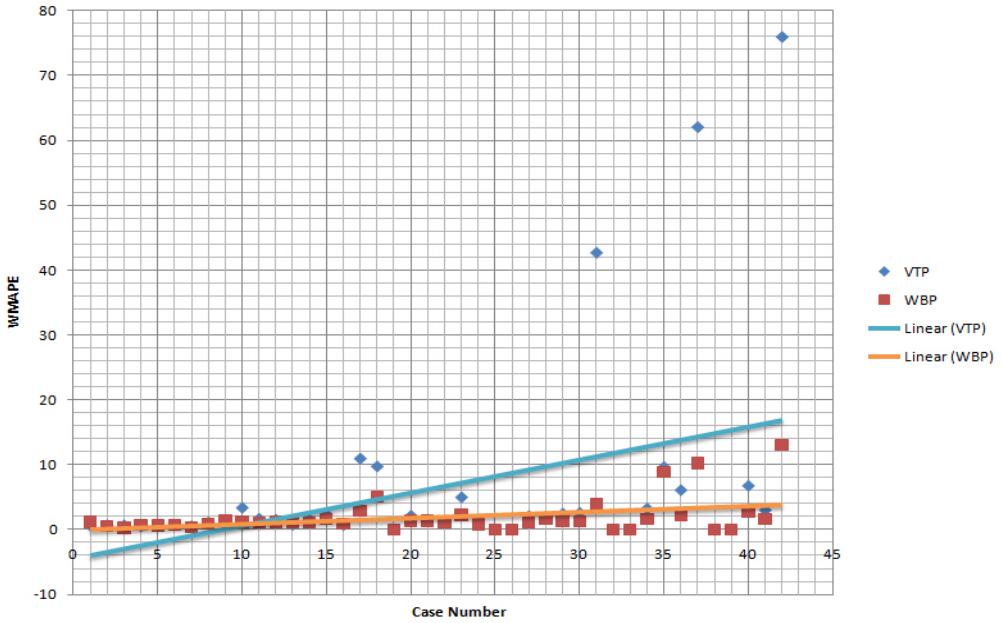

Figure 32. WMAPE for Project X

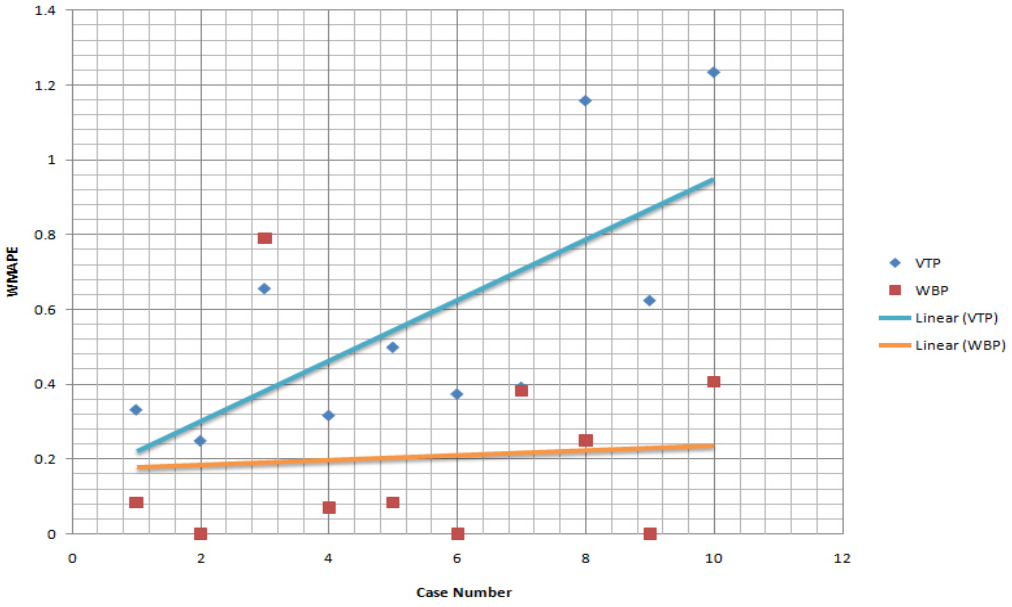

Figure 31. WMAPE for project Y

these figures, VTP stands for Velocity Trend Prediction and WBP for Work Behavior Prediction. Linear(VTP) and Linear(WBP) are the linear regression representations.

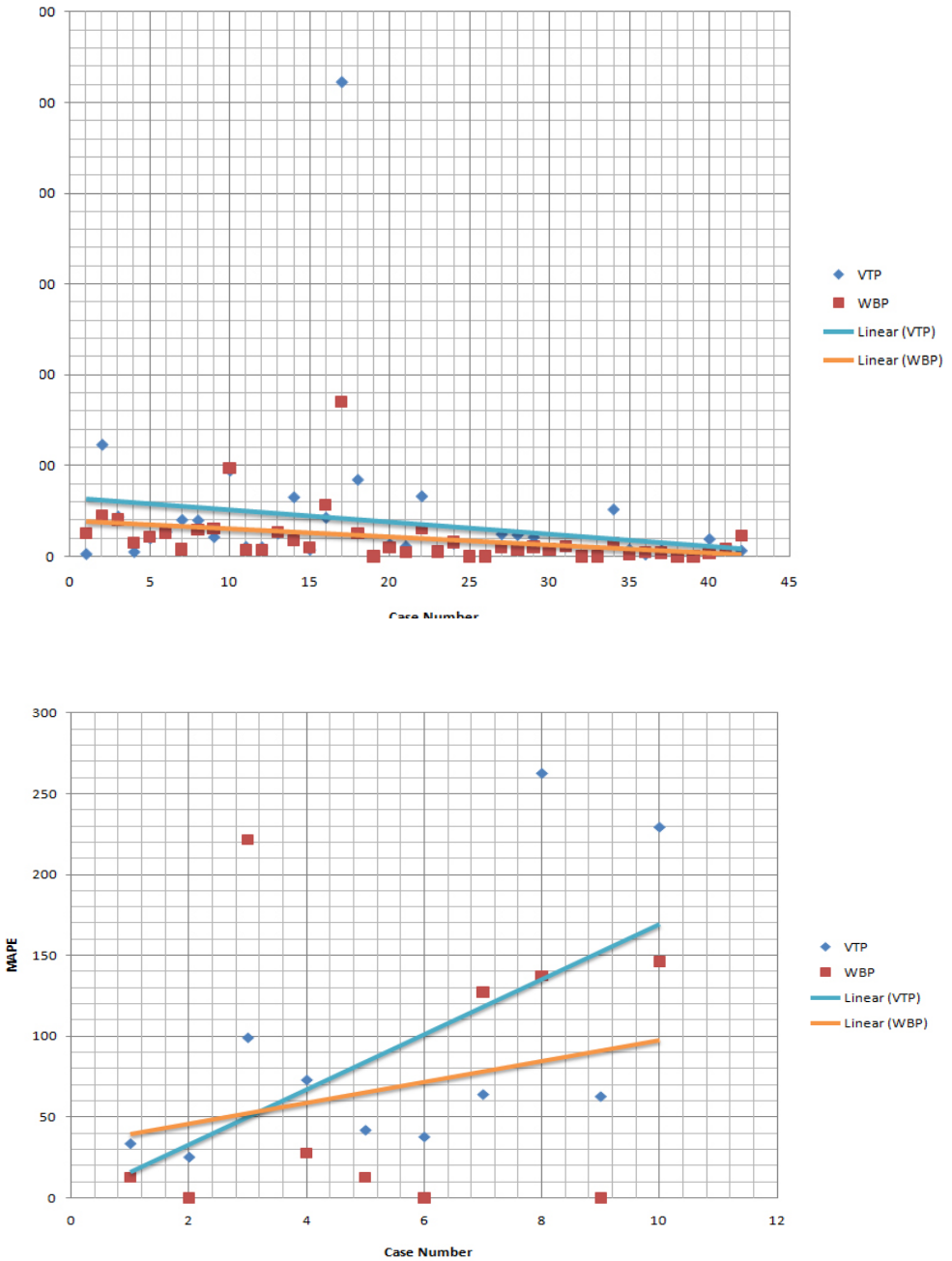

Figure 33. MAPE for project Y

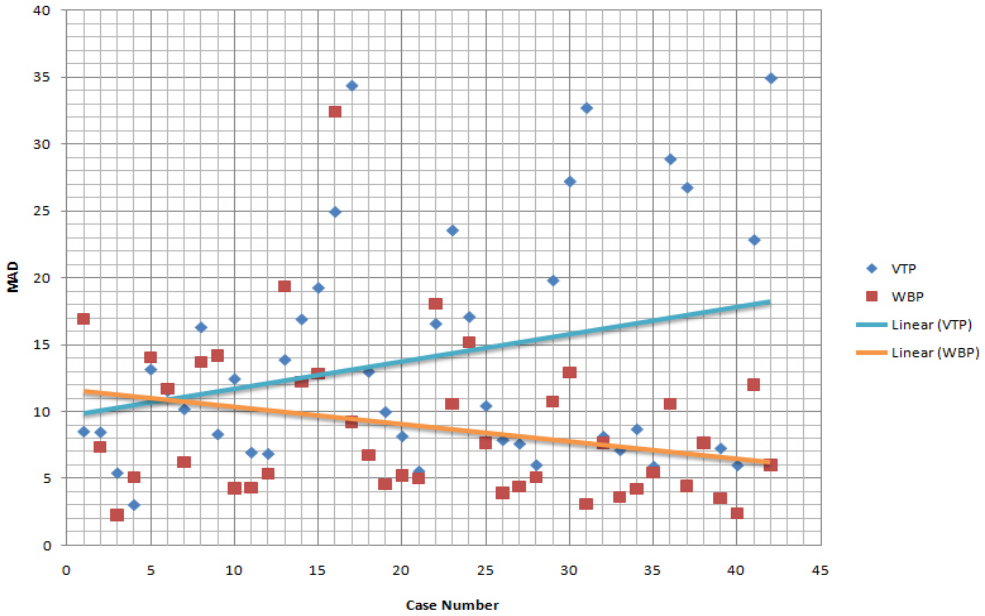

Figure 35. MAD for project X

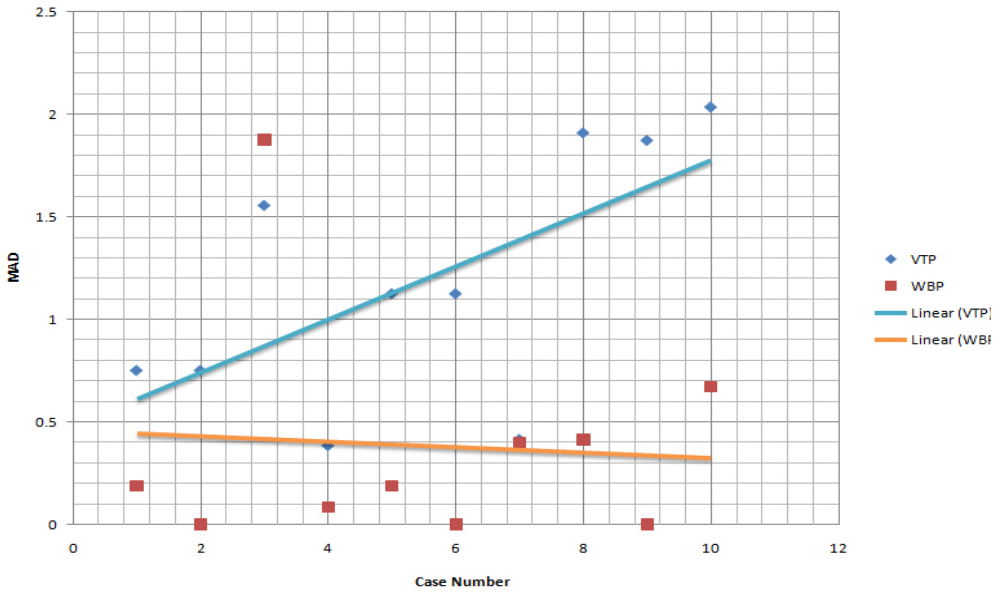

Figure 36. MAD for project Y

Although we evaluated our prediction method, Work Behavior Prediction, only on two real-world software project development data, we believe the results are valuable in the context in which such project data is very hard to get, considering its confidential nature. Even for those two projects, according to Table 3 and Table 4,

our method shows an evident superiority to a very popular prediction method, which is implemented by most ALM tools, Velocity Trend Prediction.

This prediction method is the only one against which we compared our Work Behavior Prediction so far. However, because Velocity Trend Prediction is so widely used, requiring for forecasting similar type and amount of data as our method, this was our first option for comparison.

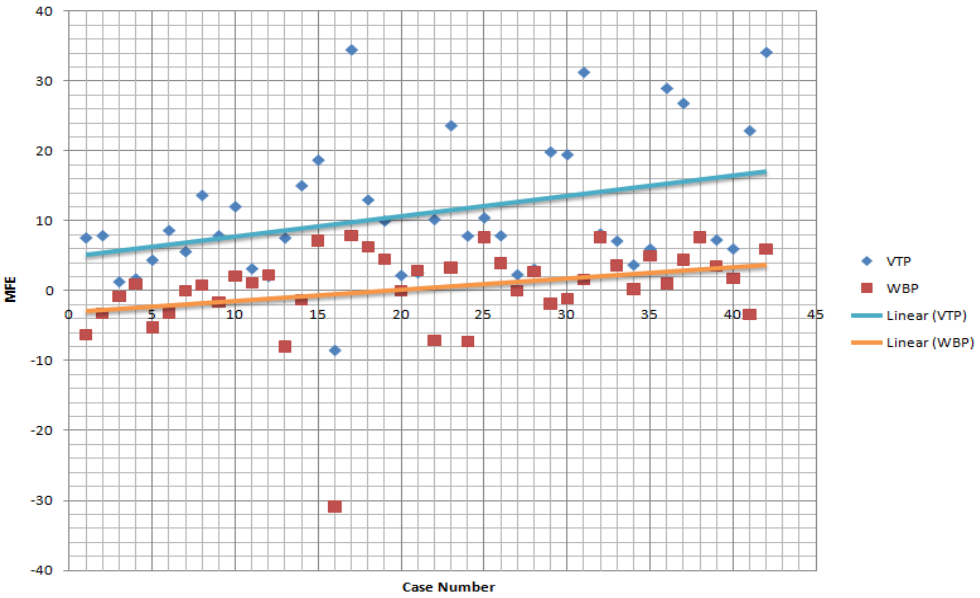

Figure 37. MFE for project X

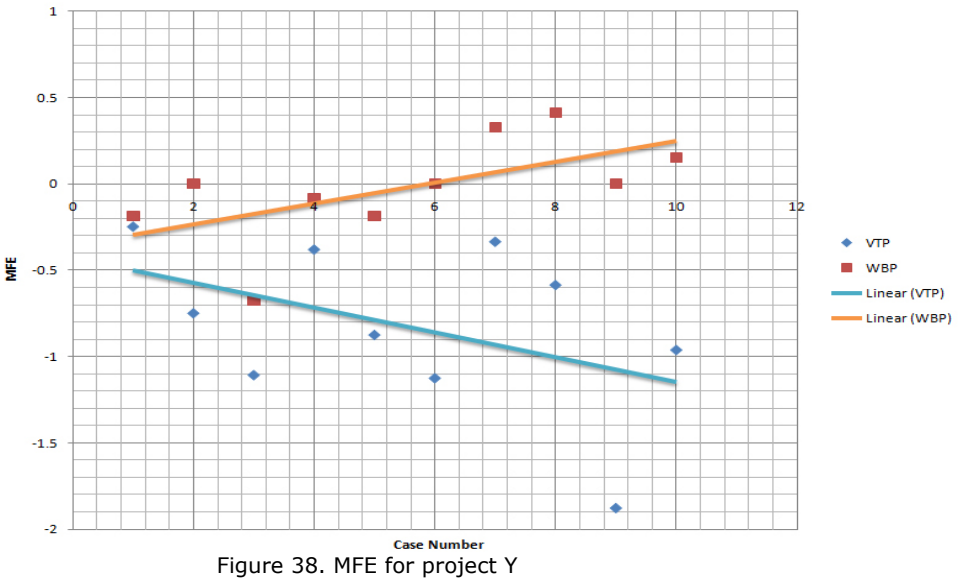

Figure 38. MFE for project Y

5.5. Conclusions

In this chapter, we present the experiments that we performed so far for the validation of the proposed Behavioral Monitoring Framework. The data used in the presented experiments are provided by two software development companies and regard the implementation of two real-world commercial projects with distinct characteristics: one is an automotive project, while the other is a pure software project; one has a heterogeneous team, while the other has an homogeneous team; one has a execution time of several years, while the other has an execution time of several months. Even though the two projects are very different overall, the data regarding their progress was provided by the companies that developed those projects in the same manner: as Microsoft Project Plan files elaborated regularly, on a monthly or weekly basis, depending on the project.

The goal of these experiments was basically to understand if the core of the Behavioral Monitoring Framework is reliable for its purpose for the particular cases represented by the projects considered in those experiments. As stated throughout this thesis, the core of the Behavioral Monitoring Framework is the model that provides the dynamic perspective over the project progress, which is the Work Behavior Prediction Model. Because the comparison between a forecast and a real value is not sufficiently meaningful due to the fact that some errors are almost always present in forecasts, we decided to consider a competing prediction methodology. Consequently, we selected for comparison the Velocity Trend Prediction which is part of the very popular Scrum management framework and which is implemented by most ALM tools (and those that don't implement it, don't provide the prediction feature at all).

The experimentation methodology assumed the utilization of the software that we developed specially for those experiments. This software prototype is able to import the Microsoft Project Plan files and to make predictions for various time intervals with both Scrum's Velocity Trend Prediction and our Work Behavior Prediction. Finally, the software prototype is able to compute several error metrics, obtaining a set of values for each of the two prediction methods, the results being provided in the form of a report.

Centralizing and analyzing the values obtained for the error metrics for the two considered prediction methods, the conclusion is that even though the projects used in these experiments were very different in type (automotive vs. pure software), execution time (several years vs. several months), and type of working teams (heterogeneous vs. homogeneous), the Work Behavior Prediction was more accurate than the very popular Velocity Trend Prediction of the Scrum management framework in most of the cases (over 80%). Moreover the trends shown by the linear regression for the considered error metrics' values suggest that the degradation of accuracy in the case of our Velocity Trend Prediction, the core of the Behavioral Monitoring Framework, is much less prominent as for the Scrum's Velocity Trend Prediction.

The results of these experiments empower us to state that the primary validation of the Behavioral Monitoring Framework, with its main feature of providing a dynamic perspective over project progress, is achieved, the results encouraging us to continue the validation process. For this, we will further try to find more companies interested in experiencing our approach to monitoring.

6. Behavioral Monitoring Applicability

We developed the Behavioral Monitoring Framework especially for tracking the progress in software projects, but it has a wide applicability as we will further show in this chapter. Into the center of every system that uses our Behavioral Monitoring Framework is the human resource.

The object of the monitoring process is represented by artifacts produced directly or indirectly by the human resource involved. Of the Behavioral Monitoring Framework component models, the Work Behavior Prediction Model is framework's core, while the Project Status Model and the Project Status Analysis Model can be seen as adapters between the operational space of the project and the monitoring framework and between the framework and the management informational needs, respectively.

In this chapter, only the simplified Work Behavior representation is used (not the generalized one), that uses remaining effort logs and no other meta-information regarding project tasks.

6.1. Software projects

Software tools are naturally used within software projects. Because the human resource involved in software projects are the most familiar with such tools, the main applicability domain of our Behavioral Monitoring Framework is represented by software projects, where it can be integrated in existing tools or implemented separately.

6.1.1. Project development tracking

The main intended application of our Behavioral Monitoring Framework is represented by the tools used for tracking the project development processes, especially those that are used in large-scale software project.

The object of the monitoring that employs the Behavioral Monitoring Framework in a project development tracking scenario is represented by the reports on the remaining effort for the assigned tasks provided by the project team members generally through automated tools like JIRA for example.

Those reports might be more or less accurate with many or few corrections, being the output of a subjective reasoning, but they define the behavior towards work of the subject of this reporting process. For example, if a worker reports a remaining effort of x, $x>2$, days for a task from day 0 to a day n and in day n+1 the worker decides to correct the reported remaining effort to x-2 (even though from day n to day n+1 the worker couldn't spent more than a day of effort on the task), this corrective behavior will be reflected in the components of the Work Behavior modeled as shown previously in this report.

The meanings of the Work Behavior components (ST, DV, and VL) are those presented widely in this report, as well as the output of the prediction process and the concepts behind the monitoring framework component models.

The main reason for using our Behavioral Monitoring Framework in project development tracking is to understand where the project is heading, being as early as possible aware of the time-constraint issues of the project.

6.1.2. Versioning systems

The object of the monitoring that employs the Behavioral Monitoring Framework in a versioning system scenario is represented by the project files updated by the human resource involved in the project through a versioning system like TortoiseSVN [96].

In such a system, Project Status Model's role is reduced in that it will only provide information regarding the changes and their frequency in the code files of a project to the Work Behavior Prediction Model. Data regarding a versioning system user's file updates can be gathered from the versioning systems because such systems store this kind of information.

The Work Behavior Prediction Model uses the same forecasting methodology as described earlier in this report, with the amendment that the completed tasks are replaced by files that were not updated lately, on a defined period of time. There are also some differences in the interpretation of the Work Behavior components as presented next:

a) The ST component of the Work Behavior represents the probability that a file is not updated in a given period of time.

b) The DV component of Work Behavior is the probability that a file is updated in a working time unit (e.g., day) that follows a time unit in which no update was made to that file or vice versa.

c)The VL component of Work Behavior is the mean number of updates per day observed for a file.

The output that is presented in a human readable form by the Project Status Analysis Model will show the predicted evolution of a file updates.

The purpose of integrating a versioning system with our Behavioral Monitoring Framework is to provide the project management the information regarding where the work tends to concentrate in what concerns the coding process of project development. In the most problematic software projects, the large-scale ones, this is information that cannot be intuitively observed and understood by project management because of the high amount of data produced by such projects. Consequently, in this kind of projects, using the Behavioral Monitoring Framework in correlation with the employed versioning system provides project management with supplementary information regarding the most dynamic parts of the coding process. Having this understanding, the project management is able to better guide the efforts of the project management team to what is really important for the project at a particular moment in time.

6.1.3. Code review

The object of the monitoring that employs the Behavioral Monitoring Framework in a code review scenario is the same as for the versioning system scenario presented above but the main concern is the number of TODO marks present in the updated code files.

In such a scenario, Project Status Model's role is to provide information regarding the changes in the number of TODO marks in a code file to the Work Behavior Prediction Model. The Work Behavior Prediction Model uses the same forecasting methodology as described earlier in this report, with the amendment that the completed tasks are replaced by files that had at some point in time TODO marks, but that currently don't have such marks. There are also some differences in the interpretation of the Work Behavior components:

a) The ST component of the Work Behavior represents the probability that a file's TODO marks number is unchanged in a given period of time.

b) The DV component of Work Behavior is the probability that a file contains more or fewer TODO marks in a working time unit (e.g., day) that follows a time unit in which no update was made to the number of TODO marks in that file or vice versa.

c)The VL component of Work Behavior is the mean number of TODO marks per day observed for a file.

The output that is presented in a human readable form by the Project Status Analysis Model will show the predicted evolution of the number of TODO marks in a code file.

The purpose of a code review system that uses the integration between a versioning system and the Behavioral Monitoring Framework is to provide the project management the information regarding where the work is needed to concentrate in what concerns the coding process of project development as a result of a code review. As explained in the previous subsection, this is especially useful in large-scale software projects. Consequently, using our Behavioral Monitoring Framework in correlation with the employed versioning system in analyzing the results of a code reviewing process and the following operational actions provides project management with supplementary information regarding the most problematic parts of the coding process. Having this understanding, the project management is able to better guide the efforts of the project management team to solving the existing coding problems within the project at a particular moment in time.

6.1.4. Task assignation

This application of the Behavioral Monitoring Framework uses the Work Behavior representation to decide the best assignee for a project task. The information used for this decision refer to work progress reports provided by project team members through software tools like JIRA or Microsoft Project (in this final case, project plan updates elaborated after discussions between project management and project team members).

In JIRA, for example, each task has an assigned priority. As shown in the previous chapter, Work Behavior as modeled in the Behavioral Monitoring Framework can be used in the evaluation of the project team members work performance by primarily considering a criterion of this evaluation. Such a criterion might be the task established priority, which is defined by the project management at task creation.

By grouping for a project team member the previously completed tasks from the projects in which this worker was or is involved and by computing a median Work Behavior for each defined priority (e.g., low priority, medium priority, high priority, critical, blocker), each worker will be assigned a score for their performance

for each type of project task. Of course, other criteria can be used for such an analysis.

The purpose of a system for task assignation that uses the Behavioral Monitoring Framework is to help project management to make the best distribution of human resources on the open project tasks considering the history.

6.2. Other domains

The Behavioral Monitoring Framework can be used in any other domain within activities organized as projects, in that they have time constraints, a structure of linked actions, and human resources involved.

Although software tools are more heavily used in software projects, where the involved human resource is more accustomed to such tools, their usage is spreading rapidly to other domains like engineering and construction, retail, or industrial manufacturing.

For example, Primavera [70], which is a portfolio/project management tool, is very popular with the engineering and construction domain. The Behavioral Monitoring Framework can work for construction projects just like it would for software projects. A difference between these types of projects may be the quantity of the information that can be collected: in software projects, where the workers spend most of the time in front of computers, more information regarding work progress is expected (and with more frequency) than in the case of construction projects. However, as shown in the previous chapter, the Behavioral Monitoring Framework is expected to perform well even when little information is available.

6.3. Conclusions

In this chapter, we discuss the applicability of our Behavioral Monitoring Framework, regarded as a set of concepts and methodologies concretized in the three component models presented in detail in Chapter 3: the Project Status Model, the Work Behavior Prediction Model, and the Project Status Analysis Model.

We present several applications of the Behavioral Monitoring Framework or based on particular concepts and methodologies of this framework. These applications are tightly related to the real-world activities that take place during project development like project tracking, code versioning, code review, and tasks assignation. For each of these applications, we describe how the concepts, methodologies and models of the Behavioral Monitoring Framework adapt to the application requirements and how the project manager can benefit from using these concepts, methodologies and models. Moreover, each of the presented applications can be implemented by software tools that can work together with existing tools for a better utilization of the information that is produced during project development. For example, versioning systems cannot tell were the activity within a project tends to concentrate, in terms of updates per file, unless these versioning systems work together with a software tool that partly implements the Behavioral Monitoring Framework, as shown in this chapter.

Besides the domain for which it was developed, which is the software projects domain, the Behavioral Monitoring Framework, as a collection of concepts, methodologies and models, can be adapted easily to work in other domains especially those in which software tools become more and more popular, like the construction field.

7. Conclusions

This thesis proposes a framework for a more effective project monitoring. The proposed framework, named Behavioral Monitoring Framework, is based on the concept of Work Behavior, that characterize the behavior towards work of the most valuable resource, in our opinion, involved in software projects, that is the human resource. We developed the Behavioral Monitoring Framework especially for the problematic software projects, but due to its versatility, it can be used with only little adaptations for the monitoring of other types of projects or as support for other informational tools.

7.1. Contributions

The main contributions of this thesis, in the order of their introduction, are:

- The modeling of Work Behavior as a concise and uniform representation of the work progress historical data available from reports:
  We propose a set of metrics, the Behavioral set of metrics, to characterize the behavior towards work of the human resource and to define the Work Behavior. The values of the component metrics of the Behavioral set of metrics computed on historical data regarding work progress (e.g., from progress reports) represent the Work Behavior observed for that historical data. There are three behavioral metrics. The first metric, Stagnation, measures the probability that a project team member spends time not working on a given task. The second, Diversification, measures the probability that a project team member starts or resumes another task before completing a given task. Finally, the third metric, Velocity, measures the completion speed for a given task. The values of these metrics computed for the same historical information given for a project team member create the Work Behavior of that team member. Large-scale projects produce a lot of historical information due to the large number of reports required during development especially from the project team members. A very important role of the Behavioral set of metrics is that it can compress the large amount of historical information is in a more human readable information, yet preserving the important meanings of the original information, that is Work Behavior. Consequently, the project management can make decisions based on the compact information provided by the Work Behavior, rather than on the huge amount of information from progress reports available from the involved human resources. Moreover, the Work Behavior computation and interpretation can be automated and implemented into decision support systems. We published this contribution in [95] and presented it in the 6th International Conference of Software and Data Technologies (ICSOFT) held in Seville, Spain, 2011.
- A proposal for project status accuracy classification:
  Project status can be computed at different levels, using various data and employing various tools. Some projects are more difficult to manage than others, requiring more information for equally effective control. Information

gathering is expensive, keeping very specialized human resources from work, so that for computing the project status, it is very important to require from the project team members only the relevant information considering the managed project. In this context, I propose a classification for project status accuracy in relation with the informational requirements for control of the managed project. For example, large-scale software projects require the highest level of accuracy for project status computation, so that special support tools based on models that use historical information are recommended to be used, along with detailed work reports required from the project team members. Of course, this cannot be the case of small projects. This classification is important in order for project managers to create the best approach to data gathering and analysis for an effective project control. We published this contribution in [94] and presented it in the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI) held in Timisoara, Romania, 2011.

- The definition of The Project Status Model capable of providing the key information for computing an accurate project status:
  The Project Status Model can be regarded as a data gathering model that uses information from multiple projects for a more realistic project status computation. In large organizations, the developed projects are interconnected through the employed human resources. Such a resource might be assigned for many tasks from different projects at a time. Consequently, a project manager should know how each human resource involved in the managed project prioritize his or her work, or if only one task is from the managed project, which position has this task in the sequence of tasks being currently in work for that resource. Such information is very important for computing a realistic project status. For example, if a project manager knows that a worker puts some important task from the managed project on the end of their to-do list, then the project manager would be able to change this if needed. In this context, besides the decisions of the project managers regarding their managed projects (e.g., task structure, prioritization of project work), the Project Status Model uses the decisions of each involved human resource regarding their own work that might not concern only one project (e.g., assigned tasks prioritization). We published this contribution in [93] and presented it in the 5th International Conference of Software and Data Technologies (ICSOFT) held in Athens, Greece, 2010.
- The definition of the Work Behavior Prediction Model that supports decision making in a changing project environment:
  The Work Behavior Prediction Model can be used dynamically, during project development. Although there are many static forecasting methods, regarded as estimation methods that can be used for making, for example, the overall effort estimation for a project at initiation, there are just few prediction methods suitable for making forecasts during project development. Such a capability is especially important in the ever changing environment of software projects. Many changes are likely to occur in such projects. The most important is the change in tasks priorities, for example when new features are required by the client or when new projects are started in parallel. The greatest importance in dealing with these changes has the human resource, which is regarded also as the most valuable resource in

software projects. Consequently, for predicting future project progress it is very important to understand how human resources involved in a project work. In small projects, eventually the project manager finds out how each member of the small project team relate to work. This is not the case in large-scale projects. The underlying methodology of the Work Behavior Prediction Model uses observed work behavior for involved human resources in order to compute a forecast regarding future work progress. We published this contribution in [95] and presented it in the 6th International Conference of Software and Data Technologies (ICSOFT) held in Seville, Spain, 2011.

- The definition of the Project Status Analysis Model which enables early responses to project execution problems:
  In large-scale software projects, due to the large amount of data, progress and state information regarding the managed project is hardly ever analyzed. To overcome this, we developed the Project Status Analysis Model that is capable for analyzing a project status and to provide recommended individual work plans for all the resources involved in one or more projects. Moreover, this model is able to identify project execution problems for the current or for a predicted project status. Each identified problem is signaled through warnings that are destined to the project manager or/and to the involved workers, each warning being accompanied by a proposal for problem solving. We published this contribution in [93] and presented it in the 5th International Conference of Software and Data Technologies (ICSOFT) held in Athens, Greece, 2010.
- The definition of the Behavioral Monitoring Framework, which represents an integrated monitoring methodology for software projects:
  The proposed integrated monitoring methodology provides support for computing the project status, to make predictions on future work progress, and to analyze the identified project status in the context of the available forecasts. The proposed methodology is suitable for automation since it is based on formally defined models that work together sinergically. We believe, the automation of the monitoring process is a must when dealing with large-scale software projects, which are very difficult to control partly due to the particularities of software projects (e.g., new technologies used, highly specialized human resources) and partly to the large amount of information (e.g., progress reports) that characterize the large-scale projects (this information being required to be analyzed for an effective management). The underlying framework of the proposed monitoring methodology, which is the Behavioral Monitoring Framework, consists of three models. The first model, the Project Status Model, uses concepts like project macro-universe and worker micro-universe in order to create the most accurate snapshot of the project state at a defined moment in time. The second model, the Work Behavior Prediction Model, is regarded as the core of the proposed monitoring framework since its forecasts can be used as data source for computing and analyzing the project status at a particular moment in the future. Finally, the third model, the Project Status Analysis Model mainly provides individual work prioritization recommendations to project team members and warnings regarding factual or expected time overruns. The analysis is done on actual or predicted project statuses. As shown throughout this thesis, the Behavioral Monitoring Framework is

capable of working also with incomplete and scarce datasets that are usually available in software projects. We published the structure of the framework in [92] and presented it in the IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics (ICCC-CONTI) held in Timisoara, Romania, 2010.

- The specification and the development of a software implementation of the Behavioral Monitoring Framework:
  The Behavioral Monitoring Framework allows for the automation of the project monitoring process. This is why this thesis also proposes a reference implementation for the software prototype of the proposed framework. We published the architecture of the reference implementation of the Behavioral Monitoring Framework in [92] and presented it in the IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics (ICCC-CONTI) held in Timisoara, Romania, 2010.
- The primarily validation of the Behavioral Monitoring Framework:
  Project development information (e.g., from progress reports) for real-world software projects is very difficult to get due to its confidential nature. This is the main reason for which there are only few accepted progress forecasting methodologies that can be used during project development. The most important of them is Velocity Trend Prediction, which is part of the very popular Scrum project management framework. Also, this prediction method is available in most ALM tools. In the context of large-scale software projects, a project monitoring framework must provide prediction capabilities to cope with the great number of changes occurring in such projects. Also, the forecasts must be as reliable as possible, being important for the decision making process. This is why the parallel evaluation of the Behavioral Monitoring Framework and Scrum focuses on the employed prediction method. The first results obtained by applying the two prediction methods (Work Behavior Prediction and Velocity Trend Prediction) on two real-world commercial software projects development data sets show a clear superiority of the prediction method of the Behavioral Monitoring Framework, obtaining lower forecasting errors, although requiring the same amount of data as the prediction method of Scrum. The obtained results primarily validate our Behavioral Monitoring Framework. We published this contribution in [95] and presented it in the 6th International Conference of Software and Data Technologies (ICSOFT) held in Seville, Spain, 2011.
- The specification of the Behavioral Monitoring Framework's application domain:
  The utilization of the Behavioral Monitoring Framework's concepts, methodologies and models for various applications can be done by slightly adjusting it for each application requirement. The clear specification of these adjustments for a defined number of important applications for software development domain is provided in this thesis. Moreover, the Behavioral Monitoring Framework can be used not only in the software development domain but also in others within activities organized as projects, like construction field for example, in which decision making support tools play an important role.

7.2. Future work

The intended future work is organized on two main directions:

a) Validation and improvement:
The proposed monitoring framework has passed the preliminary validation. This opens the road for new experiments on real-world software project. Because of the confidential nature of such information, data from real-world commercial software projects are difficult to get. To overcome this, the plan is to convince several companies to use the prototype of the proposed monitoring framework during project development. The results of the validation process are expected to provide clues regarding the possible improvements that can be made to the proposed monitoring framework in order for it to assure a better predictability for the managed projects.

b) Dissemination:
The dissemination of this research's outcome will continue as more results from the validation of the proposed monitoring framework are available. The main objective of this action is to gain visibility and confidence for the proposed monitoring framework in order for it to be used on large-scale improving the efficiency of project monitoring and control for a better project management.

7.3. Personal publications

The personal publications on which this thesis is based are:

- **Stanciu, C., "Work behavior prediction during software projects development"**. The 6th International Conference on Software and Data Technologies, ICSOFT 2011, Seville, Spain, 2011, pp. 47-52 (Thomson Reuters, Inspec, DBLP, EI)
- **Stanciu, C., "Project status accuracy in large-scale software projects".** The 6th International Symposium on Applied Computational Intelligence and Informatics, ISBN 978-1-4244-9107-0, pp. 217-222, Timisoara, Romania, May 2011 (ISI Proceedings)
- **Stanciu, C., Tudor, D. and Creţu V.I., "Towards modeling large scale project execution monitoring: Project Status Model".** Proceedings: 5th International Conference on Software and Data Technologies, ICSOFT 2010, Athens, Greece, ISBN 978-989-8425-22-5, Volume 1, pp. 36-41. Portugal: SciTePress, 2010 (Thomson Reuters, Inspec, DBLP, EI)
- **Stanciu, C., Creţu, V.I. and Cireş-Marinescu, R., "Monitoring framework for large-scale software projects"**. Proceeding of the IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics 2010, pp. 333-338, Timisoara, Romania, 2010 (IEEE Xplore)
- **Stanciu, C., Tudor, D. and Creţu V.I., "Towards an adaptable large scale project execution monitoring"**. 5th International Symposium on Applied Computational Intelligence and Informatics, pp. 503-508, Romania, 2009 (ISI Proceedings)

References

[1] A. Aamodt, E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", AI Communications, IOS Press, Vol. 7: 1, pp. 39-59, 1994

[2] Seiya Abe , Osamu Mizuno , Tohru Kikuno , Nahomi Kikuchi , Masayuki Hirayama, "Estimation of project success using Bayesian classifier", Proceedings of the 28th international conference on Software engineering, May 20-28, 2006, Shanghai, China

[3] Ajith Abraham, "Intelligent systems: architectures and perspectives", Recent advances in intelligent paradigms and applications, Physica-Verlag GmbH, Heidelberg, Germany, 2003

[4] J. Scott Armstrong, "Extrapolation of Time Series and Cross-Sectional Data", in Principles of forecasting: a handbook for researchers and practitioners, University of Pennsylvania, Wharton School, pp. 217-243, May 2001

[5] Jai Asundi, Rick Kazman, Mark Klein, „An Architectural Approach to Software Cost Modeling", Second International Workshop on Economics-driven Software Engineering Research, Limerick, Ireland, 2000

[6] Jai Asundi, "The Need for Effort Estimation Models for Open Source Software Projects", International Conference on Software Engineering Proceedings of the fifth workshop on Open source software engineering, St. Louis, Missouri, SESSION: Workshop on Open Source Software Engineering (WOSSE), Pages: 1 - 3, 2005

[7] Atlassian, "JIRA – issue and project tracking", http://www. atlassian.com/ software/jira/, retrieved 2 January 2011.

[8] B. de Baar, "Using stakeholder analysis in software project management". 2006. Retrieved December 14, 2010 from http://www.softwareprojects.org/ stakeholders.pdf.

[9] Barros, M., Werner, C. M. L., Travassos, G. H., "Applying System Dynamics to Scenario Based Software Project Management", Proceedings of the 18th International System Dynamics Conference, Berghen, Norway, 2000

[10] I. F. Barcelos Tronto, J. D. Simoes da Silva, N. SantAnna, "The Artificial Neural Networks Model for Software Effort Estimation", INPE , 2006, Vol. 1, pp. 2-21

[11] I.F. De Barcelos Tronto, J.D.S. da Silva, N. Sant'Anna, "Comparison of artificial neural network and regression models in software effort estimation", in: Proceedings of International Joint Conference on Neural Networks, Orlando, FL, USA, August 12-17, 2007

[12] Bekjti, S., Matta, N., "A Formal Approach to Model and Reuse the Project Memory", Proceedings of I-KNOW '03, pp. 507--514, Graz, Austria, 2003

[13] Oddur Benediktsson, Darren Dalcher, Karl Reed, Mark Woodman, "COCOMO-Based Effort Estimation for Iterative and Incremental Software Development", Software Quality Journal 11(4): 265-281, 2003

[14] C. Bodea, "Agile Software Project Management Methodologies", Economy Informatics, Vol. V, No. 1-4, 2005, pp. 27-31

[15] B. Boehm, R. Valerdi, J. Lane, J., and W. Brown, "COCOMO suite methodology and evolution". In: Crosstalk, Vol.18 i4, 2005, pp. 20-25.

[16] J. Bollinger, "Bollinger on Bollinger Bands", McGraw-Hill Professional, pp. 9-21, 2002

[17] Don Box, Anders Hejlsberg, "LINQ: .NET Language-Integrated Query", http://msdn.microsoft.com/en-us/library/bb308959.aspx, February 2007

[18] James Bullock, "The top 10 ways software projects are different", http://www.pmforum.org/library/papers/2003/Top10WaysSoftwareProjectsRDifferent.pdf, 2003, retrieved 14 December 2010.

[19] Caine, A. and A. B. Pidducks, "f2 COCOMO: Estimating Software Project Effort and Cost", Proceedings of the the 6th International Workshop on Economic-Driven Software Engineering Research (EDSER-6), Edinburgh, Scotland: IEEE, 2004

[20] Center for Software Engineering, "COCOMO II Model Definition Manual", Computer Science Department, University of Southern California, Los Angeles, Ca. 90089, http://sunset.usc.edu/Cocomo.html, 1997

[21] Chan, M.-C., Wong, C.-C., and Lam, C.-C., "Financial time series forecasting by neural network using conjugate gradient learning algorithm and multiple linear regression weight initialization", Computing in Economics and Finance, 61 (2000)

[22] Sunita Chulani, Brad Clark, Barry Boehm, "Calibration Approach and Results of the COCOMO II Post Architecture Model", International Society of Parametric Analysts, June 1998

[23] J. Clarke, J.J. Dolado, M. Harman, R. Hierons, B. Jones, M. Lumkin, B. Mitchell, S. Mancoridis, K. Rees, M. Roper, and M. Shepperd, "Reformulating Software Engineering as a Search Problem", IEE Proc. Software, vol. 150, pp. 161-175, 2003

[24] Colin F. Camerer, George Loewenstein, & Matthew Rabin, "Advances in behavioral economics". Princeton University Press, ISBN 0-691-11681-4, 2004.

[25] CollabNet. http://www.open.collab.net. Accessed 12 December 2010.

[26] Vladimir-Ioan Creţu, "Software Project Management", 2009

[27] Cronos, An Open Source Time Series Analysis Package, http://www.stat .cmu.edu/~abrock/oldcronos, 2006

[28] Daneva, M., "Approaching the ERP Project Cost Estimation Problem: an Experiment", Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, pp.500, September 20-21, 2007

[29] Arindam Das, "Using EVM in Software Projects for Monitoring and Control", Infosys Technologies Limited, Chennai, India, 2004

[30] Prag Dave, "Some Agile History", http://pragdave.pragprog.com, February 25, 2007

[31] P. Deemer and G. Benefield, "An introduction to agile project management with Scrum". 2007. Retrieved November 20, 2010 from http://www.rallydev.com/ documents/scrumprimer.pdf.

[32] DeMarco, T., "Software engineering: an idea whose time has come and gone?". In IEEE Software. Viewpoints, pp 94-95, 2009.

[33] Dvorak, D., Kuipers, B., "Model-Based Monitoring of Dynamic Systems", Proceedings of the 11th international joint conference on Artificial intelligence, Vol.2, pp. 1238--1243, Detroit, Michigan, USA, 1989

[34] Robert Engle, "GARCH 101: The Use of ARCH/GARCH Models in Applied Econometrics", Journal of Economic Perspectives, American Economic Association, vol. 15(4), pages 157-168, Fall 2001

[35] T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit, "A simulation study of the model evaluation criterion MMRE". Discussion paper. Norwegian School of Management BI. ISSN: 0807-3406, 2002.

[36] Mike Garnsey, Lacey Edwards, Kelly Ward, Dean Marvin, "COCOMO and SCORM: Cost Estimation Model for Web-Based Training", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2006

[37] Tudor Gîrba, "Modeling History to Understand Software Evolution", PhD. thesis, University of Bern, 2005

[38] Kathleen B. Hass, "Introducing the Project Complexity Model. A New Approach to Diagnosing and Managing Projects", PM World Today, Vol. IX, Issue VII, July 2007

[39] W. S. Humphrey, "Managing the software process", SEI series in software engineering, Addison Wesley Longman, pp. 301-395, August 1990

[40] W. S. Humphrey, "A discipline for software engineering", SEI series in software engineering, Addison-Wesley Publishing Company, pp. 217-219, November 1997

[41] Hunt, B. , "Parametric project monitoring and control: performance-based progress assessment and prediction". In Aerospace Conference, IEEE (pp. 1-12), 2007.

[42] IBM, "IBM Rational Team Concert", http://www-01.ibm.com/ software/ rational/products/rtc/, retrieved 2 January 2011.

[43] A. Idri, B. Griech, A. El Iraki, "Towards an Adaptation of the COCOMO Cost Model to the Software Measurement Theory", ESEC / SIGSOFT FSE 1997: 525-526

[44] Ion Ivan, Adrian Visoiu, Dragos Palaghita, "IT Project Metrics", Projects and Programs Evaluation. Risks, resources, activities, portfolio and project management, JAQM Volume 2, Issue 3, pp. 302, September 30, 2007

[45] Jingzhou Li, Guenther Ruhe, "Decision Support Analysis for Software Effort Estimation by Analogy", promise, pp.6, Third International Workshop on Predictor Models in Software Engineering (PROMISE'07: ICSE Workshops 2007), 2007

[46] Magne Jørgensen, Dag Sjøberg, and Geir Kirkebøen: "Human judgement in effort estimation of software projects", In Janice Singer et al., editors, Beg, Borrow, or Steal Multi-Disciplinary Workshop at the International Conference on Software Engineering (ICSE'2000), Limerick, Ireland, 5 June 2000

[47] M. Jørgensen and D. Sjøberg, "The importance of not learning from experience", presented at European Software Process Improvement 2000 (EuroSPI'2000), Copenhagen, 2000

[48] Magne Jørgensen and Kjetil Moløkken, "How Large Are Software Cost Overruns? A Review of the 1994 CHAOS Report" Software Practitioner, Vol. 16, no. 4&5, pp. 13-14, April 2006

[49] C. Kirsopp, M. Shepperd, and J. Hart, "Search Heuristics, Case-Based Reasoning and Software Project Effort Prediction", Proc. Genetic and Evolutionary Computation Conf., pp. 1367-1374, 2002

[50] B.A. Kitchenham, L.M. Pickard, S.G. Macdonell, and M.J. Shepperd, "What Accuracy Statistics Really Measure", IEE Proc. -Software, vol. 148, no. 3, pp. 81-85, 2001

[51] G. Liebchen G., M. J. Shepperd, „Data Sets and Data Quality in Software Engineering", PROMISE 2008, Leipzig, ACM Press

[52] Todd Little, "Context-Adaptive Agility: Managing Complexity and Uncertainty", IEEE Software, vol. 22, no. 3, pp. 28-35, May/June 2005

[53] K. Lum, J. Hihn, T. Menzies, "Studies in Software Cost Model Behavior: Do We Really Understand Cost Model Performance?", Proceedings of the ISPA International Conference 2006, Seattle, WA

[54] Lyneis, J.M., Ford, D.N., ”System Dynamics Applied to Project Management: A Survey, Assessment, and Directions for Future Research”, System Dynamics Review, Vol. 23, No. 2/3, pp. 157--189, 2007

[55] C. Mair, M. Shepperd, ”The consistency of empirical comparisons of regression and analogy-based software project cost prediction”. In: International Symposium on Empirical Software Engineering, pp.10, 2005.

[56] V. Marza, A. Seyyedi, L. F. Capretz, “Estimating development time of software projects using a neuro fuzzy approach”. In: Proceedings of World Academy of Science, Engineering and Technology, Vol. 36, 2008.

[57] A. I. McLeod, “A note on ARMA model parameter redundancy”, The Journal of Time Series Analysis, Vol. 14, No. 2, pp. 207-208, April 1991

[58] C. Meek, D.M. Chickering, and D. Heckerman, “Autoregressive Tree Models for Time-Series Analysis”, Proc. Second SIAM Int'l Conf. Data Mining (SDM '02), 2002

[59] Louis B. Mendelsohn, “Trend forecasting with technical analysis: unleashing the hidden power of intermarket analysis to beat the market”, Marketplace Books, pp. 35, 2000

[60] Microsoft Project, “Definition of Microsoft Project constraints”, 2007, http://support.microsoft.com/kb/74978/en-us

[61] Microsoft Corporation, “Microsoft Time Series Algorithm Technical Reference”, SQL Server 2008 Books Online, http://msdn.microsoft.com, March 2009

[62] Microsoft, “Microsoft Project”, http://www.microsoft.com/ project/en/us/product -information.aspx, retrieved 23 January 2011.

[63] Harish Mittal, Pradeep Bhatia, “Optimization Criteria for Effort Estimation using Fuzzy Technique”, Clei Electronic Journal, Volume 10, Number 1, Paper 2, June 2007

[64] Ahmed Shawky Moussa, “The Implementation of Intelligent QoS Networking by the Development and Utilization of Novel Cross-Disciplinary Soft Computing Theories and Techniques”, A Dissertation submitted to the Department of Computer Science In partial fulfillment of the requirements for The degree of Doctor of Philosophy, The Florida State University College of Arts and Sciences, fall 2003

[65] Ayyıldız Murat, Kalıpsız Oya, Yavuz Sırma, “A Metric-Set and Model Suggestion for Better Software Project Cost Estimation”, Proceedings of World Academy of Science, Engineering and Technology, Volume 16, November 2006

[66] Nikolaidis, Savvas; Lazos, C., “Fuzzy case identification in case based reasoning systems”, Acta Univ. Apulensis, Math. Inform. 7, pp. 327-336, 2004

[67] M. W. Nisar, Y.-J. Wang, M. Elahi, I. A. Khan, "Software Development Effort Estimation Using Fuzzy Logic", Information Technology Journal, Asian Network for Scientific Information, 2009

[68] S. Nolfi and D. Parisi, "Handbook of brain theory and neural networks", chapter "Evolution of artificial neural networks", pp 418–421. MIT Press, 2002

[69] Oorschot, K.E. van, Sengupta, K., Wassenhove, L.N. van, "Dynamics of Agile Software Development". Proceedings of the 27th International Conference of the System Dynamics Society, Albequerque, Albequerque, USA, 2009

[70] Oracle, "Primavera", http://www.oracle.com, retrieved on 23 March 2011.

[71] Parvinder S. Sandhu, Porush Bassi, and Amanpreet Singh Brar, "Software Effort Estimation Using Soft Computing Techniques", Proceedings of World Academy of Science, Engineering and Technology, Volume 36, December 2008

[72] Project Management Institute, "A guide to the project management body of knowledge (PMBOK Guide) - Fourth Edition". Project Management Institute, ISBN13:9781933890517, 2008

[73] Radice, R. A., Roth, N. K., O'Hara, A. C. Jr., Ciarfella, W. A., "A programming process architecture". In IBM Systems Journal 24 (No. 2, pp. 79-90), 1985.

[74] J. F. Ramil, "'Why COCOMO Works' Revisited or Feedback Control as a Cost Factor", FEAST 2000 Workshop, Imp. Col., London, 10-12 Jul. 2000

[75] C. Ravindranath Pandian, "Software Metrics: A Guide to Planning, Analysis, and Application", CRC Press, 2004

[76] Samir Ray, Dipesh Patel, "Managing Chaos in an Agile World", PM World Today, Vol. X, Issue XI, November 2008

[77] D. Reifer, B. Boehm, and S. Chulani, "The Rosetta Stone: Making COCOMO 81 Estimates Work with COCOMO II", Crosstalk, February 1999, pp. 11-15

[78] D. J. Reifer, "How good are agile methods?" In: IEEE Software, Vol.19, No.4, 2002, pp.16-18.

[79] Rodrigues, A. G., Williams, T. M., "System Dynamics in Software Project Management: Towards the Development of a Formal Integrated Framework", European Journal of Information Systems, 6, pp. 51--66, 1997

[80] T. Rollo, "Functional size measurement and COCOMO—a synergistic approach", Proceedings of Software Measurement European Forum (SMEF), pp. 259-267, Rome, Italy 2006

[81] W.W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques", in Proceedings of WesCon (August, 1970)

[82] Moshood Omolade Saliu, "Adaptive Fuzzy Logic Based Framework for Software Development Effort Prediction", A thesis presented to the Deanship Of Graduate Studies in partial fulfillment of the requirements for The Degree Master of Science in Computer Science, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, April 2003

[83] Scrum Alliance, "Glossary of terms". http://www.scrumalliance.org/articles/39-glossary-of-scrum-terms. Accessed 30 November, 2010.

[84] System Dynamics Society, http://www.systemdynamics.org, 2010

[85] Serkan, N. "An information system for streamlining software development process". In Turk J. Elec. Engin. (Vol.12, No.2), 2004.

[86] Serena, "OpenProj", http://openproj.org/openproj, retrieved on 4 February 2011.

[87] M. Shepperd, C. Schofield, "Estimating Software Project Effort Using Analogies", IEEE Transactions on Software Engineering, Vol. 23, No. 12, 1997, pp 736-743

[88] A. Sheta, "Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects", Journal of Computer Science v.2 n.2, p.118-123, 2006

[89] Martin Shepperd, "Software project economics: a roadmap", fose, pp.304-315, Future of Software Engineering (FOSE '07), 2007

[90] Shenoy, P., "Operating Systems. Scheduling - Lecture 7: September 23", a course for undergraduate CS students, University of Massachusetts, Department of Computer Science, 2008

[91] Ciprian Stanciu, Dacian Tudor and Vladimir-Ioan Creţu, "Towards an adaptable large scale project execution monitoring", 5th International Symposium on Applied Computational Intelligence and Informatics, pp. 503 - 508, Romania, May, 2009

[92] Ciprian Stanciu, Vladimir-Ioan Creţu and Ruxandra Cireş-Marinescu, "Monitoring Framework for Large-Scale Software Projects", Proceeding of the IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics 2010, pp. 333-338, Timisoara, Romania, May, 2010

[93] Ciprian Stanciu, Dacian Tudor and Vladimir-Ioan Creţu, "Towards Modeling Large-Scale Project Execution Monitoring: Project Status Model"; accepted at the 5th International Conference on Software and Data Technologies, Athena, Greece, July 2010 ISBN 978-989-8425-22-5, Volume 1, pp. 36-41. Portugal: SciTePress, 2010.

[94] Ciprian Stanciu, "Project status accuracy in large-scale software projects". The 6th International Symposium on Applied Computational Intelligence and

Informatics, ISBN 978-1-4244-9107-0, pp. 217-222, Timisoara, Romania, May 2011.

[95] Ciprian Stanciu, "Work behavior prediction during software projects development". The 6th International Conference on Software and Data Technologies, ICSOFT 2011, Seville, Spain, 2011, Volume 1, pp. 47-52.

[96] TortoiseSVN, http://tortoisesvn.net/, retrieved on 12 January 2011.

[97] The Standish Group, "Chaos Report", Technical report, Standish Group International, 1994

[98] The Standish Group, „New Standish Group report shows more project failing and less successful projects", April 23, 2009, http://www.standishgroup.com/newsroom/ chaos_2009.php

[99] Valerdi, R., "Cognitive Limits of Software Cost Estimation", 1st Conference on Empirical Software Engineering & Measurement, September 2007, Madrid, Spain

[100] DinDin Wahyudin, Matthias Heindl, Ronald Berger, Stefan Biffl, Alexander Schatten, "In-Time Project Status Notification for All Team Members in Global Software Development as Part of Their work environments", International Conference on Global Software Engineering (ICGSE), Workshop on Measurement-based Cockpits for Distributed Software and Systems Engineering Projects (SOFTPIT), Munich, August 2007

[101] Paul E. Wetzel (OPS Consulting), "Code Metrics, an Extensible Tool for Code Counting", Presented at the 21st International Forum on COCOMO and Software Cost Modeling, 2006

[102] T. L. Woodings, G. A. Bundell, "A Framework for Software Project Metrics", Proceedings of the 12th ESCOM Conference on Software Control and Metrics, London, 2001

[103] West Yarmouth, „Latest Standish Group CHAOS Report Shows Project Success Rates Have Improved by 50%", March 25, 2003, http://findarticles.com/p/ articles/mi_m0EIN/is_2003_March_25/ai_99169967/?tag=untagged

[104] N. Zivelin, "Forecast metrics and evaluation". Oracle. Retrieved on December 27, 2010 from http://demantrasig.oaug.org.