Usage Scenarios for Measurement-based Project Control

Jens Heidrich, Jürgen Münch, Axel Wickenkamp

Abstract

Many software development organizations still lack support for obtaining intellectual control over their development processes. This includes determining the performance of their processes and the quality of their products as well as the ability to react in a way that intended project and product goals can be fulfilled. Systematic support for detecting and reacting to critical project states in order to achieve planned goals is usually missing. One means to institutionalize project control, systematic quality assurance, and management support on the basis of measurement and explicit models is the establishment of so-called Software Project Control Centres. From a research perspective, the set-up of such control centres is a challenging task: The metrics to be used typically need to be tailored to business goals and to the development context; appropriate interpretation and control techniques need to be combined to aggregate the information; different stakeholders require quite different visualizations for decision support. From a practical viewpoint, the deployment of a control centre is challenging because it needs to be tailored and integrated into a specific development environment that is often characterized by multi-disciplinary, distributed development. This requires careful analysis of usage scenarios for such control centres and smooth integration into the management and quality assurance processes of an organization.

This article will give an introduction to basic control centre concepts and discuss their usage, exemplified by several typical scenarios from industry. Each scenario is described in the context of a related business goal, which should be achieved using control centres. The business goal is systematically mapped to selected measurement goals, and metrics are derived using the GQM paradigm. For interpreting and visualizing the measurement data, a Visualization Catena is defined, which formally describes what kind of data analysis has to be applied and how the data should be visualized. This way, statements about achieving the measurement and business goals can be made.

1. Introduction

The complexity of software development projects requires efficient ways to organize project control. On the one hand, the quality of a project's software products (e.g., in terms of reliability and maintainability) has to be controlled in order to fulfil customer requirements and time-to-market constraints. On the other hand, controlling the project state is essential in order to detect deviations from the time and cost plan and to react appropriately. Software is often developed in different distributed locations, which, in turn, makes it even harder to develop the system according to plan (i.e., matching time and budget constraints), because data has to be integrated to get a complete picture of the current project state. So, support for project management and control is one of the key factors for project success [5].

Project control approaches from other domains, like business process control approaches, are not directly applicable because of certain software development-specific characteristics, like highly creative and non-stable processes. [13] give an overview of control approaches in the software development domain and proclaim a classification scheme with respect to goal-oriented presentation and visualization of measurement data on the one side, and flexible data processing on the other side. The basis of all control approaches is the same: the collection and effective usage of measurement data. But, what does effective interpretation and visualization of measurement data look like and what are current obstacles?

Many organizations have problems with establishing efficient project control processes because there is no documented relationship between business goals and project goals, which complicates the tracking of goal fulfilment. There is no guidance on how to derive key performance indicators for project success and concrete measures from project goals, which makes it very hard to determine which measurement data to collect. It is often not clear how to interpret the collected data (selected control techniques, data visualizations, and models), which makes it impossible to determine the right response actions in case of project plan deviations. Finally, collected data are provided and visualized in an unstructured way, independent from their intended purpose, so that information overload is created, which decreases the efficiency of project control activities.

One way to support the effective control of software development projects is the use of basic engineering principles [5], [19], with particular attention being paid to the monitoring and analysis of actual product and process states, the comparison of actual states with planned states, and the initiation of any necessary corrective actions during project execution. Effectively applying these principles requires experience-based project planning [16] and the use of explicitly defined models in order to plan a project. Furthermore, it requires the collection, interpretation, and presentation of measurement data according to a previously defined plan in order to provide stakeholders with up-to-date information about the project state. Moreover, it requires experience packaging after project completion so that future projects can profit from the experiences gained.

One well-proven way to document the relationship between measurement goals and data to be collected is the Goal Question Metric (GQM) paradigm [1], [15]. However, interpretation and visualization of the collected measurement data in a goal-oriented way (along the GQM plan) is also needed in order to relate them to statements about a previously defined measurement goal. One means to institutionalize measurement on the basis of explicit models (i.e., process models, product models, resource models, and quality models) is the development and establishment of so-called *Software Project Control Centres (SPCC)* for systematic quality assurance and management support [13]. In the aeronautical domain, air traffic control systems are used to ensure the safe operation of commercial and private aircraft. Air traffic controllers use these systems to coordinate the safe and efficient movement of air traffic (e.g., to make certain that planes stay a safe distance apart or to minimize delays). The system collects and visualizes all critical data (e.g., the distance between two planes, the planned arrival and departure times) in order to support decisions by air traffic controllers. Software project control requires an analogous approach that is tailored to the specifics of the process being used (e.g., its non-deterministic, concurrent, and distributed nature). An SPCC is basically a specific implementation of the classic concept of a Project Management Office (PMO)[1] [14].

An SPCC is a control system for software development that collects all data relevant to project control, interprets and analyzes the data according to the project's control needs, visualizes the data for different project roles, and suggests corrective actions in the case of plan deviations. An SPCC can also support the packaging of data (e.g., as predictive models) for future use and contribute to an improvement cycle spanning a series of projects. However, the support of current SPCC implementations for all these activities varies. Setting up project control processes and providing information in a goal-oriented way, in particular, currently do not receive much support [13].

Controlling a project means ensuring the satisfaction of project objectives by monitoring and measuring progress regularly in order to identify variances from the plan during project

[1] According to [14], Section 1.6.4, a PMO is an "organizational unit to centralize and coordinate the management of projects".

execution, so that corrective action can be taken when necessary [14]. Planning is the basis for project control and defines expectations that can be checked during project execution. Project control is driven by different role-oriented needs. We define control needs as a set of role-dependent requirements for obtaining project control. A project manager needs different kinds of data, data of different granularity, or different data visualizations than the kind of data needed by a quality assurance manager or a developer. For example, a manager is interested in an overview of the project effort in order to compare it to previously defined baselines, while a developer is interested in the effort spent on a certain activity.

The article will present an approach for establishing goal-oriented project control (Section 2) and illustrate its usage via several practical scenarios (Section 3). Each scenario is described in the context of a related business goal, which should be achieved using control centres. The business goal is systematically mapped to a few measurement goals, and metrics are derived using the GQM paradigm. For interpreting and visualizing the measurement data, a Visualization Catena is defined that formally describes what kind of data analysis has to be applied and how the data should be visualized. This way, statements about achieving the measurement and business goals can be made. Furthermore, some related approaches will be described (Section 4) and the future usage and implementation of the scenarios will be discussed (Section 5).

2. Goal-oriented Project Control

In this section, we want to illustrate the goal-oriented SPCC approach (called G-SPCC for short), which is a state-of-the-art framework for project control developed at the University of Kaiserslautern and the Fraunhofer Institute for Experimental Software Engineering (IESE) [6], [7], [8], and which integrates project control into the Quality Improvement Paradigm (QIP) [1] and the GQM paradigm [15]. The aim of this approach is to interpret and present the collected data in a goal-oriented way in order to optimize a measurement program and effectively detect plan deviations.

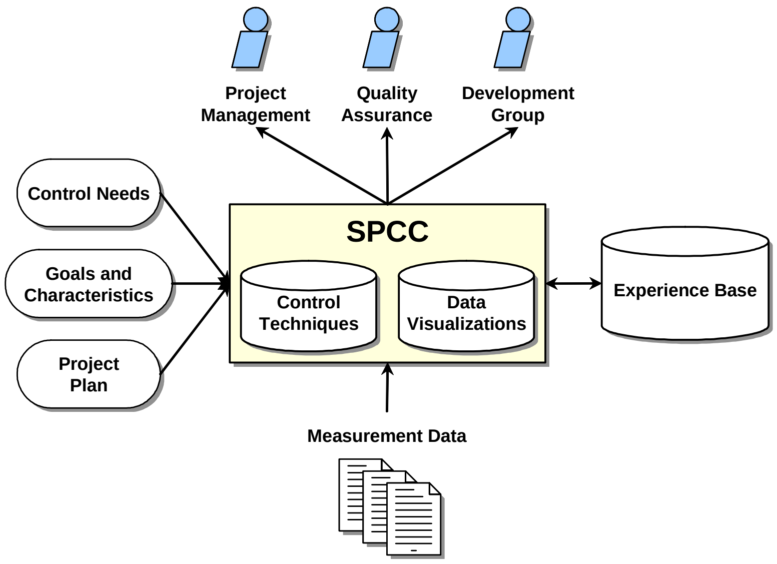

Figure 1: G-SPCC Inputs and Outputs.

Figure 1 gives an overview of the high-level G-SPCC architecture illustrating basic input and output information. Measurement data is collected during project performance and interpreted and visualized by the SPCC with respect to the goals and characteristics of the project as well as project plan information and control needs. The core of an SPCC is a set of integrated project control techniques (for interpreting the data in the right way) and data visualization mechanisms (for presenting the interpreted data in accordance with the role interested in the data). The control techniques usually cover different purposes, such as monitoring project attributes, comparing attributes to baselines, or predicting the course of an attribute. The data visualization mechanisms provide role-specific insights into the process (e.g., insights suitable for project managers, quality assurance personnel, or the development group). The SPCC interpretation and visualization process is supported by an experience base in order to reflect data from previous projects and to store experience gathered after project completion.

A so-called *Visualization Catena (VC)* [8] formally describes the relation between data collected, control techniques applied, and visualization mechanisms used. A VC consists of different components. (a) Data entries represent the content of measurement data itself, like concrete effort baselines or defect data. We distinguish between external and internal entries. The former have to be read from an external source, like an XML file or a data base. The latter are internal processing results that can act as an input for further processing or presentation. (b) Functions represent a certain control technique or method used to process data entries. A function covers at least one purpose, like monitoring a project's attribute or analyzing data, and has to be adapted to the project context. Function instances apply a certain function to a set of data entries. They process external and internal data entries and produce a new set of (internal) data that could be further used by other elements. (c) Views provide a certain way to present the data, like charts or tables. Usually, a view is associated to one or more project roles, and has to be adapted to the project context. View instances apply a certain view to a set of (internal and/or external) data entries. A view instance may refer to other views in order to build up a hierarchy of views.

So, what is different compared to other project controlling approaches? First, the G-SPCC approach is a dynamic approach for project control. That means that controlling measures and indicators are not predetermined and fixed for all projects. They are dynamically derived from measurement goals at the beginning of a development project. Existing control techniques and data visualizations (stored in an Experience Base) can be used to compose a Visualization Catena, or alternatively, new project-specific techniques and visualizations may be added to the repository and used.

Second, data is provided in a purpose- and role-oriented way. That means that a certain project role sees only the visualization of measurement data that is needed to fulfil the current measurement purpose (e.g., a QA manager does not need to see the project effort for all activities if she/he is only interested in product quality for the purpose of improving over time).

Third, project control activities are defined explicitly and are systematically collected for project roles and organizations. That means that the relationships between measurement data to collect, data processing, and data visualization are explicitly specified and provided in the form of a context-specific construction kit, so that elements with a matching interface may be combined to support a holistic approach for project control.

3. Usage Scenarios

In the following section, we describe practical usage scenarios for project control and illustrate how they can be tailored to specific goals and contexts. These scenarios are based on experience from planning initial case studies that will be performed by the authors in industrial environments. The goal of the description is to illustrate necessary steps to introduce efficient mechanisms for project control into practice by giving typical examples. In this article, the intention is not to give a comprehensive and detailed procedure for what should be done in general to establish project control. However, the illustrated procedure can be seen as a systematic means to derive a measurement plan from project goals and to specify controlling activities in order to monitor the defined goals.

First, we describe control needs and relate them to business goals. Second, all control needs are mapped to a set of concrete measurement goals (according to the GQM paradigm). Via this procedure, a kind of goal hierarchy is provided, which allows us to connect all measurement activities to explicitly stated business goals, so that the motivation for measurement is presented in a transparent way. This procedure was motivated by [2], where a business goal is mapped to explicitly stated software goals (i.e., the goals of the software unit of an organization) and later on to measurement goals. Third, we derive a couple of questions and, finally, metrics following the GQM paradigm. Fourth, a visualization catena, describing the controlling methods and data visualizations applied, is specified for each control need.

Table 1 gives three examples for control needs motivated from practice: controlling the costs of the project; the dates (end dates of project activities and milestones); and the efficiency of defect detection processes (like reviews and tests performed). Each control need has an assigned, explicitly stated, business goal. For instance, the business goal behind controlling the costs was (for the sample presented) the maximization of profit by improving cost plan adherence. We distinguish four different areas of control needs: time and schedule adherence, effort and cost adherence, project progress monitoring, and, finally, quality control [3].

Time and schedule adherence controls the compliance with explicitly defined dates of the project. Deadlines for each activity are checked with respect to exceeding or falling short of the deadline. If a deadline is exceeded significantly, the project will be delayed, which causes additional costs and affects customer satisfaction. To fall short of a deadline is usually not that problematic. However, if there is a significant gap, the overall project planning could have been more effective with a closer deadline. Schedule control identifies such plan deviations and analyzes their reasons and effects. Note that not every plan deviation of a single activity must automatically cause plan deviations of the overall project. The control flow of activities as well as tolerance ranges have to be considered.

Effort and cost adherence controls personnel effort and development costs during the enactment of the project. Usually, both types of control are strongly interrelated. The basis for controlling effort is typically the number of person hours of all project employees. Cost control usually weights the number of person hours with a certain factor and adds general costs such as equipment needed, accommodation, and various materials. Often, both terms are used inter-changeably.

Table 1: Example Control Needs.

ID	**Description**	**Area**	**Business Goal Description**
COSTS	Control of all project costs	Effort and Cost Adherence	Maximize profit by improving cost plan adherence
DATES	Control of project dates	Time and Schedule Adherence	Customer satisfaction through punctual delivery
EFFICIENCY	Improving efficiency of defect detection activities	Quality Control	Customer satisfaction through improving the reliability of the final software product

Project progress monitoring checks whether a concrete output was produced for all costs expended (e.g., an artefact like design document, code, testing documentation). The difficulty lies in determining the progress itself, because there are usually no direct project parameters that make statements about the general progress. The progress usually has to be concluded from indirect parameters such artefacts completed for certain activities.

Quality control is defined as the continuous monitoring and verification of the state of a certain unit (e.g., code) and its analysis in order to assure predetermined quality requirements (such as reliability or maintainability) [4]. The difficulty with quality control consists in finding the right metrics that allow making statements about certain quality aspects.

Table 2 derives selected measurement goals for defined control needs. Each goal is characterized according to the GQM goal template; this means that the measurement object, the purpose, the quality focus, and the viewpoint are described. We do not specify an explicit context for each measurement goal. In general, the context of the samples provided is a standard software development project in a medium-sized enterprise that has explicitly stated development processes and project goals.

For controlling costs, a measurement goal is defined that checks adherence to the cost plan. For controlling project dates, two measurement goals are defined. One goal is defined for controlling the deadlines of individual project activities and one for controlling whether general project milestones can be met. In order to control the efficiency of defect detection activities, a measurement goal is defined that monitors the slippage rate of defects found.

Table 2: Example GQM Measurement Goals.

ID	**Description**	**Control Need**	**Object**	**Focus**	**Purpose**	**Viewpoint**
COST-PLAN	Project cost plan adherence	COSTS	Project	Costs	Comparison	Project Manager
TIME-PLAN	Project time plan adherence	DATES	Project	Time	Comparison	Project Manager
MILESTONES	Meeting project milestones	DATES	Project	Milestones	Monitoring	Project Manager
SLIPPAGE	Defect slippage of reviews and tests	EFFICIENCY	Review and test processes	Defects found	Monitoring	QA Manager

Table 3 presents a summary of indicators for which a GQM question was derived. We give an indicator name for each corresponding question. For example, the indicator name “Cost Variance” corresponds to the question “Does the project vary from planned costs?” The key performance indicators support answering all specified GQM questions. For each indicator, we define a pseudo formula, which illustrates the computation of the indicator and the base measures needed, and give the reference to the corresponding GQM goal definition.

With respect to cost and time plan adherence, the Earned Value approach is used to define basic indicators. The Earned Value approach is a management technique used to assess the current state of a project. It was first defined at the end of the 19th century when engineers decided to determine cost efficiency by comparing the actual cost of work performed (ACWP) with the earned value (budgeted cost of work performed, BCWP) and the planned value (budgeted cost of work scheduled, BCWS). As work is performed, the earned value of this work is measured on the same basis as for the planned values. Several techniques may be used to compute the earned value for a certain work package, e.g., if a work package is complete, its earned value is equal to the planned value, whereas if a work package has not started yet, its earned value is zero (0/100 method).

When controlling a project, project managers are interested, e.g., in the schedule variance (that is, the earned value minus the planned value) and the cost variance (that is, the earned value minus the actual cost), which allows them to make statements about plan deviations. Performance indexes of the whole project can also be computed. With this approach it is also possible to compute the estimated costs at project completion based on the current performance, so that project managers are able to detect cost plan deviations quickly and are able to react accordingly.

For controlling milestones, a technique called Milestone Trend Analysis is used, which is able to identify the trend of milestone dates by providing a certain graphical presentation of all project milestones. For making statements about defect slippage, the number of found defects is computed per review and test activity as well as the proportion of defects that could have been found earlier in project lifecycle. The assumption is that the later a defect is found (compared to the time it was introduced into the system), the more costs it will cause. For instance, if a design defect is found during system testing, the design has to be modified, which can have an enormous effect on the entire code and can cause high rework effort.

In addition, the developer experience is measured (in number of projects) as a variation factor for defect slippage. The assumption behind this measure is that the more experienced the developers who perform the review and/or test, the more defects will be found.

Table 3: Example Indicators.

ID	**Indicator Name**	**Pseudo Formula**	**Measurement Goal**
CV	Cost Variance	BCWP - ACWP	COST-PLAN
TV	Time Variance	STWP - ATWP	TIME-PLAN
SV	Schedule Variance	BCWP - BCWS	COST-PLAN
CPI	Cost Performance Index	BCWP / ACWP	COST-PLAN
SPI	Schedule Performance Index	BCWP / BCWS	COST-PLAN
TPI	Time Performance Index	STWP / ATWP	TIME-PLAN
ETC	Estimate to Completion	(BAC-BCWP) / CPI / SPI	COST-PLAN
EAC	Estimate at Completion	ETC + ACWP	COST-PLAN
VAC	Variance at Completion	BAC - EAC	COST-PLAN
TAC	Time at Completion	PAC / TPI	TIME-PLAN
DAC	Delay at Completion	PAC - TAC	TIME-PLAN
TCPI	To Completion Productivity Index	(BAC - BCWP) / ETC	COST-PLAN
MOT	Development of milestone dates over time	F_{MOT}(DEADLINES)	MILESTONES
DFPT	Defects found per review/test	F_{DFPT}(DEFECTS)	SLIPPAGE
FDR	Degree of defects that could have been found earlier in project lifecycle per review/test	F_{FDR}(DEFECTS)	SLIPPAGE
DE	Developer Experience	F_{DE}(EMPLOYEE-PROJECTS)	SLIPPAGE

Table 4 presents some metrics that are needed to compute the indicators above. For each metric, we specify a pseudo formula that illustrates the computation. For base measures (which are directly collected from a data base and not computed by the control centre), no formula is provided.

Figure 2 illustrates the Visualization Catena for controlling project costs. On the left hand side, the VC is shown and on the right hand side, a sample visualization of the “Cost Plan Controlling View”. A VC consists of a data interpreting part, containing a number of function instances, which process data entries, and a data visualizing part, containing all views defined on data entries. In the example, we have three function instances: “Compute Earned Value” computes the overall earned value of the project in its current state from the earned values of all activities of the project and their state according to the 0/100 method. “Compute Actual Costs” computes the current project costs from the employees’ effort data and a table containing the costs per employee as well as additional cost items of the project. “Compute Planned Costs” computes the overall planned costs of the project from all activities. Finally, “Earned Value Analysis” computes all indicators as defined above according to the Earned Value method. The upper part of the VC defines a view that illustrates the processed data entries accordingly. In the sample, we have a simple line diagram that graphically illustrates the relationships between different indicators.

Table 4: Example Metrics.

ID	Description	Pseudo Formula
ACWP	Actual Cost of Work Performed per week	F_{ACWP}(EFFORT, EMPLOYEE-COSTS, ADDITIONAL-COSTS)
BCWP	Budgeted Cost of Work Performed per week	F_{BCWP}(BCWP-ACTIVITY, ACTIVITY-STATE)
BCWS	Budgeted Cost of Work Scheduled	F_{BCWS}(BCWS-ACTIVITY)
ATWP	Actual Time of Work Performed per activity and per week	-
STWP	Scheduled Time of Work Performed per activity	-
BAC	Budget at Completion	-
PAC	Plan at Completion	-
DEADLINES	Updated milestone deadlines per week	-
DEFECTS	Number of defects per review/test phase and per defect source	-
EMPLOYEE-PROJECTS	Number of projects per employee	-
EFFORT	Effort per employee and project activity	-
EMPLOYEE-COSTS	Costs per employee	-
ADDITIONAL-COSTS	Costs per additional cost position	-
BCWP-ACTIVITY	Budgeted Cost of Work Performed per activity	-
ACTIVITY-STATE	State of project activities per week	-
BCWS- ACTIVITY	Budgeted Cost of Work Scheduled per activity	-

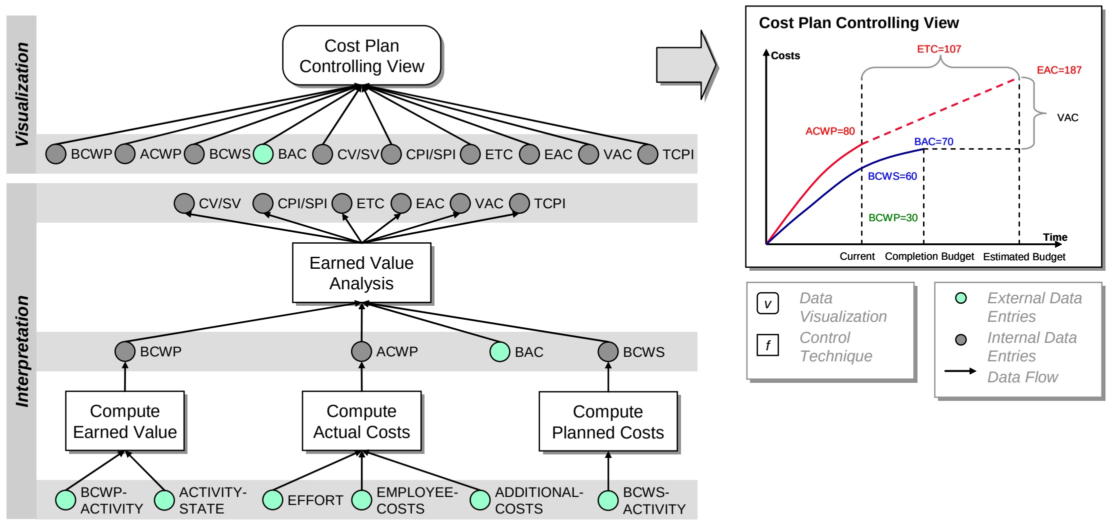

Figure 2: Sample Visualization Catena.1: Project Cost Control.

Figure 3 presents the VC for controlling project dates. Again, a function instance called “Earned Value Analysis” applies the Earned Value approach to three base measures and computes all indicators that make statements about the project time. Independently, a second function instance called “Milestone Trend Analysis” computes the development of all milestone dates over time in order to identify positive and negative trends. By means of the slope of the curve for a certain milestone, we can determine whether it will be possible or not to meet the milestone date. The visualizing part of the VC defines three views. The “Time Plan Controlling View” integrates Earned Value time information with the “Cost Plan Controlling View” from the first VC. Note that in order to be complete, the VC definition should also include the function instances needed to compute all other indicators shown in the chart. However, for simplicity reasons, this information is left out in the figure. The “Milestone Controlling View” displays a triangular chart to illustrate the development of milestone trends. Finally, the “Project Dates Controlling View” simply references the two others in order to provide a convenient entry point for controlling project dates.

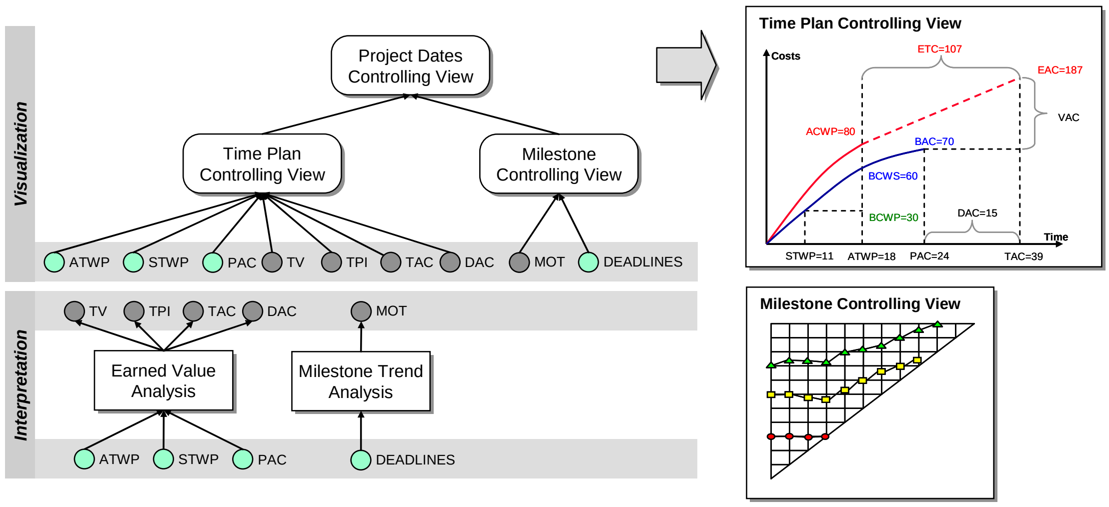

Figure 3: Sample Visualization Catena.2: Project Dates Control.

Figure 4 shows the VC for controlling the efficiency of defect detection processes. The VC consists of a function instance called “Defect Slippage Analysis” that computes the specified indicators from the defect tracking system, and a function instance called “Experience Distribution” that computes different classes of experiences for project members based on the number of projects they participated in. On the left hand side, the “Defect Slippage Controlling View” graphically presents the defect slippage between project phases. On the right hand side, the distribution of team member experience is shown in order to provide adequate support for interpreting the slippage rates.

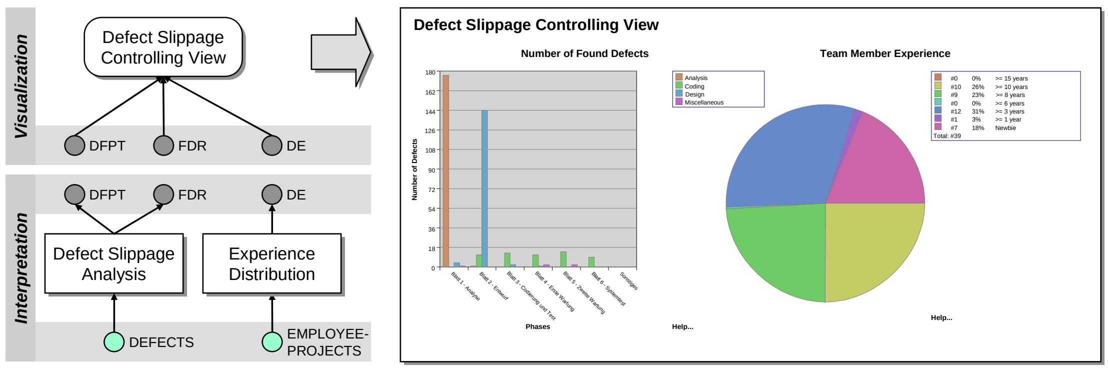

Figure 4: Sample Visualization Catena.3: Process Efficiency Control.

4. Related Work

The following section presents an overview of selected SPCC approaches. Here, the scope has been defined as on-line data interpretation and visualization on the basis of past experience. A more detailed description of the strengths and weaknesses of the approaches presented can be found in [13]. We explicitly do *not* include project dashboards in this overview. Many companies develop their own dashboards (mainly based on Spreadsheet applications) or use dashboard solutions that provide a fixed set of predefined functions for project control or solely focus on a very limited controlling focus (e.g., only deal with product quality or solely focus on project costs).

Provence [11] is generally a framework for project management. It informs managers about state changes of processes and products and allows them to initiate dynamic replanning steps. The basic architecture is focused on openness and adaptability with respect to integration into a specific software development organization.

Amadeus [18] is a metric-based analysis and feedback system embedded into the process-centred SDE Arcadia. The focus is on the integration of measurement as an active component within software development processes. Amadeus is based on a client/server architecture and uses active agents for interpreting user-defined scripts. It is possible to add new functionality by creating a new script or expanding the server's or client's tool kit. The data is processed with respect to the kinds of usage purposes implemented by appropriate techniques and methods, like classification trees or interconnectivity analysis.

Ginger2 [22] implements a computer-aided empirical software engineering (CAESE) environment and is focused on experimental aspects of software development, so-called in vitro studies. Ginger2 presents a framework for empirical studies, consisting of a life cycle model, data collection models, and a basic CAESE framework. It fulfils its tasks through a multitude of predefined data collection and analysis techniques, such as techniques for gathering audio and video data of the experiment participants, data about mouse and window movements, and data about tool usages and program changes.

The SME (Software Management Environment) [9] is an automated management tool that provides a predefined pool of techniques to observe, compare, predict, analyze, assess, plan, and control a software development project. It was developed within NASE/SEL [12]. The SME distinguishes three types of databases: one captures information about previous projects, a second one provides research results from studies of software development projects, and a third one includes management rules for guiding a project manager.

WebME (Web Measurement Environment) [21] is a successor to the SME approach and provides similar functionality. It enhances the capabilities of the SME in terms of supporting distributed development of software. It is built upon a three-layered, mediated architecture, which uses a web browser to access one of the WebME servers and data wrappers to collect data from distributed SDEs. It uses the Data Definition Language (DDL) for specifying scripts, which are used to determine data sources and data transformations.

The PPM (Process Performance Manager) [10] is a tool that is used to support the management of business processes. It is discussed here because of its open and portable architecture. The basic idea is to close the feedback gap between the specification and the execution of business processes. A number of source systems can be integrated via an XML interface. So-called key performance indicators can be visualized with the help of different views and filters, for instance, in order to identify deviations from nominal performance guidelines.

PAMPA [17] is a tool that is especially designed for data collection and visualization. It uses intelligent agents to reduce the overhead of data collection and analysis. For that, it provides a set of predefined objects with attributes and relationships between each other, which are instantiated for each project.

5. Conclusion and Future Work

This article has presented the G-SPCC approach for establishing goal-oriented project control by formalizing the way measurement data are interpreted and visualized according to a previously defined measurement goal. The basic project control set-up approach was illustrated with several usage scenarios related to explicitly defined business goals. For each control need, a measurement goal is specified and metrics are derived according to the GQM paradigm. After that, project control techniques and visualizations are defined using a Visualization Catena that formally describes what kind of data analysis has to be applied and how the data should be visualized. This way, statements about achieving the measurement and business goals could be made. Furthermore, some related approaches were presented. Existing approaches offer mostly partial solutions. Notably, goal-oriented usages based on a flexible set of techniques and methods are not comprehensively supported [13].

The expected benefits of the G-SPCC approach include: (1) improvement of quality assurance and project control by providing a set of custom-made views of measurement data, (2) support of project management through early detection of plan deviations and proactive intervention, (3) support of distributed software development by establishing a single point of control, (4) enhanced understanding of software processes, and improvement of these processes, via measurement-based feedback, and (5) prevention of information overload through custom-made views with different levels of abstraction.

The further development and evaluation of the G-SPCC approach will be conducted in the context of the so-called Soft-Pit project funded by the German Federal Ministry of Education and Research (*http://www.soft-pit.de*). One important topic of this project is how to operationally introduce an SPCC into a company and into a specific software development project, that is, how to determine which processes are affected and which adaptations have to be made in order to effectively make use of an SPCC. The aim of Soft-Pit is to develop a holistic project control centre that integrates information from different data sources in a goal-oriented way. An important research issue in this context is the elicitation of information needs for the roles involved and the development of mechanisms for generating adequate role-oriented visualizations of the project data. Another issue is support of change management. When the goals or characteristics of a project change, the real processes react accordingly. Consequently, the control mechanisms, which should always reflect the real world situation, must be updated. This requires flexible mechanisms that allow for reacting to process variations. One long-term goal of engineering-style software development is to control and forecast the impact of process changes and adjustments on the quality of the software artefacts produced and on other important project goals. The G-SPCC approach is expected to be a valuable contribution towards reaching this goal.

6. Acknowledgements

We would like to thank Sonnhild Namingha from Fraunhofer IESE for reviewing a first version of the article. This work was partly funded by the Deutsche Forschungsgemeinschaft as part of the Special Research Project SFB 501 “Development of Large Systems with Generic Methods”. In addition, the work was partly funded by the Federal Ministry of Education and Research (BMBF) in the context of the “Soft-Pit” project, a “Software Engineering 2006” research initiative.

7. References

[1] Basili, V.R.; Caldiera, G.; Rombach, D: The Experience Factory. Encyclopaedia of Software Engineering 1, 1994, pp. 469-476.

[2] Basili, V. R.: Matching Software Measurements to Business Goals. Keynote Address at the 2003 Software Management Conference, San Jose, California, February 2003.

[3] Burghardt, M.: Projektmanagement: Leitfaden für die Planung, Überwachung und Steuerung von Entwicklungsprojekten. Publicis MCD Verlag, 2000.

[4] ISO 8402. Quality management and quality assurance-Vocabulary. August 1995.

[5] Gibbs, W.W.: Software's Chronic Crisis. Scientific American, 1994, pp. 86-95.

[6] Heidrich, J.; Soto, M.: Using Measurement Data for Project Control. In Proceedings of the Second International Symposium on Empirical Software Engineering (Vol. II), Rome, 2003, pp. 9-10.

[7] Heidrich, J.: Effective Data Interpretation and Presentation in Software Projects. Technical Report 05/2003, Sonderforschungsbereich 501, University of Kaiserslautern, 2003.

[8] Heidrich, J.: Custom-made Visualization for Software Project Control. Technical Report No. 06/03, Special Research Project 501, University of Kaiserslautern, 2003.

[9] Hendrick, R.; Kistler, D.; Valett, J.: Software Management Environment (SME)— Concepts and Architecture (Revision 1); NASA Goddard Space Flight Center Code 551, Software Engineering Laboratory Series Report SEL-89-103, Greenbelt, MD, USA, 1992.

[10] IDS Scheer AG: Optimieren Sie Ihre Prozess Performance – Process Performance Manager®; IDS Scheer AG: White Paper, 2000.

[11] Krishnamurthy, B.; Barghouti, N.S.: Provence: A Process Visualization and Enactment Environment. Proceedings of the 4th European Software Engineering Conference, Lecture Notes in Computer Science 717; Springer: Heidelberg, Germany, 1993, pp. 451-465.

[12] McGarry, F.; Pajerski, R.; Page, G.; Waligora, S.; Basili, V.R.; Zelkowitz, M.V.: An Overview of the Software Engineering Laboratory; Software Engineering Laboratory Series Report SEL-94-005, Greenbelt, MD, USA, 1994.

[13] Münch, J.; Heidrich, J.: Software Project Control Centers: Concepts and Approaches. Journal of Systems and Software, 70 (1), 2003, pp. 3-19.

[14] Project Management Institute: A Guide to the Project Management Body of Knowledge (PMBOK® Guide) 2000 Edition. Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073-3299 USA, 2000.

[15] Rombach, H.D.: Practical benefits of goal-oriented measurement. Software Reliability and Metrics, 1991, pp. 217-235.

[16] Rombach, H.D.; Verlage, M.: Directions in Software Process Research. Advances in Computers 41, 1995, pp. 1-63.

[17] Simmons, D.B.; Ellis, N.C.; Fujihara, H.; Kuo, W.: Software Measurement – A Visualization Toolkit for Project Control and Process Improvement; Prentice Hall Inc: New Jersey, USA, 1998.

[18] Selby, R.W.; Porter, A.A.; Schmidt, D.C.; Berney, J.: Metric-Driven Analysis and Feedback Systems for Enabling Empirically Guided Software Development. Proceedings of the 13th International Conference on Software Engineering, 1991, pp. 288-298.

[19] Shaw, M.: Prospects for an Engineering Discipline of Software. IEEE Software 7(6), 1990, pp. 15-24.

[20] Soto, M.; Heidrich, J.: Process Modeling and Plan-based Execution. In Proceedings of the Second International Symposium on Empirical Software Engineering (Vol. II), Rome, 2003, pp. 7-8.

[21] Tesoriero, R.; Zelkowitz, M.V.: The Web Measurement Environment (WebME): A Tool for Combining and Modeling Distributed Data. Proceedings of the 22nd Annual Software Engineering Workshop (SEW), 1997.

[22] Torii, K.; Matsumoto, K.; Nakakoji, K.; Takada, Y.; Takada, S.; Shima, K.: Ginger2: An Environment for Computer-Aided Empirical Software Engineering. IEEE Transactions on Software Engineering 25(4), 1999, pp. 474-492.