Practical Guidelines for Introducing Software Cockpits in Industry

Marcus Ciolkowski, Jens Heidrich, Jürgen Münch

Abstract

Software project control is an essential component for project success. The basis of all control approaches is roughly the same: the collection and effective usage of measurement data to allow for quantitative decision making. But many organizations have problems with establishing project control mechanisms. On the one hand, this has something to do with the complexity of today's software development projects; that is, the number of people involved, the number of distributed development locations, the number and difficulty of functional and non-functional (quality) requirements, as well the experience of the people controlling and steering the project. On the other hand, quantitative project control mechanisms have to be widely accepted within an organization and have to be part of daily life in order to assure high-quality und up-to-date data for project control. This requires the integration of control mechanisms into the development process as well as strategies on how to introduce controlling mechanisms and provide training in how to use them efficiently. Software cockpits, also known as Software Project Control Centers, support the management and controlling of software and system development projects and provide a single point of project control. They provide a more systematic way of deriving and integrating control mechanisms into the development process. This article briefly describes an approach for holistic project control mechanisms developed as part of the public "Soft-Pit" project making use of reusable, customizable control components, which are instantiated according to different organizational goals and characteristics. Furthermore, it describes practical guidelines for introducing such control mechanisms into an organization based on our experience gathered in the Soft-Pit project as well as based on expert opinions collected during a workshop on software cockpits. The article concludes with a brief summary and an outlook on future research.

1. Introduction

One factor that is crucial for successfully conducting a development project is the existence of well-specified and coordinated development processes. Therefore, it is necessary to have efficient management and controlling mechanisms in place. Many companies are currently establishing so-called software cockpits for systematically supporting quality assurance and management. Software cockpits, also known as Software Project Control Centers (SPCC) [13] or Project Management Offices (PMO) [14], centrally integrate all relevant information for monitoring and controlling purposes. A project manager can use them to get an overview of the state of a project and a quality assurance manager can use them to check the quality of the software product. In addition to these primary users of an SPCC, basically any role in a project may benefit from making direct or indirect use of the SPCC functionality. For instance, a developer can use the SPCC to keep track of code quality or to trace quality issues. The benefit provided by an SPCC for a certain project role depends on the functionality and services offered. However, the needs with respect to project control differ between organizations, projects, and roles. They depend on organizational goals (business goals), process maturity, the experience of the project team, and many other factors. For instance, for multi-disciplinary, distributed software development, measurement data has to be collected from different sources (locations) and formats. In this case, integration of data is crucial for getting a consistent picture of the project state.

In practice, a variety of simple dashboard approaches for project control exist. More comprehensive approaches that support advanced management techniques and allow for organization-wide data collection, customized interpretation, and visualization are rarely implemented. Despite significant progress in recent years, setting up and establishing software cockpits such that organizations get sustainable benefits is still a challenging task: First, a measurement culture has to be established within an organization and has to be part of daily life in order to assure high-quality and up-to-date data for project control. This requires goal-oriented derivation of key performance indicators, integration of data collection procedures and control mechanisms into the development processes, and analysis and interpretation of the results in the context of project goals and organizational goals. In addition, strategies are needed that define how to effectively introduce controlling mechanisms (e.g., by appropriate training of all team members) and gain acceptance by team members. Second, successful deployment of software cockpits needs to take into account many factors, some of which change frequently. These factors include the number of organizations and people involved, the number of distributed development locations, the heterogeneity of development platforms and techniques, and the complexity of functional and non-functional requirements as well the experience of the people responsible for controlling and steering a project. Especially in the area of global development projects (across different organizations in different countries), heterogeneous processes resulting in incompatible workflows and data make it even harder to control and steer a project. Third, to create a sustainable measurement program, gaining management support for software cockpits is crucial. This requires empirical evidence on the benefits, risks, and costs of software control mechanisms compared to projects without quantitative project control. However, currently only very few empirical studies address these issues (see [4]). Empirically validated guidelines can support the successful deployment of software cockpits by addressing the challenges mentioned above. This article presents guidelines derived from experiences made with setting up and establishing software cockpits in industry. They cover issues related to defining appropriate measurement systems, implementing and tailoring software cockpits, deploying software cockpits, controlling distributed and global projects, analyzing success factors, and benefiting from software cockpits. The guidelines were initially derived from industrial case studies (several implementations of software cockpits in industrial practice, including software development for logistics systems, information systems, and web applications) and from two large research projects (the special research project SFB 501 and the applied research project Soft-Pit). We enriched the guidelines by illustrating typical problems and solutions encountered during our industrial case studies. In addition, we used expert opinions, collected during a workshop on software cockpits, to interpret and enrich the findings from the case studies.

Section 2 of the article introduces the concept of Software Project Control Centers and discusses related work, Section 3 presents the control cockpit used as part of the Soft-Pit project and illustrates empirical results, and Section 4 lists practical guidelines for successfully introducing a control cockpit. The article concludes with a summary and an outlook on future research.

2. Software Project Control Centers

A Software Project Control Center is a control system for software development that collects all relevant data for 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 case of plan deviations. An SPCC could also support packaging of data (e.g., as predictive models) for future use and contribute to an improvement cycle spanning a series of projects (see [5]). An SPCC has to provide a variety of interfaces for interacting and integrat-

ing with other project management and development tools, such as project planning tools, the Software Development Environment (SDE), measurement tools, data analysis tools, process enactment machines, and experience bases.

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 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. The gathered experience can be packaged for future projects after the project is completed in order to support organization-wide improvement cycles.

A *Software Project Control Center* is defined as a means for interpreting and visualizing measurement data during process performance and therefore supports project control. An SPCC has a logical architecture that clearly defines the interfaces to its environment, especially to all project members relying on SPCC information, as well as a set of underlying techniques and methods that support the controlling of a project. From a more technical perspective, an SPCC utilizes data from the current project (e.g., the project's goals, characteristics, baselines, and measurement data) and experiences from previous projects (e.g., information captured in quality, product, and process models) and produces a visualization of measurement data by using the incorporated techniques and methods for interpreting the data.

An SPCC is a general approach to project control and is not necessarily tool-supported. However, in order to efficiently carry out control activities such as monitoring defect profiles, detecting abnormal effort deviations, cost estimation, and root-cause analyses of plan deviations, a certain amount of tool support is necessary and inevitable. In general, tool support for the different logical tasks of an SPCC varies. Most of the existing, rather generic, SPCC approaches offer only partial solutions. Especially purpose- and role-oriented usages based on a flexible set of techniques and methods are not comprehensively supported. For instance, SME (Software Management Environment) [8], [11] offers a number of role-oriented views on analyzed data, but has a fixed, built-in set of control indicators and corresponding visualizations. The SME successor WebME (Web Measurement Environment) [18] has a scripting language for customizing the interpretation and visualization process, but does not provide a generic set of applicable controlling functions. Amadeus [15] and Ginger2 [19] offer a set of purpose-oriented controlling functions with a certain flexibility, but lack a role-oriented approach to data interpretation and visualization. A more detailed overview of the state of the art in Software Project Control Centers can be found in [13].

Usually, state-of-the-art approaches provide a certain methodological foundation addressing the customization of an SPCC to specific application environments. In state-of-the-practice approaches, this methodological support is widely missing. Many companies develop their own dashboards (mainly based on Spreadsheet applications) or use dashboard solutions (e.g., test dashboards [17]) that provide a fixed set of predefined functions for project control (e.g., dealing with product quality only or solely focusing on project costs) and are very specific to the purpose for which they were developed.

3. The Soft-Pit Control Cockpit

Soft-Pit is a project funded by the German Federal Ministry of Education and Research (*http://www.soft-pit.de*). The project focuses on getting experience and methodological support for introducing control centers into companies and projects. 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 make effectively use of an SPCC. The aim of Soft-Pit

is to develop a holistic project control center that integrates information from different data sources in a goal-oriented way.

The indicators used to control a development project depend on the project's goals and the organizational environment. There is no default set of indicators that is always used in all development projects in the same manner [12]. The concrete indicators that are chosen should be derived in a systematic way from the project goals [10], making use of a goal-oriented derivation process such as the Goal Question Metric (GQM) paradigm [2]. Some examples from indicators used in practice can be found in [1] and [11]. With respect to product quality, there exists even an ISO standard [9]. However, the concrete usage of the proposed measures depends upon the individual organization. The ideas behind GQM and the Quality Improvement Paradigm (QIP) [2] are well-proven concepts that are widely applied in practice today.

The approach taken by Soft-Pit is to make use of the QIP and GQM for introducing an improvement-oriented software project control cycle (see [6], [7]):

- I. Characterize Control Environment: First, project stakeholders characterize the environment in which project control shall be applied in order to set up a corresponding measurement program that is able to provide a basis for satisfying all needs.
- II. Set Control Goals: Then, measurement goals for project control are defined and metrics are derived determining what kind of data to collect. In general, any goal derivation process can be used for defining control objectives. For practical reasons, we focus on the GQM paradigm for deriving indicators.
- III. Goal-oriented Composition (Choose Process): Next, all control mechanisms for the project are composed based on the defined goals in order to provide online feedback on the basis of the data collected during project execution; that is, control techniques and visualization mechanisms are selected from a corresponding repository and instantiated in the context of the project that has to be controlled.
- IV. Execute Project Control Mechanisms: Once all control mechanisms are specified, a set of role-oriented views is generated for controlling the project. When measurement data are collected, the control mechanisms interpret and visualize them accordingly, so that plan deviations and project risks are detected and a decision-maker can react accordingly. If a deviation is detected, its root cause must be determined and the control mechanisms have to be adapted accordingly. This, does, for example, require data analyses on different levels of abstraction in order to be able to trace causes for plan deviations.
- V. Analyze Results: After project completion, the resulting control mechanisms have to be analyzed with respect to plan deviations and project risks detected in time, too late, or not detected at all. The causes for plan deviations and risks that were detected too late or that were not detected at all have to be determined.
- VI. Package Results: The analysis results of the control mechanisms that were applied may be used as a basis for defining and improving control mechanisms for future projects (e.g., selecting the right control techniques and data visualizations, choosing the right parameters for controlling the project).

The evaluation of the approach is currently being conducted in the context of several industrial case studies with German companies from different domains, in which the developed control center and its deployment are evaluated. The project is mainly organized intro three iterations focusing on different controlling aspects. In the case studies, the Specula Project Support Environment (PSE) tool is used for implementing the basic concepts. Specula PSE is a research prototype that basically automates the specification and packaging of all control mechanisms and is also able to automatically execute them. SPCC users may use the tool for

collecting measurement data and for utilizing the generated visualizations for project control. The control mechanisms contained in the Specula repository depend on the organization (and the very project that should be controlled). Some components may be more general and applicable for several companies and projects, whereas others may be very specific and implement organization-specific control strategies. This is also related to the different *kinds* of components in the repository. For instance, one control component may implement a (fairly) complex control technique (like an Earned Value Analysis) and another component may just provide some simple data processing functionality for supporting other functions (like scaling a time series or converting between different data types). Figure 1 gives an overview of the high-level architecture. Specula PSE has a classical three-layered architecture:

- The *data collection layer* deals with accessing different data sources. Project data and measurement data need to be collected automatically by accessing different existing databases, or semi-automatically by using web forms for importing data from files or for entering data manually.

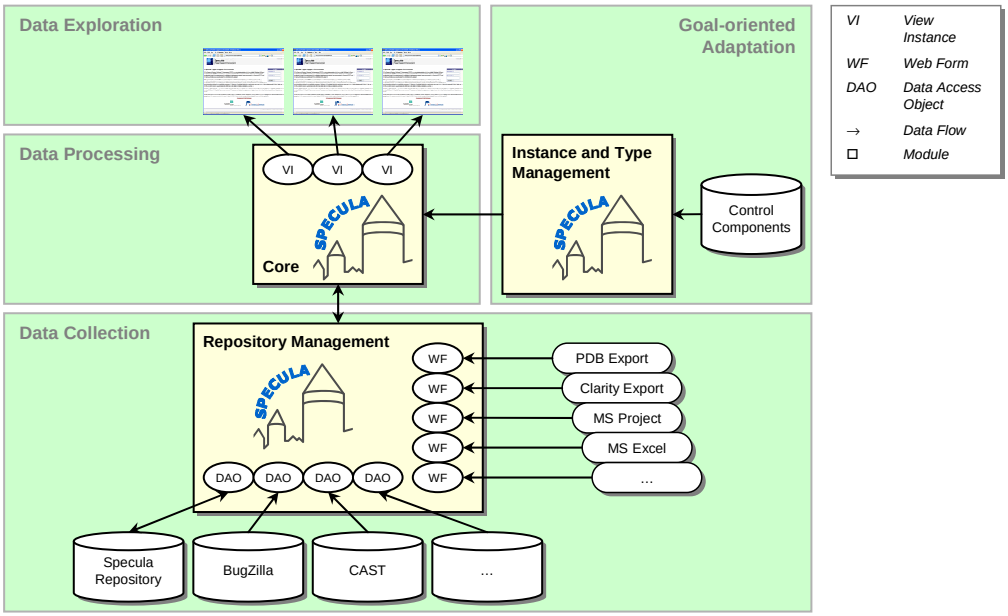

Figure 1: Specula PSE High-Level Architecture.

- The *data processing layer* uses the data collection layer for accessing data from different sources in a unique way, processing them according to the control mechanisms defined, and finally providing services upon the processing results. Different services are offered for user management, checking the consistency of control mechanisms, accessing data repositories and VC specifications, and importing/exporting functionality. In order to adapt the Specula PSE functionality to the context of the respective project that needs to be controlled, a corresponding module manages all control components; that is, it supports the definition of new control components, the reuse of existing components, and the parameterization of control components according to the project context.

- The Servlet-based *data exploration layer* uses the services of the data processing layer for providing a graphical user interface for accessing all Specula PSE functionality, including displaying visualizations, managing data, administering control mechanisms, importing/exporting data, and changing user profiles.

Specula PSE can be used as a framework for systematically composing project control mechanisms based on reusable control components; it provides a core functionality for pro-

ject control and clearly defines interfaces for specifying additional modules that can be freely enhanced with respect to specific needs. Customization includes specification of types, instances, and administration information (users and groups), implementation of data access object packages for accessing different repositories, implementation of data types for defining logical data containers, implementation of functions for processing measurement data, implementation of views for displaying data, and implementation of web forms for managing (importing, exporting, adding, removing) data.

Figure 2 shows the user interface of the Specula PSE tool. On the left side, the overall navigation bar can be seen. The menu close to the navigation bar displays all available views for controlling the project. On the right side, one can see the selected view for analyzing effort data. The tool is used within the Soft-Pit project for evaluating the basic approach in an industrial environment. Evaluation included: (a) the perceived usefulness and ease of use of the approach, (b) found plan deviations and project risks, and (c) costs for setting up and applying an SPCC. The results of the first iteration of case studies can be found in [4]. The second iteration consisted of four software development projects from three different organizations, which actually used the Specula PSE tool for controlling the projects. Preliminary results illustrate the general usefulness and ease of use of the SPCC and show that following a structured process for setting up and applying an SPCC does result in a significantly improved detection rate of plan deviations and project risks. The costs for setting up and applying an SPCC were around 10% of the overall development effort for a medium-sized project (approx. 10 team members). Empirical results of the second iteration will be published soon.

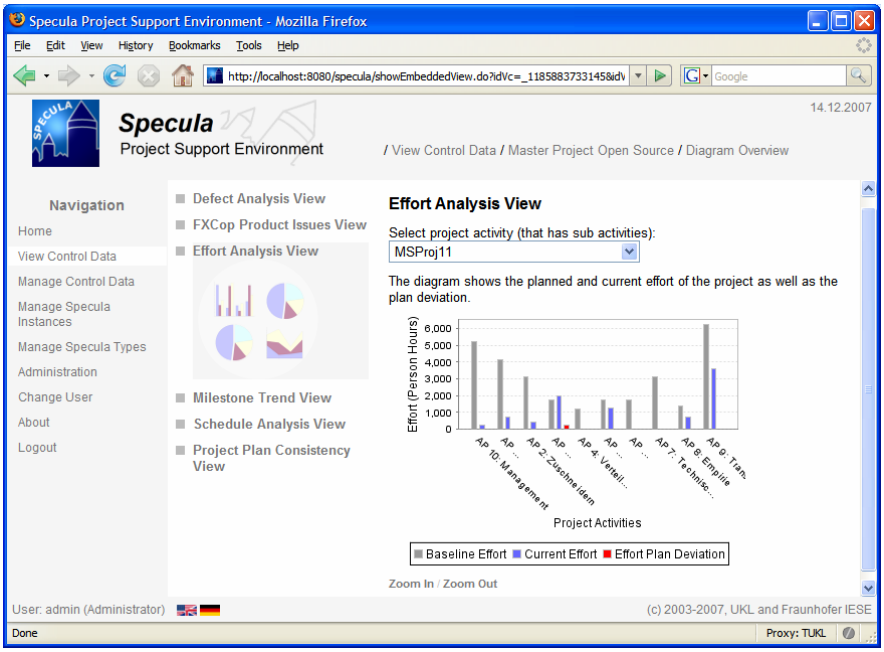

Figure 2: User interface of the Specula PSE tool.

4. Practical Guidelines

In the following, practical guidelines for introducing Software Project Control Centers in an industrial setting are described. This includes technical as well as organizational aspects. The guidelines are based on industrial case studies conducted in the context of the "Soft-Pit" project [4] as well as on discussions in the context of the 1st Workshop on Measurement-based Cockpits for Distributed Software and Systems Engineering Projects (SOFTPIT 2007, available via [16]). Even though the guidelines were derived with a focus on project control-

ling issues, a lot of them could also be applied to implementing general measurement programs. The guidelines are grouped according the six basic phases of the Quality Improvement Paradigm (QIP) [2] that were already used as a foundation of the Soft-Pit project control approach. Figure 3 gives an overview of all guidelines. In the following, we will describe all items of the list of practical guidelines in more detail, discuss practical solutions (e.g., as applied in the Soft-Pit project), and present open issues. The latter are the starting point for creating a future research roadmap in the field of Software Project Control Centers. The top issues will be summarized as part of Section 5.

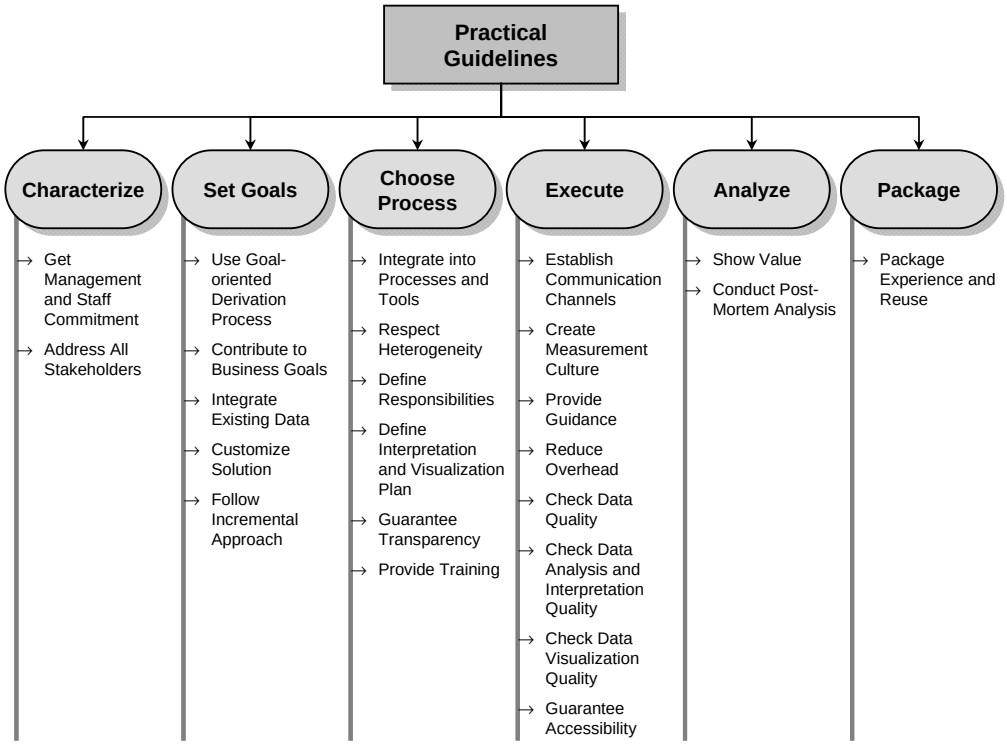

Figure 3: Overview of practical guidelines for project control.

4.1. Characterize

(G1) Get Management and Staff Commitment: As a general rule for implementing successful measurement programs, it is important to obtain the commitment of all parties involved. This primarily includes the management, which has to provide funding for setting up a measurement program and actually executing the program (for data collection, analysis, and interpretation), but also includes the staff who actually have to execute the measurement program and keep it alive. Issues of sensitivity and trust are critical factors for the general acceptance of a measurement program. For creating "trust", it is important to guarantee that data will not be used against people, but for transparently supporting certain explicitly stated goals of the project or the organization. As stated above, Soft-Pit is a *research* project, so the project partners committed themselves to introducing and evaluating mechanisms for project control. In a real application scenario, the benefits of project control and the contribution to higher-level goals have to be made clear in order to get commitment (see also G4). Furthermore, industrial studies on the practical benefits and the return on investment of project control mechanisms (like [4]) help to obtain commitment.

(G2) Address All Stakeholders: Before setting up project control mechanisms, it is important to clarify the potential stakeholders who are involved. Generally, this depends on the purpose and the goals that are to be achieved by introducing quantitative project control

mechanisms. Stakeholders make use of the analyzed and interpreted measurement data (e.g., for controlling the project, generating reports, or assuring quality). In general, we may distinguish between primary and secondary stakeholders of an SPCC. The primary stakeholders are project managers, quality assurance managers, and (financial) controllers, who mainly use the SPCC to actively control different aspects of the software development project and initiate countermeasures in case of plan deviations and detected project risks. The secondary stakeholders are developers and other technical staff, who use the SPCC to enter measurement data as well as to detect root causes for plan deviations and project risks. Usually, the SPCC supports secondary stakeholders, but is not the primary tool for their daily work. In Soft-Pit, primary as well as secondary stakeholders have been addressed when defining project control mechanisms. In addition, measurement experts helped to focus the efforts by introducing quantitative control mechanisms (see also G7).

4.2. Set Goals

(G3) Use Goal-oriented Derivation Process: There is no universal set of control mechanisms that can be applied to all kinds of projects. Therefore, it is important to explicitly specify goals with respect to project control (which should be in conformance with an organization's higher-level goals). Usually, there are quite general goals, which are applicable for different kinds of projects in an organization, but there are also individual goals for a certain project. Explicit definition of all these goals allows for systematic derivation of metrics (e.g., using a measurement paradigm like GQM [2]) and avoids collecting extraneous data that are not needed for assessing and achieving the goals. The latter is especially important with respect to the (organizational) overhead introduced by measurement, especially for small and medium-size enterprises. It is important to involve all affected parties in the metric derivation process in order get meaningful and applicable measurement data (as stated in G2). Most organizations have problems in identifying the "right" set of metrics and deriving the "right" set of indicators that make statements about achieving control goals. Soft-Pit provides a structured method that makes use of the QIP and a goal-oriented metrics derivation process for composing project control mechanisms based on control goals and environmental characteristics. Especially these environmental characteristics are important for deriving the "right" project control mechanisms. This includes, but is not limited to, the characteristics of the development process (e.g., V-Modell® XT[1], RUP, or agile development), organizational characteristics (e.g., whether software is developed in distributed locations, in-house, or whether there is a contractor/supplier relationship), and so on.

(G4) Contribute to Business Goals: A variety of different business goals exist for different organizations, with a tremendous effect on the project and on control goals that have to be met. For some organizations, time to market is important (e.g., companies that develop application software with many competitors); for others, it is reliability of the software product (e.g., embedded software in the medical domain), and so on. The goals with respect to project control should show a clear contribution towards the higher-level goals of the organization. This guarantees that the project control mechanisms are not an end in itself, but used to actually steer and control all projects of an organization. For instance, GQM$^+$Strategies [3] is a measurement approach that builds on the well-tested GQM approach for planning and implementing software measurement. GQM$^+$Strategies provides mechanisms for explicitly linking measurement goals to higher-level goals of the organization, and further to goals and strategies at the level of the entire business.

(G5) Integrate Existing Data: Derivation of metrics and indicators should not be a pure top-down process, but should take into account the capabilities of the environment in which

[1] The V-Modell® XT is a German system and software development standard, see *http://www.v-modell.iabg.de*.

measurement takes place. The environment has to be analyzed with respect to already collected data that may contribute to the explicitly stated control goals or data that could easily be collected within the corresponding environment and contribute to the stated goals. This is particularly important when introducing project control mechanisms for the first time and when changing the development process (which may be necessary for getting valid measurement) is not an option. In this case, project control mechanisms should be introduced step by step. Their first execution should introduce a minimum of changes and use the existing infrastructure as far as possible (see also G7). This eases the integration of project control mechanisms and helps in getting the necessary acceptance by the staff.

(G6) Customize Solution: There exists no off-the-shelf set of project control mechanisms that fits all purposes. It is important to customize the functionality provided to the specific needs and goals with respect to project control. Different stakeholders, goals, and environmental characteristics have to be considered when customizing project control mechanisms. For most organizations, it is difficult to come up with reliable upper and lower bounds and baselines for project control indicators that fit to their specific characteristics. One option is analyzing past project data to come up with reliable baselines. However, this requires an already existing database of former projects and corresponding measurement data. Expert interviews may also be an appropriate means for defining initial baselines. If absolute baselines are available, trends may also be used for assessing the current project state (whether the project improved with respect to the last status report). Soft-Pit provided control mechanisms for all three options depending on the available data and expert knowledge. Customizing project control mechanisms is a recurring process: Baselines and other customization parameters of control mechanisms have to be adapted (for instance, if the project plan is changed or the set of baselines is not applicable to a certain type of project) to reflect the current project situation appropriately.

(G7) Follow Incremental Approach: When introducing project control mechanisms in an organization, it is easier to start small and broaden the scope step-by-step. The complexity of control goals addressed, indicators computed, and metrics collected may be increased over time. Soft-Pit started with a quite generic SPCC in the first iteration, which provided basic functionality that mainly relied on data already existing at the different case study providers. The second and third iterations added more specific goals and made use of more customization options. This also required changes of the development process. For instance, for computing defect density per code module, developers have to refer to an already filed defect when committing code to the versioning system. It would have been difficult to introduce such kinds of changes when introducing the control center for the first time.

4.3. Choose Process

(G8) Integrate into Processes and Tools: In order to be used efficiently, measurement has to become a core component for controlling a project. Therefore, it is important that the data collection process as well as the data interpretation and visualization processes are integrated into the (organizational) development process. This includes institutionalizing defined metrics as well as making use of collected data. Some control mechanisms also imply requirements for organizing project work (e.g. an Earned Value Analysis requires explicitly defined work products and their actual/earned value). If controlling mechanisms are already in place, it is important to integrate them. For instance, if tools are already used for project planning or analyzing code quality, these tools should be integrated into the SPCC and vice versa, if possible. The tool used in the Soft-Pit project provided a variety of interfaces for collecting data from various management and development tools.

(G9) Respect Heterogeneity: Software is increasingly being developed at distributed locations, which usually have heterogeneous development processes. This may result in incompatible workflows and data provided by different development locations. Harmonizing development processes may work within one company, but is difficult across different organizations (e.g., collaborating on one common project). It is important to address this issue when setting up project control mechanisms. This may require creating process abstractions and interfaces in order to guarantee compatibility of workflows and measurement data collected. This also has an influence on the security mechanisms provided by an SPCC. It may not be desired that a project manager of company A should be able to access detailed project data of company B. This may require setting up corresponding accessibility roles for different granularities of data.

(G10) Define Responsibilities: For quantitative project control, it has to be clear which project member has to provide which data for which purpose at which point in time of the process, and who the stakeholders of the collected data are. It has to be specified when the collected data should be interpreted and visualized and how to detect and react to critical project states. Such kinds of information are usually specified as part of a measurement plan (as done in the Soft-Pit project). However, this requires training of all people participating in the project (see also G13).

(G11) Define Interpretation and Visualization Plan: If data are collected for controlling purposes, critical project states have to be detected, root causes for such states have to be derived, and corresponding countermeasures have to be defined. This requires an efficient data analysis and interpretation process on the one hand (e.g., mechanisms for data abstraction as well as the possibility to drill down into data), and efficient data visualizations on the other hand (e.g., graphical detection of plan deviation as early as possible). An SPCC should support the efficient and effective derivation of countermeasures in case of detected plan derivations and project risks. It should support the detection of root causes for plan deviations and project risks, so that an experienced project manager is able to derive corresponding countermeasures. Manual interpretation of measurement data can be difficult depending on the underlying complexity of a metric or an indicator and on the experience of the person interpreting the data. If a formal model is used for data interpretation, calibration and setting realistic target values and baselines is crucial for getting acceptance (see also G6).

(G12) Guarantee Transparency: Transparency is important for getting staff commitment. People providing data should know about the purpose and goals related to collecting exactly this kind of data. For getting reliable data, it should be clear who will use the data in the end for what purpose. The resulting decision-making process should be as transparent as possible. In the Soft-Pit project, a formal model was derived describing the relationships between control goals, related indicators, and metrics. The model was discussed with all participants of the projects.

(G13) Provide Training: If quantitative project control mechanisms shall be applied, it is important to spent resources on successfully introducing such mechanisms in practice. People need to be trained in collecting valid measurement data, stakeholders need to be trained in using an SPCC, and managers also need to be trained in how to define the project control mechanisms needed for covering the higher-level goals of an organization. In the Soft-Pit project, trainings were conducted with all SPCC users. External measurement experts supported managers in defining goals and deriving metrics and indicators that were suited for the specific project and organizational context.

4.4. Execute

(G14) Establish Communication Channels: Especially for a distributed development project (in which different organizations participate), it is difficult to establish communication channels and clearly state who should communicate with whom in which case. This is a general problem of such projects, but has a strong influence on the conception of control centers. For instance, it has to be specified at which point in time data is valid (e.g., everybody has finished entering effort data) and can be analyzed for project control purposes. Furthermore, it has to be specified how to transparently communicate countermeasures in case of plan deviations. Empirical examples of what such communication channels should look like for establishing efficient control mechanisms is widely missing.

(G15) Create Measurement Culture: A measurement culture is necessary for establishing a successful measurement program in general and project control mechanisms in particular. Acceptable practices have to be clear from the beginning (i.e., blame the process, not the people) in order to create an atmosphere of trust. This also includes feedback sessions and the discussion of data analysis and interpretation results. In the Soft-Pit project, the control center was used differently across the case studies. Some organizations already had a kind of measurement culture and were aware of the problems related to data collections. In order to support creation of such a culture, every case study had a so-called coach (a measurement expert from the research partners of the project), who was responsible for supporting the setup and execution of project control mechanisms.

(G16) Provide Guidance: In practice, it is difficult for a project manager to determine whether the project is in a good or bad state. This largely depends on his/her experience. An SPCC can support a project manager (and other project roles) in detecting plan deviations and projects risks. The aim is not to replace the project manager, but to help him/her to make decisions on a quantitative basis. However, further guidance is needed to support detection of root causes for plan deviations and project risks or to provide a list of countermeasures that may save the project in case of plan deviations and risks. In Soft-Pit, we implemented control mechanisms that provide drill-down and abstraction functionality for detecting root causes. A list of countermeasures was not provided, because this usually requires building up an extensive experience base [2] of appropriate actions that could be applied in a certain project situation. This was beyond the scope of the project.

(G17) Reduce Overhead: For acceptance of an SPCC, it is important to reduce the overhead introduced by applying project control mechanisms. An SPCC should provide a central point for data analysis and interpretation, so that consulting a variety of other project control tools can be avoided. An SPCC should not re-implement their functionality, but integrate (data of) these tools and provide access to them (e.g., via a link) if related plan deviations or project risks are detected. Importing data from other tools (like MS Project, BugZilla, or FXCop) should be automated as far as possible in order to avoid unnecessary overhead. For instance, the SPCC used in the Soft-Pit project provided generic interfaces for collecting data from other tools and integrating them into a common decision base.

(G18) Check Data Quality: High quality of the collected data is critical for making it possible to draw the right conclusions. If data are collected automatically (e.g., code metrics), at least the syntactical correctness of the data is assured; that is, data can be generated in the required format. If data are collected manually (e.g., effort data), carefully defined data collection procedures need to ensure the correct syntax; for example, the effort data may contain typing errors (e.g., letters) that have to be dealt with. Checking and assuring semantic correctness requires even more advanced techniques, such as triangulation (i.e., cross-checking data from different sources).

(G19) Check Data Analysis and Interpretation Quality: If data analysis and interpretation mechanisms for project control are specified, their correct functionality should be checked regularly. In Soft-Pit, this included simple plausibility checks on whether control techniques like Milestone Trend Analysis or Earned Value Analysis provided correct results. This was especially done at the beginning of the case studies and was done less frequently towards the end.

(G20) Check Data Visualization Quality: The visualizations provided by an SPCC should be checked regularly with respect to whether all data is displayed correctly. Scalability of diagrams, especially, is important when dealing with real project data. Moreover, the functionality of drill-down and abstraction mechanisms (as important means for detecting root causes of plan deviations and project risks) should be checked regularly.

(G21) Guarantee Accessibility: In order to work effectively, all stakeholders must be able to access the SPCC functionality easily. The Soft-Pit SPCC was implemented using web-based technologies, which guaranteed easy client access (using a web browser). But accessibility is not only a technical issue. For an SPCC user, it is important to have access to the data that is needed for his specific role and only to those data. This should be done in order to avoid information overload and guarantee data privacy. In Soft-Pit, individual access rights for visualizations and data collection mechanisms were provided. However, it was not possible to specify access rights on different levels of data granularity, which is important for fully supporting distributed projects in which multiple organizations take part.

4.5. Analyze

(G22) Show Value: When analyzing the results of the project control mechanisms applied, it is important to show the value to all affected parties as early as possible. Often, the return on investment of project control mechanisms cannot be measured directly. There are a lot of indirect benefits, related, for instance, to the overall process maturity of an organization. Therefore, the benefits should be analyzed carefully (not only quantitatively, but also qualitatively). Demonstrating the value of project control mechanisms may already start during the execution of the project under control, for instance by demonstrating how data are actually used for effectively controlling the project.

(G23) Conduct Post-Mortem Analysis: After the end of the project, the effectiveness of the applied project control mechanisms should be analyzed. For instance, the complete lists of plan deviations and project risks have to be analyzed. This includes deviations and risks detected by the SPCC as well as deviations and risks that were detected by other means or that were detected after the end of the project. All deviations and risks should be classified as to whether they were detected in time by the SPCC, whether they were detected too late, or whether they were not detected at all. For all deviations and risks that were not detected at all, the measurement plan has to be analyzed with respect to missing goals or other missing parts of the measurement model.

4.6. Package

(G24) Package Experience and Reuse: If measurement is to be established as a means for quantitative project control, it has to be introduced in a sustainable way. On the one hand, this includes that (qualitative and quantitative) experience is used to define and reuse project control mechanisms and to adapt them to the current projects. On the other hand, models (e.g., quality models) that already work for a project need to be generalized and made available to “comparable” projects of the same type. Control mechanisms may look different, or may even not make sense at all, if reused in different contexts. A meta-model for control indicators might contain information on dependencies of control components (e.g., a rule-based

specification including restrictions and dependencies of components). However, this problem is only partly addressed by the Soft-Pit project and needs further research.

5. Conclusion and Future Work

This article provided a general introduction to the field of Software Project Control Centers. A specific goal-oriented control center approach, developed and evaluated as part of the Soft-Pit project, was briefly described. After that, practical guidelines were listed addressing challenges in the field of setting up and introducing software cockpits in industry. Social and organizational issues related to the deployment of software cockpits (such as overcoming acceptance problems) were addressed. For selected guidelines, we illustrated how to overcome those issues based on experience that was systematically gathered in several industrial case studies and research projects. The benefits experienced from following the guidelines include, but are not limited to: being able to identify and reduce risks related to introducing software cockpits, being more efficient in setting up and adapting project controlling mechanisms, allowing for more transparent decision-making regarding project control, being able to achieve higher acceptance of the underlying measurement program within a company, reducing the overhead of data collection, increasing data quality (especially for manually collected data), and, finally, achieving projects that are easier to plan and to control.

Further development and evaluation of the approach will take place in the context of the Soft-Pit project. Important research questions to be tackled from a practitioner's viewpoint include the following aspects:

- Setting up a *holistic control center* that integrates more aspects of engineering-style software development. The starting point for setting up such a control center are usually high-level business goals, from which measurement programs and controlling instruments can be derived systematically. Thus, it would be possible to transparently monitor, assess, and optimize the effects of business strategies performed (such as CMMI-based improvement programs).
- Supporting *distributed multi-organization development projects*. In order to successfully conduct projects that need inter-enterprise collaboration, one crucial success factor is the existence of well-specified and coordinated distributed development processes. In order to guarantee intellectual control over such a process, it is necessary to have efficient management and controlling mechanisms in place. On the level of global multi-domain inter-enterprise development projects, project controlling is aggravated by many factors, such as heterogeneous development processes, incompatible workflows and data, slow technological infrastructure, cultural factors, and lack of trust. The existing software cockpits mainly focus on intra-organizational processes and therefore, the emerging area of globally distributed software projects remains poorly supported.
- In the area of small or medium-size enterprises (SME), technical development processes are often seen as being more important than management processes and are therefore not integrated well into the organizational culture. *Supporting project control proactively* based on quantitative analyses and goal-oriented project control is often missing. Such systems explicitly have to address the specifics of an SME.

6. Acknowledgements

We would like to thank Sonnhild Namingha from Fraunhofer IESE for reviewing a first version of the article. This work was supported in part by the German Federal Ministry of Education and Research (Soft-Pit Project, No. 01ISE07A).

7. References

[1] Agresti, W.; Card, D.; Church, V.: Manager's Handbook for Software Development; SEL 84-101. NASA Goddard Space Flight Center. Greenbelt, Maryland, November 1990.

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

[3] Basili, V.R.; Heidrich, J.; Lindvall, M.; Münch, J.; Regardie, M.; Rombach, D.; Seaman, C.; Trendowicz, A.: GQM+Strategies®: A Comprehensive Methodology for Aligning Business Strategies with Software Measurement In: G. Büren, M. Bundschuh, R. Dumke (ed.): MetriKon 2007, Praxis der Software-Messung, Tagungsbad des DASMA-Software-Metrik-Kongresses, Nov. 15-16, 2007, Kaiserslautern, Germany, pp 253-266.

[4] Ciolkowski, M.; Heidrich, J.; Münch, J.; Simon, F.; Radicke, M.: Evaluating Software Project Control Centers in Industrial Environments; International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Madrid, 2007, pp. 314-323.

[5] Heidrich, J.; Münch, J.; Riddle, W.E.; Rombach, D.: People-Oriented Capture, Display, and Use of Process Information. In: Silvia Teresita Acuña, Natalia Juristo (ed.): New Trends in Software Process Modelling. World Scientific, ISBN 981-256-619-8, Feb. 2006, pp. 121-180.

[6] Heidrich, J.; Münch, J.; Wickenkamp, A.: Usage Scenarios for Measurement-based Project Control; In: Ton Dekkers (ed.): Proceedings of the 3rd Software Measurement European Forum, May 10-12, 2006, Rome, Italy. Smef 2006, pp. 47-60.

[7] Heidrich, J.; Münch, J.: Cost-Efficient Customisation of Software Cockpits by Reusing Configurable Control Components. In: Ton Dekkers (ed.): Proceedings of the 4th Software Measurement European Forum, May 9-11, 2007, Rome, Italy. Smef 2007, pp. 19-32.

[8] 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.

[9] ISO 9126: Software Engineering – Product Quality; Technical Report. ISO/IEC TR 9126. Geneva, 2003.

[10] Kitchenham, B.A.: Software Metrics; Blackwell. Oxford, 1995.

[11] 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.

[12] McGarry, J.; Card, D.; Jones, C.; Layman, B.; Clark, E.; Dean, J.; Hall, F.: Practical Software Measurement – Objective Information for Decision Makers, Addison-Wesley Professional; 1st edition, ISBN 4-320-09741-6, October 15, 2001.

[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] 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.

[16] Soft-Pit Project Homepage (*http://www.soft-pit.de*). Last visited Feb. 27th, 2008.

[17] SQS AG, SQS-Test-Professional, Component Process Performance Management to Monitor Test Quality and Error Rates, http://www.sqs.de/portfolio/tools/tools_ppm.htm, last checked Jan 11, 2006.

[18] 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.

[19] 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.