JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
*J. Softw. Maint. Evol.: Res. Pract.* 2010; **22**:381–405
Published online 28 April 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.458

# Goal-oriented customization of software cockpits

Jens Heidrich*, † and Jürgen Münch

*Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany*

**SUMMARY**

**Software cockpits, also known as Software Project Control Centers, support the management and controlling of software and system development projects and provide means for quantitative, measurement-based project control. Currently, many companies are developing simple control dashboards that are mainly based on Spreadsheet applications. Alternatively, they use solutions providing a fixed set of project control functionalities that cannot be sufficiently customized to their specific needs and goals. Specula is a systematic approach for defining reusable, customizable control components and instantiating them according to different organizational goals and characteristics based on the Quality Improvement Paradigm (QIP) and the Goal Question Metric (GQM) approach. This article gives an overview of the Specula approach, including the basic conceptual model, goal-oriented composition of control centers based on explicitly stated measurement goals and a conceptual architecture supporting the approach. Related approaches are discussed, a practical usage example is given, and evaluation results from using Specula as part of industrial case studies are presented. Copyright © 2010 John Wiley & Sons, Ltd.**

*Received 11 February 2010; Accepted 11 February 2010*

KEY WORDS: Software Project Control Center; project control; measurement; QIP; GQM

## 1. INTRODUCTION

The complexity of software development projects continues to increase. One major reason is the ever-increasing complexity of functional as well as non-functional software requirements (e.g., reliability or time constraints for safety-critical systems). The more complex the requirements, the more people are usually involved in meeting them, which further increases the complexity of controlling and coordinating the project. This, in turn, makes it even harder to develop the system according to plan (i.e., matching time and budget constraints). According to a study by the Standish Group [1], 15% of all software development projects are canceled and 51% are late and over budget. Many software development organizations still lack support for obtaining intellectual control over

*Correspondence to: Jens Heidrich, Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany.
†E-mail: jens.heidrich@iese.fraunhofer.de

Contract/grant sponsor: German Federal Ministry of Education and Research; contract/grant number: 01ISE07A

their software development projects and for determining the performance of their processes and the quality of the produced products. Systematic support for detecting and reacting to critical project states in order to achieve planned goals is often missing.

Companies have started to introduce so-called software cockpits, also known as Software Project Control Centers (SPCC) [2] or Project Management Offices (PMO) [3], for systematic quality assurance and management support. A software cockpit is comparable to an aircraft cockpit, which centrally integrates all relevant information for monitoring and controlling purposes. A project manager can use it to get an overview of the project state and a quality assurance manager can use it to check the quality of the software product. In addition to these primary users of an SPCC, basically any role in a project may profit 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 different 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 general, an important success factor in the software engineering domain is that these solutions are customized to the specific goals, organizational characteristics and needs, as well as the concrete project environment. Specula (Latin for *watchtower*) is an approach for composing project control functionality out of reusable control components [4,5]. It was mainly developed at the Fraunhofer Institute for Experimental Software Engineering (IESE) and makes use of the Quality Improvement Paradigm (QIP) for integrating project control activities into a continuous improvement cycle. Furthermore, the GQM approach [6] is used for explicitly specifying measurement goals for project control. The goals of the Specula approach are as follows:

- *G1*: *Provide an effective SPCC*: The main task of an SPCC is to detect plan deviations and potential project risks, so that a decision maker can initiate appropriate countermeasures, if necessary. The goal is to detect these deviations and risks effectively, that is, as soon as possible.
- *G2*: *Simplify setting up and applying project control*: Currently, project control highly depends on expert knowledge and the tools determine the indicators that have to be used for project control. It is very difficult to set up an SPCC, find the right set of indicators to use, and effectively use the SPCC during project execution. The goal is to facilitate and simplify setting up and applying project control mechanisms.
- *G3*: *Provide an easy-to-extend and easy-to-adapt SPCC*: The control mechanisms used for a specific software development project must be derived in a goal-oriented manner based on explicitly defined measurement goals. Nowadays, the tools determine the mechanisms to be used and the provided functionality is taken as it is. The goal is to provide an easy-to-extend and easy-to-adapt SPCC with respect to the specific needs of an organization and the project that is being controlled.

Section 2 of the article presents the related work in the field of SPCC and key performance indicators for project control. Section 3 introduces the Specula approach exemplified by a usage example, describes the underlying conceptual model and its relationship to goal-oriented measurement, and



DOI: 10.1002/smr

finally presents the basic steps of the methodology for composing control components (encapsulated, packaged techniques for project control) based on explicitly defined measurement goals. Section 4 presents empirical evaluation results based on industrial case studies conducted. The article concludes with a brief summary, a discussion of the utility of the presented approach, and an outlook to future work.

## 2. RELATED WORK

An overview of the state of the art in SPCC can be found in [2]. The scope was defined as generic approaches for online data interpretation and visualization on the basis of past experience. The article presents a reference model for concepts and definitions around SPCCs using five dimensions (purpose, technical, improvement, role, and tool dimension), a characterization scheme, and a classification of essential approaches contributing to this field. In practice, 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 (e.g., deal with product quality only or solely focus on project costs) and are very specific to the company for which they were developed. Most of the existing, rather generic, approaches for control centers 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) [7] 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) [8] has a scripting language for customizing the interpretation and visualization process, but does not provide a generic set of applicable controlling functions. Unlike Provence [9] and PAMPA [10], approaches such as Amadeus [11] and Ginger2 [12] offer a set of purpose-oriented controlling functions with a certain flexibility, but lack a role-oriented approach to data interpretation and visualization.

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. According to McGarry *et al.* [13], a 'good' indicator has to (a) support analysis of the intended information need, (b) support the type of analysis needed, (c) provide the appropriate level of detail, (d) indicate a possible management action, and (e) provide timely information for making decisions and taking action. The concrete indicators that are chosen should be derived in a systematic manner from the project goals [14], making use of, for instance, the Goal Question Metric (GQM) approach. Examples from indicators used in practice can be found in Agresti *et al.* [15]. With respect to controlling project cost, the Earned Value approach provides a set of commonly used indicators and interpretation rules. With respect to product quality, there exists even an ISO standard [16]. However, the concrete usage of the proposed measures depends upon the individual organization. Moreover, there is no unique classification for project control indicators. One quite popular classification of general project management areas is given by the Project Management Body of Knowledge (PMBoK) [3]. The PMBoK distinguishes between nine areas, including project time, cost, and quality management.

The ideas behind GQM and the QIP [6] are well-proven concepts that are widely applied in practice today. An approach based on GQM and QIP to create and maintain enhanced measurement plans, addressing data interpretation and visualization informally, is presented in Differding [17].



DOI: 10.1002/smr

## 3. THE SPECULA APPROACH

Specula is a state-of-the-art approach for project control. It interprets and visualizes collected measurement data in a goal-oriented manner to detect plan deviations. The control functionality provided by Specula depends on the underlying goals with respect to project control. If these goals are explicitly defined, the corresponding functionality is composed of packaged, freely configurable control components. Specula provides four basic components (see [2] and [18]): (1) a logical architecture for implementing software cockpits, (2) a conceptual model formally describing the interfaces between data collection, data interpretation, and data visualization, (3) an implementation of the conceptual model, including a construction kit of control components, and (4) a methodology of how to select control components according to explicitly stated goals and customize the SPCC functionality.

The methodology is based on the QIP and makes use of the GQM approach [6] for specifying measurement goals. QIP is used to implement a project control feedback cycle and make use of experience and knowledge gathered in order to reuse and customize control components. GQM is used to drive the selection process of finding the right control components according to defined goals. Large parts of the approach are supported by a corresponding prototype tool, called Specula Project Support Environment (PSE), which is currently also being used as part of industrial case studies (see Section 5). Specula basically addresses the following roles that make use of the provided functionality:

- *Primary users*: Project manager, quality assurance manager, and controller, who mainly use an SPCC to control different aspects of the software development project and initiate countermeasures in case of deviations and risks.
- *Secondary users*: Developers and technical staff who use an SPCC to enter measurement data as well as to detect root causes for deviations and risks.
- *Administrators*: Administrators who install and maintain an SPCC.
- *Measurement experts*: Experts who define measurement goals, support derivation of control components, and help to customize and effectively use the SPCC.

Section 3.1 gives a brief overview of the conceptual model upon which Specula is built. Section 3.2 addresses the connection of the conceptual model to goal-oriented measurement, Section 3.3 illustrates the logical architecture for implementing a control center tool supporting the approach, and Section 3.3 provides a brief overview of all steps necessary to apply the Specula approach as a whole.

### 3.1. Cockpit concepts

The conceptual model of the Specula approach formalizes the process of collecting, interpreting, and visualizing measurement data for software project control. The derived structure for operationally controlling a development project is called a *Visualization Catena* (*VC*) [4], which defines components for automatically and manually collecting measurement data, processing and interpreting these data, and finally visualizing the processed and interpreted data. The processing and interpretation of collected measurement data are usually related to a special measurement purpose, such as analyzing effort deviations or guiding a project manager. A set of techniques and methods (from



DOI: 10.1002/smr

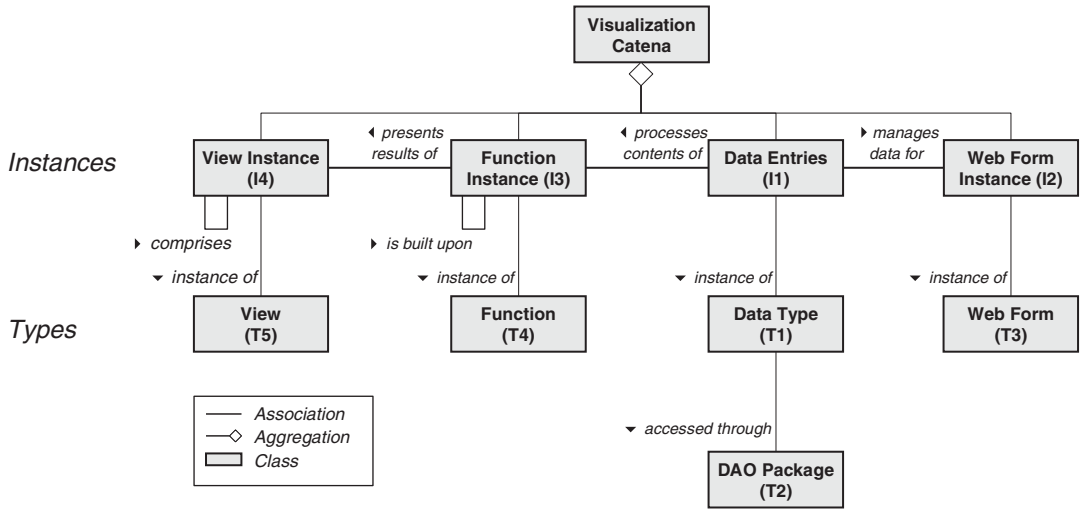


Figure 1. Overview of the elements of the conceptual model.

the repository of control components) is used by the VC for covering the specified measurement purpose. The visualization and presentation of the processed and collected measurement data is related to roles of the project that profit from using the data. The VC creates a set of custom-made controlling views, which presents the data according to the interests of the specified role, such as a high-level controlling view for a project manager and a detailed view of the defects found for a quality assurance manager. The whole VC has to be adapted in accordance with the context characteristics and the organizational environment of the software development project currently being controlled.

Figure 1 gives an overview of all VC components and their corresponding types. Specula distinguishes between the following five components on the type level from which a concrete VC is instantiated for a certain project:

- (*T1*) *Data types*: describe the structure of incoming data and data that are further processed by the VC. For instance, a time series (a sequence of time stamp and corresponding value pairs) or a project plan (a hierarchical set of activities having a start and end date and an effort baseline) could be logical data types that could either be directly read-in by the system or be the output of a data processing function.
- (*T2*) *Data access object packages*: describe the different ways that concrete data types may be accessed. For instance, an XML package contains data access objects for reading data (having a certain data type) from an XML file, writing data to an XML file, or changing the contents of an XML file. A special package may be used, for instance, to automatically connect to an effort tracking system or bug tracking database. A data access object contains data type-specific parameters to access the data repositories.
- (*T3*) *Web forms*: describe a concrete way of managing measurement data manually, involving user interaction. A web form manages a concrete data type. For instance, new data may be added, and existing data may be changed or completely removed. A web form also refers to



DOI: 10.1002/smr

other data types that are needed as input. For instance, in order to enter effort data manually, one needs the concrete activities of the project for which the effort is tracked. Web forms are needed if the data cannot be automatically retrieved from an external data source.

- (*T4*) *Functions*: represent a packaged control technique or method, which is used to process incoming data (such as Earned Value Analysis, Milestone Trend Analysis, or Tolerance Range Checking). A function needs different data types as input, produces data of certain data types as output, and may be adapted to a concrete context through a set of parameters.
- (*T5*) *Views*: represent a certain way of presenting data, such as drawing a two-dimensional diagram or just a table with a certain number of rows and columns. A view visualizes different data types and may refer to other views to create a hierarchy of views. The latter may, for instance, be used to create a view for a certain project role consisting of a set of sub-views.

In addition, the following components are distinguished on the instances level:

- (*I1*) *Data entries*: instantiate data types and represent the concrete content of measurement data that are processed by the SPCC. We basically distinguish between external and internal data. External data must be read-in or imported from an external location, or manually entered into the system. Each external data object has to be specified explicitly by a data entry containing, for instance, the start and end times and the interval at which the data should be collected. In addition, the data access object package that should be used to access the external data has to be specified. Internal data are the outcome of functions. They are implicitly specified by the function producing the corresponding data type as output and therefore need no explicit specification and representation as data entry. External as well as internal data may be used as input for instances of functions or views if their corresponding data types are compatible.
- (*I2*) *Web form instances*: provide web-based forms for manually managing measurement data for data entries. All mandatory input data type slots of the instantiated web form have to be filled with concrete data entries and all mandatory parameters have to be set accordingly.
- (*I3*) *Function instances*: apply the instantiated function to a certain set of data entries filling the mandatory input slots of the function. A function instance processes (external and internal) data and produces output data, which could be further processed by other function instances or visualized by view instances. All mandatory function parameters have to be set accordingly.
- (*I4*) *View instances*: apply the instantiated view to a certain set of data entries filling the corresponding mandatory data type slots of the view. A view instance may refer to other view instances in order to build up a hierarchy of views.

Each component of a VC and its corresponding type contains explicitly specified checks that may be used to test whether the specification is complete and consistent, whether data are read-in correctly, whether function instances can be computed accurately, and whether view instances can be created successfully. A VC consists of a set of data entries, each having exactly one active data access object for accessing incoming data, a set of web form instances for managing the defined data entries, a set of function instances for processing externally collected and internally processed data, and finally, a set of view instances for visualizing the processing results. A formal specification of all the components may be found in [19].



DOI: 10.1002/smr

### 3.2. Mapping cockpit concepts to GQM

For a goal-oriented selection of control components, a structured approach is needed that describes how to systematically derive control components from project goals and characteristics. GQM provides a template for defining measurement goals, systematically derives questions that help to make statements about the goals, and finally derives metrics to help answer the stated questions. In order to complete such a measurement plan for a concrete project, each metric can be further described by a data collection specification (DCS) basically making statements about who or which tool has to collect the measurement data at which point in time of the project from which data source. In [5], usage scenarios on how to derive a GQM plan from a control goal and how to define a VC that is consistent with the defined goals are described.

Figure 2 presents an overview of all the relationships between a GQM plan, its DCS, and a visualization catena (cf. [18]). On the left side, one can see the components of the VC. On the right side, one can see the structure of a GQM model and a corresponding DCS.

- *Data entries*: collect measurement data for GQM metrics according to the DCS. If the data have to be collected manually, a *web form instance* is used to implement the DCS in addition. For instance, if the DCS states that the start and end dates of an activity shall be collected from an MS Project file, a corresponding data entry is defined and a web form instance implements importing the project plan from the file.
- *Function instances*: compute metric values if a metric has to be computed from other metrics. For instance, if a cost performance index is computed for an Earned Value Analysis, the budgeted costs of work performed and the actual costs of work performed are needed. A function instance could also compute answers to GQM questions by taking into account

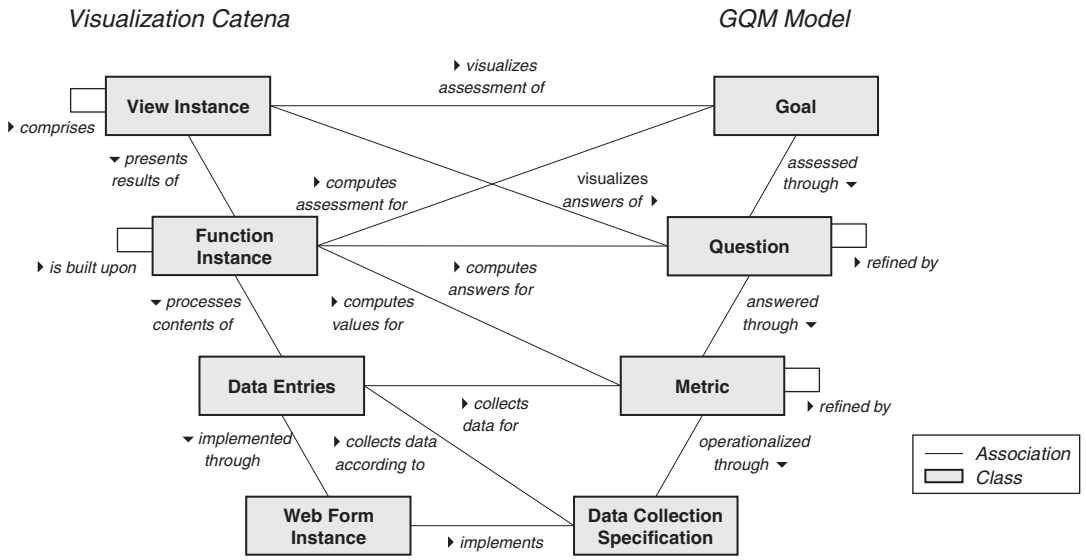


Figure 2. Mapping the conceptual model to GQM.



DOI: 10.1002/smr

all the metrics assigned to the question and applying an interpretation model to all the metric values. In analogy, a function instance could assess the attainment of a GQM goal by assessing the answers of all assigned questions using an interpretation model.

- *View instances*: visualize the answers to GQM questions. A chart is produced or tables are displayed illustrating the metric results of the corresponding questions and the interpretation model used to answer the question. For instance, the cost performance and schedule performance indexes could be visualized as a line chart in which good and bad index values are marked accordingly. A view instance could also visualize the assessment of the GQM goal.

### 3.3. Conceptual architecture

Figure 3 illustrates the conceptual modules of the Specula approach. The conceptual architecture is a basis for actually implementing control center tools following the Specula approach. Large parts of the conceptual architecture are already implemented by a corresponding tool that was used in industrial case studies for evaluating the underlying approach. Three basic modules are distinguished:

- The *planning module*: is responsible for selecting and adapting the control components according to project goals and characteristics and defined measurement (control) goals. It is possible to include past experience (e.g., effort baselines, thresholds) in the selection and adaptation process. This experience is stored in an experience base. A VC is created, which formally describes how to collect, interpret, and visualize measurement data. The set of reusable control components from which the VC is instantiated basically consists 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.

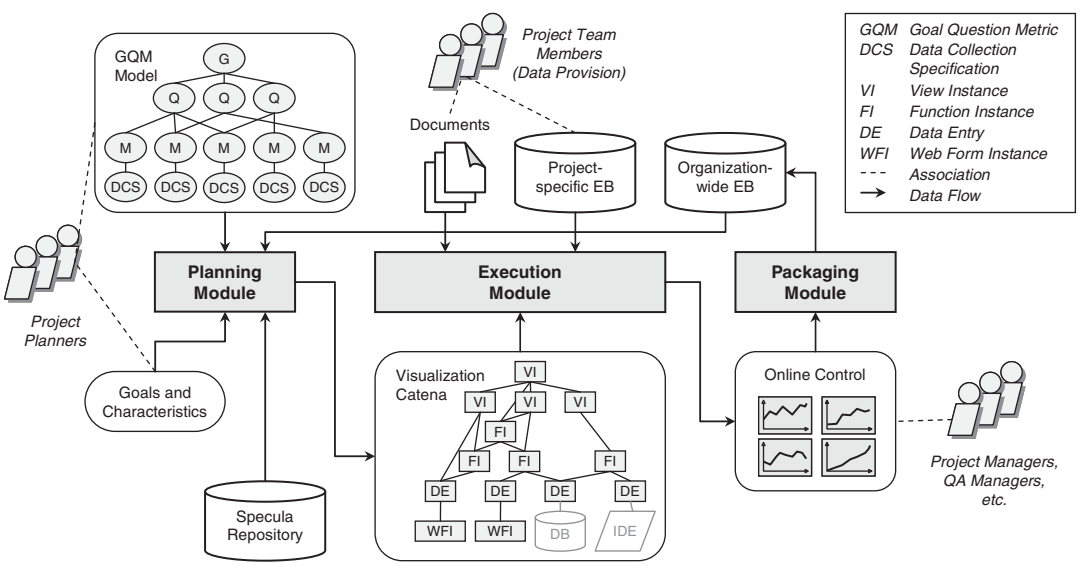


Figure 3. Conceptual Specula architecture.



DOI: 10.1002/smr

- The *execution module*: collects measurement data during project performance and interprets and visualizes them according to the VC specification. Measurement data can be retrieved (semi-)automatically from project repositories using data entries or manually from data collection forms and (semi-)formal documents using web form instances. For instance, project plan information is imported from an MS Project file, including a list of all project activities, start and end dates, and the planned effort for each activity. The collected data are processed by function instances in order to allow for data aggregation, data analysis, and data interpretation. For instance, the actual effort of project team members is aggregated along the activities and compared to the planned effort using a certain tolerance range in order to detect plan deviations. Finally, the processing results are visualized for different stakeholders of the project and charts and tables are produced to allow for online project control. For instance, a project manager is appointed to plan deviations using a bar chart displaying all project activities together with their planned and actual effort. If the actual effort exceeds a tolerance range, the corresponding activity is marked accordingly so that the project manager can initiate countermeasures.
- A *packaging module*: collects feedback from project stakeholders about the application of the control mechanisms and stores it in an Experience Base (e.g., whether a baseline worked, whether all plan deviations were detected, or whether retaliatory actions had a measurable effect).

Using these modules, the Specula approach is able to specify a whole family of project control centers. A concrete control center is composed on-the-fly based on explicitly specified project goals and characteristics. The Specula PSE is a tool that implements large parts of the conceptual architecture. It completely automates the execution module as well as parts of the planning module (defining control components and creating a VC). The process of deriving a VC from a GQM plan, project goals, and characteristics is currently not automated by the tool and must be performed manually using a set of concrete guidelines for ensuring completeness and consistency of the derivation process. Packaging is currently only supported by the tool as far as maintenance of control components from the Specula repository is concerned (e.g., creating new components, changing existing ones, and adapting parameter settings).

### 3.4. Composing control components

The Specula methodology for goal-oriented composition of SPCC is largely based on the QIP. It distinguishes six different phases in analogy with the QIP. Each of the six phases consists of a couple of more detailed steps, which are described as part of separate subsections. Following each step, an example will be used for illustrating it from a practical point of view.

#### *3.4.1. Phase I: characterize control environment*

First, project stakeholders characterize the environment in which project control shall be applied to set up a corresponding measurement program.

- *Describe the project context*: Important characteristics for setting up project control mechanisms have to be defined.



DOI: 10.1002/smr

- *Discuss the overall organization*: Organizational characteristics have to be clarified. This includes roles and responsibilities, potential stakeholders—such as managers of the organization, project managers, quality assurance manager, developers—and team organization.

*Example.* A practical course called 'Master Project Open Source' was conducted at the University of Kaiserslautern. There, the Specula project control approach was applied. The aim was to develop mobile services for creating a virtual office of the future. There were 17 team members. The project manager and quality assurance manager were supposed to use an SPCC to control different aspects of the project. In addition, an administrator (not part of the project team) participated, who was familiar with the SPCC tool.

### *3.4.2. Phase II*: *set control goals*

Then, measurement goals for project control are defined and metrics are derived that determine what kind of data to collect.

- *Elicit control goals*: The Specula approach makes use of GQM for defining measurement goals in a structured manner. GQM already provides a systematic approach for defining measurement goals, systematically deriving questions that help to make statements about the goals, and finally deriving metrics in order to help answer the stated questions.
- *Clarify relationships to higher-level goals*: The relationships with higher level goals have to be modeled. For this purpose, all measurement goals are connected to higher-level software and business goals using the GQM$^+$Strategies® approach [20].
- *Derive indicators*: Based on the measurement goals defined for project control, questions and metrics have to be derived using GQM.
- *Define GQM model*: A GQM model is created containing the project-specific measurement goals, corresponding questions that make statements about achieving goals, and metrics that support answering the questions.

*Example.* For the practical course, a measurement expert conducted structured interviews with the project manager and quality assurance manager to retrieve the measurement goals with respect to project control that were to be achieved. The following goals were retrieved:

1. Analyze the project plan for the purpose of evaluating effort plan adherence from the point of view of the project manager.
2. Analyze the project plan for the purpose of evaluating schedule adherence from the point of view of the project manager.
3. Analyze the project plan for the purpose of monitoring effort tracking regularity from the point of view of the project manager.
4. Analyze the project plan for the purpose of monitoring the consistency of the plan from the point of view of the project manager.
5. Analyze the source code for the purpose of evaluating the quality from the point of view of the quality assurance manager.
6. Analyze the defect detection activities for the purpose of evaluating their efficiency from the point of view of the quality assurance manager.



DOI: 10.1002/smr

### 3.4.3. *Phase III: goal-oriented composition*

Next, a VC is composed based on the defined goals to provide online feedback on the basis of the data collected during project execution. More details about this process can be found in Heidrich and Munch [18].

- *Derive measurement plan*: A comprehensive measurement plan has to be derived based on the GQM model, including a DCS.
- *Define interpretation models*: Interpretation models are used to aggregate measurement data to answer a GQM question or make a statement about achieving a GQM goal.
- *Derive data entries and web form instances*: Next, matching data types are identified based on the metric definition, the object to be measured, and the quality attribute. For each simple metric (which is not computed from other metrics), the data type is instantiated and a corresponding data entry is created. The DCS is used to determine the start time, the end time, and the interval when the data should be collected. If the metric has to be collected manually, a web form is identified based on the data source and the instantiated web form is attached to the data entry.
- *Derive function instances for complex metrics*: For each complex metric (which is computed from other metrics), a function is identified that is able to compute the metric based on the metric definition, the object to be measured, and the quality attribute. The identified functions are instantiated by first filling all input data slots with data entries or results of other function instances. Then, the function instances are parameterized according to the metric definition.
- *Derive function instances for GQM questions*: If an interpretation model is described in the GQM plan that defines how to formally answer a question, a function implementing this model is identified based on the object and quality attribute addressed to compute the answers to the question. The functions are instantiated by filling all input data slots with data entries or results of other function instances assigned to the question. The function instances are parameterized according to the interpretation model.
- *Derive view instances for GQM questions*: The answers to the question are visualized by identifying a set of views based on the kind of answers to the question and the data visualization specifications of the measurement plan (if any). The identified views are instantiated by filling all input data slots with data entries or results of function instances assigned to the question. The view instances are parameterized according to the data presented (e.g., title and axis description, size, and color).
- *Derive function instances for GQM goals*: If an interpretation model is described in the GQM plan that defines how to formally assess goal attainment, a function implementing this model is identified and instantiated based on the object and quality focus addressed to attain the measurement goal.
- *Derive view instances for GQM goals*: Goal attainment is visualized by identifying and instantiating a set of views based on the kind of assessment of the goal and the data visualization specifications of the measurement plan (if any).
- *Check consistency and completeness*: After defining the whole VC for controlling the project, the consistency and completeness of the mapping process are checked.



DOI: 10.1002/smr

- *Configure SPCC*: If the VC has been defined and checked, it has to be transferred to a corresponding tool (Specula tool prototype).
- *Provide training*: Training is provided for all SPCC users in order to guarantee the effective usage of the SPCC.

*Example.* For all GQM goals defined in phase II, a corresponding VC has to be created. For example, if the goal is to evaluate effort plan adherence (goal #1), the corresponding control components can be selected as follows. Figure 4 presents the GQM model for this goal on the left side and the corresponding excerpt of the resulting VC on the right side. The left side shows the GQM plan to be implemented by an SPCC. According to the information specified in the GQM plan, components are identified from a reuse repository and instantiated to create a VC. In the example, the one and only question asked for goal #1 was about absolute effort deviation per activity. A complex metric (that is, a metric that is made up of other metrics) defined the deviation as the amount that an actual effort value is above an effort baseline. Three simple metrics were consequently defined and operationalized by corresponding DCSs. The baseline should be extracted from a project plan stored in an MS project file, hence a corresponding web form collecting project plan information and data types representing the project activities and the effort baseline were instantiated. The actual effort data should be extracted from the company-wide effort tracking system including effort per person and activity. A data type was instantiated that accesses the tracking system using a corresponding data access object. A function was applied to aggregate the effort data for each activity across all persons. In order to compute the complex metric 'effort plan deviation', a tolerance range checking function was applied that computes the deviation accordingly. Finally, a view was instantiated in

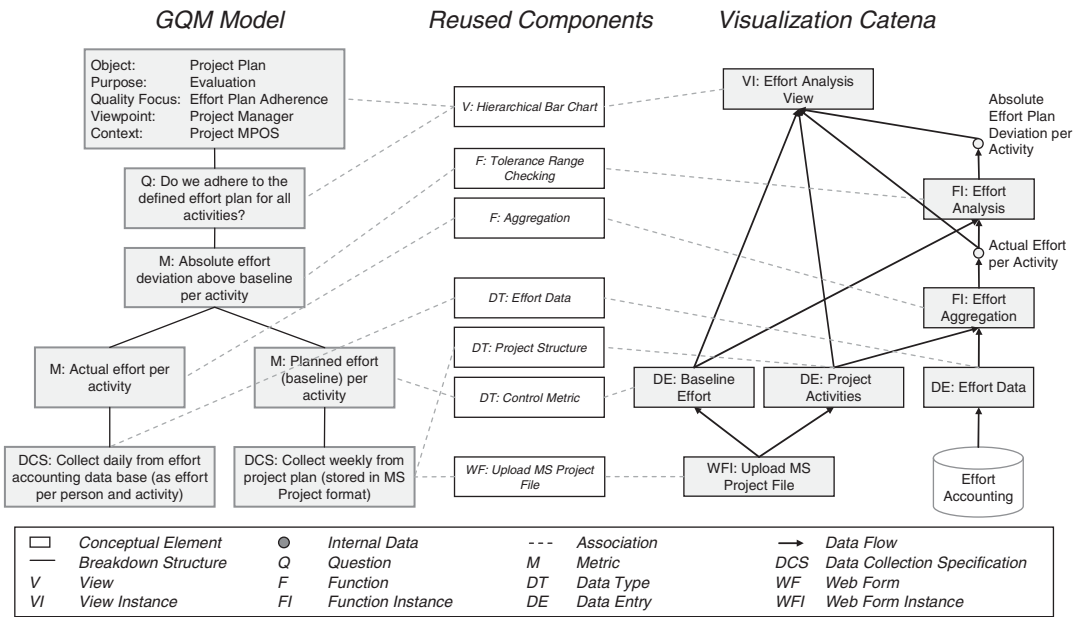


Figure 4. Composing the VC from reusable components.



DOI: 10.1002/smr

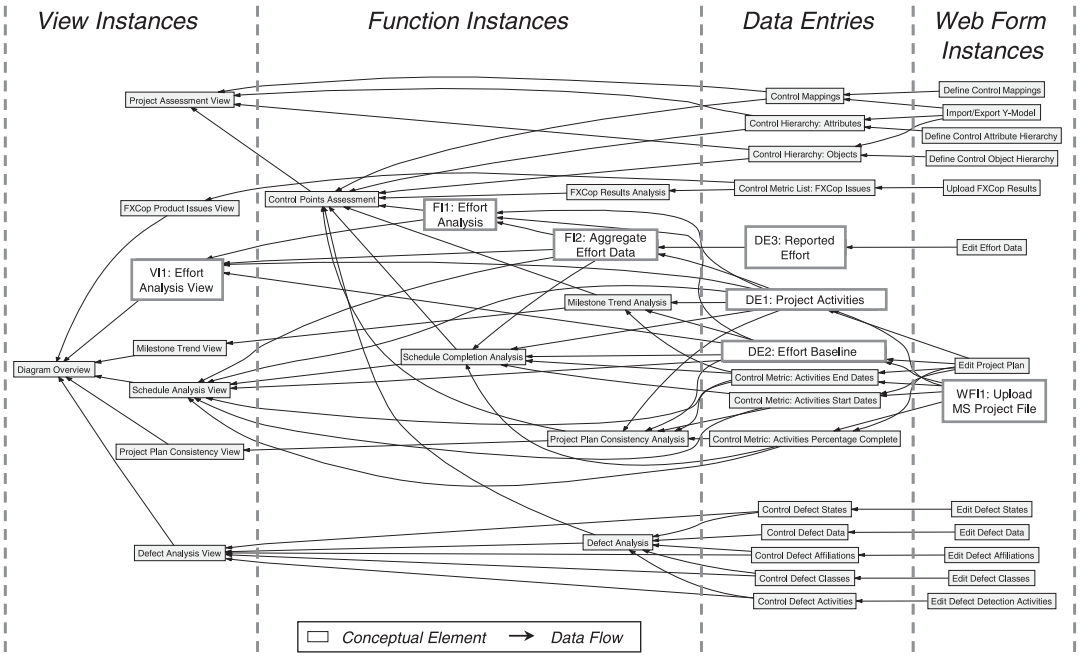


Figure 5. Example visualization catena.

order to graphically display the results of the assigned function instances and data entries. Figure 5 presents the complete VC that was derived for all goals defined as put out by the Specula PSE tool. One can see all input and output data of all control components used for constructing the VC. 13 web form instances provide input for 15 data entries, which are processed by eight function instances, and visualized by eight view instances. As can be seen, the logical dependency of components is quite high, even for a limited number of control components. The excerpts of the VC discussed above are highlighted accordingly.

### *3.4.4. Phase IV*: *execute project control mechanisms*

Once the VC is specified, a set of role-oriented views is generated by the SPCC for controlling the project based on the specified VC. If a plan deviation (e.g., schedule or milestone delay) or project risk (e.g., product quality issues) is detected, its root cause must be determined and the control mechanisms have to be adapted accordingly.

- *Perform data collection*: The SPCC users have to perform data collection activities according to the measurement plan defined.
- *Use control views for GQM questions*: The SPCC users have to use the view instances offered to get answers for the GQM questions of their GQM model.



DOI: 10.1002/smr

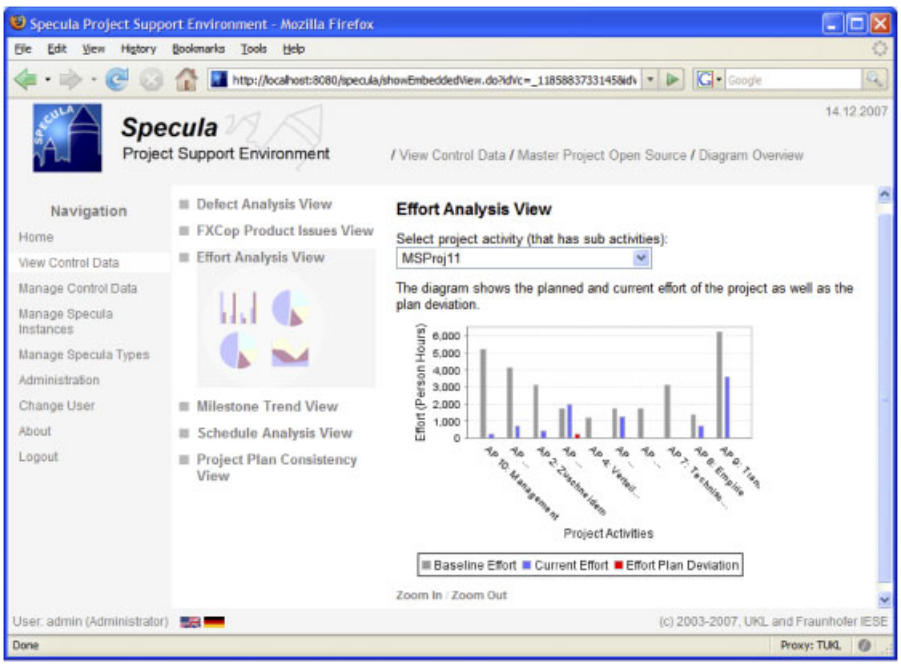


Figure 6. User interface of the Specula prototype tool.

- *Use control views for GQM goals*: The SPCC users have to use the view instances offered to get a general answer with respect to achieving a certain goal of the GQM models.
- *Check SPCC functionality*: The SPCC users should check the correct functionality of the Project Control Center regularly.

*Example.* Figure 6 presents a visualization of the effort analysis view generated by the Specula prototype tool (as implemented by the instantiated control component 'Hierarchical Bar Chart'). On the left-hand side, one can see the overall navigation bar. The menu close to the navigation bar displays all available views for controlling the project. On the right-hand side, one can see the selected view for analyzing effort data. During the execution of the project, the team members entered their effort data using the corresponding Specula web form. The project manager regularly updated the project plan using MS Project and imported the plan into the SPCC. The quality assurance manager used a static code analysis tool to analyze code quality and imported a corresponding report into the SPCC.

### 3.4.5. *Phase V: analyze results*

After project completion, the resulting VC has 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 have been detected too late or not all have to be determined.



DOI: 10.1002/smr

- *Analyze plan deviations and project risks*: The complete lists of plan deviations and project risks have to be analyzed after the end of the project.
- *Analyze measurement plan*: 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 GQM models.
- *Analyze interpretation models*: For all deviations and risks that were not detected at all or that were detected too late, the interpretation models have to be checked to see whether they work as intended or whether metrics or answers to questions need to be interpreted in a different way.
- *Analyze visualization catena*: For all deviations and risks that were detected too late, those components of the VC that helped in detecting them have to be analyzed to see whether they can be improved to support earlier detection in future projects.

*Example.* In our example, general deviations from the effort baseline were detected, including, but not limited to, the observation that the requirements phase took a lot more effort than planned. If we assume that the plan deviation was detected on time and the project manager updated the project plan accordingly, no further actions have to be taken. On the other hand, although no negative milestone trend was detected at all by the control center, an important milestone was missed. If the deviation was supposed to have been detected by the control center, the corresponding control component would have to be analyzed and adapted accordingly.

#### *3.4.6. Phase VI: package results*

The analysis results of the VC that was applied may be used as a basis for defining and improving the control activities for future projects (e.g., selecting the right control techniques and data visualizations, choosing the right parameters for controlling the project).

*Example.* If we assume that the control component for detecting milestone trends used a wrong parameter setting, it will have to be adapted for future use in subsequent projects.

## 4. EMPIRICAL EVALUATION

Large parts of the evaluation of the Specula approach were conducted in the context of several industrial case studies as part of the Soft-Pit research project funded by the German Federal Ministry of Education and Research (see http://www.soft-pit.de). The project focused on getting experience and methodological support for operationally introducing control centers into companies and projects. One important topic of this project was 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 a control center. The aim of Soft-Pit was to develop a holistic project control center that integrates information from different data sources in a goal-oriented manner. The project included performing several industrial case studies with German companies from different domains, in which the Specula project control center and its deployment were evaluated together with other parts of a more comprehensive Soft-Pit



DOI: 10.1002/smr

methodology, which is beyond the scope of this article. The general hypotheses evaluated in the case studies were as follows:

- *H1*: The Specula approach will detect critical plan deviations and project risks earlier than the traditional project control approach used by case study projects. Moreover, it will detect more plan deviations and project risks than the traditional project control approach. H1, therefore, supports provision of an effective SPCC (G1).
- *H2*: The Specula approach will be perceived as useful and easy to use according to the Technology Acceptance Model (TAM). H2 mainly supports simplifying the setup and application of project control mechanisms (G2), because the usefulness and ease of use of a technology are key factors when adopting a new technology.
- *H3*: The Specula approach will allow for a cost-efficient SPCC setup and application phase; that is, the overall costs for setting up and applying the project control mechanisms defined (excluding implementation efforts) will be within a reasonable order of magnitude (for the case study projects around 10%) of the overall effort spent on the project. H3 mainly supports provision of an easy-to-extend and easy-to-adapt SPCC (G3).

The Soft-Pit project was organized into three iterations focusing on different controlling aspects.

- The goal of the *first iteration* was to show the principal applicability of the control center approach in an industrial environment.
- The goal of the *second iteration* of case studies was to broaden the scope of the control center with respect to the addressed measurement goals as well as to provide more support in customizing the SPCC to the specific needs of the case study projects.
- The goal of the *third iteration* was to provide an aggregated view of the overall project status summarizing different kinds of information into one holistic representation.

This section will briefly give an overview of the design and execution of the industrial case studies, highlight some interesting results, and discuss threats to validity. A more comprehensive discussion including some statistical analyses can be found in Ciolkowski *et al.* [21,22].

### 4.1. Case study design and execution

The contexts in which the control center was applied differed quite a lot depending on the case study provider. They included small and medium-sized companies as well as a large organization from different domains. The case studies ran between one and three months, during which the Specula PSE tool was used for controlling real development projects of the case study providers.

The case study projects of organization O1 were in the area of web application development. O1 used the control center for controlling three projects in iteration 1, two in iteration 2, and another two in iteration 3. The organization already had controlling mechanisms in place (as part of an integrated project management tool) and used the control center as an addition to the already existing control functionality.

The case study projects of organization O2 were in the area of information system development. O2 used the control center for controlling enhancement activities of their core product over the three iterations. The company used the control center for proving an integrated view mainly on the already collected project data.



DOI: 10.1002/smr

The case study projects of organization O3 were in the area of web application development. O3 controlled one project each in iteration 1 and 2 and two in iteration 3 using the control center. The company also had some control mechanisms in place and used the control center for extending those mechanisms and proving an integrated view.

For each case study project, a couple of control goals were defined. Overall, the following control goals were addressed over all three iterations:

- *Effort plan deviation*: Determines whether the actual effort exceeds the planned effort as specified in the project plan.
- *Schedule completion*: Determines whether an activity consumes its effort and progresses according to the schedule.
- *Milestone plan adherence*: Determines whether the milestones will most probably be met.
- *Project plan consistency*: Determines whether the quality of the project plan is sufficient with respect to consistency (e.g., whether the start date of a sub-activity is later than or equal to the start date of the activity containing the sub-activity).
- *Project performance/efficiency*: Determines whether the project performance is sufficient with respect to cost and schedule (using the Earned Value approach).
- *Code quality*: Determines whether the quality of the source code is sufficient with respect to a set of coding guidelines and quality indicators.
- *Effort tracking consistency*: Determines whether the team members' actual effort was tracked regularly (so that up-to-date information can be provided for project control).
- *Defect density*: Determines whether certain parts of the code are more error-prone than others.

Note that not all control goals were addressed during all iterations and not all control goals of one iteration were controlled by all case study projects. In all cases, a subset was chosen that fit the context and characteristics of the iteration and the case study project. The goals were broken down into concrete indicators and metrics. After that, corresponding control mechanisms were selected in order to explicitly specify how to interpret and visualize the gathered measurement data according to the specified goals. For each goal, a corresponding visualization (called view instance) was provided, displaying all related measurement data and allowing for drill-down, abstraction, and filtering of data. In addition, an aggregated project status view was provided in order to summarize the state of the project according to the defined control goals.

For a previously defined evaluation period, the control center was used for each case study project. After the end of the evaluation period, a questionnaire was used to gain feedback from all participants. The participants of the questionnaire included all SPCC users as well as the case study coaches (supporting SPCC users) and all administrative staff, who were mainly responsible for customizing the control center to the specific organizations and projects. The focus of the evaluation was on setting up a control center based on explicitly defined goals (steps I–III of the methodology) and applying the tailor-made control center for detecting plan deviations and project risks (step IV of the methodology), called *setup phase* and *application phase*, respectively. Overall, the following constructs were evaluated:

- *Perceived usefulness*: is defined as the degree to which a person believes that a control view provides support for effective project control. *Perceived ease of use* is defined as the degree to which a person believes that using a particular system would be free of effort. The underlying model is based on the TAM [23], which is an information systems theory that models how



DOI: 10.1002/smr

Table I. Detailed overview of questionnaire participants.

| | Iteration 1 | Iteration 2 | Iteration 3 |
|---|---|---|---|
| *Data collection sheet* | | | |
| Coaches | — | 4 | 3 |
| Administrators | — | 4 | 3 |
| Primary users (e.g., PM) | — | 4 | 9 |
| Secondary users (e.g., Developers) | — | 7 | 9 |
| Number of responses | — | 19 | 22 |
| *Evaluation questionnaire* | | | |
| Coaches | — | 2 | 3 |
| Administrators | — | 2 | 4 |
| Primary users (e.g., PM) | 6 | 8 | 10 |
| Secondary users (e.g., Developers) | 2 | 11 | 9 |
| Number of responses | 8 | 23 | 24 |

users come to accept and use a technology. Perceived usefulness and ease of use are each measured via a set of questions. Each of the questions is rated on a five-point Likert scale in our case. The score for the aggregate metrics perceived usefulness and ease of use is defined as the average score of each answer on a scale from 1 (perceived as not useful/easy to use) to 5 (perceived as useful/easy to use). Perceived usefulness and ease of use were measured separately for the setup phase and the application phase of the control center.

- *Detected plan deviations and project risks*: is defined as the set of all plan deviations and project risks that occurred during the project with respect to the specific areas a certain primary or secondary SPCC user is working in. Note that this construct was only evaluated in iterations 2 and 3.
- *Effort spent on setting up and applying an SPCC*: is defined as the overall effort related to setting up and applying an SPCC spent by primary and secondary users as well as administrators and coaches. This construct, too, was only evaluated in iterations 2 and 3.

A questionnaire was sent out to all participants of the case study after the start of the evaluation period to document detected plan deviations and project risks and track the effort (data collection sheet). At the end, another questionnaire was sent out in order to evaluate the perceived usefulness and ease of use of the control center (evaluation questionnaire). Table I gives an overview of the number of responses to the two questionnaires and analyzes the roles they had in the projects. Note that one participant could have more than one role in the project.

### 4.2. Case study analysis and interpretation

In this section, we will highlight some of the evaluation results of the three Soft-Pit iterations.

*Usefulness and ease of use* (*H2*): Table II and Figure 7 give an overview of the average perceived usefulness and ease of use of the control center's setup and application phase and their distribution[‡].

[‡]In contrast to the results of the first iteration presented in Ciolkowski *et al.* [21], one case study project and the corresponding questionnaires had been removed form the analyses presented here, because it did not apply the Specula approach as presented in Section 3.



DOI: 10.1002/smr

Table II. Detailed results of perceived usefulness and ease of use.

| | Application | | | Setup | | |
|---|---|---|---|---|---|---|
| | Iteration 1 (A-I1) | Iteration 2 (A-I2) | Iteration 3 (A-I3) | Iteration 1 (S-I1) | Iteration 2 (S-I2) | Iteration 3 (S-I3) |
| *Usefulness* | | | | | | |
| Mean | 3.46 | 3.32 | 4.05 | 3.38 | 3.94 | 4.33 |
| Standard deviation | 0.76 | 0.83 | 0.59 | 0.30 | 0.68 | 0.69 |
| No. of valid answers | 7 | 17 | 19 | 7 | 6 | 8 |
| *Ease of use* | | | | | | |
| Mean | 3.21 | 3.71 | 3.40 | 2.71 | 3.67 | 3.14 |
| Standard deviation | 1.19 | 0.50 | 0.75 | 0.57 | 0.58 | 0.94 |
| No. of valid answers | 7 | 18 | 20 | 7 | 6 | 7 |

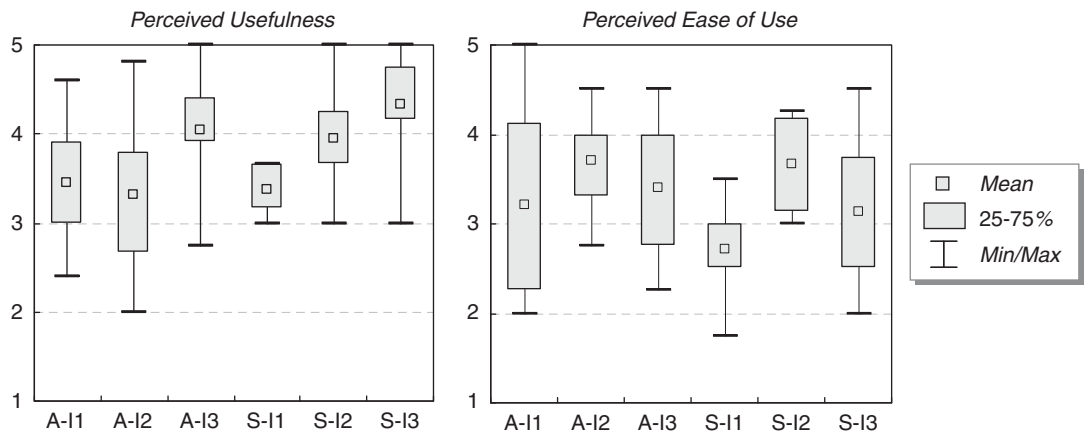


Figure 7. Perceived usefulness and ease of use.

A statistical analysis of the first two iterations containing a more in-depth analysis can be found in Ciolkowski *et al.* [21,22], respectively. In general, participants perceived the usefulness and ease of use of the Specula control center as positive (i.e., an average score above 3), with the exception of the perceived ease of use of the setup phase during the first iteration. As can be seen, the results for ease of use are not as positive as the results for usefulness. This is not a surprising result given the fact that a prototype was used during the case studies. All users received basic training in using the control center, but depending on their familiarity with such tools, the results varied. The general usefulness and ease of use also varied across the different case study providers depending on the state of the practice before introduction of the Soft-Pit control center solution. As can be observed, the usefulness of the application phase decreased from the first to the second iteration. This can be explained by the broadening of the scope of the control center from the first to the second iteration. During the first iteration, only three of the control goals mentioned above were actually controlled (depending on the concrete case study, even less). During the second iteration, at most seven goals



DOI: 10.1002/smr

Table III. Analysis of detected plan deviations and project risks.

| Property | Iteration 2 | Iteration 3 |
|---|---|---|
| *Reported deviations/risks* | | |
| Overall number | 18 | 26 |
| Found by the control center | 18 | 26 |
| Potentially found by traditional approach | 15 | 16 |
| *Detected deviations/Risks per diagram type* | | |
| (a) Aggregated project status view | 0 | 8 |
| (b) Code quality view | 8 | 5 |
| (c) Effort consumption bar view | 1 | 4 |
| (d) Effort consumption hierarchical view | 4 | 5 |
| (e) Earned value view | 3 | 0 |
| (f) Milestone trend view | 2 | 5 |
| (g) Schedule completion view | 6 | 12 |
| (h) Project plan consistency view | 0 | 2 |
| (i) Defect density view | Not available | 1 |
| *Alternative detection using traditional approach* | | |
| Not possible | 3 | 10 |
| Possible with delay | 12 | 4 |
| Possible without delay | 3 | 12 |
| *Results* | | |
| Deviations/risks found earlier | 12/15=80% | 4/16=25% |
| Additionally found deviations/risks | 3/15=20% | 10/16=63% |

were addressed, and during the third one, at most eight goals were controlled. Whereas the three goals of iteration 1 were quite basic goals, the goals of iterations 2 and 3 were more advanced goals, which were not perceived as useful as the goals in iteration 1 when using them for the first time. We continuously improved the method for setting up the control center and provided concrete guidelines for the case study providers (and their coaches) on how to perform concrete tasks. As a consequence, the usefulness increased continuously over the three iterations (see Table II).

*Detected plan deviations and project risks* (*H1*): Table III shows an overview of the detected plan deviations and project risks for iterations 2 and 3 (this aspect was not evaluated in iteration 1). Overall, 44 deviations and risks were detected during the two iterations (18 in the second and 26 in the third). All deviations and risks were detected by the SPCC. Potentially, the approach used for controlling the project before introducing the control center would have detected 15 deviations and risks in iterations 2 and 16 in iteration 3. Sixteen deviations and risks were detected earlier than it would have been possible using the traditional approach to project control (12 in the second and 4 in the third) and 13 would most probably not have been detected at all (3 in the second and 10 in the third). The deviations and risks were rated on a scale from 1 (not critical at all) to 5 (most critical). Large parts of the detected deviations and risks were rated higher than or equal to 4 (13 in the second iteration and 11 in the third iteration). That is, they would have had a strong influence on the future course of the project, had they not been detected. As can be seen, the number of detected plan deviations and project risks depends on the concrete visualizations used. That is, it depends on the concrete goal that was controlled using a certain visualization. In our cases, most deviations and risks were detected using the code quality view and the schedule



DOI: 10.1002/smr

Table IV. Effort analysis.

| Property | Iteration 2 | Iteration 3 |
|---|---|---|
| *Estimated effort of an SPCC (per month)* | | |
| 1 Coach (person-hours) | 6.0 | 15.4 |
| 1 Administrator (person-hour) | 34.2 | 27.8 |
| 1 PM/1 QA Manager (person-hour) | 35.6 | 48.9 |
| 8 Developers (person-hour) | 100.7 | 112.9 |
| *Effort analysis (per month)* | | |
| SPCC setup effort (person-months) | 0.2 | 0.2 |
| SPCC application effort (person-months) | 0.9 | 1.0 |
| SPCC effort (person-months) | 1.1 | 1.3 |
| Overall development effort (person-months) | 11.1 | 11.3 |
| SPCC portion of effort | $1.1/11.1 = 9.9\%$ | $1.3/11.3 = 11.4\%$ |

completion view. Please note that a deviation or risk can also be detected by using more than one single view. In the third iteration, the focus was shifted towards proving a better top entry point to project control and presenting the overall project state in one visualization. This resulted in improved usage of the aggregated project status view and a significantly improved number of detected plan deviations and project risks. The approach was able to detect between 25 and 80% of the listed plan deviations and project risks earlier than the traditional approaches to project control. More than 20% of the plan deviations and project risks were found that would not have been detected at all without using the control center. Although the results are quite promising, no further statistical analysis was possible because of the small sample size (four case study projects reported deviations/risks in iteration 2 and five projects in iteration 3). Therefore, the results can only be seen as provisional.

*Effort spent on setting up and applying an SPCC (H3)*: The average effort per month that was needed for setting up and applying an SPCC during the case studies was analyzed across the different roles of the case studies. The effort was normalized by dividing the reported effort by the duration of the specific case study in months. Assuming an 8-hour work day and 20 days per month, a corresponding effort in person-months can be calculated for setting up and applying an SPCC and the overall effort needed. The overall development effort is computed based on the number of project team members. Table IV gives an overview of the main results. The coaches were mainly involved in defining the measurement program and in accompanying the case study project. Compared to primary and secondary users, the effort for administrators was quite high. This may be related to the fact that the control center tool used was only a prototype. The ratio of control center costs to the overall development costs varied between 10 and 12% for a team size of 10 team members (one project manager, one quality assurance manager, and eight developers). The principles used in Table IV can be generalized for computing the SPCC portion of the overall development effort depending on the team size of a project. The ratio decreases if the number of developers increases: For instance, the ratio drops to 9–10% for a team size of 17 team members (15 developers). This still relatively high ratio might have been related to the fact that a prototype tool was used, that some tasks had to be performed manually, and that the evaluation period was too short, so that activities that usually have to be performed just once had a bigger impact on the overall effort figures.



DOI: 10.1002/smr

### 4.3. Threats to validity

In this section, we briefly address the most important threats to validity with respect to generalizing the results of the case studies. First, all main results were achieved in industrial case studies in a real setting. Unfortunately, the data set gained in that manner was too small to perform statistical analyses for some of the evaluated constructs. In the future, it would be good to perform further case studies and/or controlled experiments in order to obtain a larger sample size.

Second, for some companies, the concept of a control center was basically new; for others, sophisticated tools for project control were already in place. The scope of the control goals addressed was broadened during the second iteration and every organization rated the usefulness of the control center quite positively. However, the additional value still varies depending on the organization. The overall picture is a mixture of different organizations and it is hard to isolate the effects. Therefore, it is hard to make a general statement.

Third, the control center was evaluated for about two to three months as an addendum to the already existing controlling mechanisms. The evaluation period was too short for getting representative results (especially regarding the cost figures). The duration of the case study projects was much longer than this evaluation period. Thus, the overhead of introducing the control center is expected to be lower in a real application scenario.

Fourth, the evaluation instruments for perceived usefulness and ease of use were used for all three iterations without adaptations, and their reliability was evaluated systematically (using Cronbach's alpha[§]). In general, the reliability of these instruments was sufficiently high. However, other aspects such as evaluating found deviations and risks as well as costs were addressed using a set of open questions. Thus, a quantitative evaluation of their reliability was not conducted. In future evaluations, the questions addressing deviations and risks as well as cost should be modified, so that an evaluation can be performed that ensures that the questions measure the constructs reliably.

## 5. CONCLUSION AND FUTURE WORK

This article presented the Specula controlling approach for setting up a project control mechanism in a systematic and goal-oriented manner. Reusable control components were defined and instantiated to illustrate how to define measurement-based project control mechanisms and instantiate them for the software development projects of a concrete organization. A high-level process was presented that provides guidance on how to select the right control components for data collection, interpretation, and visualization based on explicitly defined measurement goals. Moreover, a simple example was presented of how to apply generically defined control components. The Specula approach implements a dynamic approach for project control; that is, 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 components can be systematically reused across projects or defined newly from scratch. Data are provided in a purpose- and role-oriented manner; that is, a certain role sees only those measurement data visualizations that are needed to

[§] For instance, in iteration 2, all alpha values were above 0.7 except ease of use of the setup phase ($\sim$0.6). That is to say, for all areas except this one, usefulness and ease of use were measured quite reliably.



DOI: 10.1002/smr

fulfill a specific purpose. Moreover, all project control activities are defined explicitly, are built upon reusable components, and are systematically performed throughout the whole project. A context-specific construction kit is provided, so that elements with a matching interface may be combined. The qualitative subjective benefits of the approach (extracted from practitioners' comments in the questionnaires) include: ability to identify and reduce risks related to introducing software cockpits; more efficiency in setting up and adapting project controlling mechanisms, allowing for more transparent decision-making regarding project control; reduction of the overhead of data collection; increase in data quality; and, finally, easier planning and control of a project.

The industrial case studies revealed the following evaluation results:

- Preliminary results show that the Specula approach was able to detect plan deviations and project risks earlier and more completely than the traditional approaches to project control in different case study projects.
- People perceived the usefulness and ease of use of the Specula control center as positive. This includes setting up a control center as well as actively applying a control center during the execution of a software development project.
- The ratio of SPCC costs to the overall development costs varied between 10% and 12% of the overall development effort for a medium-sized project (10 team members). This relatively high ratio might have been related to the fact that a prototype tool was used, that some tasks had to be performed manually, and that the evaluation period was too short, so that activities that usually have to be performed once had a bigger impact on the overall effort figures.

The future work will also concentrate on setting up a control center that integrates more aspects of engineering-style software development (e.g., monitoring of process-product dependencies and linking results to higher-level goals). 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. The first steps in this direction have been taken as part of the GQM$^+$Strategies® approach [20], which aligns business strategies and measurement initiatives.

**ACKNOWLEDGEMENTS**

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

**REFERENCES**

1. Standish Group International. *CHAOS Report 2007*: *The Laws of CHAOS*. Standish Group International: Boston MA, U.S.A., 2007.
2. Münch J, Heidrich J. Software project control centers: concepts and approaches. *Journal of Systems and Software* 2004; **70**(1–2):3–19.
3. Project Management Institute. *A Guide to the Project Management Body of Knowledge* (PMBOK® *Guide*) (3rd edn). Project Management Institute: Newtown Square PA, U.S.A., 2004.
4. Heidrich J, Münch J. Goal-oriented data visualization with software project control centers. *MetriKon 2005*, *DASMA-Software-Metrik-Kongress*, Büren G, Bundschuh M, Dumke R (eds.). Kaiserslautern, Germany, Shaker Verlag: Aachen, Germany, 15–16 November 2005; 65–75.



DOI: 10.1002/smr

5. Heidrich J, Münch J, Wickenkamp A. Usage scenarios for measurement-based project control. *Proceedings Third Software Measurement European Forum*, Dekkers T (ed.), Rome, Italy. Smef, 10–12 May 2006; 47–60.
6. Basili VR, Caldiera G, Rombach D. The experience factory. *Encyclopaedia of Software Engineering* 1994; **1**:469–476.
7. 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, U.S.A., 1992.
8. Tesoriero R, Zelkowitz MV. The Web Measurement Environment (WebME): A tool for combining and modeling distributed data. *Proceedings 22nd Annual Software Engineering Workshop* (*SEW*), NASA/GSFC, Greenbelt MD, U.S.A., 1997.
9. Krishnamurthy B, Barghouti NS. Provence: A process visualization and enactment environment. *Proceedings Fourth European Software Engineering Conference* (*Lecture Notes in Computer Science*, vol. 717). Springer: Heidelberg, Germany, 1993; 451–465.
10. Simmons DB, Ellis NC, Fujihara H, Kuo W. *Software Measurement—A Visualization Toolkit for Project Control and Process Improvement*. Prentice-Hall: Englewood Cliffs NJ, U.S.A., 1998.
11. Selby RW, Porter AA, Schmidt DC, Berney J. Metric-driven analysis and feedback systems for enabling empirically guided software development. *Proceedings 13th International Conference on Software Engineering*, Austin TX, U.S.A. IEEE Computer Society Press: Los Alamitos CA, U.S.A., 1991; 288–298.
12. 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* 1999; **25**(4):474–492.
13. McGarry J, Card D, Jones C, Layman B, Clark E, Dean J, Hall F. *Practical Software Measurement—Objective Information for Decision Makers* (1st edn). Addison-Wesley: Reading MA, 2001 (ISBN: 4-320-09741-6).
14. Kitchenham BA. *Software Metrics*. Blackwell: Oxford, 1995.
15. Agresti W, Card D, Church V. *Manager's Handbook for Software Development*. SEL 84-101, NASA Goddard Space Flight Center: Greenbelt MD, 1990.
16. ISO 9126: Software engineering—Product quality. *Technical Report*, ISO/IEC TR 9126, Geneva, 2003.
17. Differding C. Adaptive measurement plans for software development. *PhD Thesis* in Experimental Software Engineering, 6, Fraunhofer IRB Verlag, 2001 (ISBN: 3-8167-5908-4).
18. Heidrich J, Münch J. Cost-efficient customisation of software cockpits by reusing configurable control components. *Proceedings Fourth Software Measurement European Forum*, Dekkers T (ed.), Rome, Italy. Smef, Libreria CLUP Soc. Coop.: Milan, Italy, 9–11 May 2007; 19–32.
19. Heidrich J. Custom-made visualization for software project control. *Technical Report 06/2003*, Sonderforschungsbereich 501, University of Kaiserslautern, 2003.
20. Basili VR, 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. *MetriKon 2007*, *DASMA-Software-Metrik-Kongress*, Büren G, Bundschuh M, Dumke R (eds.). Kaiserslautern, Germany, 15–16 November 2007; 253–266.
21. Ciolkowski M, Heidrich J, Münch J, Simon F, Radicke M. Evaluating software project control centers in industrial environments. *Proceedings First ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (*ESEM 2007*), Madrid, Spain, 2007; 314–323.
22. Ciolkowski M, Heidrich J, Simon F, Radicke M. Empirical results from using custom-made software project control centers in industrial environments. *Proceedings Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (*ESEM 2008*). Kaiserslautern: Germany, 2008; 243–252.
23. Adams DA, Nelson RR, Todd PA. Perceived usefulness, ease of use, and usage of information technology: A replication. *MIS Quarterly* 1992; **16**:227–247.

## AUTHORS' BIOGRAPHIES

**Jens Heidrich** is department head at the Fraunhofer Institute for Experimental Software Engineering. His interests include project management, quality assurance, and measurement. Heidrich received a PhD in Computer Science from the University of Kaiserslautern.



DOI: 10.1002/smr

**Jürgen Münch** is division manager for quality management at the Fraunhofer Institute for Experimental Software Engineering. His interests include quality assurance, process management, and measurement. Münch received a PhD in Computer Science from the University of Kaiserslautern.




DOI: 10.1002/smr