TU Clausthal
Clausthal University of Technology

Mastering Project-Controlling Using a Process-Integrated Project Cockpit

Marco Kuhrmann[1], Jürgen Münch[2] and Andreas Rausch[3]

IfI Technical Report Series IfI-10-06

Impressum

Publisher: Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
Editor of the series: Jürgen Dix
Technical editor: Michael Köster
Contact: michael.koester@tu-clausthal.de
URL: http://www.in.tu-clausthal.de/forschung/technical-reports/
ISSN: 1860-8477

The IfI Review Board

Mastering Project-Controlling Using a Process-Integrated Project Cockpit

Marco Kuhrmann[1], Jürgen Münch[2] and Andreas Rausch[3]

[1]Technische Universität München, Boltzmannstr. 3, 85748 Garching,
kuhrmann@in.tum.de
[2]Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern,
muench@iese.fraunhofer.de
[3]Technische Universität Clausthal, Julius-Albert-Str. 4 38678 Clausthal-Zellerfeld
andreas.rausch@tu-clausthal.de

Abstract

Measurement has been emphasized as an effective method for gaining control and insight into software development projects. For that reason many organizations have incorporated measurement based controlling mechanisms into their software development processes. However, an integrated coupling between controlling mechanisms and the organization wide standardized process definitions on different application levels, like for instance on engineering and managing level, is still missing. Thereby standardized de-escalation and exception procedures could be provided to project and executive management in case of abnormal project situation. These procedures support help to perform corrective actions within the projects. Therefore the process definition has to be extended to incorporate standardized controlling measurements and de-escalation and exception procedures. To demonstrated and apply the proposed concepts the standard process - the V-Model XT - is used as application sample.

1 Introduction

In the face of the high impact of computer science on economy and administration, the creation of IT systems still comes with a high level of uncertainty. A quarter of all IT projects are canceled before completion with dramatic consequences for all participants. Almost half are significantly behind schedule, budget or suffer from incompleteness [1], [2]. This current situation is not acceptable. The success rate, the productivity, and the cost efficiency in terms of development, maintenance, and operation of IT systems

have to be increased. Standard process models like the Rational Unified Process (RUP) [3] or the V-Model XT [4], [7] can contribute to alleviating this situation.

Those process models capture best practice knowledge offering the possibility to acquire and use this knowledge, and last but not least provide an outline for training young software engineers. Several studies give empirical evidence on such benefits. Once a standardized process model is correct implemented within an organization quality, productivity, and success rates are significantly increased [5]. Obviously a couple of key factors, like for instance proper tool support and well trained people, are needed to successfully implement a process model as the organization wide accepted and applied model. However to establish, control, and improve the application of the process model at least the following three elements are required:

- A standardized process description organization wide accepted
- Adequate measure and controlling instruments for observing the application of the process model
- A catalog de-escalation and exception procedures based on experiences to find solutions for abnormal situations

Explicitly defined standardized processes that are applied in real development projects are necessary for implementing adequate controlling mechanisms. The required measurements have to be performed on standardized data input defined by the development process. Moreover controlling can be a means to increase adherence between the documented processes and the real processes. This is especially helpful if the data that is monitored during controlling provides information about process deviations. Thus measure and controlling instruments can also be seen as a means to support the successful deployment of process standards.

Project cockpits - to control and direct projects - are well known in other domains such as production engineering or business process engineering. In software engineering, typically two types of project cockpits can be observed. The first one is mainly used by the executives and provides only coarse-grained information. For each project to be monitored, there are usually two or three indicators comparable to traffic lights. These indicators are normally used to symbolize aspects such as timeliness, resource consumption, or degree of completeness of products. The second variant is a cockpit for the development project itself. Such cockpits usually focus on one project and visualize information such as defect profiles, product states, or test progress. Thus they are often project-driven and very specific. Nowadays, executive and management cockpits are usually not coupled, as shown in the comprehensive overview of concepts and approaches of project cockpits [6]. This leads to some serious problems, e.g. it is difficult to understand

what colored indicators of executive cockpits mean in detail. As a consequence, it is difficult to react appropriately on the executive and in particularly on the project management level. Moreover project management cockpits are usually project specific. If one wants to adapt a cockpit for a new project, it often has to be created from scratch. So, on the one hand, the controlling and directing capabilities of such cockpits are very suitable, but on the other hand, they are almost not reusable.

Finally we have to take into account that de-escalation and exception procedures - emergency rules describing corrective actions for anticipated or unexpected situations - are typically not part of the deployed process models. Those rules are a very important help in many critical settings. As IT projects are complex and as well often critical those de-escalation and exception procedures must be part of the deployed process model.

To sum up, more or less powerful isolated concepts for process models exist. However there is no integrated process model with respect to process model definition, measure and control instruments, and de-escalation and exception procedures. Therefore process models have to be improved towards more sophisticated integrated process models. Such an integrated process model is based on a common meta-model. This meta-model must be powerful enough to capture not only the process definition but also - in an integrated manner - the measure and controlling as well as the de-escalation and exception procedures.

In the next section we show, based on sample scenarios, how such an overall project controlling environment can be used for project management and risk-driven controlling. In section three, we introduce the meta-model of a usual standard development process - the V-Modell XT. Based on this meta-model we show in section four how to extend the meta-model to enable process engineers to model not only the development process itself but also related information that have to be integrated. In particularly measure and controlling instruments and de-escalation and exception procedures have to be integrated. Based on these elaborated concepts, we introduce in section six the design and implementation of a project cockpit to apply the presented approach within an organization and within project management.

2 Applying a Project Cockpit

As we pointed out, a project cockpit should be suitable for both management and project execution. At first, we want to motivate Project Cockpits using a real world scenario.

2.1 What we can learn from pilots and air traffic controllers

To point out our approach more clearly, we want to take a look at an area where cockpits work well. Looking at air traffic, we find controlling and monitoring on different levels.

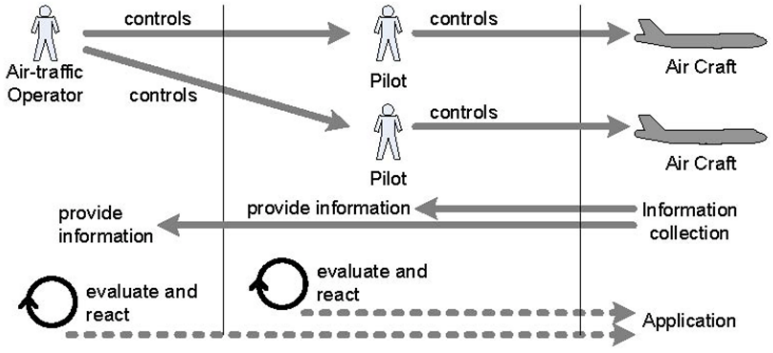

Figure 1: Controlling relations and information flow in air traffic

At first there is the aircraft. During a flight it produces status data and needs to be controlled. The aircraft is controlled by a pilot, who evaluates the collected and presented data. The relevant data are shown on instruments in the aircraft's cockpit. According to the current state, the pilot has to make decisions like correcting the heading because of stormy weather or initializing emergency procedures defined in emergency rules. Finally, the pilot gets information and instructions from air traffic controllers who monitor the overall sky. They know how many aircrafts are in the air, where they are, at which altitude, etc. Operators have to coordinate the air traffic.

In Figure 1 we sketched the relations between aircrafts, pilots, and controllers as well as the information flow between them. Especially the right part of the figure is of special interest. Pilots as well as controllers have so-called emergency rules available that define how to react in a crisis. So, for example, if a controller realizes that two aircrafts are heading for collision, he has to alternate the heading of one of the two machines to avoid a collision. Another very critical scenario is the case of an air crash. Here, operators have to find alternative routes for incoming aircraft and have to redirect them to other destinations.

As we have pointed out, pilots have emergency rules as well. So if there is an engine failure, a pilot will have detailed steps described in an emergency checklist. This checklist describes what he has to do in what order and so on. Based on the collected data, the pilot has to take action, as do the operators.

But the kind of information presented might be different. So, for example, an operator on the ground does not need to know that the air conditioning system works well. The pilot should know because a failure could endanger the passengers' safety.

2.2 Adaptation to project management

We consider this scenario to be a good parable, and we want to adopt this to project management. According to our scenario, we want to define the scope as the first step. We can identify two relevant scopes. The first one is the Project Management Scope/View and the second one is the Executive Management Scope/View. Each view addresses different roles. The first one is the view the project manager has on his project and is comparable to the pilot controlling the aircraft. The second addresses the executive management that is responsible for several projects. This is comparable to the controllers working in the control centers.

Thus the project manager becomes the pilot in this scenario. He needs detailed data about his project and has to be able to react in appropriate ways according to the project's current situation. On the other hand, the executive management becomes some kind of operator who needs information about several projects. The executive management has to be able to rate several projects related to the objectives of the whole company. In fact: both need information to make decisions, but they have different scopes. Decisions made by a project manager usually affect only one project but can have impacts on the organizational environment.

Decisions made by the executive management always target the whole organization and give rules and frames to particular projects as well.

2.3 Applying de-escalation procedures

Before switching to development process models, we want to present an example scenario comparable to the pilot - operator setting shown in section 2.1. We assume that a project manager evaluates the project's current state for reporting. The manager is responsible for a client-side software development project. He realizes that the document summarizing the system's requirements is not in an adequate state (meaning showing the document's state indicator is yellow or red). Because of the project's current situation he does not have a chance to correct this for himself. Thus he has to report to the executive management for escalating the situation. Together with the executive management he opens a Project Cockpit tool, which controls the currently running projects (Figure 2, upper right). The situation is as follows: The generated report shows that the product Requirements Specification Sheet is six days over schedule and two quality testing procedures were

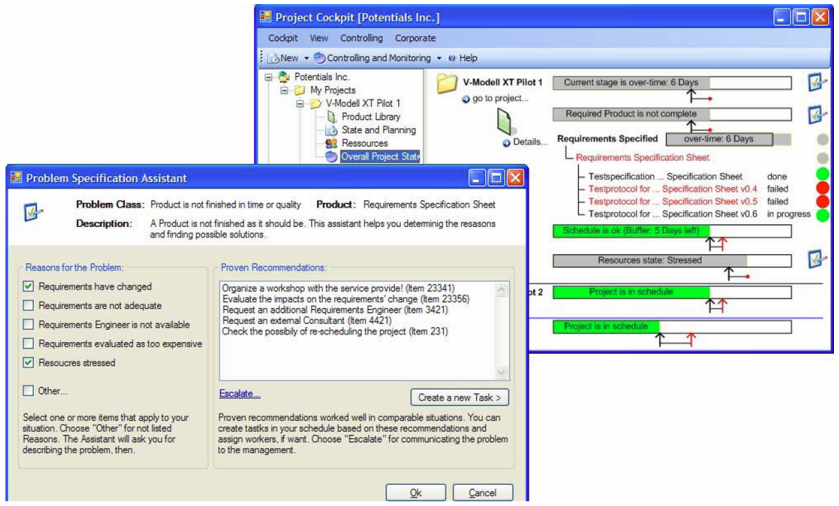

Figure 2: Controlling relations and information flow in air traffic

failed. The third test is currently running but may fail, too.

The Cockpit tool now provides some help - a Problem Specification Assistant (Figure 2, lower left). This assistant sums up information stored in a knowledge base, consisting of experiences, best practices and project reports. For the current situation, the tool identifies the problem based on predefined indicators or performance counters. The managers at first have to determine the problem (diagnostic mode). The list shown in the Reasons for the Problem-section is based on experiences. They can select applying reasons. Possible reasons for this kind of problem could be:

- Requirements have changed (because of new environment conditions etc.)
- Requirements are not adequate (because of possible misunderstanding etc.)
- Requirements Engineers are not available (maybe someone is ill)
- ...

During the organization's project experience, best practices were collected. The assistant now provide recommendations that could be a possible solution for the current problem. Because every project is different, several alternatives are presented, such as:

- Request an external Consultant (this could apply if the Requirements Engineer is not available or project resources are stressed)
- Request an additional Requirements Engineer (this could work, if the re-sources are stressed, but other projects possibly have free resources for compensating this project)
- Organize a requirements workshop (this could work, if requirements have changed or requirements were identified as not realizable)
- ...

It is not possible to anticipate all situations that can occur. So the presented solutions are only recommendations! The assistant have to couple with this and have to provide alternatives, like adding new problems, reasons, best practices etc. Furthermore it should be possible to implement adequate solutions in the problematic project e.g. by adding new tasks, rescheduling and so on.

Views on projects. Both, the project managers and the executive project management need support to master a variety of situations like the one shown above. In the previous section we sketched a conceptual tool prototype outlining the shown scenario.

A Cockpit has to support a project manager by analyzing the current problems and presenting the right information. It has to consider that a project manager can make decisions limited to his scope. So it should only present actions and recommendations possible in this context. A Project Cockpit acts context-sensitive. In the same way as the project manager's options, the ones for the executive management are also context-sensitive. But the scope is a different one. Decisions, made for a particular project might affect other projects as well. Thus the executive management always has to have possible impacts in mind. An adequate Cockpit has to consider this as well. It should always present a critical situation in the context of all other projects, the executive management is responsible for (Figure 2, upper right). A supporting tool automatically alerts responsible people that projects reached warning points and presents support using a checklist based on predefined experiences, emergency rules and guidelines, as sketched in previous section[1].

[1]The Project Cockpit cannot be omniscient! It has to have knowledge about reference and actual values, information about objects to be monitored, information about risk or quality indicators and references to a knowledge base associating actual values with problem descriptions, experiences and possible solution descriptions.

3 Standardized Development Process Model

For a project cockpit as presented in the previous sections a standardized development process model is required as the basis. One prerequisite for applying the presented approach is that the standardized development process model must be based on a process meta-model. Usually process engineers use the language defined by the process meta-model to describe a company's standardized development process model (see also [10], [11]).

In this section, we will introduce the meta-model of the used process model. In the next sections, the process meta-model will be extended to enable the process engineers to model not only the standardized development process model but also related measure and controlling instruments and de-escalation procedures. Once these aspects are modeled based on an overall meta-model an integrated tool support as presented in the previous section can be provided.

3.1 Sample process model - V-Modell XT

In this paper we use the V-Modell XT [7] as sample process model[2]. The V-Modell is the German system development and lifecycle process model standard for federal administration and defense engineering projects. The V-Modell regulates the system development and maintenance process; it defines binding sets of activities and artefacts, and accompanying processes such as quality assurance, configuration management, and technical project management. It is publicly available and many companies have successfully adopted it.

In 2002, the most recent update of the V-Modell dated back to 1997. Many new innovations in software engineering were missing, as were quality attributes of the process model. Consequently, a project was initiated to analyze the drawbacks and improvement potentials for the current V-Modell, and to develop a new redesigned version called V-Modell XT. At the end of 2004 the new V-Modell XT was established within the federal administration, military engineering projects, and companies as the new development process standard [9].

3.2 Development process meta-model

The V-Modell XT is a generic process model based on a well-defined formal meta-model. The V-Modell can be applied to different project constellations. Hence, it is necessary to customize the V-Modell to certain project settings.

[2] Any other process model based on a meta-model could be used to apply the presented approach.

This customization - called tailoring - is one of the first and most critical activities of a V-Modell user. Tailoring in the context of the V-Modell XT means the selection of one of the supported project types, the process modules to be applied, and the strategy for project operation to be used. All these metamodel concepts will briefly be introduced in the following.

Process Modules: As shown in Figure 3 (right side), a process module encapsulates a set of products, activities, and roles. Products represent the What of a project, meaning all (provisional) results, like documents, source codes or physical components. Activities are the How part. They are directly assigned to products. Every activity completes exactly one product. There are no activities defined that do not complete any product. A process module contains a specific set of products, activities, and roles relevant to a specific process area. All contents of a process module have content dependencies on each other. Thus, for example, the relevant contents for project management are collected in a separate process module; relevant contents for software development are collected in another process module, and so on. Each process module is an independent unit. It is individually changeable or extendable. The essential, static contents of the V-Modell XT are contained in the process modules. All products, activities, and roles do not contain any formal specification or limitation for the project's execution or the product's creation process.

Strategies for Project Operation: A project's execution is usually very complex. To enable reliable planning and controlling of a project, an ordered process has to be worked out. To support the users, the V-Modell XT includes a catalog of so-called strategies for project operations (SPO). SPOs contain the dynamic parts of the V-Modell XT. An SPO defines the coarse frame for the traceable project operation. The V-Modell XT contains a set of so-called decision points. Each decision point is a milestone defining a set of products. These products have to be in the finished state, which shows that these products were tested during the quality assurance process. Looking at these products, the project management has to decide whether the current stage can be finished successfully or not. Typical decision points are "Project engaged" or "System specified". Hence, an SPO defines a set of decision points and an order across this set to outline how the project has to be operated. Decision points can be reached multiple times and mark the finalization of a project stage.

Project Types: Obviously there exist dependencies between SPOs and process modules. The decision points of an SPO refer to the products to be finished. These products can be located in different process modules. As mentioned above, the V-Modell XT is a generic process model that has to be tailored before it can be applied in a project. During tailoring it has to be guar-

anteed that a reasonable combination of process modules and SPOs will be selected. Therefore the V-Modell provides project types. A project type characterizes typical project settings like server-side system development. Each project type defines a reasonable combination of mandatory and optional process modules as well as a selection of corresponding SPOs.

4 Integrating Measurement and Controlling Instruments and Emergency Guidelines

Quality and risk management should be directly integrated in the process model. Necessary extensions can be integrated in a meta-model to give a guideline for projects operated according to the process model. So a formalized process model that builds on a meta-model is advantageous. The meta-model can be extended with standardized measurement points. Standardized processes define uniformed project operation strategies. Furthermore such projects include standardized connection points for collecting data. So these projects are comparable. In the following sections we introduce such meta-model extensions.

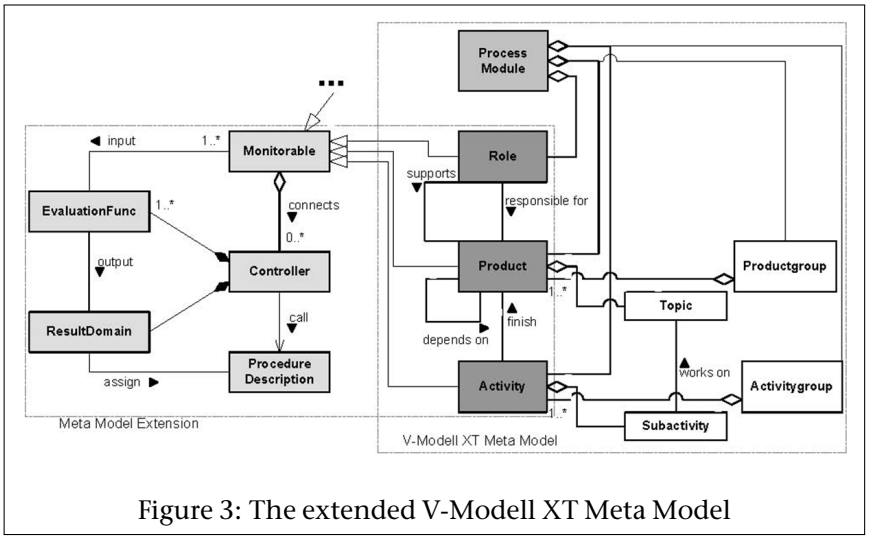

Figure 3: The extended V-Modell XT Meta Model

4.1 Project quality and progress controlling

The V-Modell XT contains meta-model elements appropriate for measurement and controlling. The elements we are interested in are products, ac-

tivities, and roles - concepts included in almost all available process models. Products and activities are promising candidates because their monitoring can be automated.

How to use controlling extensions? In section 2.3 we introduced a sample scenario and sketched a tool supporting the management on different levels. The tool monitors the products' states and the scheduled activities. In the underlying process model[3], connection points for miscellaneous controlling and monitoring instruments are defined. The process engineer, who made a concrete process model from this meta-model, has defined concrete metrics. Such metrics can be time-frame calculations, product state calculations, and associations to warning levels and so on.

Controlling extensions, defined in the meta-model, have to be known to the supporting tool. To schedule and control a project it is necessary, to have information about reference values and actual values (see also section 2.3). Reference values can be extracted during the scheduling of a project based on the process model [13]. Continuously reporting provides the required actual values. These are necessary to evaluate the project's current state. To rate the current state, metrics are needed, which can be derived by the GQM [16] approach. Metrics usually provide only raw data, which have to be interpreted. So some kind of knowledge base (including best practices or experience reports) is required. In the previous sections, we used traffic light indicators as metric. In the following section we want to describe this more in detail.

Sample: Traffic light indicators. To be able to interpret metrics, it is necessary to define result domains. Result domains consisting of the values produced by an evaluation function. An evaluation function gets the objects to be monitored (products, activities and so on) as input and calculates a result from the domain using predefined expected values and current values. Let us give an example. The state of a product can be in progress, presented and finished [7]. The question to be answered is: what states this indicator related to a product's state? A possible definition of the particular indicators made by a process engineer might look as follows:

1. Green:
    - A Product is still being edited/an Activity is still in progress. The Product must not have to be finished. Project is on time.
    - A Product is finished/an Activity has been carried out.

[3]The process model can only provide concepts preparing an organization to introduce sophisticated project-controlling. Concrete implementations, such as specific metrics have to be defined by a process engineer during the instantiation and customization of the process model for the organization or particular projects. So not only a tool for supporting the management is required, but an adequate tool for process definition as well!

- A Product/an Activity failed the quality testing procedure with minimal defects, meaning the number of errors is quite small or the importance is marginal.

2. Yellow:
   - A Product/an Activity failed the quality testing procedure with medium defects that have to be corrected immediately.
   - A Product/an Activity should have been finished, but is minimal over time, meaning within a scheduled buffer.
3. Red:
   - A Product/an Activity failed the quality testing procedure with heavy defects.
   - A Product/an Activity should have been finished, but is extremely over time, meaning the scheduled buffer was exceeded.

This is a very simple metric for product states. Every value defined in the domain is associated to at least one reason. Other interpretations or models are possible. Result domains may be specific to an organization or a single project.

Collecting controlling model elements. As shown in the previous sections, we need several elements to extend a process meta-model in an appropriate way. First of all, the meta-model objects to be monitored have to be identified. The second point is to define some kind of architecture that describes the controlling and monitoring engine. As shown in the previous section, we need an evaluation function realizing some kind of metric. This function has to calculate the monitored object's state according to the project's goals. The calculation leads to a result from a defined domain, which to our understanding has to be assigned to de-escalation procedures. As shown in section 2.3, de-escalation procedures can be checklists executed by someone. In the next section we want to refine our modeling and recommend an extension to the V-Modell XT meta model including all points mentioned above.

4.2 How to integrate Measurement Points into the Process Development Model?

The V-Modell XT's meta-model defines products, roles, and activities as basic concepts. One connection point in our understanding is a meta-model element appropriate for measurement.

Existing connection points in the meta-model. Here we want to give some examples for potential connection points already available in the chosen development process model. Connection points address queries over the set of meta-data associated with every meta-model element. Taking a closer look at the products, we can ask: "Who is responsible?" or "What is the current state?" and so on. In the same way, we are able to give information about activities as well. The V-Modell XT is rich in structured meta-data. So it is possible to identify lots of connection points for evaluating a project's current state. As we pointed out earlier, relevant connection points are: products, activities and roles. Products are the main results and contain lots of meta-data, like dependency information or completeness. Activities, on the other hand, are a matter of scheduling the project. They can be observed and associated with resources, like time or people. Roles may contain information about people assigned to a particular role like an architect who works under stress or a developer, who is fired.

Connection points for controlling and monitoring on the meta-model level. What are connection points in the context of project controlling and monitoring capabilities? Looking at Figure 3 (left side) we integrate the controlling and monitoring capabilities by creating a new super-class called Monitorable in the meta-model and deriving all relevant meta-model elements from this class. This way we express the concept of a connection point in a formal way. This super-class is also the access point for the controlling and monitoring engine outlined above.

The Monitorable class connects a Controller class, which contains at least one evaluation function, and a result domain (e.g., red, yellow, and red, as pointed out earlier). The evaluation function gets one or more monitorable objects as input and delivers a result from the domain. Each result is assigned to a de-escalation procedure, which is called by the controller. A de-escalation procedure may result in showing a checklist comparable to the scenario shown in section 2.3. It should be possible to assign more than one controller to a monitorable object on the one hand, and it is possible to get more than one monitor able object as input for evaluating on the other hand, too. So it is possible to define complex monitoring operations by combining several objects.

4.3 De-Escalation and Exception Procedures

After having defined connection points in the process model in the last section, it is necessary to define de-escalation and exception procedures. We want to refer to the scenario shown in section 2.3 again.

Scenario refinement and procedures. De-escalation and exception pro-

cedures are dynamic parts of projects that have to be activated if tolerances are exceeded. As pointed out in section 2.3, the project manager as well as the executive management should have checklists and recommendation-lists available. Furthermore, we noted that these lists are context-sensitive. We want to take a closer look at this: the project manager realizes that a product is too late in the schedule. This is shown by a tool implementing the monitoring capabilities shown in the last section. So, the list shown in the lower left of Figure 2 is the result of an executed de-escalation function, called by a controller observing the requirements sheet. Based on the example sketched in section 2.3, we can identify at least two points where exception procedures can be fired by a controller. The first point is the schedule that shows whether products are over-time or not. The second one is the testing result associated with a product's quality assurance definition. In the following two sections we want to briefly discuss about these two points.

Activity-based exception procedures. Activity-based exception procedures can be initialized on two levels. The first one is look ahead like, the second one is an emergency. Look ahead exception procedures apply e.g., if buffers are defined for work packages. If the schedule is near the buffer, the Cockpit is able to warn the user automatically, enabling the user to react in an early stage. This should be the preferred way. The emergency exception procedures can be fired automatically, too. These procedures apply e.g., if particular products are over-time. Different from the look ahead procedures, serious consequences for the whole project/for the organization are possible in this case. An immediate reaction is required.

As mentioned, corresponding lists are context-sensitive. Furthermore, a Project Cockpit can provide support for look ahead and emergency scenarios. Looking at the scenario shown in section 2.3, the presented list is a look-ahead checklist, because we think a buffer is still available.

Quality-based exception procedures. Quality-based exception procedures can be caused by quality testing procedures. To fire a quality based exception it is necessary to assign products to testing protocols. Quality based exceptions are caused by the product's state model and therefore can be fired automatically, too. An adequate controller observes a product. The associated evaluation function requires the product and the test result as input and is able to calculate the result, which possibly causes an exception.

5 Project Cockpit Design

After having introduced the basic concepts of a Project Cockpit, as well as the necessary extensions of a meta-model based development process model and their integration, we want to give an idea of the design of an appropriate

tool for management support. In this section, we want to give an overview of an appropriate tool design. We look at the scenarios sketched in section 2.3.

5.1 Collecting requirements

Projects create status data and results. The project leader has to understand, evaluate, and react in his scope - he has to avoid a crash. The project's status has to be known to management as well. But the scope is a different one. The state of one project has to be seen in the context of all projects that are currently running. This requires that projects are comparable. Comparability means:

- All projects should implement a similar process (unified development process model).
- All projects should produce comparable data at predefined points and should implement comparable methods for quality management (measurement points and quality gate definitions).
- All projects should have catalogs for emergencies available (unified emergency rules/correcting action/de-escalation procedures) that apply for the whole organization, including guidance for their application in particular projects.
- All projects should implement an interface that enables the project leaders as well as management to collect all information they need to realize the current state of a particular project as well as the state of all projects in the enterprise's scope. The interface should also include an implementation of a controlling engine.

These requirements based on the scenario in section 2.3 and our modeling in section 4 implicate that components are necessary for an organizational Project Cockpit. In Figure 4 we outline the big picture of an organizational Project Cockpit including all the mentioned points. The core of Figure 4 is a unified organizational development process model. This model should be based on a well-defined meta-model. We think this is the best way to include the required capabilities in accordance with measurement points, basic quality management, and basic integration concepts of emergency behavior as outlined in section 4.

As we could learn from the V-Modell XT's introduction, the generic process has to be customized to specific organizations first. A team consisting of process engineers and experienced project workers has to integrate organizational best practices during the customization phase. Thus, every point we sketched in section 4 would be integrated in a concrete process model.

The organizational process model should provide an interface for controlling and management. As shown in section 4.2, this integration can directly be included in the process meta model.

5.2 Views

An appropriate tool for supporting managers in project operation is always based on a unified organizational process model. Tasks that are specific to managers and the executive management are only views in such a tool.

Project management view: As shown in section 2.3, managers require detailed information about one project. Existing tools like microTOOL's inStep provide such views. Managers need guidance in an emergency that focuses on basic questions like finishing products quickly. Measures based on best practices are collected in a knowledge base and are assigned to specific tasks. Thus the tool supports controlling while its goal is to achieve the target of the project.

Executive management view: As shown in section 2.3, the executive management needs a different point of view. Beside the successful operation of single project, higher ranked targets of the organization are in their scope, too. An adequate view first provides coarse-grained information about general states. In case of an emergency, the executive management needs information about global resources to be applied to a critical project for possible compensation. They need information that supports them in estimating the impact of possible measures on the whole organization.

5.3 Architecture considerations

The requirements sketched above imply some kind of centralized project controlling. An organization has to provide project servers that observe all running projects. As we could learn in [12], this procedure not only enables good controlling, but distributed working as well. Centralized project servers have the advantage that the organizational process model, which is one of the most basic requirements, is available to all participating people. So all projects are executed on a comparable basis, which on the other hand, is a basic requirement for overall project controlling. The Project Cockpit is thus a simple distributed application, where the server is an organizational process model server, including a repository, a knowledge base, and the controlling engine, sketched during the previous sections.

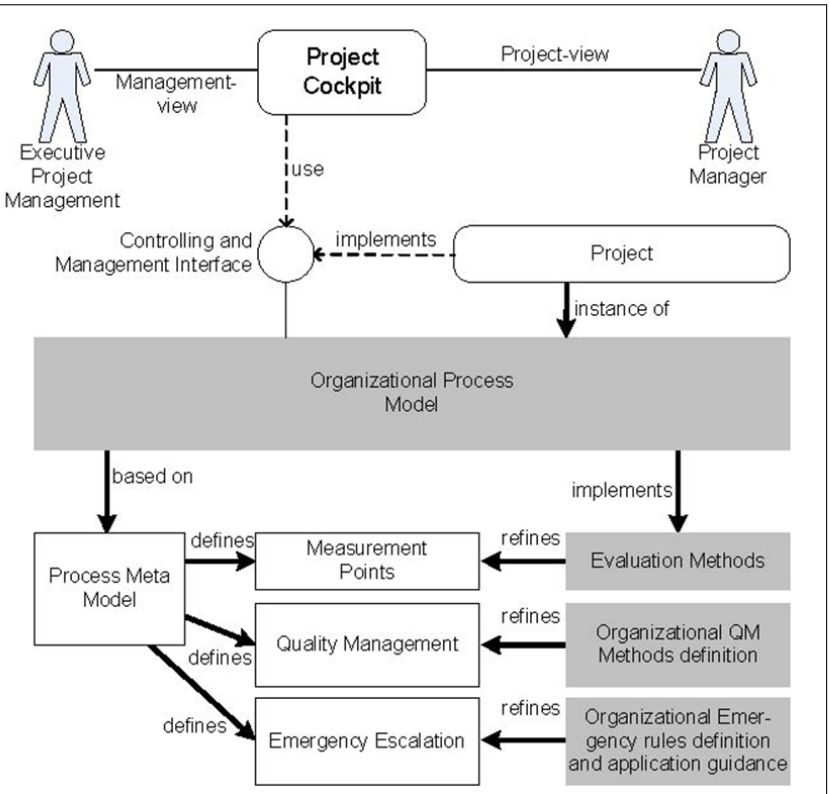

Figure 4: Concept of the organizational Project Cockpit

6 Related Work

Relevant approaches that are related to the presented approach for instrumenting process standards can be seen in the following areas: Process modeling notations, standard processes and software project control centers.

Process modeling notations or languages (such as IDEF0) usually comprise elements to represent activities, products, resources, and product flow. In addition, some notations (such as the Marvel Strategy Language) offer constructs for representing control flows (usually in a constraint-based or imperative way). Quality aspects are often represented via attributes that characterize activities, products, or resources. Examples are product complexity, duration, or availability. In addition, products and processes usually have status attributes such as 'non-existent', 'incomplete', and 'complete' for products, and 'disabled', 'enabled passive', and 'enabled active' for processes. Only few notations allow for more advanced definitions of quality models. MVP-L, for example, allows for capturing explicit relations between dependent and independent as quality models and to describe relation between attributes on different aggregation levels by attribute mappings. This supports a goal-oriented interpretation of the collected data in a Project Cockpit. Attributes are a good means to attach metrics to processes and process elements. However, they need to be coupled with controlling mechanisms during project execution.

Manifold process standards are available. Examples are the V-Modell XT, the MIL standards, or company-specific standards. Most of these standards have their unique meta-model that does not adhere to a meta-model standard (such as SPEM). Furthermore, most of the standards do not explicitly support instrumentation and controlling. Sometimes, as for example in the ECSS Software Development Standards of the European Space Agency standards, explicit instrumentation is addressed mainly in the verification activities. Applying standards for project tracking requires that they offer mechanisms to cope with replanning. When the goals or characteristics of a project change, the processes must react accordingly. Consequently the process standards, which should always reflect the real-world situation, must be updated. This requires flexible mechanisms for change management. Only a few software process modeling notations support updating the process models (e.g., Slang [14]). General problems exist manipulating states and keeping them consistent with the type information, and partial backtracking of process tracks in the case of erroneous process performance. Approaches for Project Cockpits that are tailored to software engineering mainly exist on the level of research prototypes. A comprehensive framework for feedback-oriented Cockpits can be found in [15]. Professional Project Cockpits usually stem from the business performance management domain. A recent survey of existing software project control centers can be found in [5]. Experience with

performance management is described in [17].

Finally, it should be mentioned that measurement should be done in a goal-oriented way so that predefined metric sets are only of a limited value [18]. Therefore, process standards need to offer a concept that allows integrating metrics without prescribing them in detail.

7 Conclusion

Project controlling and monitoring is quite complex - especially if many projects have to be monitored. In this article we presented an approach, which is based on a standardized organizational development process model. An integrated approach of instrumenting processes needs meta model-based processes. Mechanisms for the instrumentation can be directly integrated in the meta model and enriched by tailoring the process to specific needs. We identified possible so-called connection points in V-Modell XT's meta model appropriate for the injection of controlling capabilities. Controlling needs objects to be controlled, reference values and actual values. The observation of these values can be automated by observing selected objects and the schedule. To create added values it is necessary to provide additional support, like problem analysis and solution recommendations, too. A meta model-based integrated approach is strongly required, because it is the basis of a standardized project operation and controlling. Using this approach, projects become comparable. Thus not only instrumentation and controlling is possible, but benchmarking, too.

Standardized meta model-based Project Cockpits have high potentials for increasing projects' quality. It is possible to establish semi-automatic risk management: risks can be identified and associated to indicators, which can be controlled by the Cockpit. The Cockpit allows sophisticated information flow. Context-sensitive views, automatically created tasks and so on enable to optimize processes and their implementation. Standardized quality metrics and indicators combined with proven methods and best practices can be modeled and directly integrated in the project controlling environment. A Project Cockpit, based on our approach, is able to support responsible people by making suggestions and presenting recommendations. If there is a standardized process including standardized and homogenized process pattern, quality metrics, and indicators for controlling and benchmarking and a knowledge base including best practices, all projects in a multi-project scenario can be operated in the same manner. They are comparable and easier to handle.

In this article we sketched our idea of a tool supporting the Project Cockpit idea. We assume that it is not necessary to implement such a tool from the scratch. Existing tools can easily be extended. So it is possible to extend established project management tools like in-Step with Cockpit support. It is

also possible to think of extending team-supporting solutions like Microsoft Share Point Team Services with appropriate services.

8 References

1. Standish Group International, Inc. CHAOS: A Recipe for Success. (1999)
2. K. Bergner, M. Broy, K. Moll, M. Pizka, A. Rausch, T. Seifert. Erfolgreiches Management von Softwareprojekten. Informatik Spektrum 27:5, Springer-Verlag (2004) 419-432
3. P. Kruchten. The Rational Unified Process, An Introduction. Addison-Wesley, 2nd Edition, (2000)
4. A. Rausch, M. Broy. V-Modell XT - Grundlagen, Erfahrungen, Werkzeuge. dpunkt.Verlag (2006)
5. H. Watts. Three Process Perspectives: Organization, Teams, and People. Annals of Software Engineering, Kluwer Academic Publishers 14 (2002) 39-72
6. J. M"unch and J. Heidrich. Software project control centers: concepts and approaches. The Journal of Systems and Software 70 (2004) 3-19
7. V-Modell XT. Definition and documentation on the web. http://www.v-modell-xt.de
8. BWB-IT I-5, AU 250, Entwicklungsstandard fr IT-Systeme des Bundes, Vorgehensmodell. (1997)
9. Projekt WEIT - Weiterentwicklung des Entwicklungs-Standards fr IT-Systeme des Bundes auf Basis des V-Modell-97. http://www.v-modell-xt.de (2003)
10. M. Gnatz, F. Marschall, G. Popp, A. Rausch, W. Schwerin. The Living Software Development Process. Software Quality Professional, 5 (2003)
11. Object Management Group (OMG). Meta Object Facility (MOF) Specification. http://www.omg.org, document number: 99-06-05.pdf (1999)
12. M. Kuhrmann, D. Niebuhr, A. Rausch. Application of the V-Modell V-Modell XT - Report from a Pilot Project. Software Processes Workshop 2005, Beijing (China) (2005)
13. M. Gnatz. Vom Vorgehensmodell zum Projekplan. PhD Thesis, Technische Uni-versit"at M"unchen (2005)

14. Sergio C. Bandinelli, Alfonso Fuggetta, and Carlo Ghezzi. Software process model evolution in the SPADE environment. IEEE Transactions on Software Engineering, 19 (1993) 1128-1144

15. Christopher M. Lott. Measurement-based feedback in a process-centered software engineering environment. PhD thesis, Department of Computer Science, University of Maryland (1996)

16. V.R. Basili, G. Caldiera, and H.D. Rombach. Goal Question Metric Paradigm. "Encyclopedia of Software Engineering", 1 (1994) 528-532

17. S. Miller and W. Riddle. Experience Defining the Performance Management Key Process Area of the People CMM . . . and the Link to Business Strategy. Software Engineering Process Group Conference (SEPG 2000), Seattle, Washington, (2000)

18. Tesoriero R, Zelkowitz MV. The Web Measurement Environment (WebME): A Tool for Combining and Modeling Distributed Data. 22nd Annual Software Engineering Workshop (SEW), (1997)