A Learning Support for the Definition of Process Flexibility

O. Jaufman, M. Stupperich, K. Schneider, A. Dold, N. Kleiner

Abstract —At present, software development in the automotive industry is characterized by frequent changes caused by new innovations, fast-growing system complexity, growing software portion in cars, changing business relationships. This dynamical environment demands for flexible software processes. In order to improve a software development process with respect to flexibility, it is necessary to characterize what kind of flexibility is required. Therefore, we defined a set of requirements for desired processes based on our process analysis in DaimlerChrys-ler's engineering departments and analysis of related contributions proposed in the literature. Based on this requirement the existed processes can be analyzed to identify its improvement potential. The application of the requirements is illustrated in the context of a case study.

Index Terms — Process flexibility, requirements, case study, software development

1 Introduction

Nowadays, the automotive industry is confronted with a highly dynamic environment which is characterized by frequent changes due to innovations, business relationships or new product structures. One of the reasons for this trend is the fact that software has come to play a more and more important role in the automotive industry and will be the major source of innovation in the future. As software is a relatively new science, the corresponding new effective and efficient processes for software development which consider the mechanical and electrical engineering domains are needed. Especially important in such dynamic environments is flexibility: software engineers desire flexible software development processes in order to be able to quickly react to the changes in the development environment.

The problem is that the processes proposed for development of large, complex, and safety-critical software are not flexible enough for our ever-changing environment. For example, last year our prescriptive issue tracking process was changed significantly three times. The reasons for the changes were newly gained knowledge during the application of the "old" process and the need for three departments to work together collaboratively.

The research question arising from the problem is basic: What should a flexible software development process look like? In order to define a flexible software development process, the requirements as to the process flexibility are first needed. These requirements should be defined in such a way as to be measurable enabling a systematic evaluation of the process flexibility. Consequently, a suitable support for the definition and evaluation of the process flexibility is called for.

In this paper, such a support is presented. It consists of an emergent knowledge base and a flexibility definition process. The emergent knowledge base provides generic knowledge related to the definition and evaluation of the process flexibility. The flexibility definition process defines the steps to be performed in order to define the process flexibility for a specific software development process.

The benefit of our approach is support, first, through the definition and evaluation of the process flexibility and, second, through the identification of process weaknesses and improvement potentials with respect to process flexibility.

This paper sets out both the emergent knowledge base and the flexibility definition process and introduces a case study for validation. It therefore targets project managers, software engineers, and process quality assurance staff who are interested in defining or evaluating process flexibility.

The remainder of the paper is structured as follows. Chapter 2 discusses related work. Chapter 3 defines the objectives of our research work. Chapter 4 presents our approach, viz. the support for the definition and the evaluation of process flexibility. In Chapter 5 our approach is illustrated by means of a case study. Finally, Chapter 6 summarizes the most important contributions of the paper and addresses future work.

- *O. Jaufman is with the DaimlerChrysler AG, Ulm, HPC U800, E-mail: Olga.Jaufman@DaimlerChrysler.com.*
- *M. Stupperich is with the DaimlerChrysler AG, Ulm, HPC U800, E-mail: Michael.Stupperich@DaimlerChrysler.com.*
- *K. Schneider is with the University of Hanover, Hannover, E-mail: Kurt.Schneider@Inf.Uni-Hannover.de.*
- *Dold is with the DaimlerChrysler AG, Ulm, HPC U800, E-mail: Axel.Dold@DaimlerChrysler.com.*
- *N. Kleiner is with the University of Ulm, Ulm, E-mail: nikolaus.kleiner@informatik.uni-ulm.de.*

2 Related Work

There are two chief ways to define the quality of a process. One is to employ a process assessment standard such as SPICE [5][6]. Another way is the application of a define-your-own-model approach such as the GQM measurement method [9].

The process assessment standards [5], [6], [8] characterize processes based on a reference model. This reference

model typically defines the process capability levels and the criteria that have to be fulfilled for a considered level. The process assessments are then performed on the basis of the criteria. The advantage of the standards lies in the fact that they define the necessary criteria to be fulfilled in order to develop a product of the desired quality. The weak point of the standards considered is that they provide neither an explicit definition for process flexibility nor the criteria or metrics for the evaluation of process flexibility.

To define process flexibility in a measurable way, a measurement approach can be applied. As different contexts have different understandings of the process flexibility, a define-your-own-model approach should be applied [2]. The representative for define-your-own approaches is the GQM method.

The GQM method is a goal-oriented measurement method according to which a measurement goal should be refined into metrics via questions. The advantage of the GQM approach is that it allows the context and the project stakeholders' views of the process flexibility to be considered. The key drawback of the method, however, is that it calls for comprehensive knowledge and experience from measurement personnel. Unfortunately, a lot of companies do not have the experts with the necessary knowledge and experience.

The missing knowledge and experience needed to define and evaluate process flexibility can be gained by learning from past knowledge and experience. Several approaches for supporting the learning process have been proposed in the area of artificial intelligence. For example, deriving a process model based on logging of the activities of the people involved in the process [4] is one such method. These approaches address more the implementation aspect of learning; yet before addressing the implementation aspect of process flexibility by learning, concrete requirements for process flexibility are needed. If process flexibility is to be evaluated systematically, it needs to be measurable. Therefore we aim at supporting a measurable definition and the systematic evaluation of process flexibility by defining and providing the following:

- The generic knowledge related to the process flexibility definition and evaluation.
- A process for the usage of the knowledge provided.

3 THE GOALS

The goals of our proposed concept are threefold:

- To define the term "process flexibility" informally.
- To organize generic knowledge and experience related to the definition and evaluation of the process flexibility in the emergent knowledge base.
- To define a process for the use of the emergent knowledge base.

We aim to achieve the first goal with our learning support, since the term "process flexibility" is not at all explicitly defined in the process assessment standards considered; however the industry needs such support for the definition and evaluation of process flexibility.

We aim to achieve the second goal with our learning support: our objective is to be able to effectively and efficiently define and evaluate process flexibility in a measurable way. "Effective" means that the measurement program covers all the aspects that are important for process flexibility from the project stakeholders' point of view. "Efficient" means that the effort required for the definition of process flexibility should be as minimal as possible. We firmly believe that reuse of the available knowledge and experience would contribute to a more effective and efficient evaluation of process flexibility.

We aim to achieve the third goal to make it clear how the knowledge provided in the emergent knowledge base can be reused for the definition and evaluation of a concrete project.

The benefit of our approach is a support for software engineers, first, through the definition of requirements placed on process flexibility and, second, through the evaluation of process flexibility.

4 LEARNING SUPPORT FOR THE EVALUATION OF PROCESS FLEXIBILITY

4.1 The Informal Definition of Process Flexibility

Process flexibility is defined as the ability of a process to be easily and quickly adaptable to a new situation. A new situation can be caused through an innovation, a new business connection, changes in product structure, or other factors.

In order to make this key definition clear, the history of a process, called issue tracking, is considered. The issue tracking process targets capturing of issues such as changes in the product requirements, faults in the realization of the requirements or a decision to realize a specific feature earlier than planned. Figure 1 illustrates the issue tracking process with respect to its four different phases. The process is considered to be in another phase, if any process changes are occurred. In the first phase, the issue tracking process is described by two statuses "issue open" (i.e., a new issue that needs to be remedied is identified) and "issue closed" (i.e. the issue has been remedied).

During the application of the issue tracking process (phase 1) for tracking the issues arising during the software development of an electrical device, for example, new experience is gained. Based on this experience, the issue tracking process is refined. The following steps are added: analyse an issue, confer with an expert, evaluate an issue, refuse the correction of the issue, assign the issue to a person responsible for it, remedy the issue, and verify the correction.

During the application of the issue tracking process in accordance with phase 2, the following modifications were found to be needed:

The assign step is modified: in this step not only the assignment of an issue to a responsible person is to be performed. Additionally, whether the issue is a software issue, a mechanical issue, or a hydraulic issue should be defined.

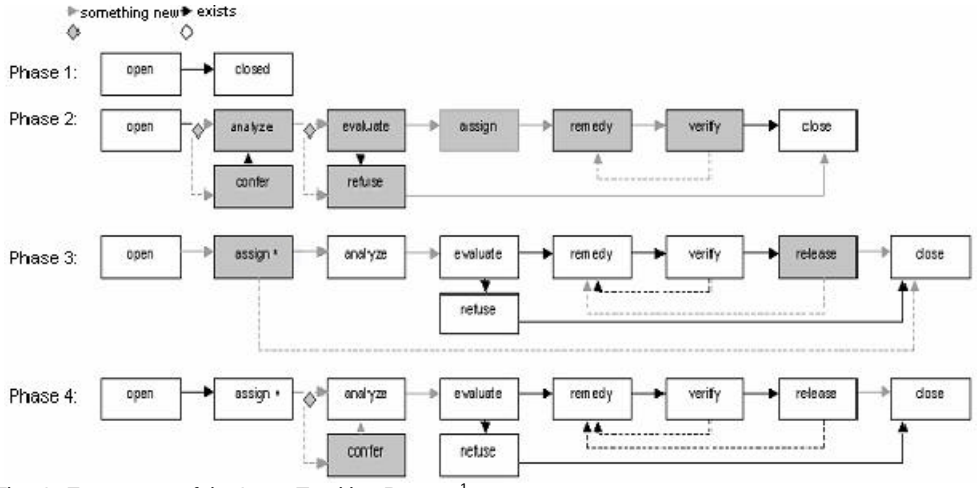

Fig. 1. Emergence of the Issue Tracking Process[1]

The order of the activities is changed: the assign step is performed before the analyse and evaluate steps. The confer step is omitted. A new step is added - release: during this p the integration is performed.

The process in the third phase shows the modifications. The emergence of the issue tracking process (phase 4) is caused by a need for inter-departmental collaborative working. To achieve a consistent process, an additional process step, confer, is first added and a transition from the assign* step to the close step is then omitted.

Referring to this example, process flexibility can be informally characterized by the ease to quickly

- identify the need for a process modification.
- understand what kind of process modification is needed.
- perform a required process modification.

Identification of the Need for a Process Modification

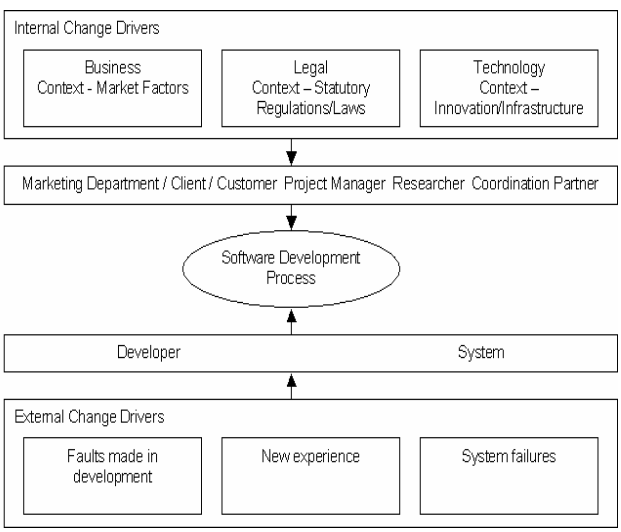

Fig. 2. Drivers for a Process Modification

In order to support the identification, the causes for the process modification are analysed. As an input for the identification of change drivers, the Aalst et al. autonomy of the workflow change [1] is used. This autonomy is abstracted to the autonomy of software development changes and modified based on the experience gained in a case study. The modification encompasses the introduction of roles who initiate the process modification. The case study is performed in the context of the issue tracking process. Figure 2 indicates that our autonomy of changes distinguishes between the following change driving factors: market, software development legislation, new knowledge, technical experience, errors made during software development, and system failures.

These drivers are introduced by the roles: marketing department, client, customer, project manager, researcher, cooperation partner, development team or the system platform.

Based on the knowledge taken from Figure 2, six issue classes are distinguished:

Issue class 1: issues caused through availability of new knowledge, experience or laws (initiated by researchers or developers).

Issue class 2: issues caused through a change in the internal organizational structure (initiated by the project manager).

Issue class 3: issues caused through a change in the external organizational structure (initiated by cooperation partners or by the project manager).

Issue class 4: issues caused through an incorrect implementation of the requirements from the product concept catalog (initiated by a product developer).

Issue class 5: issues caused through a change in product requirements (initiated by a customer / client or a marketing team).

Issue class 6: issues caused through system failures (initiated by the system platform).

The issue classes identified here are not to be seen as a silver bullet but as an emergent shopping list for the identification of the relevant process-specific issue classes. “Emergent” means that this set should be updated in line with the newly acquired knowledge and the experience gained from its application. Both the autonomy of the change drivers and the issue classes are intended to first

[1] The purpose and the content of the figure will be explained later.

help to quickly identify a need for a process modification (e.g. by process monitoring with respect to relevant issue classes). Second, the knowledge helps to define the issue classes that a given process should be especially flexible for.

Understanding the Type of the Process Modification

In order to understand what kind of process modifications are needed, the software development process at hand should be understandable for people involved in the process. This means, for example, that everybody should be able to evaluate which activities have been carried out, which ones are to be performed next, who should do which activities, when the activities to be performed should be started, and what kinds of resources are needed to perform the activities.

Carrying out the Process Modification

Looking back at our issue tracking example (Fig. 1), we note that the modification is performed through executing one or several of the following modification activities:

1. Removing an existing process step.
2. Adding a new process step.
3. Modifying an existing process step.
4. Changing the order of the process steps.
5. Removing a transition between two process steps.
6. Adding a new transition.

Consequently, a flexible process is characterized by ease and speed of performance of modification activities.

4.2 Emergence Knowledge Base

In order to define and evaluate process flexibility systematically, it is important to understand four aspects:

1. The changes that can cause the process modification.
2. The changes to which the process should be especially sensitive.
3. What the relevant stakeholders understand by a flexible software development process (i.e. what are the criteria and metrics for the evaluation of the process flexibility).
4. The constraints that are to be fulfilled (e.g. quality assurance, documentation, product preparation).

The fourth aspect is essential as process flexibility is desired but may not negatively affect the product quality. There are process steps that have to be performed for this. The focus of the paper is, however, on the first three aspects. Consequently, the emergent knowledge base should provide the issue classes, the criteria and the metrics for the measurable definition and evaluation of process flexibility, and the relevance of the criteria with respect to the issue classes. To achieve this knowledge, the criteria and metrics for the process flexibility are to be defined.

Defining the Criteria and Metrics

A set of criteria and metrics that might be relevant for the definition and evaluation of the process flexibility is needed. Before defining the desired set, existing process assessment standards [5], [6], [8] are considered. This is done because process assessment standards aim at the improvement of software development processes to achieve enhanced product quality. Nevertheless, improvement of software development processes with respect to flexibility is a pivotal aspect, especially in such dynamic environments as we are faced with today. Unfortunately, such standards do not define the criteria or metrics for the evaluation of the process flexibility.

It is for this reason that the GQM method [9] is applied for this purpose. We decided to utilize the GQM method, as it is a widely used measurement method that allows a goal-oriented identification of the desired criteria and metrics for process flexibility. To derive the desired criteria and metrics, the following activities are performed:

The GQM goal is described with respect to its attributes (*object*: process, *purpose:* to define and to evaluate, *focus:* process flexibility, *viewpoint:* researcher, developer, manager, *context:* software development).

Based on the knowledge presented in Section 4.1 and focusing on the GQM goal, the process flexibility is described by four high-level questions. Then different aspects of each of the four questions are described by sub-questions. The modification activities, autonomy of the drivers, and the issue classes presented in Section 4.1 are taken into account in doing so with the focus on the GQM goal. This "decomposition" (i.e. refinement of different aspects of the high-level question through sub-questions) is performed until questions that are not concrete enough from our point of view are derived. The defined questions are the desired criteria.

Finally, the metrics are assigned to the "concrete enough" questions. Metrics propose a scale for "answering" the questions. We proposed both qualitative and quantitative metrics.

In this way, all the criteria and the metrics were defined. The proposed set of criteria and metric is a generic, reusable set. The approach for the reusage of the set will be discussed later.

Organizing the Knowledge in the Knowledge Base

The knowledge and experience gained during the analysis is stored in the emergent base with respect to the scheme depicted in Figure 4. The scheme is adapted from the define-your-own-model, the so-called SQUID model [7]. The SQUID model is employed, since it supports top-down decomposition as defined by the GQM method. Figure 3 shows that each criterion can either be decomposed in sub-criteria that describe different aspects of the criterion in more detail or be made measurable through assignment of a measurement model. Each criterion is captured in the emergent knowledge base as shown in Table 1.

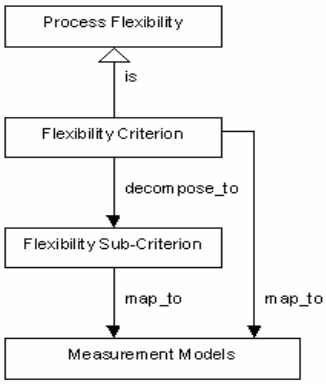

Fig. 3. Schema of the Emergent Knowledge Base

TABLE 1 EXERPT FROM THE EMERGENT KNOWLEDGE BASE

Father Question	Reference Number	Question
To L1_Q1 Question: How quickly (time aspect) /easily (needed experience aspect) the need for change and the kind of process change can be identified?	L1.Q1_L2.Q1	How complex is the software development process?
	L1. Q1_L2.Q2	How often do the validation and verification activities have to be performed in the process?
	L1. Q1_L2.Q3	How understandable is the software development process?
	L1.Q1_L2.Q4	How much knowledge and experience should the role involved in the software development process have to be able to quickly identify a need for change?

The table indicates that each criterion in the emergent base is characterized by the reference to the father question.

4.3 Using the Emergent Knowledge Base

The emergent knowledge base provides the issue classes, criteria, and metrics for the definition and evaluation of process flexibility. In order to keep the contents of the emergent base up to date, it should continually be evaluated and updated. Consequently, the corresponding processes are needed. The focus of the paper is, however, the *process flexibility definition process.*

The objective of the process is to select and identify suitable criteria and measures for a given software development process by reusing the knowledge from the emergent knowledge base. An overview of the process is set out in Fig. **4**.

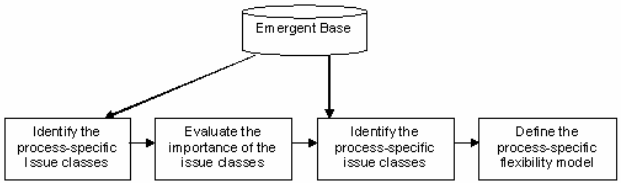

Fig. 4. The Flexibility Definition Process

In the first process step, the relevant issue classes from the knowledge base are to be selected and the missing issue classes identified for a given specific software development process are to be identified.

The task of the second process step is to evaluate the issue classes for which the process should be especially flexible. The evaluation is to be performed by all relevant process stakeholders (e.g. software developers, process quality assurance staff, etc.). The importance of the issue classes should be evaluated on the basis of the four-point scale: very important, important, somewhat important, and not important at all. Before evaluating the issue classes, the project stakeholders should understand which effect the representatives of the issue classes have and how often the changes occur.

In the third step, the appropriate criteria and metrics for the definition of the process flexibility are to be selected and identified. Which criteria are relevant for a given software development process is to be decided contingent on the importance of the issue classes. The relevant criteria and metrics are to be selected from the emergent base. The missing criteria can be identified in a top-down manner by application of the GQM method.

In the fourth step, a process flexibility model is to be produced. This should be done by first deleting the criteria and metrics not relevant for a given software development process from the decomposition tree. Second, the identified missing criteria and metrics have to be inserted into the decomposition tree with respect to the decomposition structure. Third, by specifying the criteria in the context of a given process. Finally, the relevance of missing criterion and metrics is to be evaluated on the same scale. The evaluation has to take the importance of the issue classes into account.

5. APPLICATION EXAMPLE AND EXPERIENCE

In order to clarify our approach and its benefits, a case study in the context of the issue tracking process was performed.

5.1 Application of Our Approach

The following sets out the four steps taken in defining the flexibility of the process.

Step 1: Identify the process-specific issue classes.
As cited in the history of the issue tracking process, the knowledge and experience we gain from the definition of issue tracking process lets us identify the following drivers for a process modification:

1. New experience, knowledge.
2. A change in the internal organizational structure.
3. A change in the external organizational structure.

Consequently, when referring to the generic issue classes, the following issue classes are identified as issue tracking specific:

Issue class 1: issues caused through availability of new knowledge or experience (initiated by developers).

Issue class 2: issues caused through a change in the internal organizational structure (initiated by the project manager or developers).

Issue class 3: issues caused through a change in the external organizational structure (initiated by cooperation partners or by the project manager).

Step 2: Evaluate the importance of the issue classes.
Based on the history of the issue tracking process, it is assumed that the issues from the first issue class have the greatest impact and the highest likelihood of occurrence.

Therefore the first issue class is evaluated as *very important*.

The effects of the second and third issue classes are assumed to be relatively small, because, for example, due to the wish to cooperate with each other the cooperation partners usually try to achieve a consistent process with as few process modifications as possible. The occurrence likelihood for the issues from the second and third issue classes is low, as a rule. Therefore the second and third issue classes are evaluated as *somewhat relevant*.

Consequently, the issue tracking process should be especially flexible with respect to the newly gained knowledge and experience.

Step 3: Define the process-specific criteria and metrics

First, the relevance of the criteria and metrics provided by the emergent knowledge base is evaluated based on the scale very relevant, relevant, somewhat relevant, and not relevant at all. Second, the missing criteria and metrics are identified by application of the GQM approach in a manner similar to that set out in Section 4.2 for the context issue tracking process. The criteria and metrics provided in the emergent knowledge base serve as a support.

Step 4: Define a process specific flexibility model

The process specific flexibility model is defined by

- omitting criteria evaluated as non-relevant from the decomposition tree. For example, the criterion "Does the process have a high level of information hiding? (i.e., only the absolutely necessary information produced during a process step should be passed on)" is removed. The criterion is not relevant because, in the issue tracking process, the people involved in the process should be able to see the information of interest to them even if they do not necessarily have anything to do with the information.
- adding missing criteria and metrics in the decomp osition tree with respect to its relationships to other tree criteria. For example, the criterion "How easy it is to quickly identify the risk and attractiveness of the implementation of a considered issue?" is identified as missing and is, therefore, added in the decomposition tree.
- specifying the criteria and metrics in the context of the issue tracking process, for example, the specification of the criterion "How complex is the process?" (based on our experience related to the issue tracking process):
    - An issue tracking process is very complex if it has more than ten process steps, more than eight different roles are involved in the process, and it is not defined hierarchically.
    - An issue tracking process is complex if it defines more than five process steps and has more than five different roles involved in the process.
    - An issue tracking process is non-complex if it has fewer than six process steps and has at most six different roles involved in the process, a clear responsibility concept, and few dependencies between process steps.

Thus, a flexibility model specific for the issue tracking process described in Fig. 1 is provided. Fig. 5 portrays an excerpt from the model.

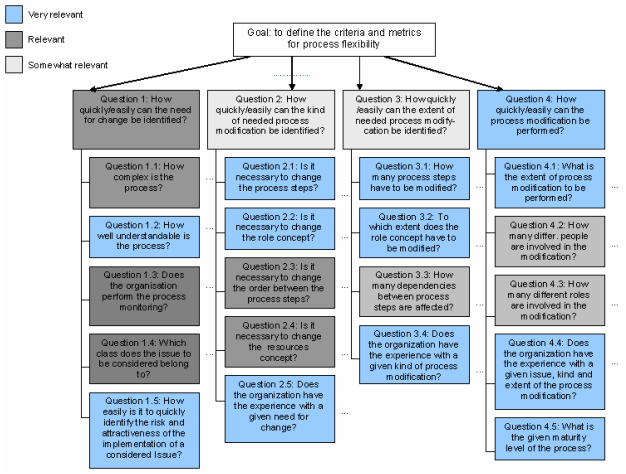

Fig. 5. An Excerpt from the Process Flexibility Model

5.2 Benefits of the Flexibility Model

The benefits of the process flexibility model are outlined in the following.

1. We can evaluate the issue tracking process based on the criteria and metrics provided in the model. For example, the issue tracking process (phase 3) presented in Figure 1 can be evaluated as follows:
    - The process is complex, as it has nine activities and eight roles involved in the process; but the process is part of the process hierarchy and the maturity of the process is quite high.
    - The process is understandable, as it has a clear responsibility concept; but it has many dependencies (i.e. the synchronization and coordination points) between process steps.
    - The organization intends to perform the explicit process monitoring and has captureed the knowledge with respect to the attractiveness and the risk of an implementation of the issue.
2. The improvement potentials of the issue tracking process are as follows:
    - It would be helpful if the variants of the issue tracking process together with its context were provided in the emergent knowledge base. The reuse of the process variants would help to quickly modify the process.
    - An efficient re-reuse process would support a quick and easy identification of the kind and extent of the requisite process modification.
    - A tool-supported, regularly performed identification of the difference between the prescriptive issue tracking process and the activities performed by the developers would help to quickly identify the need for a process modification.
3. We achieve a definition of the requirements for the process flexibility. For example, in order to be flexible a process should

- define what artifacts need to be developed, when the development of the artifacts should be started, who should develop which artifact, when an artifact should be finished, etc.
- be defined hierarchically in order to be well understandable and traceable for the people involved in the process.
- explicitly define process monitoring.
- explicitly define delta analysis between the prescriptive and currently performed processes.
- systematically package and reuse the acquired knowledge and experience related to process flexibility.

6. CONCLUSION

A present trend in software development is a highly dynamic environment. Such an environment is characterized by frequent changes caused by innovations, business relationships, and other factors. In order to be able to quickly react to a change, software engineers wish to have flexible software development processes. And to be able to define flexible software development processes, the requirements placed on the process flexibility are needed. The requirements should be defined in such a way that they are measurable in order to be able to evaluate whether a process is sufficiently flexible for a given software development environment. We have hence, in this paper, outlined our approach to support the research question "How can process flexibility be defined in a measurable way?".

For this purpose, we have first informally defined process flexibility, second an emergent knowledge base has been developed, and, third, a process for the measurable definition of the process flexibility has been derived. The informal definition of process flexibility was needed, as the standards considered do not explicitly define the term. The emergent knowledge base aims at providing the knowledge related to the definition of process flexibility in a measurable way. We believe that the application of the knowledge provided would help to more effectively and efficiently define the process flexibility for a concrete process. The flexibility definition process allows us to define the issues with respect to which the process should be especially flexible and, based on the evaluation of the issue's importance, to select and to identify the criteria and measures relevant for the process flexibility of a software development process at hand.

The knowledge stored in the experience base such as the flexibility definition process is validated and illustrated by means of a case study. The activities and the results gained in the case study are elaborated. We therefore feel that the work presented in the paper can be easily transferred into other business units and organizations and enable them to define the flexibility of their processes in a measurable way.

Our next steps in context of this work are first to explicitly define the constraints to be taken into account. This is essential as, in the automotive industry application domain, flexibility is needed but may not have a negative impact on the final product quality. A further step is to analyze how much flexibility is permissible and whether the flexibility alternatives proposed are sufficient.

REFERENCES

[1] Aalst, W. M. P., Jablonski, S. Dealing with Workflow Change: Identification of Issues and Solutions, International Journal of Computer Systems, Science, and Engineering, 15(5), 2000: pp. 267– 276.

[2] Boehm, B. and Turner, R., Balancing Agility and Discipline: A Guide for the Perplexed, Addison Wesley, 2004.

[3] Fenton, N. and Pfleger, S. Software Metrics – A Practical and Rigorous Approach. International Thomson Computer Press, 2nd edition ed., 1996.

[4] Joachim Herbst, Niko Kleiner Workflow Mining: A Case Study from Automotive Industry, the 10th European Concurrent Engineering Conference, Plymouth, UK, April 2003

[5] ISO/IEC TR 15504-2:1998(E), ISO Standard. Information technology — Software process assessment — Part 2: A reference model for processes and process capability, ISO/IEC, 1998.

[6] ISO/IEC TR 15504-5:1998(E), ISO Standard Information Technology-Process Assessment-Part 5: An example process Assessment Model, ISO/IEC JTC1/SC7, 2003.

[7] Kitchenham, B., Linkman, A. Pasquini, and Nanni, V. The SQUID-Approach to Defining a Quality Model, Software Quality Journal, vol. 6, no. 3, pp. 211-233, 1997.

[8] ISO 15497, Road Vehicles – Development Guidelines for Vehicle based Software, the Motor Industry Research Association, 2000.

[9] Rombach, R. Practical benefits of goal-oriented measurement. In N. Fenton and B. Littlewood, editors, Software Reliability and Metrics. Elsevier