In: Proc. of International Conf. on Business Process Management (BPM 2006), pp. 368-377. LNCS 4102. Springer Verlag.

IT Support for Release Management Processes in the Automotive Industry*

Dominic Müller[1,2], Joachim Herbst[1], Markus Hammori[1], and Manfred Reichert[2]

[1] Dept. REI/ID, DaimlerChrysler AG Research and Technology, Germany
{uni-twente.mueller|joachim.j.herbst|markus.hammori}@daimlerchrysler.com
[2] Information Systems Group, University of Twente, The Netherlands
{d.mueller|m.u.reichert}@ewi.utwente.nl

Abstract. Car development is based on long running, concurrently executed and highly dependent processes. The coordination and synchronization of these processes has become a complex and error-prone task due to the increasing number of functions and embedded systems in modern cars. These systems realize advanced features by embedded software and enable the distribution of functionality as required, for example, by safety equipment. Different life cycle times of mechanical, software and hardware components as well as different duration of their development processes require efficient coordination. Furthermore, product-driven process structures, dynamic adaptation of these structures, and handling real-world exceptions result in challenging demands for any IT system. In this paper we elaborate fundamental requirements for the IT support of car development processes, taking release management as characteristic example. We show to which extent current product data and process management technology meets these requirements, and discuss which essential limitations still exist. This results in a number of fundamental challenges requiring new paradigms for the product-driven design, enactment and adaptation of processes.

1 Introduction

In the automotive industry, car development has been dramatically influenced by the introduction of electrical and electronic (E/E) systems. E/E-systems consist of electrical control units (ECUs), i.e., embedded systems containing hardware and software components. In modern cars, we can find up to 70 ECUs comprising more than 10.000.000 lines of code [1, 2]. Several bus systems interconnect dependent ECUs realizing joint features like safety or multimedia functions. Car manufacturers expect shorter development cycles by faster implementation, bug fixing and installation of ECU software. Process support in E/E development shall accelerate product development and transfer of new technologies into the

* This work has been funded by *DaimlerChrysler Research and Technology* and has been conducted in the *COREPRO* (configuration based release processes) project

car. However, development processes must also meet the requirements of product liability laws and industrial standards, e.g., by adopting CMMI (Capability Maturity Model Integration) to achieve process maturity in car development or by implementing IEC 61508 to meet safety requirements. Altogether, mature processes shall contribute to realize strategic goals like high quality of the developed components and thus the whole car.

These expectations have raised new challenges for car development, particularly regarding the integrated support of engineering processes in the disciplines mechanics, electronics and software [3]. The synchronization of the different development life cycles is one challenge; another arises from the handling of the complex dependencies in E/E systems due to highly networked ECUs. Finally, different departments, engineering teams and external suppliers participating in the development processes have to be coordinated (cf. Fig. 1).

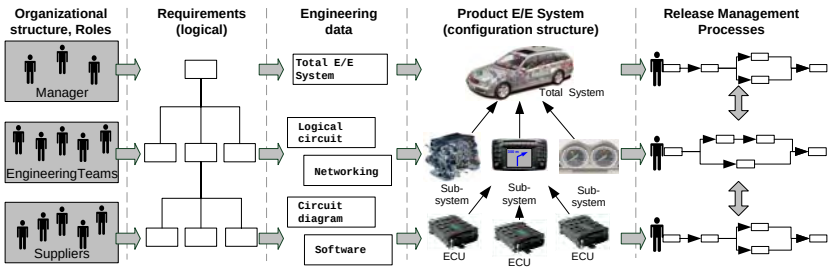

Fig. 1. E/E development with highly linked organizational structures, requirements, documents, product structures and processes [4].

The optimal coordination and synchronization of the development processes related to different car components is the key to adequate IT support. Fig. 1 illustrates the strong correlation between data and process structures. In particular, the structuring of the different development processes and their concrete dependencies are determined by the hierarchical structuring of the E/E system. Consequently, a process structure may have to be adapted if the corresponding product structure changes. For example, when adding a subsystem (e.g., a *navigation system*) to the product structure, new processes (e.g., for testing and releasing the new component) must be added and synchronized with the other ones. This is a complex task to be accomplished in a consistent and semantically correct manner. Finally, knowledge about the relations between process and data structures is helpful in the context of exception handling. When real-world exceptions related to a product component (e.g., failures in ECUs) occur, exception handling at the process level (e.g., abortion of the process) might become necessary.

This paper shows the high potential of BPM technology when being applied to product-driven processes in car development. We elaborate fundamental requirements for the IT support of automotive development processes taking *release management* (RLM) as characteristic example. To evaluate these require-

ments and to elaborate shortcomings of existing technology, we apply the process engine of a product data and process management system to RLM processes. We summarize the results of this evaluation and discuss which challenges remain with respect to the IT support of RLM processes.

Section 2 discusses characteristics of RLM processes and Section 3 elaborates basic requirements for their IT support. In Section 4 we highlight fundamental challenges based on the results of an implementation of RLM processes with the tool *UGS Teamcenter Engineering*. Section 5 discusses solution approaches in literature and Section 6 closes with a summary.

2 Release Management Processes

Release management (RLM) is an important part of the overall development process. Major goal of RLM is to systematically release the different product components at a specific point in time, for example, when a certain *quality gate* (i.e., milestone) or *production start* are reached. RLM covers *configuration management*, *testing* and finally the *release* of all necessary ECUs.

Different hardware and software versions as well as different variants of ECUs complicate RLM significantly. As an example for a product component with variants consider the *air condition unit*, where each variant is adapted to a specific climate region. These variants are realized by *ECU configurations* (when talking about ECUs we mean ECU configurations) consisting of different software or hardware. Fast implementation and change of ECU software result in about 100 changes of the total car system per day in early development phases [5]. However, proper functioning of every single variant as well as the total car system (based on combinations of all variants) have to be ensured. Thus, testing and release constitute complex tasks within the overall development process.

So far, ECUs often have been released without relying on a formal process that considers their complex dependencies. Due to the missing synchronization of the RLM processes for the different product components, costs as well as duration for integrating and testing have significantly increased. For this reason, configuration-based RLM has been introduced. The overall goal is to explicitly consider the dependencies between product components by defining hierarchical product configurations (cf. Fig. 2). These configurations represent the technical, logical, organizational or electrical view on the product [1]. The creation of configurations for E/E subsystems (e.g., the *air condition unit*) also helps encapsulating ECUs that realize functionality in common. As a result, we obtain a hierarchical configuration structure. Fig. 2 shows the encapsulation of the dependent ECUs 1, 2 and 3 by the configuration *Subsystem 2*.

Instead of performing RLM processes in an isolated fashion and solely at the level of single ECUs, we need improved process coordination and process synchronization. Case studies pointed out that the ordering structure of the RLM processes is determined by the configuration structure. We denote this phenomenon as *configuration-driven process structure*. The example in Fig. 2 shows a configuration-based release process. The creation of these RLM processes

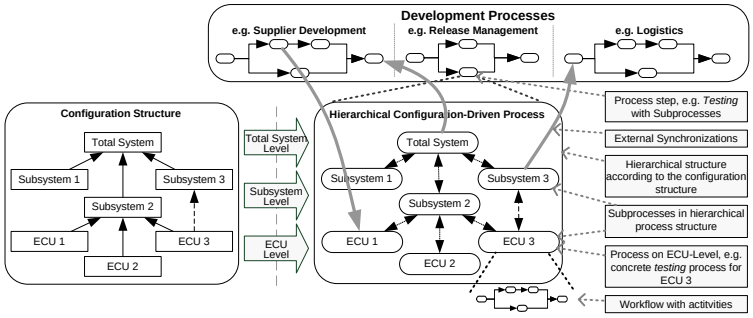

Fig. 2. Configuration-driven process structure

demands in-depth knowledge about the total car system and its configurations as well as existing dependencies between them [6]. Thus RLM processes cannot be fully automated, but consist of manually executed steps as well. The procedure to create a total system release (cf. Fig. 3), for example, starts with the following steps to gain the current ECU versions for a new release:

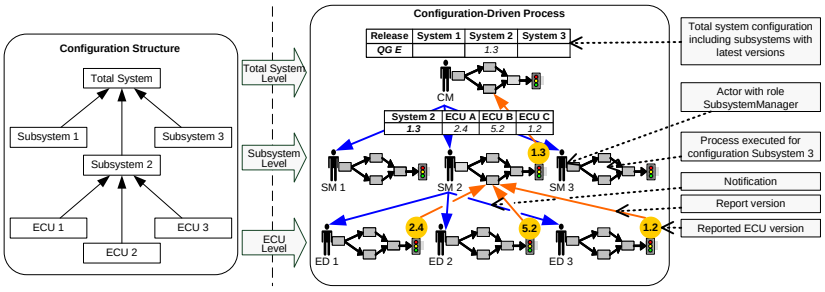

Fig. 3. Procedure to create a configuration-based release

1. With the given configuration structure, the actor with role *configuration manager* (CM) notifies the *subsystem managers* (SM) who are responsible for the subsystem level (e.g., SM1, SM2 and SM3 in Fig. 3).
2. All Actors with role *subsystem manager*(SM) also notify *ECU level actors* (ED) to retrieve the versions of their components (e.g., in Fig. 3 subsystem manager SM2 asks ECU level actors ED1, ED2 and ED3).
3. The actors working at the ECU level report the latest working ECU version to the corresponding SM (e.g., ED1 reports ECU version *2.4* in Fig. 3). After synchronizing the ECU versions, the SM generates a new version for the subsystem and reports it to the CM.
4. After completing the subprocesses and synchronizing the reported subsystem versions, further steps can be taken (e.g., triggering of external logistics processes for ordering the ECUs needed for testing).

The long duration of RLM processes amplifies the need for *flexible adaptation* of process structures during runtime. As an example, consider the removal of

the configuration *Subsystem 2* from the configuration structure as shown in Fig. 2. Several adaptations of the process structure become necessary, such as the termination and removal of the processes for *Subsystem 2* as well as its subprocesses *ECU 1* and *ECU 2*. The process for *ECU 3* is still needed since this component is also linked with *Subsystem 3*. To ensure consistent and semantically correct process results, dependent processes in the process structure have to be notified (e.g., the superior process *Total System* in Fig. 2). In addition, also external processes, which are synchronized with the configuration-driven process structure have to be informed about the change (e.g., the *logistics* department might have to cancel orders for removed ECUs). Similar reactions will become necessary if the process structure is modified by changing the *process definition* (e.g., due to optimization) or when adapting the running processes to deal with exceptional events.

Adaptation procedures must enable adequate runtime reactions to external events as well. A process exception will occur, for example, if an actor on ECU level (e.g., ED1) does not report the ECU version the actor on the superior level (SM2) in time (cf. Fig. 3). Then the actor on the superior level (SM2) has to react, e.g., by sending a notification to ED1 or by exchanging this actor. Further, exceptions caused by exogenous events (e.g., failures found in ECUs) have to be handled by the process management (cf. Fig. 4). If a minor error appears, such as a failure in the multimedia component, one possible reaction will be to stop the execution in this subtree of the process structure, to fix the error at ECU level, and to continue (or restart) the execution of the dependent processes. By contrast, a severe fault in the braking subsystem has extensive consequences, necessitating, for example, the abortion of the complete process (including all dependencies) and marking the release as *faulty*. In this case, the RLM processes for the respective ECU, the encapsulating subsystem and the total car system have to be restarted after error correction (cf. Fig. 4).

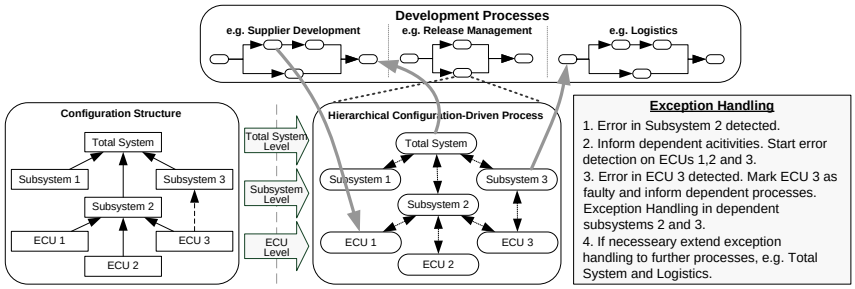

Fig. 4. Exception handling in configuration-driven process structures

The more complex configuration structures are the more difficult exception handling becomes. In case of an exception, all dependent processes have to be notified even if they have been already finished. The latter becomes necessary since external processes might also be affected by the exception. In large and highly coupled process structures, ad hoc reactions in conjunction with hierarchi-

cal and external dependencies may cause serious consequences up to deadlocks. Exception handling mechanisms must ensure process consistency as well as semantically correct and efficient process enactment.

3 Requirements for IT support

The high number of product variants, versions and component dependencies as well as dynamic adaptations of product structures make manual synchronization of related RLM processes almost impossible. The major goal for the IT support of RLM processes is therefore to assist process participants in managing the complex dependencies among configuration-driven process structures at the different configuration levels (cf. Fig. 4); the focus is less on the complete automation of all activities of a particular RLM process. Based on the experience we gained during our case studies, we distinguish four categories of requirements as shown in Table 1.

A. IT Landscape	B. Process Control
A1) Product data and configuration management functionality A2) Process management and data exchange	B1) Configuration driven process structures B2) Flexible subprocess execution B3) External synchronizations
C. Process Enactment Support	**D. Usability**
C1) Flexible adaptation of process structures C2) Exception Handling	D1) Visualization D2) Logging, monitoring and forecasts D3) Semantical merge of processes

Table 1. Requirements for IT support

3.1 IT infrastructure

IT support for RLM processes demands basic features and interfaces on IT infrastructure. First, there is a need for integrated product data management (PDM) in order to store and manage engineering data (e.g., component information, technical documents and software) and their dependencies in a consistent manner, and to make this information available for development processes (Req. A1). This includes support for product configuration management with the ability to manage a large number of product variants and versions [7]. Second, the IT system must also provide standard process management functions and support the controlled exchange of data between the PDM and the process management system (PMS) (Req. A2). This is required, for example, to transfer configuration-related results from the PMS to the PDM system (e.g., to flag a component as *released* after completing a RLM process). Further, user information needed for role resolution by the PMS is usually stored inside documents of the PDM system and therefore has to be made available for the PMS.

3.2 Process control

To enable process control, we have to implement the configuration-driven process structures (Req. B1). First, standard modeling concepts are needed for describing the different aspects of a process (e.g., control and data flow, reuse

of process fragments). Second, appropriate concepts for modeling hierarchical process structures become necessary to realize superior processes depending on the result of subprocesses; i.e., nested processes must fulfill a condition (e.g., provide a specified data quality or simply finish) until processes on higher levels are able to continue their execution. We call this mechanism *hierarchical synchronization* (cf. Fig. 4). As opposed to the common definition of hierarchical processes, where a subprocess is considered as a refinement of a superior process activity [8], we define a process activity as a placeholder for *a set of* subprocesses according to the configuration structure. These subprocesses constitute instances of different process definitions (e.g., different testing processes for multimedia and safety subsystems). Additional dependencies in hierarchical structures or exception handling constraints have to be applied to the hierarchical process structure.

Another requirement for synchronizing hierarchical processes concerns autonomy in terms of flexible execution of the subprocesses (Req. B2). As opposed to strict hierarchical process structures, there is a need to start single (and already instantiated) subprocesses independently of their superior processes. For example, the RLM process for *ECU 1* in Fig. 3 may be started independent from the RLM process of *Subsystem 2*. The synchronization of the hierarchical structure (cf. Req. B1) must be ensured for this case as well. In order to meet Req. B1 subprocesses have to fulfill the defined condition before superior processes can be started. Further, there are dependencies to external processes (e.g., *Subsystem 3* is connected to an independent process outside the hierarchical structure in Fig. 4), which we call *external synchronizations* (Req. B3).

3.3 Process enactment support

As described in Section 2, the *flexible adaptation* of configuration-driven process structures is a must. If a configuration change occurs, the hierarchical process structure has to be dynamically adapted (e.g., by adding or removing subprocesses) to ensure consistent results [9]. Thereby, hierarchical as well as external synchronizations have to be considered (irrespective of their execution state) and - if necessary - be adapted to ensure semantically consistent results (Req. C1). Changes of process definitions also affect running process instances. Due to the long execution time, proper adaptation might become necessary.

Further - and this is probably the most challenging issue - *exception handling*, in the sense of reacting on real-world exceptions, must be enabled. These exceptions are expected (to some extent), but require flexible handling mechanisms due to the large number of concurrently executed, dependent processes. Process reactions (executed automatically or by human interaction) depend on the error classification (comp. Section 2). Among other things, it must be possible to *abort*, *redo* and *restart* subprocesses in an efficient way. Thereby results of finished processes must be preserved, if they are not affected by the exception. Semantically consistent configuration structures and consistent process execution must be ensured in any case (Req. C2).

3.4 Usability

To enable the user-friendly execution of configuration-driven process structures, visualization support with partial, abstract, data- and process-centric views is required as well as the presentation of process changes and exceptions in a user friendly way (Req. D1). To ensure data privacy, authorization mechanisms with access control have to be implemented. For instance, engineers need technical views on configurations (and corresponding processes), while external suppliers shall only have restricted access to activities of assigned configurations (with exceptions being hidden). Managers want to have high-level views on the process, which are enriched with forecasts of process and product performance (e.g., execution duration, costs or the product quality). Basic to this kind of process intelligence is the creation and analysis of execution and change logs (Req. D2).

Regarding usability, we want to highlight the semantical merge of processes on ECU level (Req. D3). Generally, developers may be responsible for several ECUs. Considering the process in Fig. 3, the developer has to report the current version for every single ECU. From his point of view, it is sufficient to report all of his ECUs in one step. To realize this demand the execution of several processes has to be semantically merged.

4 Evaluation of current technology

The defined requirements in mind, we evaluated IT systems currently used in the automotive domain. For this purpose, we implemented the RLM process from Fig. 3 based on the PDM system *UGS Teamcenter Engineering* and its underlying process engine. This tool supports the management of engineering and product data, enables configuration management, and allows for process modeling and execution. Due to lack of space, we focus on the most important results of our evaluation (cf. Table 2).

A. IT Landscape	Rating	B. Process Control	Rating
A1) Product data and configuration management functionality	+	B1) Configuration driven processes structures	–
A2) Process management and data exchange	+	B2) Flexible subprocess execution	–
		B3) External synchronizations	o
C. Process Enactment Support	**Rating**	**D. Usability**	**Rating**
C1) Flexible adaptation of process structures	–	D1) Visualization	o
C2) Exception Handling	o	D2) Logging, monitoring and forecasts	o
		D3) Semantical merge of processes	–

+ = supported o = partially supported – = not supported

Table 2. Summary rating of *Teamcenter Engineering*

Teamcenter Engineering provides full product and configuration management support and meets the requirement for the exchange of data between PDM system and PMS (Req. A1 + A2). Basic mechanisms for modeling processes with sequential and parallel routing are available. Though hierarchical processes are supported, there is no possibility to create hierarchically synchronized processes as needed for configuration-driven process structures (Req. B1). Thus, flexible subprocess execution (Req. B2) also remains unsupported. Synchronization with

external processes is enabled by a predefined activity (so called *sync task*). However this concept is too inflexible (e.g., synchronization based on data quality is unsupported) to meet Req. B3.

Adaptations of (hierarchical) process structures are not supported at all (Req. C1). The same applies to flexible exception handling (Req. C2). Though the process engine supports some ad hoc actions, like aborting the execution of an activity or revoking the whole process, the consideration of dependencies to realize exception handling in hierarchical process structures remains a challenge.

Visualization mechanisms like the ones set out by Req. D1 are also not provided. While basic logging mechanisms are available (Req. D2), further concepts like automatic evaluation of the derived data and forecasts based on this data stay unsupported. The realization of semantical merge of processes (Req. D3) is also not possible using standard features.

5 Related Work

Based on the described requirements and the results of our PDM system evaluation we have investigated solution approaches from literature.

Workflow systems define fixed control flows to manage the execution of activities. In contrast, *case handling* [10] describes the coordination based on data objects. This enables less rigid process execution and shall make dynamic changes obsolete. Case handling also provides a way to create direct links between data objects and processes (denoted as *product-driven case handling*). A commercial implementation is provided by the *FLOWer* system [11]. Data-driven process modeling is an interesting approach for development processes. However, our focus is on process synchronization rather than on the coordination of single activities. Further, we identified several approaches that handle parts of our requirements. A solution approach meeting Req. B1 is *product-based workflow design* [12], a method for redesigning process structures based on product structures. Further, approaches for adaptive process management enable flexible process changes during runtime [13].

Related approaches are provided by *AHEAD* and *SIMNET*. AHEAD [14] deals with dynamic (software) development processes. It offers dynamic support for project management, process management, and engineering data management. The authors assume that development processes cannot be planned in detail in advance. Based on the modeled relationships between data and processes, dynamic task nets are generated. Even though the goals of this approach are closely related to ours, there are many differences. In contrast to software processes, car development is more complex and needs fixed processes to guarantee evolving and mature processes and thus high quality.

SIMNET [4, 15] is an approach for managing *engineering workflows*. Its goal is to enhance the communication between the participating parties in engineering processes by linking product data and workflow management (denoted as *product data-driven process*). SIMNET focuses on the provision of an evolutionary data model; extensive and flexible process control has not been considered.

6 Summary and Conclusion

Car manufacturers are more and more recognizing that process management is crucial not only for car production but also for the support of the complex development processes. Fast changes in technology and increasing complexity of development processes in the automotive domain are the challenges for an IT supported process management. As shown in this paper, current technology meets the requirements of car development processes only to a small degree. Especially the lack of flexibility and the non-availability of configuration-driven BPM tools prevent the usage of current process engines for development process support or necessitate a high degree of customization. New mechanism and paradigms for flexibility in configuration-driven process structures are required to enable IT support for process coordination not only in the automotive industry.

References

1. Knippel, E., Schulz, A.: Lessons learned from implementing configuration management within E/E development of an automotive OEM. In: INCOSE '04. (2004)
2. DaimlerChrysler AG, Research and Technology: Hightech report 01/2002 (2002)
3. VDI (Association of German Engineers): VDI 2006 - Design methodology for mechatronic systems. (2004)
4. Rouibah, K., Caskey, K.: A workflow system for the management of inter-company collaborative engineering process. Engineering Design **14**(3) (2003) 273–293
5. Wehlitz, P.: Nutzenorientierte Einführung eines Produktdatenmanagement-Systems. PhD thesis, TU Munich (2000)
6. Heinisch, C., Feil, V., Simons, M.: Efficient configuration management of automotive software. In: ERTS '04. (2004)
7. Crnkovic, I., Asklund, U., Dahlqvist, A.P.: Implementing and Integrating Product Data Management and Software Configuration Management. Artech House Publishers (2003) ISBN 1-58053-498-8.
8. Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques. Prentice-Hall PTR (2000) ISBN 0-13-021753-0.
9. Müller, D., Reichert, M., Herbst, J.: Flexibility of data-driven process structures. In: DPM '06. (2006)
10. Aalst, W., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. DKE **53**(2) (2005) 129–162
11. Aalst, W., Berens, P.J.S.: Beyond workflow management: Product-driven case handling. In: GROUP 2001. (2001) 42–51
12. Reijers, H., Limam, S., Aalst, W.: Product-based workflow design. Management Information Systems **20**(1) (2003) 229–262
13. Reichert, M., Dadam, P.: ADEPTflex: Supporting dynamic changes of workflow without loosing control. JIIS **10**(2) (1998) 93–129
14. Jäger, D., Schleicher, A., Westfechtel, B.: AHEAD: A graph-based system for modeling and managing development processes. In: AGTIVE. (1999) 325–339
15. Goltz, M., Schmitt, R.: Simnet - workflow management for simultaneous engineering networks. IMV Institutsmitteilung **23** (1998) 97–100