Integration and Complexity Management within the Mechatronics Product Development

Michael Abramovici, Fahmi Bellalouna
Dept. of IT in Mechanical Engineering, Ruhr University Bochum, Bochum, Germany

Abstract
Mechatronic products are the result of combining the engineering disciplines mechanics, electrics, electronics and IT. This requires coordinated trans-sectoral cooperation from the people developing the product as well as from the organisational unit. However, the systematic development of mechatronic systems has special demands to a multidisciplinary and holistic development process. Therefore implementing appropriate methods and tools is decisive for an effective product development. This article deals with the approach of integrating discipline-specific processes, applications and partial data models according to the SOA principle, based on the experience of developing PLM methods in the automotive industry.

Keywords:
Mechatronics; Product Lifecycle Management (PLM); Service Oriented Architecture (SOA)

1 INITIAL SITUATION

Complex mechatronic products that originate by combining and integrating solution principles from the engineering disciplines mechanics, electrical and IT, have automatically increased the complexity of development methods and processes as well as the resulting product data.

Many companies are being faced with more and more problems in view of this trend. These problems are making it difficult to cope with the development of mechatronic products regarding increasing quality, reducing development costs and time.

1.1 Process-Based Problems

The particularity of mechatronic systems is that their sub-systems are based on various technical solution principles - mechanics, electrical engineering and IT - which are combined together. Product innovations are obtained through this synergetic interaction [2] and therefore the development processes are so important to be able to realise such systems with multidisciplinary specifications.

However, in most companies established processes show large deficiencies regarding handling multidisciplinary development processes. The mechatronic product development in the involved disciplines is still carried out separately and in a rather isolated fashion, according to established, specific development methods [3]. The results are that:

- It is not possible to regard the product as an integrated mechatronic system.
- Coordinating and synchronising the different domain-specific development processes, activities, tasks and results across all fields is not sufficiently supported.
- The complex coherences and interactions between the disciplines are only considered in a later development phase.
- Comprehensive integration, configuration, change and release management across all disciplines is little or barely supported.

1.2 Data-Based Problems

Tools for developing mechatronic systems have over the years developed to domain-specific and isolated computer-based tools e.g. CAx, EES (Electrical/Electronic Engineering Solutions), CASE, PLM. These create large amounts of product data and product structures, that are only available in incompatible formats to one another. Today, for example, CAx data is stored and administrated in MPDM (Mechanical PDM) systems, EES data in EPDM (Electrical PDM) systems and CASE data in CVS (Concurrent Versions System) systems. These all have their own specific data models and structures that, in general, are incompatible with one another. This diversity of product data, data models and data formats, as well as product structures, has lead to huge problems in developing mechatronic systems:

- The interdisciplinary and functional relations between the various components and systems cannot be shown, understood, constructed as needed.
- The behaviour of the interdisciplinary components, systems and functions that are dependant upon one another cannot be displayed and analysed sufficiently.
- Interdisciplinary and coordinated changes on product data, that build on another is hardly supported.
- No adequate interdisciplinary integration of product data.
- It is hardly possible to interdisciplinarily release product data as well as functions, systems and components.
- Interdisciplinary product data configuration is hardly possible.

1.3 IT-Based Problems

PLM systems - PDM, configuration management and change management- are the hub for all IT tools and processes within

the development of mechatronic products because they supply a number of functions and data models for managing product data and for controlling and integrating development processes and activities.

Field-specific and isolated PLM island application systems have evolved in many companies over the years. These do not allow or support managing or integrating development processes along all disciplines. The classic PLM landscape, that is most commonly found in companies, consists of three different heterogeneous, incompatible and independently operating PLM platforms for the particular mechanics, electronics and IT areas. The consequence is that the tool and system support when developing products in mechatronic aspects – combining and integrating solution principles for the engineering disciplines mechanics, hydraulics, electrics, electronics, software and hardware – are inadequate, hardly available.

2 INTEGRATION REQUIREMENTS

Mechatronics has the potential of being successful in creating future products thanks to the close collaboration of mechanical-, electrical engineering and IT. It also has particular requirements for the development process: mechatronic products are characterised by high complexity and they integrate components from various disciplines (heterogeneity) [4].

Integration approaches and mechanisms in process, data and application system levels within the development of mechatronic systems are very necessary in view of this situation to be able to control inter- and multidisciplinary development processes.

The requirements regarding processes, data and IT architecture are illustrated in the following to be able to master the complexity and integration management within the development of mechatronic systems.

2.1 Process Based Requirements

The development of mechatronic products requires a holistic view of the product as an integrated mechatronic complete system. This requires multidisciplinary cooperation and coordination between involved disciplines, to realise an optimised whole solution. [4]

A multidisciplinary and holistic development approach model is needed to be able to work in a multidisciplinary fashion between the various domains and for them to agree on the conditions concerning time, costs and quality. This must fulfil the following requirements:

- The established development processes and activities – already existing sub-processes specified by the company philosophy e.g. mechanic, Electric/Electronic and software development processes – are not to be excluded but should be incorporated in the multidisciplinary development process, with as few alterations as possible. [5]
- It should force integrating all departments at the beginning of the development project. [5]
- The overall system specification, description and definition of solutions in the earlier development stages is to be supported. Consequently the dominance of few departments can be avoided.
- This development approach should contribute to minimising the development risk, by coordinating the domains at an early stage and thus securing their complex connections.

2.2 Data Based Requirements

Viewing the product as an integrated complete mechatronic system not only makes it necessary to synchronise between involved departments, but also between the specific disciplines' partial data models and along the entire development process. Therefore a multidisciplinary and abstract integration data model has to be developed to master the complex coherences in the development of mechatronic systems.

The data model acts as an integration platform for the development of mechatronic systems. Therefore the following criteria need to be fulfilled:

- The data model must allow comprehensive, neutral and abstract mapping of the product functions, their dependencies and their behaviour regarding the mechanical, electrical and IT aspects. The details for the individual disciplines' system designs are to be derived from this model. This data model is to act as an integration model, to merge the disciplines' own development results.
- The data model has to link all domain specific data models e.g. mechanical, electrical/electronic and IT data models with one another, whereby the comprehensive interdisciplinary coherences are to be mapped on the meta-model level. Here the systems' complex and interdisciplinary interdependencies between one another can be displayed and visualised more transparently and with less organisational effort.
- The data model must allow internal changes or further developments within the discipline specific data models, without adjusting other areas data models.
- The data model has to support implementing comprehensive versioning, configuration, change and release management, that allow integrating the various specific development results to a functioning entire system.

2.3 IT Based Requirements

The integration of discipline-specific IT landscapes plays a key role, in order for the above mentioned requirements for the process and data landscape to be satisfied. The comprehensive development methodology requires a comprehensive federative data model, but also a comprehensive IT integration platform. Which must fulfil the following requirements:

Integrating the specific IT integration platforms

The federative integration platform has to link up the already existing and established domain-specific IT integration platforms, which in general are specific PDM systems e.g. mechanical PDM, electrical PDM, CVS [5]. A comprehensive IT integration platform is to be developed, that enables comprehensive controlling, coordination and cooperation of mechatronic systems' development processes. This platform has to enable comprehensive systematic versioning, configuration, change and release management.

Protecting existing IT investments

Lately, many investments have been made in the divers development departments for building up own IT systems and these are already very valuable, so that it is out of the question to entirely replace these IT applications. In addition, such IT systems contain many years of company knowledge and experience that are indispensable for the companies. Therefore the federative IT integration platforms cannot replace the available domain-specific IT integration platforms, but rather build on these.

Flexibility

Mechatronic products and the corresponding technologies are very short-lived, which leads to high change dynamics in the development environment of such products. Companies often have to carry out changes in the process landscape or introduce new technologies and IT applications or shut down old IT systems at short notice in view of this situation. This often involves adapting the entire IT environment e.g. adapting the system interfaces. It is therefore important that the federative IT integration platform does not interfere with the change dynamics in the departments, but allows these with as little effort as possible. The comprehensive IT integration platform needs to be very flexible to satisfy this challenge. This in turn leads to increasing the innovative ability and agility of the company in terms of being able to react quickly according to market requirements and therefore increasing the competitiveness.

The domain specific IT integration platforms have to be integrated according to the principle of loose coupling in the comprehensive IT integration platform, to reach this flexibility in a federative IT integration platform. Unnecessary dependencies and tight couplings between IT components, that lead to a rigid IT architecture, can be avoided in this manner.

3 CONCEPT FOR THE INTEGRATION OF DOMAIN-SPECIFIC MANAGEMENT PLATFORMS

The three main, classical fields – mechanics, electrical engineering and IT – that have grown historically in companies, are usually involved in developing mechatronic systems. This organisational structure is mirrored in the IT architecture, which has lead to the development of three main domain-specific integration platforms that link the domains' specific computer-based tools with one another: CAx tools in mechanics, EES tools in electrical engineering, CASE tools in IT, as well as the sub-processes in the individual domains. Lately product data management (PDM) systems have established themselves in the mechanical and electrotechnical development areas and concurrent version systems (CVS) in IT development. These systems enable integrating the various computer-based tools, managing, organising and steering their created data with the aid of versioning, configurating and release methods and functions [7].

A concept for integrating the above mentioned integration platforms in terms of eliminating the deficits mentioned at the beginning and managing the challenges that arise in mechatronics is described in the following. This concept comprises a generic integration architecture model and a federative data model.

The generic integration architecture model was designed according to the service orientated architecture (SOA) principle. This is a layer architecture that encapsulates the various applications' functions in a service layer as services, which can be used within the process integration, thus the various application systems are loosely coupled. This not only helps to increase the technical connectivity of heterogeneous applications and reduces the diversity of interface technologies, but also permits enhancing and optimising the existing IT, data and process landscapes.

Mechatronic products are a synergy of components, which generally come from the areas mechanics (including hydraulics and pneumatics), electrical engineering and IT. As the electrotechnical components (hardware components) and the IT components (software components) are closely connected and dependant upon one another, it is necessary to describe their functions together (E/E function description) and to later integrate them together (E/E integration). This leads to a two-step function description in the product development – an whole entire and an E/E function description – as well as two-step integration – whole and E/E system integration. Whole system and E/E system integration platforms need to be developed to be able to manage this approach (Figure 1).

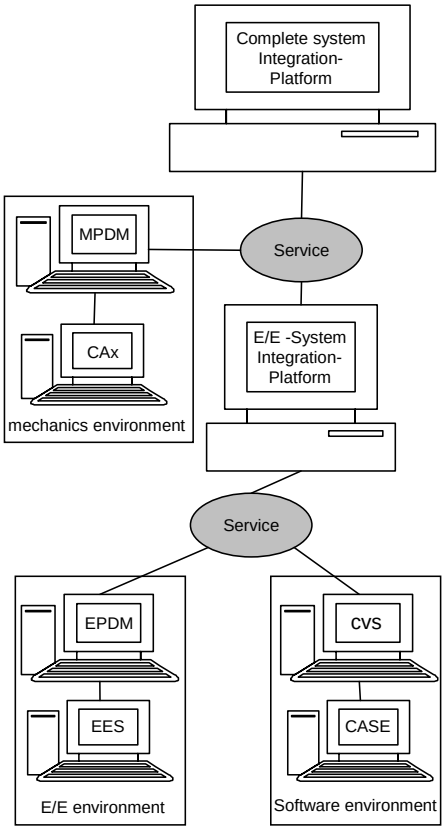

Figure 1: Concept for the integration domain-specific Integration Platforms.

3.1 Generic Integration Architecture Model

An integration model standardises the integration of existing domain-specific application systems and establishes a cross

platform integration infrastructure. The aim is to allow a holistic view of the system and thus to establish the basic prerequisite for product development including various disciplines.

The integration model builds on the available specific application landscapes and comprises five layers the application, service, workflow, integration platform and comprehensive process layers (Figure 2).

Application layer

The application layer contains domain-specific application systems that automatically treat, manage and steer information and data in the individual fields. Such applications are to be seen as resources, which provide their own data and functions in encapsulated form using suitable interfaces.

Service layer

Standardised services, which are implemented in the application layer underneath, are offered in the service layer. This decouples the applications and achieves high flexibility in the entire IT landscape.

Services can be split into two separate classes, atomic services and composite services. The first are services that encapsulate the application functions as adapters and make them available for the service layer. Such services are designed bottom-up. The definition of composite services, on the contrary, comes from the business process requirements. Such services support carrying out process activities and sub-processes. Hence composite services are put together from other composite and atomic services [8].

Workflow layer

Development activities and tasks covering various domains generally need functions for various applications. This requires adequately controlling and coordinating the cooperation between the participating applications, in order to achieve the continuity when processing business activities concerning various applications and to avoid the break of information along the development processes [8].

Workflows that assemble, coordinate and steer activities covering various applications are defined on the workflow layer. The individual activities in turn use the composite services on the lower service layer to carry out the required process tasks.

Integration platform layer

All necessary functions and data that enable product development incorporating various domains are provided in the integration platform layer. The functions in this layer are defined by the domain-spanning business processes' requirements and normally make up numerous activities, which make up workflows on the lower workflow layer.

The integration platform layer counts on a multidisciplinary federative data model that makes the communication between the integration platform and the specific applications on the metadata level, as well as a whole system description, possible. The metadata model is a model that abstracts all necessary specific metadata and maps these homogeneously.

Comprehensive process layer

Multidisciplinary process cycles are defined and mapped on the process layer. A number of multidisciplinary tasks, that are to be carried out in a set operation sequence and where required, supported by applications, are merged to one process unit. Functionalities and data, which are provided by the integration platform layer, are used to carry out these process tasks.

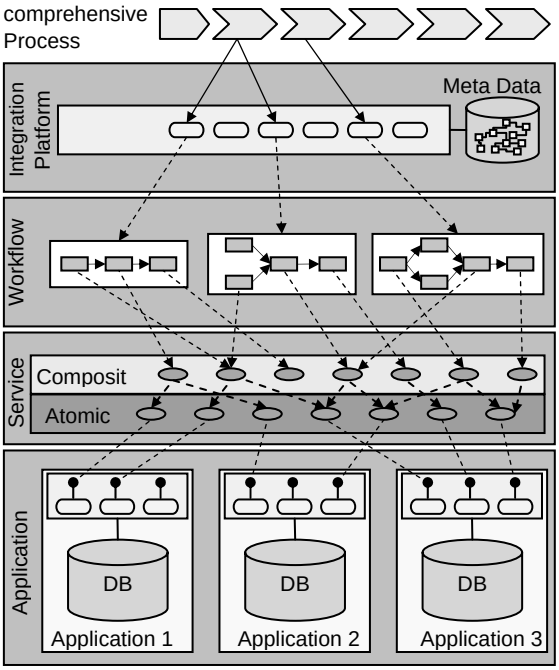

Figure 2: Generic Integration Architecture Model.

3.2 Meta model for the Generic Integration Architecture

A meta-model was designed using the UML modelling language to describe the elements of the generic integration model and its connections. The following classes are intended for this and their ties are shown in figure 3:

- The **Integration Platform** class describes an application that supports domain-specific processes coordination and cooperation in the development of mechatronic products.
- The integration platform disposes of numerous **functions** to take care of process tasks spanning various domains and applications. Here information objects are created, edited, steered, managed or deleted.
- The **Information Object** class describes all metadata that is needed for linking the discipline-specific partial data models. It makes continuous data processing along all processes possible. This class is described in detail by means of the federative data model (see subsection 3.3).
- A spanning-application function can define one or more activities, which are described in one or more self-contained performance units that are realised in one or more domain-specific applications. The **Activity** class is used here to represent all required activities to describe functions spanning applications.

- The **Workflow** class is used to describe all workflows, which synchronise and steer automatically carried out activities. The aim is to carry out domain-spanning process tasks that include various domains.
- One or more Composite-Services are used to carry out activities. This behaviour is described using the association between the Activity class and the **Composite-Service** class. A composite-Service can also use the services of one or more other Composite-Services, but it is required to use the services from at least one Atomic-Service.
- Every domain-specific application function is abstracted and presented on the service layer using an Atomic-Service. This is displayed through the association between the **Atomic-Service** class and the Application class. The Atomic-Service class abstractly describes domain-specific application functions as well as an interface that allows accessing the domain-specific applications.
- The **Service** class is an abstract class that is a generalisation of the Atomic- and the Composite-Service classes. This class serves the purpose of describing the common behaviour and attributes of the two service-types.
- Services communicate among one another and their environment (i.e. applications and activities) by sending and receiving messages that contain data and information concerning the execution of process tasks. This form of communication is mapped in the **Message** class.
- The **Application** class presents all domain-specific application systems that offer several functionalities to create, process and manage data objects. Therefore domain-specific applications are responsible for the technical implementation of Atomic-Services by providing data and functionalities.

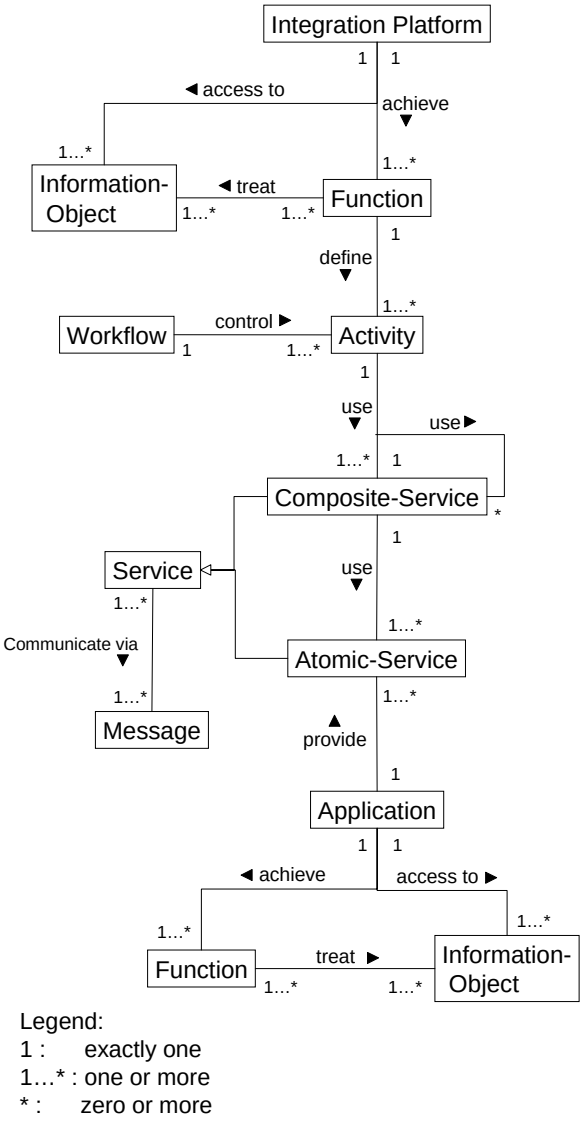

Figure 3: Meta Model for the generic integration architecture

3.3 Federative Date Model

The federative data model is introduced in this section. It allows top-down product structure and description covering various domains, as well as linking discipline-specific partial data models on the meta level. The therefore required classes and their ties are described in the following (Figure 4):

- The **Item** class depicts all kinds of elements from a mechatronic product that represent an object from the development process. Objects in this class are (sub-) systems, (main) functions, (sub-)modules, and components (mechanical, pneumatic, hydraulic, electrotechnical, IT).
- The Item class is further classified using the **Classification** class, to be able to distinguish between the item objects, i.e. if a function, system, module or component is concerned.

- An object from the Item class can have one or more variants that are depicts in the **Variant** class. Variants describe specific object adjustments that differ in complexity and in the technical development and production.
- An object from the Variant class can be versioned using the **Item_Version** class. This makes it possible to discribe a product's specification or its instance at a certain point in time. A clear validity for a new object version can be determined and displayed using the association between the Item_Version class and the **Effectivity** class.
- Objects from the Item_Version class are assigned to one or more domains (e.g. mechanics, hydraulics, pneumatics, electrics, electronics, IT) using the **Context** class. In this way different domain-specific views can be generated for certain tasks and users.
- Relationships between the objects from the **Item_Version** class are described in the association class, **Relationship**. These classes help to define product, system, function and component structures across all disciplines.

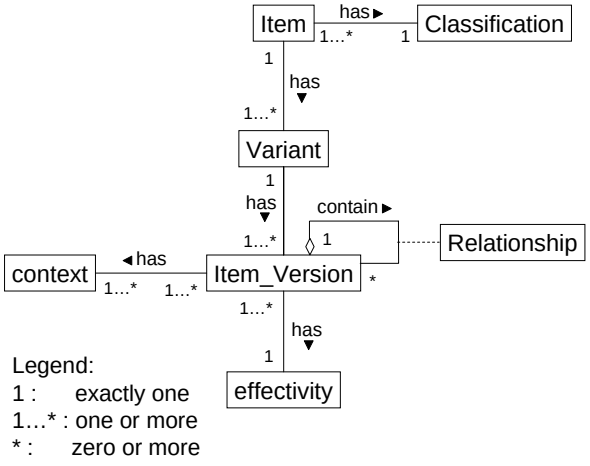

Figure 4: Federative Data Model

4 SUMMARY

The described and discussed problems in this work give an overview of the difficulties and challenges faced by many companies when developing mechatronic products. Mechatronics offers many fascinating possibilities and large success potentials, but at the same time has special requirements not only concerning the development process but also the IT-landscape and information handling [9].

The development of mechatronic products within time, financial and qualitative constraints makes a holistic view of the product as a mechatronic system indispensable. This leads to applications and partial data models needing to cooperate and be coordinated between the domain-specific processes that are concerned with the product development.

The concept presented in this article offers a generic architecture, which makes the synergetic cooperation of all disciplines, including processes, applications and data possible. A meta-model was developed for the generic integration architecture that supports flexible and loose coupling of the domain-specific applications. A federative data model was also developed that allows continuous product structures and descriptions and therefore contributes to linking the domain-specific partial data models on the meta level.

The advantage of this concept is that it is based on already existing and established domain-specific processes, application systems and partial data models. This preserves long standing and valuable company investments in the process, IT and data landscapes. In future, special fields can conserve their internal authority, through the principle of loose coupling, so that they can continue to further develop their internal processes, applications and data models autonomously, without being influenced by other fields. This leads to companies being able to introduce new products, technologies and innovations quickly and economically.

This concept is currently being implemented within the scope of internal work at the Chair of IT in Mechanical Engineering at the Ruhr-University Bochum, where the PDM-System Teamcenter Engineering and the web service technology based on the .net framework is being used.

5 REFERENCES

[1] Ebbesmeyer, P., Gausemeier, J.; Kallmeyer, F., 2001, Produktinnovation, Hanser Fachbuch München.

[2] Kleiner, S., 2003, Föderatives Informationsmodell zur Systemintegration für die Entwicklung mechatronischer Produkte, Dissertation TU Darmstadt, Shaker.

[3] Asklund, U., Dahlqvist, A.P., Crnkovic, I., 2003, Implementing and Integrating Product Data Management and Sofware Configuration Management. Artech House Computing Library, 17-31.

[4] Entwicklungsmethodik für mechatronische Systeme, VDI 2206, Beuth Berlin.

[5] Stuetzel, B., Russ, M., 2005, Orientierungshilfe im mechatronischen Entwicklungsprozess – das 3-Ebenen- Vorgehensmodell, atp, 47 ; 46-50.

[7] El-Khoury, J., Redell, O., Törngern, M., 2005, A Tool Integration Platform for Multi-Disciplinary Development, 31st Euromicro Conference on Software Engineering and Advanced Applications.

[8] Schemm, J., Heutschi, R., Vogel, T., 2006, Serviceorientierte Architekturen: Einordnung im Business Engineering, University St. Gallen.

[9] Möhringer, S., 2004, Entwicklungsmethodik für mechatronische systeme, Heinz Nixdorf Institut Paderborn.