Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2008
August 3-6, 2008, Brooklyn, New York, USA

DETC2008-49738

PML IN APPLICATION – AN EXAMPLE OF INTEGRAL SHEET METAL DESIGN WITH HIGHER ORDER BIFURCATIONS

Reiner Anderl / Department of Computer Integrated Design, TU Darmstadt

Jochen Raßler / Department of Computer Integrated Design, TU Darmstadt

Thomas Rollmann / Department of Computer Integrated Design, TU Darmstadt

Zhenyu Wu/ Department of Computer Integrated Design, TU Darmstadt

ABSTRACT

Processes are very important for the success within many business fields. They define the proper application of methods, technologies, tools, and company structures in order to reach business goals. Not only manufacturing processes have to be defined from the start point to their end, also other processes like product development processes need a proper description to gain success. For example in automotive industries complex product development processes are necessary and defined prior to product development.

Over the last decades several product modeling languages have been developed moving to object oriented modeling languages, such as UML, but the used process modeling languages are still procedural. The paradigm shift caused by object oriented description within product modeling languages has to be transferred to process modeling languages.

This paper describes an object oriented approach for process modeling, referred to as PML and its application to a complex product development and production process for integrated sheet metal design of higher order bifurcations. Using UML as a starting point an object oriented process modeling method is differentiated. The basic concepts which are needed for process modeling are put into an object oriented context and are explained. The paper gives an outlook, what can be achieved by the new approach.

INTRODUCTION

As the tendency of enterprises to collaborate growths steadily, industry faces new challenges managing business processes, product development processes, manufacturing processes and much more. Furthermore, discipline spanning product development processes are increasing, e. g. desired mechatronical products are in the need for knowledge from mechanical, electrical as well as software engineers. Humanists and economists also play a huge role in modern product development processes. Each individual discipline has its own, well-defined and specific processes, which typically are based on well-tried methodologies. These process descriptions are very powerful within the traditional discipline or the original enterprise, they were invented in. On the other side, they lack for flexibility, due to the reason that most existing process descriptions are based on a procedural approach. These are not powerful enough to meet requirements of describing cross collaboration. A short example is given to illustrate the problem.

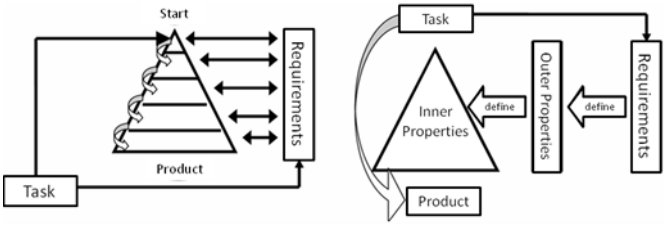

Fig. 1: Conventional and algorithm-based design process

Integral sheet metal products with higher order bifurcations represent an innovative product genus of sheet metal products. Product development for integral sheet metal products of higher order bifurcations requires a novel kind of development process utilizing knowledge of several disciplines like mechanical engineering, applied mathematics and material science. This new approach of product development is called algorithm-based product development and does not fit into the traditional VDI 2221 process (Fig. 1). The conventional product development process of VDI 2221 reflects the human thinking processes in product development. Following this conventional product development approach, the designer

begins his work by means of assumptions and first solution ideas, based on his former design experience. This solution is further intuitively refined and iteratively changed until an adequately suitable solution based on the designer's subjective assessment has been created. Concluding, a successful product is dependent on its developer's heuristic capabilities [4]. The potency of human mind is to draw coherent conclusions even on the basis of vague or fragmentary information chunks. Then again, humans only have limited cognitive resources. These facts, combined with the non-existence of basic examples for such products humans could orient on, make it impossible for specialized applications to realize the product development process in the conventional way.

Taking everything into account, the need for a new scientific approach in the development process was affirmed and the algorithm-based approach was established. The new algorithm-based product development approach does not rely on individual knowledge and subjective experience, but ideally generates complete and palpable solutions systematically. Still there is no proper description for this kind of flexible processes. To meet all these needs a new process modeling language is developed and demands the following requirements:

- Support of hierarchical structures.
- Support of flexible interpretation of a defined process without getting incompatible – support of generalization and specification.
- Robust process definition for flexible proceeding sequences of activities without losing process comparability – support of interchangeability of processes.
- Support of different integration scenarios and levels without changing process description at any time.— support of flexibility of processes.
- Easy to learn and read – audience of those process definitions are very broad.

This paper proves the sustainability and performance of a recently developed process modeling language means by the application to a complex product generation process example.

A conclusion closes this paper.

BENCHMARK OF EXISTING PROCESS MODELING LANGUAGES

In this chapter some existing process modeling languages are covered. It is briefly described why they do not meet the requirements of modern process definitions. For an in depth analysis and further particular details we refer to [13].

IDEF0 / SADT and Event Driven Process Chains (EPC) are procedural process modeling languages and support modeling processes with different levels of details. Both process modeling languages lack for transparency and clarity if they are applied to complex processes. Moreover, they are not very flexible regarding changes to the proceeding sequence of activities. [1], [2], [3], [4]

The Unified Modeling Language (UML) offers an all spanning modeling language. Regarding data and information model the language is object oriented. If UML is utilized to describe processes, UML reveals several disadvantages. UML is not an object oriented language for process modeling, because processes are still modeled procedural. Each activity is seen as an object. Relations between activities still base on logical states. Processes defined with UML are not very flexible regarding changes in the proceeding sequence of activities. [5], [6], [7]

Business Process Modeling Notation (BPMN) representation of processes is quite similar to the UML activity diagram. It is a standardized graphical notation for drawing business processes in a workflow. Processes are defined as a sequence of activities in swim lanes. Again it is a state based connection between object oriented activities. Therefore the evaluation result upon BPMN is similar to the UML verdict.

The Integrated Enterprise Modeling as a refinement of SADT enables users to generate views on the complete enterprise, not only on its processes. [9], [2] Processes are still in a SADT kind of style. Due to its retaining on logical sequence of activities it has no real advantage in modeling flexible processes. It still lacks a powerful support of process flexibility.

The Process Specification Language (PSL) basically is an ontology for describing processes. As PSL's objective is to serve as an Interlingua for integrating several process-related applications without formal and graphical constructs, it is therefore not capable for process modeling. [10]

The Semantic Object Model (SOM) methodology allows flexible and robust process modeling, based on the division of an enterprise model into several model layers, each of them describing a business system from a specific point of view. Within the process model the activity objects are connected with events. In comparison, SOM is most progressive regarding the definition of relations, but its constructs are difficult to understand due to the complex, integrated approach.

The modeling languages still describe relations on state based, proceeding sequences of activities. Taken together these results evoke the need for a new process modeling language facing the requirements of the paradigm change.

BASICS OF PML

A new approach for a process modeling language will be introduced in this chapter, which uses object oriented techniques and hence meets all requirements.

This approach uses the well known and widely used modeling language UML, that applies object oriented techniques to obtain modularization, reuse, flexibility and easy maintaining, among others, in the field of software and system modeling. UML is a technology that has a wide acceptance by users, developers and managers. Ongoing developments on the basis of UML, like SysML, prove the sustainability of the UML metamodel. Thus UML is a good starting point for the development of an object oriented process modeling language.

Fig. 2 shows the definition of an UML class diagram including class name, attributes and methods.
The class itself is time invariant as it is a generic description of the content of the context. But the instance of a class, an object, is time variant, because it holds characteristic values that can be checked to given times and can change over time. This means, the values can change, but the general structure of an object (number and kind of attributes) can not change.

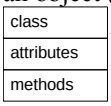

Fig. 2: UML class diagram

Having a time variant object it can be derived by time regarding to [11]

$$\lim_{T \to T_0} \frac{Object(T) - Object(T_0)}{T - T_0} = \frac{dObject}{dT} = \dot{O}bject \,. \tag{1}$$

Equation (1) shows that the content of an object, which means the attributes of an instance of a class, may change over time. Given a rule to change the attributes of an object one can express the change of the object's content as a process instance, which is shown in (2). It is necessary to mention that we use a discrete time T instead of continuous time t to implement "time steps". This is due to the result of the derivation as different process instances may need different time intervals to execute.

$$\dot{O}bject = \text{Process instance} \tag{2}$$

As we have derived the object we now have to derive the object's content. Fig. 2 uses the word attributes as defined in UML, in equation (3) we will derive the attributes, but using the word information to make the meaning clearer and more generic.

$$\lim_{T \to T_0} \frac{Information(T) - Information(T_0)}{T - T_0} = \\ = \frac{dInformation}{dT} = \dot{I}nformation \tag{3}$$

The derivation of information shows that the information may change over time. So the change of information, the change of attributes or data can be expressed as a method, which is shown in (4).

$$\dot{I}nformation = Method \tag{4}$$

The last field of a UML class diagram and thus in the object holds the methods, which act on the attributes. In the following we use the term operation for UML methods to differentiate between UML and our introduction. Operation and the just derived method are quite similar and are the same in several cases. In the following we derive the operation, which is shown in equation (5).

$$\lim_{T \to T_0} \frac{Operation(T) - Operation(T_0)}{T - T_0} = \\ = \frac{dOperation}{dT} = \dot{O}peration \tag{5}$$

The meaning of the derivation of an operation is quite complex. To express this mathematically we can use equations (3) to (5), which show, that dOperation / dT is the first derivation of an operation or the second derivation of information. This means dOperation / dT is the gradient of an operation or the curvature of information. The expression gradient of an operation seems quite handsome and opens the question: what does result in the change of an operation? Or, more exact, what does result in a change of the quality of the execution of a method? Think also of the similarity of operation and method. This question directly leads to the answer to the problem, which is

$$\dot{O}peration = Resource \,. \tag{6}$$

Resources influence the execution of an operation. The use of more or less resources leads to faster or slower execution, influences the quality of the output, may lead to more innovation and so on.
Equations (1) through (6) have shown the derivation from a time variant object to a time variant process instance. Generalizing the process instance we get a process class, which again is time invariant. The diagram of a process is shown in Fig. 3.

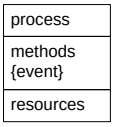

Fig. 3: PML Process class diagram

UML uses assurances to guarantee the range for its attributes. We need assurances too, but not as static ranges. Deriving a static value by time normally leads to zero. But knowing that the integral of a delta impulse δ(t) is defined as 1 [12] and we derive the constant with this definition in mind, the derivation of the static assurance leads to the delta impulse, which can be interpreted as an event. This means, an assurance for the processes is an event, a constraint becoming true, a set of data becoming available, time is elapsed and so on.
We introduce the term PML, which stands for Process Modeling Language and can be seen as an extension to UML, as SysML is. Thus the known techniques of inheritance, association, and cardinalities can be used. Implementing those techniques processes can be modeled hierarchically with modularization, structure, exchangeability and reusability.
Fig. 4 shows the used way to derive PML. Starting from the time invariant UML class we have instantiated a time variant object. This is derived by time and leads to a process instance, or project, which is time variant, and finally generalized to a time invariant process.

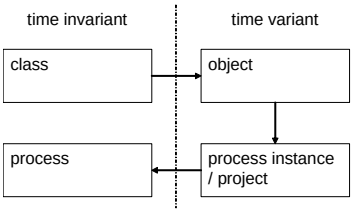

Fig. 4: Derivation loop of PML

The class therefore describes the product in a generic way, while the real contents are stored in its instantiation.
The same is true on process level. The PML process class describes the process in a generic way. It allows one to define all methods with assurances and resources needed for the process. The instantiation of a process is a project. This means, the instance of a process defines the current occurrence of resources, used data models etc.
Regarding connections and dependencies between single process classes, PML features the well known UML-concepts of inheritance and associations. The concepts for inheritance of process classes follow the notation of standard UML classes through simple object-oriented means like generic super-processes, sub-processes, overwriting and inheritance of methods and resources. Structural and hierarchical modeling is supported by using the concepts of associations, aggregations, and compositions with or without specifying the number of relation by the cardinalities. [13]

EXAMPLE APPLICATION OF PML

The deployability of PML for simple processes was proven in [13].
Now we want to illustrate the capability of PML means by a complex product development process and a manufacturing process. This example application stresses out the advantages of PML regarding flexibility of defined processes, reusability, clarity and understandability.
In the introductory chapter of this paper we mentioned the product generation process of integral sheet metal parts with higher order bifurcations. We will now briefly acquaint you with the challenging process of developing and manufacturing integral sheet metal products with higher order bifurcations: Engineers utilize the numerous advantages of the principle of bifurcation when constructing sheet metal products. Many building components have branched structures to enlarge surfaces, increase stiffness, influence flows, transport material, and so on.
Moreover they are adopted in the areas of lightweight design, optics and acoustics. In many applications the branching can serve several purposes at the same time. So far branched sheet metal structures were mainly produced in differential style, i.e. by gluing, welding or similar procedures or by material sheeting. These procedures have several disadvantages as they are heavier, have lower thermal conductivity, higher disposition to corrosion and are more likely to break due to the instability at the connecting piece or the double layer. To avoid these disadvantages, the new techniques of linear flow splitting and linear bend splitting have been developed. Fig. 5 is illustrating these new techniques.

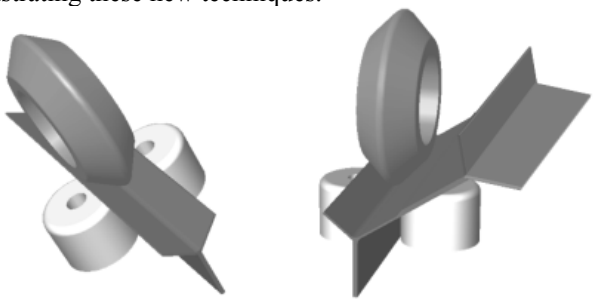

Fig. 5: Linear flow splitting and linear bend splitting

They enable engineers to generate bifurcations without the need to produce in differential style and make it possible to bail out all advantages of advanced bifurcated structures. Fig. 6 exemplarily illustrates a five-chamber integral sheet metal product with higher order bifurcations. Notice this slide trail is made out of sheet of metal, a benefit of the new production techniques just introduced. At the same time the number of welding spots decreased to only three.

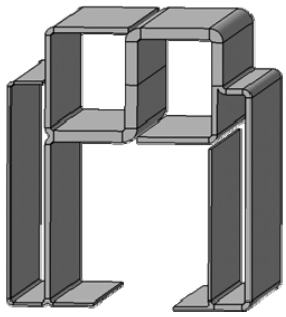

Fig. 6: Integral sheet metal product example: slide rail

The product development process for integral sheet metal products consists of the following activities:
The beginning of the development process is a clarifying-task phase, where the requested specifications are created to describe the properties of the product the customer is looking for (e.g. “acceptable weight”). These properties, called outer-properties, cannot be established in a direct way by the engineer. He has to look for parameters which are related to the outer-properties and which can be established in a direct way (e.g. material and geometry parameters). These new parameters are called inner-properties. They are derived from the cumulated customer requirements. In Fig. 7 the processes of collecting customer requirements and deriving inner-properties are represented through the processes *RequirementsCollection* and *SpecificationDerivation.*
Both processes together aggregate to the process *RequirementsManagement.*

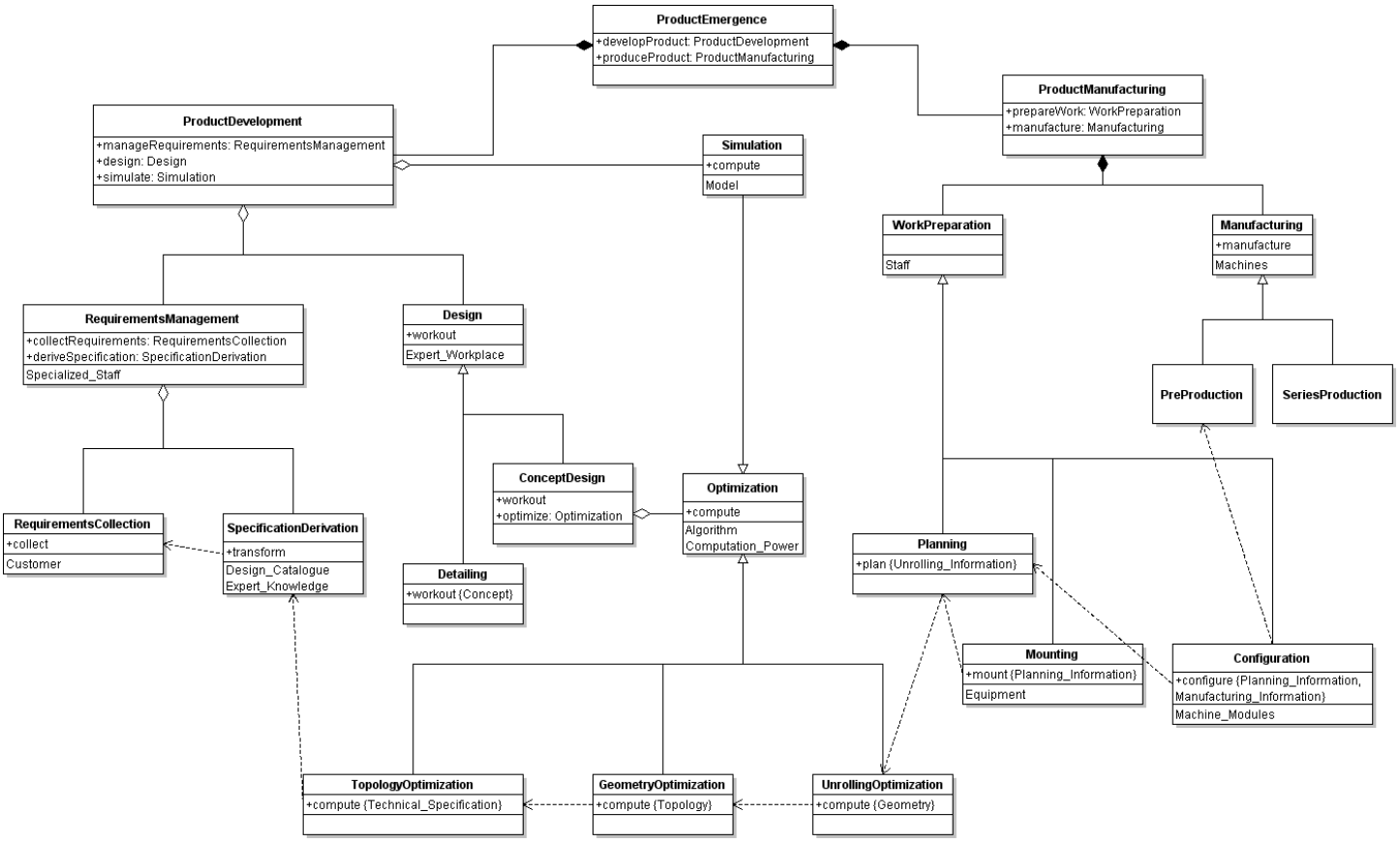

Fig. 7: Process diagram for product emergence of integrated sheet metal products with higher order bifurcations

The interrelation of design parameters and requirements on branched sheet metal products can be expressed in terms of functional equations. These equations are normally non-linear, such as elasticity or heat transfer, and make the corresponding model most complex and highly difficult to solve. To find out feasible solutions very quickly, the mathematical optimization process is decomposed into three stages. The aim of phase one is to calculate the overall topology of the product by solving a coarse mixed-integer programming (MIP) model with linearized functional relations. In stage two, a more detailed product geometry should be obtained by solving a non-linear continuous shape optimization model.

Finally the third optimization stage calculates valid unrollings, i.e. solution trees for the sheet metal product. The process activities *TopologyOptimization*, *GeometryOptimization* and *UnrollingOptimization* represent these activities. They are inherited from the super-process *Optimization*, merely the respective methods are overwritten. Notice the association between the processes indicating dependencies between them. Furthermore the process *Simulation* is also inherited from the super-process *Optimization*. The process *Optimization* itself is a part of *ConceptDesign*, which is inherited from the super-process *Design.*

The result of the mathematical optimization is a set of solution trees, which has to be reduced to the most suitable alternative. A 3D-representation helps the engineer to certify his selection decision. Therefore a 3D-model is automatically generated. The engineer can now manually add further geometrical features to the 3D-model using a CAD-system. The process of generating and refining the 3D-modell is represented by the process *Detailing* which is also inherited from the super-process *Design*.

The generic process *ProductManufacturing* of manufacturing integrated sheet metal products is composed of *WorkPreparation* and *Manufacturing.* The heritages of WorkPreparation are the sub-processes *Planning*, *Mounting* and *Configuration.* In the planning process the sequence of production steps for the facility is determined, depending on the computed unrolling for the sheet metal product. In the mounting process the facility is set up by placing the respective production units in correct order onto the machine bed. Each production unit then is set up individually by specifying parameters and selecting machine settings. Notice the association relations between several process activities, expressing close relationships and dependencies. The super-process *Manufacturing* inherits the sub-processes

PreProduction and *SeriesProduction* and is representing the production process of sheet metal parts.
As the complete process model is expressed in PML, the generic process description remains at a level of utmost flexibility and is clearly structured. This enables all processes dealing with the product emergence of integral sheet metal products to be modeled with this generic process model by instantiating it. The model itself does not alter through instantiating it and remains unchanged. An instantiated process embodies exactly one project representing a specific integral sheet metal product.
UML supports instance diagrams, which basically are class diagrams with the instances built in to show the relations between instances and classes. The instance diagram additionally shows the actual object's occurrence and hence the used resources in our process models. As instance diagrams are not very handsome for complex models we will not use them here. Instead we use other diagrams to show the instances and – more interesting to processes – their timely and logical occurrences. These will be the sequence diagram and the activity diagram respectively. As announced in [13] it is possible to derive sequence and activity diagrams with PML.
Fig. 8 illustrates the activity diagram for the emergence of an integral sheet metal product. In the UML, an activity diagram represents the logical workflows of components in a system and shows the overall flow of control. This approach also fits for PML.

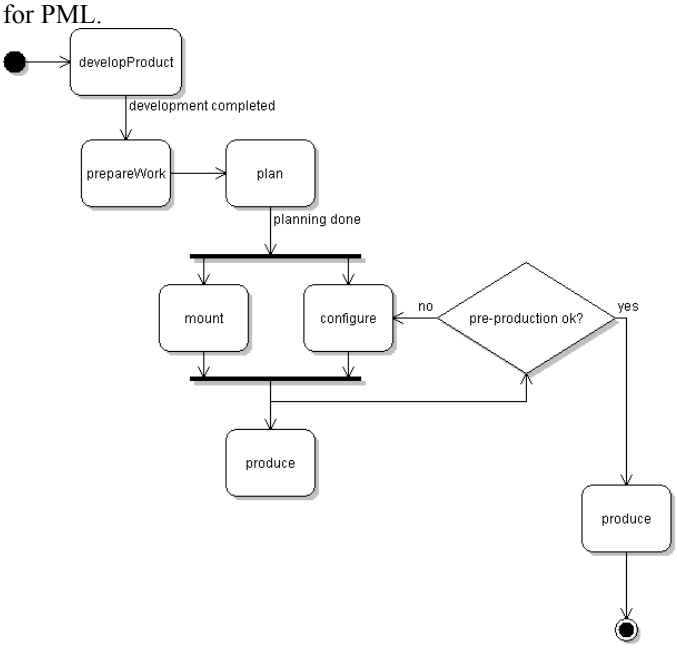

Fig. 8: Activity diagram of manufacturing process

The starting point in the activity diagram is the filled circle in the left top corner. Then the process passes the product development processes, which aren't modeled here. After that the work preparation starts, which is splitted in three phases, planning, mounting, and configuring. The latter two can be processed in parallel order. Both results are needed for a pre-production process, which is checked for high quality outputs. If the pre-production output is ok, the actual production can start. If it's not ok, the configuration process starts in its next iteration to reconfigure the used machines to optimize the production. After the production process the manufacturing is completed, which is shown by the filled circle that has a larger circle around itself in the lower right corner.
Another way to show the instances is by using sequence diagrams. Fig. 9 shows the appropriate sequence diagram for the manufacturing process. As in UML the sequence diagram shows the life time of objects with its construction and destruction events and signals.
Fig. 9 shows the current instances with instance names and the corresponding classes they are instantiated of. *RollerTrailManufacturing* is active for the whole manufacturing process and activates different sub-processes as planning, mounting, configuring, and the series production. The pre-production is constructed and controlled by the configuration to allow iterations to optimize the machine configuration.

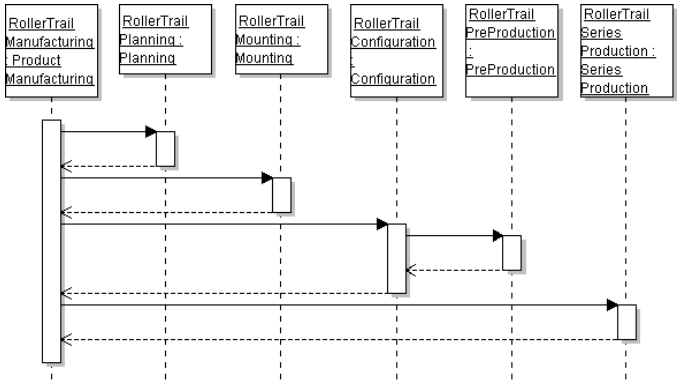

Fig. 9: Sequence diagram of manufacturing process

To show the real strength of PML Fig. 10 details the manufacturing processes. *Manufacturing* consists of *SplittingProcess*, which has three sub-processes: *LinearFlowSplitting*, *LinearBendSplitting*, and *LinearBending*. To instantiate *Manufacturing* different *SplittingProcesses* are instatiated. But, using the concept of object orientation, not *SplittingProcess* but the sub-processes will be instantiated. The same is true for the resources in *Manufacturing*. Machines should hold the specialized machines, which are linear flow splitting machines, linear bend splitting machines, and linear bending machines. Thus having a generic process description, the product and its manufacturing process is dependent of the used machines, the splitting processes and their order in the manufacturing line.
The method names *manufacture* are used in the process classes *Manufacturing*, *SplittingProcess*, and its sub-processes. The *manufacture* method in the *SplittingProcess* and its sub-processes can just be implemented and used, but the *manufacture* method in *Manufacturing* has to be

implemented on its own in the actual instance to specify the manufacturing process for the actual product.

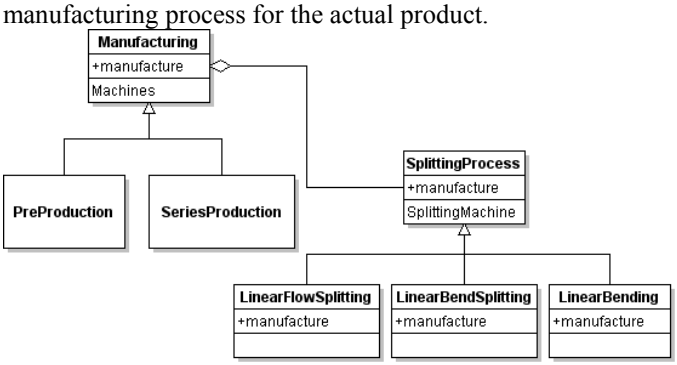

Fig. 10: Detailed manufacturing processes

So the production depends on the resources and the order of the usage of the resources.

CONCLUSION

PML enables users to model processes in a generic, reusable, modularized and object-oriented kind of style. The example application of integrated sheet metal design proves the capability of PML to flexibly handle complex processes and even represent them clearly arranged. This is obtained through the use of the well-know constructs of hierarchical structuring and linking between single processes. The object oriented approach of process modeling introduces a paradigm change not only to the view of process and project management, but also enables new possibilities for interoperability. Heavy use of modularization enables exchangeability and process reusability and hence strengthens the integration of third-party processes. This leads to more powerful cross-enterprise collaboration.

In summary, we briefly benchmarked existing process modeling languages and discovered several deficiencies regarding to proven object-oriented standards, flexibility, exchangeability and many more. Hence we introduced a new approach for an object oriented process modeling language called PML. PML is an extension of UML and was formally introduced. The practicability was pointed out by an example of integral sheet metal products. The paper demonstrates the use of the well-tried sequence and activity diagram also with PML.

In future works the derivation and application of further diagram types in PML will be covered. Furthermore, an approach for a novel CA-toolbox to model processes with PML will be presented.

ACKNOWLEDGMENTS

The results presented here were determined within the framework of a subproject A5 "information model for integrated sheet metal design with higher order bifurcations" of the collaborative research center CRC 666 supported by the German Research Foundation.

REFERENCES

1. IEEE Std 1320.1-1998. IEEE Standard for Functional Modeling Language—Syntax and Semantics for IDEF0. New York: IEEE, 1998.
2. Bernius, P.; Mertins, K.; Schmidt, G. (Eds): Handbook on Architectures of Information Systems, 2nd Edition. Springer Verlag Berlin, Heidelberg (2006)
3. Scheer, A.-W.: ARIS – Business Process Frameworks, 2nd Edition, Berlin, 1998
4. Scheer, A.-W.: ARIS – Business Process Modeling, 2nd Edition, Berlin, 1999
5. OMG: Unified Modeling Language: Superstructure v2.1.1, of Feb 2007, www.omg.org, 2007
6. Eriksson, H.-E.; Penker, M.: Business modeling with UML: business patterns at work. John Wiley & Sons, Inc, New York (2000)
7. Burkhardt, R.: UML – Unified Modeling Language: Objektorientierte Modellierung für die Praxis. Addison-Wesley-Longman, Bonn (1997)
8. OMG: Business Process Modeling Notation Specification, of Feb 2006, www.omg.org, 2006
9. Spur, G.; Mertins, K.; Jochem, R.; Warnecke, H.J.: Integrierte Unternehmensmodellierung Beuth Verlag GmbH (1993)
10. International Standards Organization (ISO): ISO 18629 Series: Process Specification Language of 2004, www.iso.org, 2004
11. Luh, W.: Mathematik für Naturwissenschaftler, Bd.1, Differentialrechnung und Integralrechnung, Folgen und Reihen, Aula, Wiesbaden (1987)
12. Clausert, H., Wiesemann, G.: Grundgebiete der Elektrotechnik 2. Wechselströme, Leitungen, Anwendungen der Laplace- und Z-Transformation, Oldenbourg, München (2000)
13. Anderl, R., Raßler, J., Malzacher, J.: Proposal for an Object Oriented Process Modeling Language, Proceedings of the 4th International Conference on Interoperability for Enterprise Software and Applications (I-ESA), Berlin.

ANNEX A

PUT ANNEX TITLE HERE

Put text of Annex here