Database Scheme Configuration for a Product Line of MPC-TOOLS

Benjamin Klöpper, Tobias Rust,
Bernhard Vedder, and Wilhelm Dangelmaier

Heinz Nixdorf Institute, University of Paderborn,
Fürstenallee 11, 33102 Paderborn, Germany
kloepper@hni.upb.de
http://www.hni.upb.de/cim

Abstract. Data model, planning restrictions, and objectives related to manufacturing planning and control (MPC) strongly depend on the given production process and workshop. In our opinion, these individual properties are the reason, why standard software fails to properly support decisions in MPC. In this paper, we introduce a platform, which enables a configuration process to create affordable individualized MPC software from existing software components.

1 Introduction

Manufacturing Planning and Control (MPC) is a critical task in many industries and it encompasses a large variety of decision problems [1]. MPC posses features which differ from other application areas of decision support and optimization. These special features are the reason for a large number of individual problem formulations and heuristics. The first essential feature of MPC is the high frequency of decision-making. Decisions about production quantities and the matching of quantities to the available production capacities are made, or at least revised, several times a week or sometimes even several times a day. The reason for this high frequency is the changing environment: incoming orders, unexpected scrap rates or machine breakdowns. The second feature is the lack of clearly defined objectives. From the view point of cost-effectiveness the overall purpose of MPC is to achieve a predetermined output performance at minimal costs [2]. Unfortunately, the exact determination of costs in a production environment is often not possible, because the costs either are not continuously documented or cannot be documented (e.g. the long-term costs of delayed delivers are hard to estimate). Thus, usually alternative objectives and constraints are used in MPC. As the alternative objectives bear conflicts, it is not possible to describe a common objective function for MPC problems [3]. On the other hand, experienced human planners are able to modify existing plans in such a way, that they comply with soft factors, which can hardly be included in an automated decision making process (e.g. in-plant work time agreements). For this purpose, MPC tools must

B. Vallespir and T. Alix (Eds.): APMS 2009, IFIP AICT 338, pp. 57–64, 2010.

provide interactive interfaces [4]. Finally, every production system and process has some unique properties. These properties may result from organizational or technical issues. Examples are buffers for decoupling of productions stages (organizational), set-up times or mandatory productions sequences (technical). These individual properties have to be included in order to achieve feasible or even high quality decisions. This opinion is supported by the fact that materials management and production planning are among the most frequently customized modules of ERP systems. Olhager and Selldin [5] present a survey about Swedish manufacturing firms, where 60.7% of the material management and 69.2% of the production planning modules were modified during the implementation.

For these reasons, a custom-made MPC solution for every production system (plant or workshop) is required while there is also a need for intuitive graphical user interfaces and effective planning procedures. Thus, a desirable platform enables the required individuality and cost-effective development process at the same time. The reuse of software may offer a custom-made and affordable MPC software.

2 Software Reuse

"Software reuse is the process of creating software systems from existing software rather than building software systems from scratch." [6] The main motivations for software reuse are gains in productivity by avoidance of redevelopment as well as gains in quality by incorporating components, whose reliability has already been established [7]. Obviously, the modularization and formation of encapsulated software components is an important issue in software reuse. An important step towards this vision was object orientation. According to Nierstrasz et al. "objects provide a paradigm for decomposing large application into cooperating objects as well as a reuse paradigm to compose application from pre-packaged software components" [8]. Szyperski and Pfisters deliver a more accurate definition of a software component and an important differentiation against pure objects: "A software component is a unit of composition contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties" [9].

The paradigm of generative programming goes beyond the idea of component-based software. The idea is not to focus on single software systems, but to create families of software products, from which custom-made variants are generated [10]. The generative domain model is the basis of the concept. It consists of the problem space, configuration knowledge and the solution space. The problem space encompasses domain specific concepts and features, which are described in domain specific languages (DSL). A DSL provides comprehensible but formal methods to describe the customer's requirements regarding the software product. The solution space is the power set overall components of a software product family, from which an appropriate configuration is selected.

3 OOPUS WEB - A Development Platform for MPC

OOPUS WEB is a platform to efficiently create individualized planning and scheduling tools. It provides flexible and adaptable master data functions as well as smart planning interfaces, which enable maximal information transparency for dispatchers. OOPUS WEB is intended to provide components and tools for a MPC tool product line. The basic principle of OOPUS WEB is to decouple MPC algorithms and user interfaces from the data model of the platform. In this way, the large variety of planning algorithms and models is available and may be selected depending on the current application scenario. OOPUS WEB focuses on the area of serial production in line production fashion. This limitation enables the definition of a lean but extensible data model. This data model is the basis of the OOPUS WEB platform. The Model of Serial Manufacturing is described in detail in [11].

Figure 1 shows the task-structure of OOPUS WEB. The task structure is the fundamental architectural concept in OOPUS WEB. The overall task, the detailed planning and scheduling of a given production process is decomposed in several subtasks, each working on a section of the overall model. Such a section or sub model consists in partial subsets of production stages (multiple stage planning methods), a single production stage (single stage planning methods) or even a single line (single machine planning methods). Furthermore, the overall model is also divided into sub models regarding the granularity of planning periods. In that way, it is possible to perform a planning and scheduling on the level of months, weeks, days or shifts or to directly create a machine scheduling down to the minute. To solve a subtask, modules are combined and applied, which can be selected from a toolbox.

These modules are defined task oriented. Two different types of modules and components are distinguished in OOPUS WEB: user interfaces and algorithms. All modules and algorithms are only loosely coupled by the data they

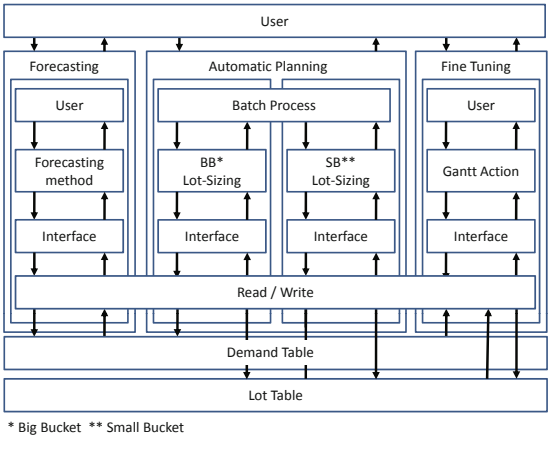

Fig. 1. OOPUS WEB Task Model

visualize and manipulate. Details about the OOPUS WEB task model and the semi-automated generation of the interface layer between modules and database models can be found in [12]. A detailed use case can be found in [13].

Finally, one important part of an OOPUS WEB product is left without tool support so far: the database. The database has to be modified for most companies and plants in order to meet their individual properties and requirements.

4 OOPUS Database Configurator

The OOPUS Database Configurator (ODC) is a tool to create individual database schemes for different applications from the OOPUS WEB family. The system enables the user to make several businesses, technical and organizational oriented decisions in order to generate a matching database model. The next section introduces the concept of ODC. Subsequently the prototype is presented.

4.1 Concept

The basic idea of generative programming is to automate the development of software products. The generative domain model is the basic element of generative programming. It consists of a problem space, a solution space and the configuration knowledge, which defines a mapping between the to spaces (shown in figure 2, for details cf. [10]).

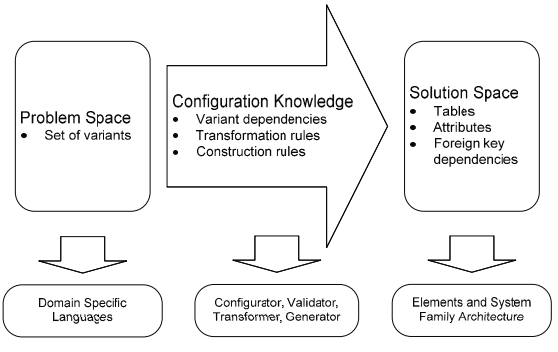

Fig. 2. Generative Domain Model

The user selects different variants to create an individual configuration. These variants provide a high-level domain specific language that enables business or technical oriented users like managers or dispatchers to generate a database scheme. The user requires no technical experiences in database modeling or SQL (Structured Query Language) but can focus on the business and organizational needs of the current plant or workshop. Thus, the problem space consists of a set of variants provided in the configuration tool. The solution space consists of the tables, attributes and foreign key dependencies that represent the database scheme. This database scheme has to be well formed. To ensure this, a validator checks the combination of selected. The required information about the

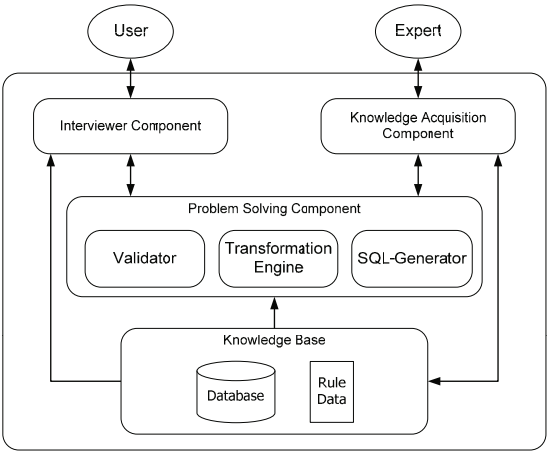

Fig. 3. Architecture of OOPUS Database Configurator

dependencies of the variants is stored in the knowledge space. Another important function of the configuration system is to map the users' point of view on a MPC system on a technical one. Thus, the configuration knowledge contains transformation rules and construction rules. A transformer converts the selected variants into a database model and the generator creates the resulting database scheme in SQL. The ODC follows the general architecture of expert systems introduced by Puppe [14]. It encompasses four basic components (Figure 3): the interviewer component, the problem solving component, the knowledge base, the knowledge acquisition component.

The validator checks the consistency of the selected configuration. Between variants, positive or negative conflicts can exist. A positive conflict arises, if a variant A requires another variant B that is not currently selected. A negative conflict denotes that a variant C must not be selected together with a variant D at the same time. If conflicts arise between two or more variants, they are directly reported to the interviewer component. The user is responsible to resolve them by changing the configuration. When the configuration is finished and no conflicts remain, the database scheme can be generated. This is done by the second component of the problem-solving component, the transformation engine. The transformation engine is the most complex element of the problem-solving component. It has to convert a given set of consistent variants into a set of database tables that define a specific and well-formed database scheme figure.

The knowledge applied in the transformation engine is represented in rules. To simplify the administration of the rules, a second DSL was developed. Compared to the business-oriented variants, this DSL is closer to technical problems. However, it enables the user to define rules in a natural language (cf. figure 4). The transformer generates a database scheme represented as Java objects. This is the input for the SQL-Generator that generates the database scheme in data definition language (DDL). The knowledge used in the problem solving components is stored in the knowledge base.

```
1       when
2        Variant "team-based processing time" is selected
3      then
4       Create table "team"
5       Create field "id_team" in table "times"
6       Create foreign key  from table "times" to table "team"
7    end
```

Fig. 4. Example rule from the knowledge base

4.2 Prototype

The core component of the application is the rules engine. In this case, the open-source software JBoss Rules Engine is used. It encompasses three basic elements (figure 5). The rule base contains the rules that are transformed by a compiler. The working memory contains the facts. The inference engine controls the application of the rules to the facts. To start the process, rule file and the DSL file are imported into the rule base. The DSL file contains the mapping information. Afterwards the application data (variants and table definitions) are loaded into the working memory. When firing the inference engine matches the facts against the rules. Depending on the selected variants, several elements like tables, attributes and key dependencies are activated. Therefore, every element has a flag that marks whether an element is active or not. Finally, the set of marked elements forms the resulting database definition. The prototype enables a user to select domain specific features of the desired product and visualizes possible conflicts. Figure 6 shows a screenshot from the prototype. The left hand side of the screenshot shows the selectable features in a hierarchical structure. The right hand side enables the user to select different product features. The red exclamation marks denotes a conflicts, which is explained by the message box. When the user resolved all conflicts, the rule engines uses the selected features as facts and generates the database definition.

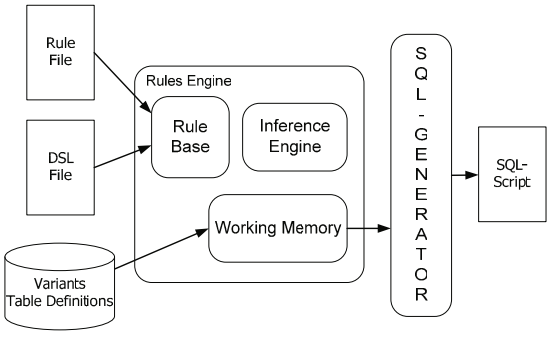

Fig. 5. Sketch of the Prototype

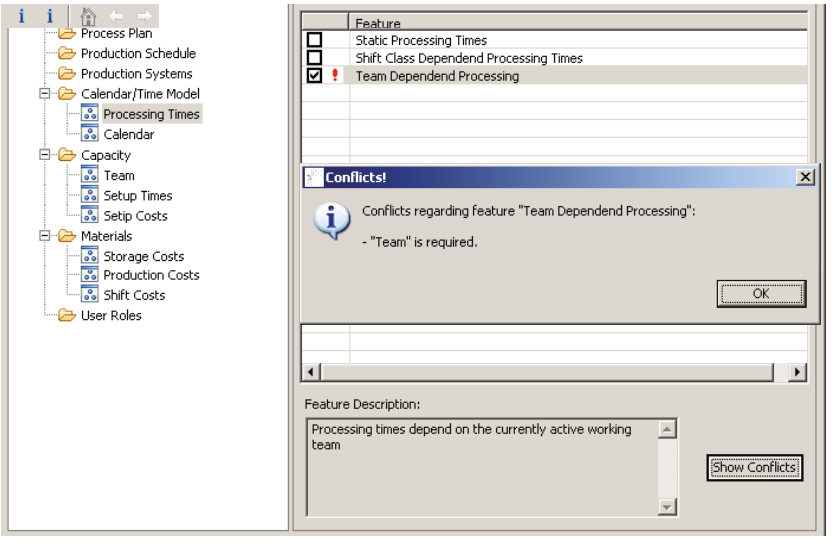

Fig. 6. Screenshot from the Prototype

5 Conclusion

Standard software applications provide unsatisfactory performance in many manufacturing planning and control use cases. The main reason for the poor performance is ignorance of individual properties of manufacturing environments. Only individual solutions can offer the performance required in today's competitive environment. Of course, individual solutions have the drawback of high investment costs. Thus, we introduced a platform, which enables the fast and lean development of custom-made MPC applications for serial manufacturing. The basic idea is the definition of a family of software products, where reusable components are combined in order to meet the requirements of a specific company or workshop. An important part of any MPC system is its database. To model a production system in an appropriate way, the database scheme is a crucial issue. Thus, in order to compose individual MPC solutions, a flexible database is essential. In this paper, we introduced a concept and a prototypical implementation of database-scheme configurator, which enables the definition of individual, well-defined database schemes by selecting the required business-related, technical, or organizational properties. Currently the OOPUS Database Configurator is used in two development projects at German automotive suppliers.

References

1. Tempelmeier, H., Kuhn, H.: Flexible Manufacturing Systems: Decision Support for Design and Operation. Wiley Interscience, San Francisco (1993)
2. Vollmann, T.E., Berry, W.L., Whybark, D.C., Roberts, R.J.: Manufacturing Planning and Control for Supply Chain Management. Taylor-Francis, New York (2005)

3. Fleischmann, B., Meyr, H., Wagner, M.: Advanced Planning. In: Stadtler, H., Kilger, C. (eds.) Supply Chain Management and Advanced Planning. Springer, Berlin (2005)
4. Stadtler, H.: Production Planning and Scheduling. In: Stadtler, H., Kilger, C. (eds.) Supply Chain Management and Advanced Planning. Springer, Berlin (2005)
5. Olhager, J., Selldin, E.: Enterprise resource planning survey of Swedish manufacturing firms. European Journal of Operational Research 146, 365–373 (2003)
6. Krueger, C.W.: Software Reuse. ACM Computing Survey 24(2), 131–183 (1992)
7. Selby, R.W.: Enabling Reused-Based Software Development of Large Scale Systems. IEEE Transactions on Software Engineering 31(6), 495–510 (2005)
8. Nierstrasz, O., Gibbs, S., Tsichritzis, D.: Component-Oriented Software Development. Communications of the ACM 35(9), 160–165 (1992)
9. Szyperski, C., Gruntz, D., Murer, S.: Component Software. In: Beyond Object-Oriented Programming. Addison-Wesley Professional, Reading (2002)
10. Czarnecki, K., Eisenecker, U.W.: Generative Programming: Methods, Tools and Applications. Addison-Wesley Longman, Amsterdam (2000)
11. Klöpper, B., Timm, T., Brüggemann, D., Dangelmaier, W.: A Modelling Approach for Dynamic and Complex Capacities in Production Control Systems. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 626–637. Springer, Heidelberg (2007)
12. Klöpper, B., Rust, T., Timm, T., Dangelmaier, W.: A Customizing Platform for Individual Production Planning and Control Solutions. In: Proceedings of Wirtschaftsinformatik 2009, vol. 2, pp. 77–86 (2009)
13. Klöpper, B., Rust, T., Timm, T., Brüggemann, D., Dangelmaier, W.: OOPUS WEB: A MPC Customizing Platform with Ergonomic Planning Interfaces. In: Proceedings of the 1st International Conference on Business Innovation and Information Technology. Logos Verlag (2009)
14. Puppe, F.: Systematic Introduction to Expert Systems. Springer, Berlin (1993)