2009

A CUSTOMIZING PLATFORM FOR INDIVIDUAL PRODUCTION PLANNING AND CONTROL SOLUTIONS

Benjamin Klöpper
Universität Paderborn

Tobias Rust
Universität Paderborn

Thorsten Timm
Universität Paderborn

Wilhelm Dangelmaier
Universität Paderborn

A CUSTOMIZING PLATFORM FOR INDIVIDUAL PRODUCTION PLANNING AND CONTROL SOLUTIONS

Benjamin Klöpper, Tobias Rust, Thorsten Timm,
Wilhelm Dangelmaier[1]

Abstract
In production planning and control, decision support has to face the special properties of production processes and systems. To find an optimal solution for decision problems in this domain is usually np-hard and a proper definition of objective functions and constraints is extremely difficult. One reason for the problem of proper model definition for classical optimization methods is the individual character of production systems. Thus, in many cases only individual solutions for PPC offer satisfying results. In this paper, we introduce a framework for efficient implementation of PPC tools in serial manufacturing.

1. Introduction

Production planning and control (PPC) encompasses a large variety of decision problems [15]. These decision problems range from the design of new production systems to the determination of production quantities and the sequencing of production lots on a production line. For these problems a large variety of models and algorithms is available (for instance, cf. Afentakis [1], Kimms [16] and Haase [14]), which should be selected regarding the current application scenario. Within this contribution, we will define the special requirements of PPC and introduce a software framework called OOPUS WEB, which enables the fast and efficient development of custom-made PPC tools.

1.1. Problem Statement

PPC possesses special features, which differ from other application areas of decision support and optimization. The design and implementation process of PPC tools has to consider these special features. The first essential feature of PPC is the high frequency of decision-making. Decisions about production quantities and the matching of the quantities to the available production capacities have to be 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 have to be considered.
Another feature is the lack of clearly defined objectives. From the viewpoint of cost-effectiveness, the overall purpose of PPC is to achieve a predetermined performance at minimal costs [26].

[1] Universität Paderborn, Germany

Unfortunately, the exact determination of costs in a production environment is often not possible, because either the costs are not documented continuously or they cannot be documented. Thus, PPC usually persuades alternative objectives. As the alternative objectives bear conflicts, it is not possible to describe a common objective function for PPC problems [12].
On the other hand, experienced human planners are able to describe a decision making process, which leads to good or at least acceptable results [24]. Human planners are also 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.
Finally, every production system and process has some unique properties. These properties may result from organizational or technical issues. Anyways, these individual properties have to be included in order to achieve feasible or even high quality decisions. The following design assumptions are concluded from the features described above:

- Since fast decision making is required, PPC tools should use heuristics
- These heuristics have to be selected with respect to individual properties and objectives
- Thus, the data model must include the individual properties of the production system
- The system must enable human planners to analyze and alter automatically generated production plans

For these reasons a custom-made PPC 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 has always been a major concern of software engineering. Current trends in software engineering like component-based software engineering [7] or service oriented architectures (SOA) [23] place emphasis on the reuse of encapsulated components or services. Especially SOA promises flexible interaction of services by semantic orchestration and choreography. In our opinion**,** these mechanisms are not suitable for applications such as PPC, where the processing of a large amount of data in a short time period is mandatory (cf. next section).
We think that approaches like generative programming [8] are more promising for the domain of PPC. The idea is to model software system families such that, given particular requirements specification, a customized and optimized product can be automatically generated from reusable components. The foundation of the concept of generative programming is a sound domain definition, which is the link between different implementations in the software families.

Reusable components in PPC require a well-defined model of tasks, and interaction of tasks as well as a generic but customizable data model to easily adapt to the current application scenario. Finally, methods for integrating the customized data models with ready-made software components are required. This paper introduces our first step towards a platform for a family of highly customized and optimized PPC applications.

2. Related Work

The common way to solve the various PPC problems is to either run centralized ERP or PPC software. These systems (such as, SAP ECC 5.0 and SAP APO [2], SAGE [27], or Navision [9]) cover many PPC tasks like demand planning, lot sizing, and scheduling using standard algorithms. The task of modeling manufacturing systems in such an ERP system is complex and requires expensive experts. Without remarkable effort and expertise, many ERP projects fail [25]. Since standard ERP and PPC software provide standard algorithms and standard models, it is hardly possible to accurately model a production system in standard software. Hence, these software

products are likely to provide merely inaccurate and inadequate solutions for PPC decision problems.
Brehm et al. [4] suggest more flexible ERP solutions with the application of Federated ERP systems on the basis of Web Services and P2P networks. The basic idea is to choose the current best web service for a given business task. This approach has several shortcomings. It relies on a centrally defined database model, which has to be accepted by all Web Services and ERP systems [5]. This severe prerequisite complicates the inclusion of workshop specific properties in the planning process. Furthermore, the XML based interaction between Web Services is not adequate for mass data usually processed in PPC systems.

On the other hand, there are various planning and optimization algorithms and methods from research and practice, of which many are well documented in literature. These algorithms can be distinguished by their purpose, their objectives and the complexity of the problems. The purpose addresses the current problem solved by the algorithm or the method. It can solve a single problem (such as quantity planning (for instance, cf. Plossl [21]) or scheduling (for instance, cf. Graves [13])) or solve several problems simultaneously (such as quantity planning and scheduling (for instance, cf. Haase [14])). The objectives address the alternative objective persuaded by the algorithm (e.g., minimize delay or minimize storage costs). The complexity finally describes if the methods solves the given problem for complex product structures (multi-level problems, cf. Afentakis [1] or Kimms [16]) and multiple periods. Finally, every algorithm supports different technical properties, like setup times. These methods and algorithms should be selected and combined regarding the current objectives and properties of a given production system.

3. The OOPUS WEB Approach

OOPUS WEB is a platform used to fast and 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.
The basic principle of OOPUS WEB is to decouple the data model from PPC algorithms and other modules of the platform. This way, a large variety of planning algorithms and models is available and can be selected depending on the current application scenario. OOPUS WEB is clearly focused 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. It is called Model of Serial Manufacturing and is described in detail in [19].
The next section outlines the technical implementation of OOPUS WEB. Subsequently**,** we introduce an approach for the flexible integration of planning and optimization algorithms and functional components like user interfaces. An example for the combination of planning methods in OOPUS WEB will be introduced in section 4. This use case describes the practical application of OOPUS WEB beginning with the forecasting of demands**,** continuing with multiple stages / multiple line quantity planning and finishing with single line detailed scheduling.

3.1. Technical Implementation

OOPUS WEB is entirely implemented in Java and all used packages and frameworks are open source. This enables an unproblematic distribution of the software for research and teaching. In the technical implementation, the focus is on flexibility as well. To limit maintenance, a thin client architecture was chosen, and realized by a web browser interface. *Figure 1* shows the overall architecture. A servlet container executes the entire business logic; the users' workstations are limited to the user interfaces. The required decoupling between the presentation and business logic and the persistent data storage is assured by the open source frameworks “Hibernate” and

"Tapestry". Tapestry enables the implementation of web-applications according to the Model-View-Controller paradigm, thus user interfaces can be efficiently adapted to changing requirements [22]. Hibernate [3] decouples the business and presentation logic from the currently used database scheme and automatically creates JavaBean representations of database schemes. Planning interfaces, which feature a more intense interaction with the user**,** are realized as Java applets. To encapsulate business logic and data management on the server, the applets interact with servlets, which implement the business logic and data access.
To embed OOPUS WEB into existing IT infrastructures, it is necessary to interact with classical ERP software. This concerns standing data (such as products and resources) as well as dynamic data (stocks, primary demands, and production schedules). The exchange of data is bidirectional. Up to now, OOPUS WEB implements an interface to the ERP system SAP ECC on basis of the JavaConnector [18].

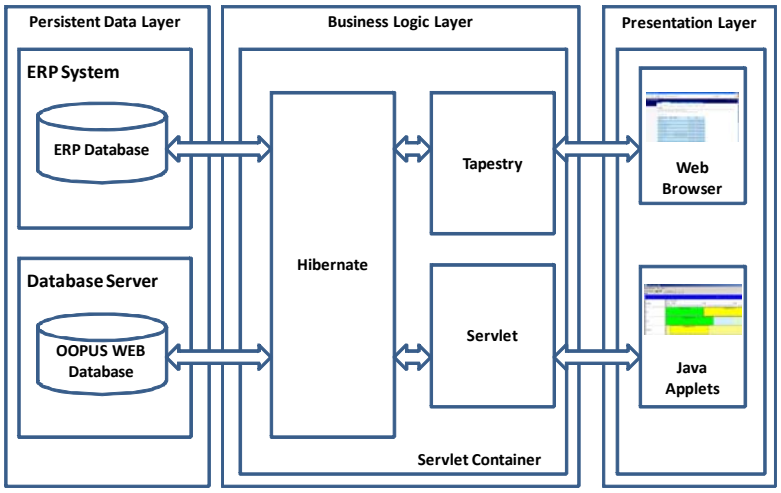

Figure 1: Architecture of an OOPUS WEB Application

3.2. Flexible Combination of Various Planning and Optimization Algorithms

Figure 2 shows the principle idea of the OOPUS WEB platform following the task structure defined in [11]. 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.

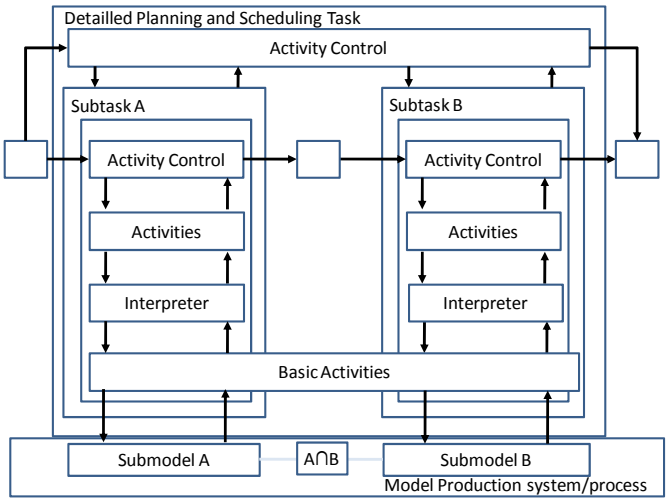

Figure 2: Decomposition of Tasks in OOPUS WEB

Such a section or submodel consists of 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 submodels regarding the granularity of planning periods. This way, it is possible to perform planning and scheduling on the level of months, weeks, days or shifts**,** or to directly create a machine scheduling down to the minute. In order to solve a subtask, planning methods and algorithms are combined and applied, which can be selected from a method toolbox.

3.3. Customizing the data model

The Model of Serial Manufacturing is just the basis of the development of application specific models. Part of the OOPUS WEB platform is a customizing concept for the data model. The basic idea is to provide different variants for the modeling of production processes within production stages. The variants enable the inclusion of different technical properties of the production stages and a varying level of detail. Thus, it is possible to include the particular properties of a planning problem according to the current application scenario.
Technically**,** the customizing concept relies on the representation of a relational database scheme as bill of material (BOM). Therefore, different technical or business oriented functions are encapsulated in a component. A component is a property an OOPUS WEB implementation supports or not. An example of a technical component is the inclusion of set-up times in the model, an example of a business-oriented property is the detailed modeling of inventory costs. The definition of plus/minus bill-of-materials [28] enables the derivation of consistent variants of a database by selecting several relevant components from the BOM representation of the Model for Serial Manufacturing. A system of rules - partially automatically generated from the BOM representation - assures the consistency of the model by forcing the user to consider dependencies and conflicts between the components during the configuration.
The customizing concept encompasses two different processes: the process of creating a new configuration depending on an application scenario and the extension of the pool of components. Two wizard-like tools support both processes. Since the configuration and extension address different people in the development process, two independent Java applications were developed. The software automatically performs many process steps - like the derivation of conflicts and dependencies from the BOM.

3.4. Automated Interface Generation

If the customizing concept is applied, different OOPUS WEB implementations may possess different database schemes. On the other hand, the methods and algorithms from the method toolbox and OOPUS WEB components possess their specific internal representations. Thus, for every combination of database scheme and method a proprietary interface is required. To restrict the implementation effort, an automated interface generation was developed. The basic idea is to define a mapping between the JavaBeans (automatically derived from the database scheme by Hibernate) and the internal model of input and output variables of the different methods and components. The mapping is defined by an XML file, where several blocks describe the derivation of input parameters from the possible JavaBeans or the re-calculation of output variables. Elements of such a block are parametrizable Hibernate queries, which specify the required data from the database and calculation instructions, which calculate the values of the input parameters from the JavaBeans. The XSLT code generation [6] creates an individual interface class. Thus, the interface classes implement the interpreters in *Figure 3*.
The automated interface generation supports two different types of interfaces. The first kind of interface treats the methods and algorithms as a black box. It derives all required data, calls the algorithm, receives the results of the calculation, and writes them into the database. This kind of interface is suitable for many planning and optimization algorithms and requires no modification or

specific programming within the implementation of the algorithm. This is an important property, if existing implementations shall be reused. The left part of *Figure 3* illustrates the processing scheme of the black box interface.
On the other hand, this kind of interface implies that all required data is known in advance. This condition is not true for every algorithm and every component of the platform. Especially when user interaction is involved (like in manual planning), it is not possible to comply with this precondition. Thus, the second kind of interface allows a more client/server-like interaction between the methods or components and the database. Of course, these interfaces require an OOPUS WEB specific programming style. The methods or user interfaces must be able to call get/set methods provided by the interface. The client/server-like interface supports interoperability and distributed computing by the application of CORBA technology.

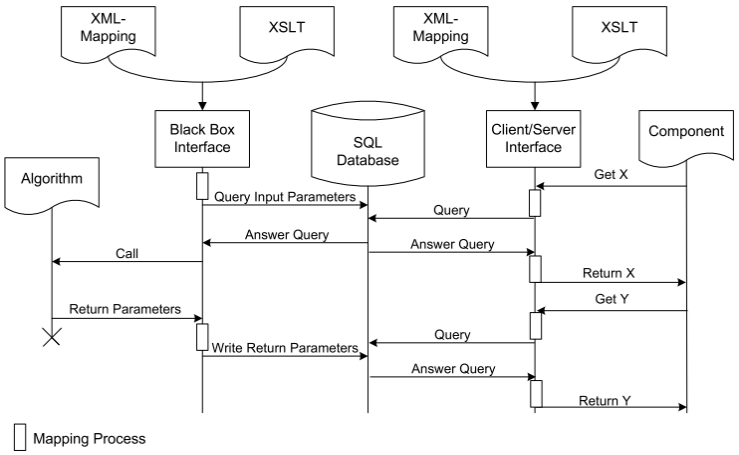

Figure 3: Interaction Scheme for Automatically Generated Interfaces

3.5. User Interface in OOPUS WEB

Experiences from earlier PPC projects were used to provide the required information transparency [20]. These projects have shown that ergonomic graphical user interfaces are crucial. Practical experience proved that a combination of a planning table and an accumulated quantity table (AQT) offer excellent support for detailed scheduling tasks.
The planning table, implemented as a gantt chart, displays lots on production lines in a chronological order. The dispatcher can directly interact with the production schedule. He is able to trigger lot-oriented functions like shifting lots or changing the lot size. The AQT presents a production schedule, handled in the planning table, in aggregated fashion as calculated cumulated quantities. The (future) production progress is set into relation with the primary and dependent material requirements. The two interfaces are implemented in a parametrizable version, which provide a basis for fast implementation of application specific user interfaces.
Both, planning table and AQT are components of several PPC applications. However, in our opinion only the integrated use of both planning tables enables the information access required for planning. To use both interfaces simultaneously, a two-display-mode is suggested. Whenever a change is made in the planning table, the long-term results are visible in the AQT. Thus, it is possible to create an initial plan (using algorithms from the toolbox) and analyze this initial plan in the AQT regarding backlog and capacity utilization**,** and tune the plan manually using the planning table. For a detailed description of the interaction between the tables, cf. [20].
While the AQT is a generic concept used to visualize quantities and delivery status in PPC detailed planning and scheduling, the current visualization in the planning table depends heavily on the properties of the presented production stage. For example**,** depending on the production stage, setup times or information about working teams has to be shown. Thus, different versions of planning tables are provided by the framework and are dynamically linked**,** depending on the type of the

current production stage. The client/server like interface (section 3.4) can be used to decouple the several planning table implementations and the changing data model.

4. OOPUS WEB use case

This section introduces a use case for OOPUS WEB. It ranges from a long-term forecasting over all products to a detailed scheduling for a single line. It is shown how several components and algorithms can be combined to fulfill a complex PPC task.
Figure 4 shows the use case according to the decomposition in section 3. The overall process is controlled by the user. The user starts the three main steps forecasting (1), automatic planning (2) and fine-tuning of production plans (3).

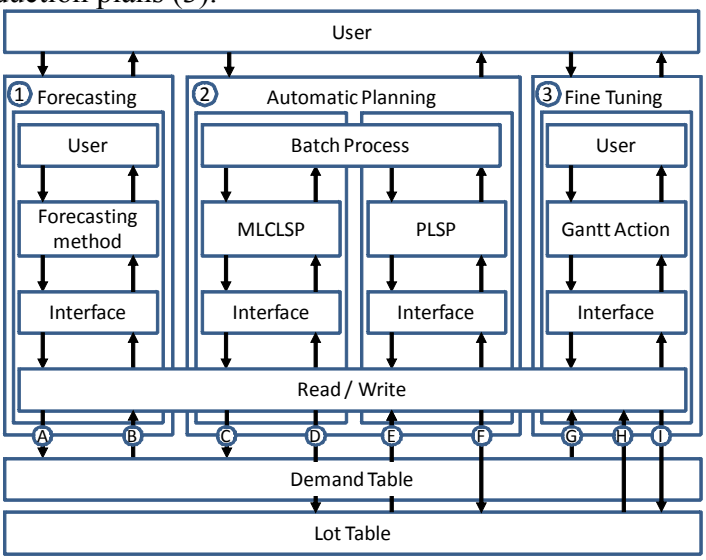

Figure 4: A use case of OOPUS WEB

The three steps are only coupled by the underlying database model. For the sake of simplicity, we just included two database tables in *Figure 5*, although the use case obviously requires information about the structure of the production system and the material as input. The demand table contains the demands resulting from actual client call-offs as well as forecasted demands. The lot table provides information about which quantity of which product is produced on which production line at what time. The time can vary regarding its granularity. Thus, it is possible to define, for example, daily lots that specify which quantity of a product has to be produced on a production line during a specific day, and on the other hand, to provide lots with to-the-minute start and end times. The varying granularity is required for the flexible combination of planning algorithms and problems.

The forecasting component reads historical demands from the demand table (A) and writes forecasts into it (B). The automatic planning process is implemented for the sake of simplicity by the two well-defined and well-known optimization problems: Multi Level Capacitated Lot-sizing Problem (MLCLSP) and Proportional Lot-sizing and Scheduling Problem PLSP. A batch process executes them. The MLCSLP accesses client call offs and forecasted demands (C) in order to determine *day lots*. These day lots are written into the lot table with a flag identifying them as *day lot (D)*. The PLSP reads the day lots (E) and determines a detailed sequence of lots. This result is also written to the lot table (F). Finally, the user tunes the automatically generated plan with the planning interfaces. In this process, information from the demand (G) and the lot tables (H) is read and created, and deleted or changed lots are written to the lot table (I).

All interfaces in *Figure 5* are generated by the automated interface generation introduced in section 3.4. The interface generator converts a model in form of a XML file to a Java class, thus just the definition between the internal model of the component or algorithm has to be defined. The current read and write operations on the database are performed by the Hibernate framework.

4.1. Forecasting

In most industries**,** demand information based on actual customer call-offs are only available within a limited time horizon. If a longer planning horizon is required, a prediction of demands in the later periods is needed. Thus, a forecasting component was developed and integrated into OOPUS WEB (*Figure 5*).
This component enables forecasting of demands with several customizable methods. Information available on the internet, like economic growth or the oil price, can be integrated in the model. The user directly controls the forecasting process. A benchmarking process supports the user in choosing the best forecasting method by comparing the available methods and suggesting the most suitable method for the given data. The result of the forecasting process can be saved in the demand table and is used as input for production planning.

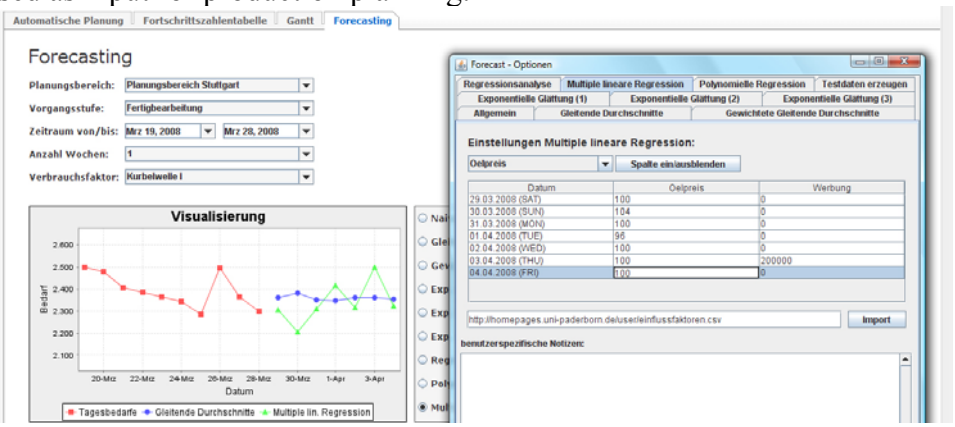

Figure 5: Forecasting Component

4.2. Automated Planning

The next step in the use case is a multi stage method to determine production quantities on an, for example, daily or weekly basis. OOPUS WEB can generate those plans either using optimal methods or heuristics. Anyway, the results of the forecasting in the previous step are used as input data (demands to meet). For the application of optimal methods, the solver is called. With a solver, problems like the MLCLSP (cf. [17]) can be solved to determine production quantities. The result of the optimization is production quantities for each period of time, each production line, and each product. These quantities are saved in the lot table as day or week lots. Since MLCLSP does not determine a production sequence, it is not possible to directly consider sequence depending setup times stored in the database. Thus, the generated interface can be used to calculate an average sequence independent setup time for each product. This average setup time can be considered in the restrictions of the MLCLSP.
The third step is the automatic planning of single production stages and single lines on a more detailed level. The day lots generated in the previous step are used as input data. An example of a problem applied in this step is the PLSP (cf. [10]). The PLSP takes the week or day lots as input and determines a detailed sequence of lots that offers the best compromise between setup and inventory costs. The result of the PLSP is lot information with exact quantities and start/end times. It is written to the lot table.

The final production-planning step is the manual adaptation of the automatically generated plans by the production planner. This is done by the user interfaces described in 3.5.

5. Conclusion and Further Work

The platform OOPUS WEB enables the derivation of data models for individual software solutions from a global master model and the continuous extension of this global model. OOPUS WEB enables the application-oriented combination of methods and algorithms from a toolbox, to provide the best solution for every serial production system. The implementation effort is limited by automated interface generation.
Based on experience from previous industrial projects OOPUS WEB provides two interacting planning interfaces for the production planning and scheduling, which provide maximum information transparency and efficient tuning possibilities for the human dispatcher. These interfaces are either parametrizable or provided in different variants, which can be combined with changing data models by the automatically generated interfaces.
Up to now, OOPUS WEB supports classical detailed planning and scheduling functions like quantity planning and machine scheduling. In the near future, more tasks from the PPC area shall be supported by OOPUS WEB. An example for this is the demand-oriented definition of shift plans or assignment of production steps to stations on an assembly line. These fundamentally different tasks require alternative user interfaces.
By continuously extending the platform - in research, teaching and application – OOPUS WEB is intended to grow to a comprehensive construction kit for PPC applications and restrain future development efforts.
The basic ideas behind OOPUS WEB - the definition of a common data model, which can be extended and modified within a customizing concept and the automated interface generation to apply existing and new components within the platform, can be transferred to other areas than production planning and control.

References

[1] AFENTAKIS, P., GAVISH, B.: Optimal Lot-Sizing Algorithms for complex product structures, in: Operations Research, Bd. 34 (1986), 237-249.

[2] BALLA, J.: Production Planning with SAP APO-PP/DS, Bonn 2006.

[3] BAUER, C., KING, G.: Java Persistence with Hibernate, Greenwich 2004.

[4] BREHM, N., MARX GOMEZ, J., RAUTENSTRAUCH, C.: An ERP Solution Based on Web Services and Peer-to-Peer networks, in: International Journal of Information Technology and Management, Bd. 1 (2007).

[5] BREHM, N., MARX GOMEZ, J.: Web Service-Based Specification and Implementation of Functional Components in Federated ERP-Systems, in: Abramowicz, W. (Hrsg.), Business Information Systems 2007, Lecture Notes in Computer Science, Bd. 4439, Berlin 2007, 133-146.

[6] CLEAVELAND, J.C.: Program Generators with XML and Java, Upper Saddle River 2001.

[7] CRNKOVIC, I.: Component-based software engineering - new challenges in software development, in: Proceedings of the 25th International Conference on Information Technology Interfaces, New York 2003.

[8] CZARNECKI, K.; EISENECKER, U.W.: Generative Programming: Methods, Tools, and Applications, Reading 2000.

[9] DIFFENDERFER, P.M., EL-ASSAI, S.: Microsoft Navision 4.0: Jump Start to Optimisation, Wiesbaden 2005.

[10] DREXL, A.; HAASE, K.: Proportional lotsizing and scheduling, in: International Journal of Production Economics Bd. 40 (1995), 73-87.

[11] FERSTL, O.K.; SINZ, E.J.: Grundlagen der Wirtschaftsinformatik, München 2006.

[12] FLEISCHMANN, B.; MEYR, H.; WAGNER, M.: Advanced Planning, in: Stadtler, H.; Kilger, C: (Hrsg.). Supply Chain Management and Advanced Planning, Berlin 2005.

[13] GRAVES, S.C.: A Review of Production Scheduling, in: Operations Research, Bd. 29 (1981), 646-675.

[14] HAASE, K.: Lotsizing and scheduling for production planning, in: Lecture Notes in Economics and Mathematical Systems, Vol. 408, Berlin 1994.

[15] HIGGINS, P., LE ROY, P., TIERNEY, L.: Manufacturing Planning and Control: Beyond MRP II, Berlin 1996.

[16] KIMMS, A.: Multi-Level Lot Sizing and Scheduling, Heidelberg 1997.

[17] MAES, J.; VAN WASSENHOVE, L.: Capacitated Dynamic Lotsizing Heuristics for Serial Systems, in: International Journal of Production Research, Bd. 29 (1991), 1235-1249.

[18] MEINERS, J.: SAP Interface Programming, Bonn 2004.

[19] DANGELMAIER, W.; TIMM, T.; KLÖPPER, B. BRÜGGEMANN, D.: A Modelling Approach for Dynamic and Complex Capacities in Production Control Systems, in: Abramowicz, W. (Hrsg.), Business Information Systems 2007, Lecture Notes in Computer Science, Vol. 4439, Berlin 2007, 626-637.

[20] DANGELMAIER, W.; RUST, T.; HERMANOWSKI, T.; BRÜGGEMANN, D.; KASCHULA, D.; DÖRING, A.; TIMM, T.: OOPUS - A Production Planning Information System to Assure High Delivery Reliability Under Short-term Demand Changes and Production Disturbances, in: Proceedings of the Ninth International Conference on Enterprise Information Systems (ICEIS 2007) - Databases and Information Systems Integration, Setúbal 2007, 423-430.

[21] PLOSSL, G.: Orlicky's Material Requirements Planning, 2nd edition, New York 1994.

[22] SHIP, H.M.L.: Tapestry in Action, Greenwich 2004.

[23] SINGH, M.P.; HUHNS, M.N: Service-Oriented Computing: Semantics, Processes, Agents, Chichester 2005.

[24] STADTLER, H.: Production Planning and Scheduling, in: Stadtler, H.; Kilger, C.: Supply Chain Management and Advanced Planning, Berlin 2005.

[25] VOGT, C.: Intractable ERP: A Comprehensive Analysis of Failed Enterprise-Resource-Planning Projects, in: Software Engineering Notes, Bd. 27 (2002), 62-68.

[26] VOLLMANN, T.E.; BERRY, W.L.; WHYBARK, D.C.; ROBERTS, R.J.: Manufacturing Planning and Control for Supply Chain Management, New York 2005.

[27] WALLACE, T.F., KREMZAR, M.H.: ERP:Making It Happen: The Implementers' Guide to Success with Enterprise Resource Planning, Chichester 2001.

[28] WOSS, W.: A rule-driven generator for variant parts and variant bills of material, in: 8th International Workshop on Database and Expert Systems Applications, New York 1997.