Multi-Layer Architecture for Transition of Business-Models to common Software-Tools and Optimization of the Model-Structure exemplified with Microsoft SharePoint 2010

Sebastian Lauck
Heinz Nixdorf Institute, University of Paderborn,
Paderborn, 33102, Germany

and

Dr. Christoph Laroque
Heinz Nixdorf Institute, University of Paderborn,
Paderborn, 33102, Germany

and

Philip Hartmann
Heinz Nixdorf Institute, University of Paderborn,
Paderborn, 33102, Germany

ABSTRACT

This paper describes a model to transfer established business models like bml models into a new dynamic structure which is immediately usable in standard software and portable between different systems without complete reengineering. Many products like Microsoft SharePoint provide toolkits for work-flow development but these workflows are stored commonly in a proprietary way and are not easy alterable and adoptable.

Aside from the possibility to embed complex business models in standard software, this approach supports runtime altering of the workflow structure in terms of optimization of the process structure to different targets like time, cost and quality. An exemplary extension to the basic model is given which describes a capacity limitation in single process steps. To respect these bounds it is necessary to embed artificial constraints in the workflow structure.

To reach the goal of a loose coupled dynamic structure a multi-layer architecture was developed, in which dynamically connected objects are used to represent the formal model structure. This approach allows a translation of process-descriptions into requirement-definitions which can be stored in relational database structures.

The approach was validated by developing a prototype based on SharePoint 2010. A short description of the prototype follows the conceptual introduction.

Keywords: Business Processes Models, Optimization, BML, Network Planning, Process-Architecture

1. INTRODUCTION

Standardization of business processes keeps being an actual topic in different types of organizations. Causes are manifold, e.g. the implementation of management systems or the establishment of a quality management system like ISO9001. Business processes can be taken as time- and space- coordinated goals of the management, while input- and output- parameters have to be known [2,4,6].

Different types of models, describing activities and business processes have been developed. Popular examples are the EPC (entity process chain) and the PDM (precedence diagramming method) described in DIN69900. EPC models are used primarily to describe logical relations between single steps or tasks. PDMs focus on the determination of durations for single activities and buffer-times resulting from the relationship between single steps [3].

Logical connections like OR and XOR aren't defined in PDM-models whereas EPCs lack of specific time definitions. The solution presented on the following pages enables the developer to describe different kinds of relationships (OR/AND/XOR) in combination with temporal properties (END-START/END-END/START-START/START-END) between tasks as well as the definition of durations for each step.

Afterwards defining and modeling business processes a common step is the implementation of these models in IT-systems. Established workflow management systems enable this step by providing toolkits. Mostly a static defined business process, which can be executed and monitored, is the result of this implementation [13].

In the end the singular definition of business processes isn't satisfying in each case: On the one hand the model can differ from the actual process, on the other hand there is the possibility that processes change over time, or have to be treated in dependence of framework conditions. Quick changes in models—at best performed by the system itself—have to be possible. Herbst distinguishes between ad-hoc and evolutionary changes to the model: Ad-hoc changes describe the modification of the structure at runtime and evolutionary modifications alter more fundamental parts of the process based on variance analyses [8].

Ad-hoc as well as evolutionary changes on business process models can be made in workflow management systems—in general—only by intervention at development level with considerable expenditure, caused by the static nature of the internal workflow toolkits. Automated structure optimization isn't considered in most systems yet.

Therefore the main goal of this work is the development of an architecture, which defines the business process in a less static structure. This structure should be customizable by (authorized) users as well as by the system itself. Furthermore it must be possible to instantiate and run the business processes with support of the workflow tools the systems have on-board. Experiences and usage data are collected and stored in an appropriate repository and can be used for further analysis and as basis for optimization.

An extension of the model, which needs automatic structure modification beyond evolutionary changes, is the introduction of additional restrictions. Exemplary the extension of capacity limits in single process steps is described later. Capacity limits can be limited transport volumes, human resources or limited throughput of plants and machines. It's clear that processes cannot be instantiated randomly in such cases. The introduction of artificial dependencies between single process-steps, as well as usable methods for finding valid solutions permit optimal schedules for new instances.

First we will present the developed architecture in the following chapter, which will be extended with exemplary optimization methods further on. The realized prototype, which is based on Microsoft SharePoint Workflows, will be presented and described in chapter 3.

2. THREE- LAYERED ARCHITECTURE

This solution is based on a three-layer architecture (figure 1), where the first layer consists of information about the global workflow structure and a repository for experienced data. Based on this information in Layer 2 a cycle-free PDM-structure is generated for each workflow-instance, representing the predicted times for each process step. Each node in Layer 2 is connected to a fixed description in Layer 3 where detailed information is provided how to handle the given task.

New instances of the workflow shall always be generated on basis of experienced runtimes and delays; therefore a static definition of desired values is unnecessary. The required forecasting-module is defined in Layer 1. Global structure definitions describing the whole process and dependencies between single steps are the second element in Layer 1. This is crucial to obtain correct and runnable instances. This description is based on relational tables and will be discussed in section 2.1.1. A possible extension using an optimization algorithm to retrieve valid solutions for a capacity-bounded system is given in section 2.1.3.

The development of a so-called meta workflow element (MWE), which describes one single step of a workflow (or just a logical connecting expression) for a unique instance (Layer 2), is a fundamental part of this approach. Each MWE describes the planned schedule, links to the logic for the connected process-step and is stored in the knowledge base where it is extended with its realized durations after the step has terminated. The full description of the MWE is given in section 2.2.

Layer 3 provides specific execution rules and structures separately for each task. The integration and connection between MWE and task-description for dedicated single steps is shown in section 2.3. Layer 3 is used to reduce the complexity of the system-independent process-description. Therefore it is differentiated between dividable and undividable process steps: undividable process steps, which have to be progressed in the same way every time, can be coupled to one Layer 3 item. This approach leads to significant faster results in terms of optimization. The choice between a high level of detail in Layer 1 (more data and details in the knowledge base) or Layer 3 (faster calculation) is left to the workflow-developer.

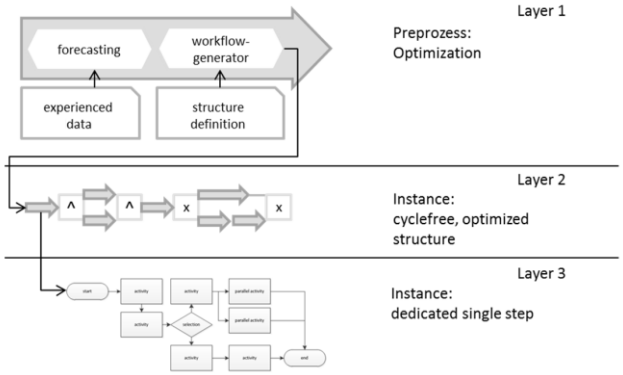

Figure 1: *Relationship between the three layers. As result of the preparation-steps in Layer 1 a unique instance is generated for execution in Layer 2. The instance in Layer 2 contains only MWEs. Each MWE, which is not a logical connection between different MWEs is bidirectional connected with a description given in Layer 3.*

2.1 Layer 1 – Network generation and optimization

Like illustrated above, Layer 1 includes two main parts to create an optimal workflow instance for a current situation. A forecasting-module is started initially, to predict the runtime for each single step according to the information stored in a repository with experienced data. This forecast is made for each undividable set of single tasks given in the structure definition. We give no proposals for different matching forecasting algorithms here, because the type of approach is strongly dependent of the concrete case. Most approaches have in common that the given information has to be transformed to time-series before analysing the deeper structure and predicting future values. Motif-based search is a possibility to gather forecasts based on time-series data like the approach developed by Lin, Keogh and Lonardi [10, 11].

Based on the calculated runtimes for each step and unambiguous dependency-descriptions, a graph algorithm (workflow-steps and dependencies as nodes) returns a complete schedule for the observed workflow-instance. An optional structure optimization can be used as pre-process of the workflow generating graph-algorithm.

2.1.1 Definition of workflows with dependency-definitions and logical connections: An adequate structure- and dependency-definition is mandatory for the network-generation. To accomplish platform independence the developed structure is based solely on lists / relational tables.

Figure 2 shows the basic approach of a list-based dependency definition. Goal of the workflow generation is an automated detection of parallelizable tasks – in figure 2 the tasks B/C and C/D.

Nevertheless this way of modeling is not sufficient to represent complex business-process models. Many business-processes contain split and join operations, representing logical OR-/ AND-relations. The dependency-definitions have to be extended with logical information accordingly like represented in figure 3.

Now the network generator has to note that B or C have to be completed before D can start. D cannot be scheduled immediately after B, because there is a possibility that B gets delayed and C ends earlier.

x	y → x	t(x)
A	Ø	3
B	A	2
C	A	4
D	A,B	3

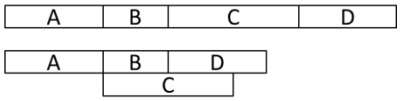

Figure 2: *Dependency-definition for single steps with given duration*

x	y → x	x → z	Typ
X_1	A	B,C	XOR
X_2	B,C	D	OR

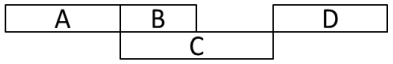

Figure 3: *Extension of the dependency-definition with logical statements*

2.1.2 Workflow-generation: Based on the de-pendency-descriptions and the computed running times a set of meta-objects is generated, which represents the optimal structure and duration for the unique instance. The generation of the elements and retrieval of the optimal start- and end- times for each process step can be realized using a graph-based algorithm like illustrated in listing 1. The explained code is strongly simplified and misapplies the check for loops and approaches to determine connected predecessors and successors. Nevertheless, it is sufficient to demonstrate the common idea. For each node *wfItem* the earliest possible start-time is defined by the latest possible end-time of all predecessors. The function *getReadyTasks* retrieves all following tasks, which become available by finishing the current progressed node (it is possible that successors have multiple dependencies and don't get available by completing *wfItem* immediately). For each succeeding node the same procedure is executed recursively.

One meta-element is computed for each processed *wfItem*, with inclusion of the given information about successors and predecessors and the calculated start-and finish-schedule. The underlying structure is explained in section 2.2.

```
   Input: Initial Node wfItem
   finishTask(Node wfItem)
1  StartTime = 0;
2  Foreach Predecessor in getPredecessors(wfItem)
3          If Predecessor.startTime + Predecessor.Duration >
             StartTime then
4            Starttime = Predecessor.startTime +
             Predecessor.Duration
5  wfItem.startTime = StartTime;
6  wfItem.endTime  = StartTime + wfItem.Duration;
7  addToFinishedTasks(wfItem)
8  Foreach WFItem Successor in getReadyTasks(wfItem)
9          finishTask(Successor)
```

Listing 1: *Recursive graph-algorithm to obtain the earliest possible start- and end-times for each task*

2.1.3 Example — Optimization with constrained resources: By extension of the given dependency-descriptions with information about resource-demands for each task, like shown by the example in figure 4, the complexity of the problem rises largely. Therefore not only the single tasks have to be considered, but rather the set of all instantiated and unfinished workflows. In the example two new instances have to be scheduled, while a single step (B) demands a bounded resource. In this case an optimal solution is obvious, but it should be clear that optimal results cannot be given in a trivial way when computing more complex models with large sets of instances.

One approach to retrieve optimal schedules facing constrained resources was published 1983 by Bartusch. It is based on the introduction of forbidden sets, which represent the parallel execution of tasks leading to the violation of one or more constraints. These forbidden sets have to be solved by creating artificial dependencies, which define valid sequences for all tasks. Using a branch-and-bound algorithm the optimal schedule can be obtained [1].

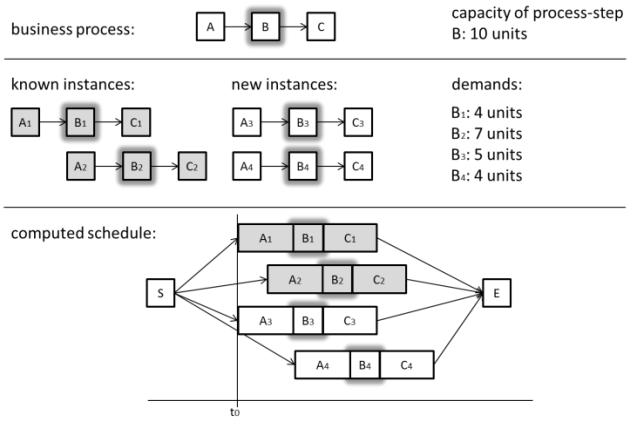

Figure 4: *Dynamic scheduling over parallel business process instances*

2.2 Layer 2 – Meta-Objects for workflow-accomplishment

To achieve the goal of a system-independent workflow structure it is necessary to model the business-logic separate from the built-in workflow functions of each standard-system. Therefore Layer 2 can be understood as abstraction layer between system-functions and the workflow model. Layer 2 represents a concrete, optimized process-instance. Each process-step is represented by a meta-object, which contains all necessary information like type, runtime-instructions, predecessors and successors as well as planned and realized times as attributes. Figure 5 illustrates the attributes of each meta-element.

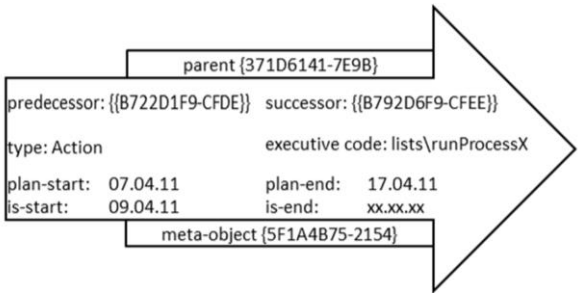

Figure 5: *MWE with attributes*

The inner logic of the meta-object can be implemented with the default toolkit of standard-workflow software. The distinction between logical connectors and task-representing meta-elements is elementary, like shown in figure 6, where the inner logic of meta-objects is given: first it is checked if the MWE represents a task or a logical expression.

A logical synchronizing object (AND) would wait until all preceding tasks have been finished and start each defined successor afterwards. A logical OR will activate the successors immediately.

Task-representing meta-objects are distinguished too: there are self-starting and external-started objects. On the one hand self-starting objects instantiate and run immediately when they are ready- they represent tasks that can be accomplished by the user himself (e.g. filling out a template with given data). On the other hand external-started objects are instantiated immediately, but cannot be ran until an external event triggers the object. These types stand for tasks which will be executed by clients or partners and the system has to wait for the results (e.g. waiting for a delivery).

By the definition of a generic object, which can represent as well tasks as logical connections, it is possible to transform workflow models with easy computation into nearly every standard workflow system. The necessary data can be stored in list-structures or simple objects. The structure of the meta-elements can be built upon standard workflow systems with given toolsets; 1:1 connections are made by setting attributes to the meta-objects.

2.3 Layer 3 – Runtime-directions for undividable process-steps

Each task–associated meta-element is connected to an explicit runtime-description which reflects exactly the way to handle the according process-step. This description can be defined and implemented with tools given by the workflow-management-system and can be instantiated and started either by the user or the preceding meta-element. An appropriate interface between meta-element and runtime-description is elementary for the interaction between these elements. The sub processes in Layer 3 have to be extended with functions which store reference-information to the meta-object at runtime and functions which return termination information when finishing the Layer 3 description to continue the workflow according to the meta-objects in Layer 2.

3. PROTOTYPE

The introduced concept has been prototypal implemented based on Microsoft SharePoint 2010. Goal of the development was to model a standard purchase-process and the abstraction of the model to the illustrated dependency-structure. The prototype has further been extended with the forecasting component which was explained with Layer 1. This component retrieves future runtime values for new instances on basis of finalized instances. SharePoint offers out-of-the-box document management and workflow capability and toolsets. This enables the implementation of Layer 2 meta-elements and the task de-scriptions (Layer 3) immediately.

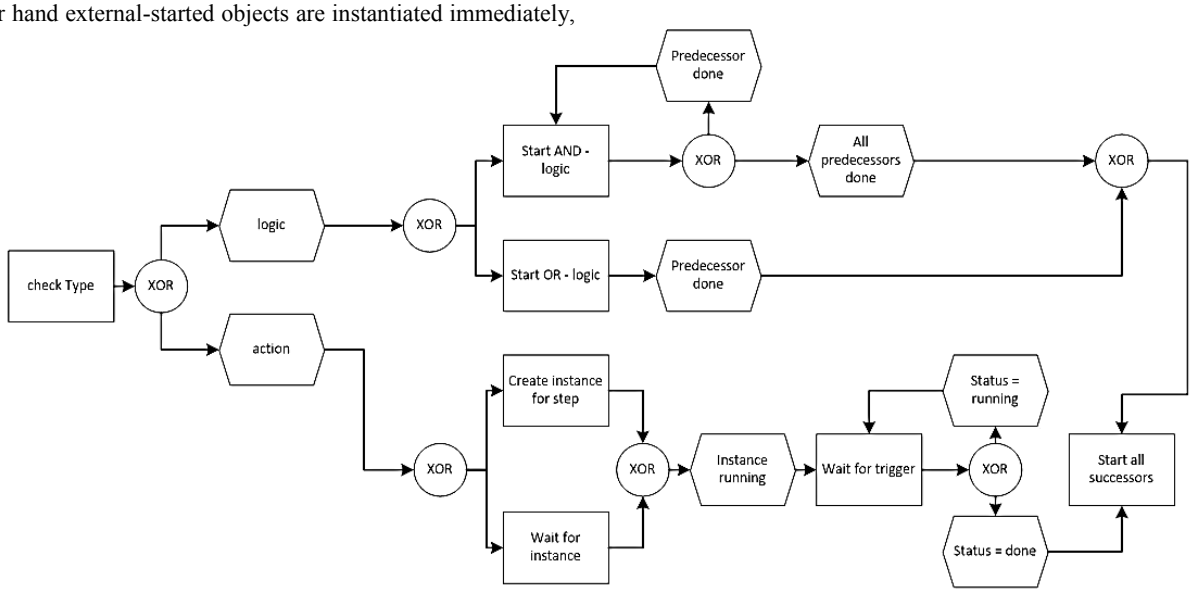

Figure 6: *Runtime logic for meta-objects*

3.1 Particular case model and logical connectors

The realized workflow is based on standardized descriptions given in VDI 4400 part 1 [15]. However, the description of the operative purchasing is branched to the steps disposition and order. To model a more complex process the steps in VDI 4400 have been extended with information in standard literature [7]. The prototype has been expanded with elements to gather deliverer-ratings and to realize an automated selection of the best deliverer for a single instance. To keep the focus on the underlying structure we won't give more details about these components in this paper. The resulting process is illustrated in figure 7; starting with a demand several steps have to be accomplished to cover this need.

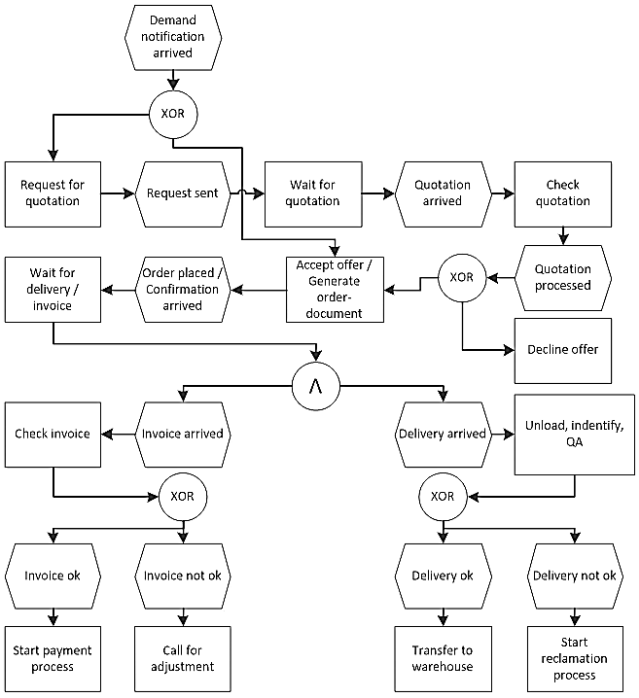

Figure 7: *Process-structure of the prototype*

The modelled process clarifies the relevance to implement different logical connectors in the software, split as well as join connectors have been used to cover different cases.

At the current development level only join-connectors are represented by meta-elements, because they don't need a deeper semantic than "trigger all successors after all predecessors have finished" for AND and "trigger all successors after any predecessor has finished" for OR connections. Split-connections have to introduce situation-dependent semantics. This case has been excluded by checking a given flag in the preceding object in all successors and terminating all paths which don't fit the implemented structure. A logical AND split is trivial, because it is the typical case of starting all available successors without any further test.

3.2 Implementation with Microsoft SharePoint 2010

The primary goal of the realisation of the prototype with Microsoft SharePoint 2010 was to implement the introduced architecture in a commonly used software product. A second goal was an intensive document usage, because most business workflows have document relationships. SharePoint supports natively a strong connection between documents and workflows. The core-tasks were:

- representation of the architecture and model with lists
- realisation of an unique structure of dependency descriptions
- development of an object, which represents the described meta-element and stores relevant data
- modelling and implementation of undividable tasks (Layer 3) and creation of connections between documents and workflows

The applied list-structure is shown in figure 8. It's clear that beginning with a demand, each task is connected and encapsulated with a unique meta-object. Looking on implemented lists, one list for each type of task was created, one list contains all meta-objects and one further list contains the main demand-objects, represented by documents. The approach to store all meta-objects in a common container eases the forecasting and optimization procedures, because the whole set of necessary information is available in the same place.

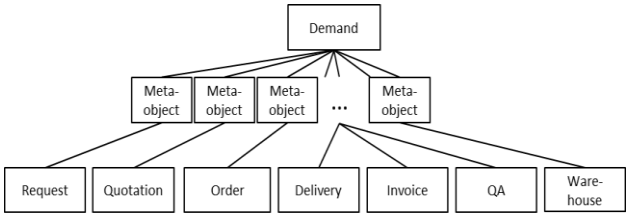

Figure 8: *List-structure in SharePoint, each box represents one dedicated list. Lines show direct connections between objects*

The implementation of the meta-object has been done using a list-entry, which is being connected to the workflow at instantiation, like described in figure 5. The runtime structure (figure 6) of the meta-object can be realized with the toolset given in SharePoint. Furthermore the list contains columns to store information about predecessors and successors, planned and realized timings and the connection to the workflow in Layer 3. Based on this information it is possible to operate the whole workflow.

Additionally there a separate workflow was implemented, which performs the necessary initialisation- and optimization-steps in Layer1 and one workflow for each task in level 3.

The explained dependency descriptions have been declared in a separate list, containing only information about the structure like before/after definitions and logical implications. This list is comparable to the tables given in figure 2 and 3 and is enriched with references to Layer 3 tasks and documents. Logical parts have to be flagged with an additional parameter defining the type of the connection (AND / OR).

The introduced information-types suffice to transfer complex business process models into the standard software. By giving a clear structure on list-basis it is possible to modify the dependencies in terms of optimization (like the creation of artificial dependencies in 2.1.3). In addition this style of modelling is much more maintainable than the static workflow models given in SharePoint 2010, because little changes in the list can alter the whole workflow without using complex development suites like visual studio.

4. DISCUSSION

The introduced concept is capable of being extended in various ways. The main goal to transform business process models to a generic form, which can be implemented in proprietary software, has been fulfilled. The possibility to add complex approaches for optimization and structure adaption is given.

The formal definition and description of the logical connectors remains an open point. By extending the concept with optimization methods which adapt the system to recognized user- or system-behavior the need for formal model checking arises. This model checking needs semantic specification for all components of the system, therefore a concrete description has to be defined. A possible extension would be a petri-net based semantic, like introduced by Hinz, Schmidt and Stahl [9] or Dijkman, Dumas and Ouyang [5]. They give the opportunity for automatic model checking while defining the initial model and dependencies as later when the structure is being altered by algorithms.

While join operations have been modeled by special Meta-Objects, this is not the case regarding split connectors. A possible extension of the concept would be the introduction of case-based reasoning methods, which choose the applicable decisions at runtime. Weber and Wild published an approach which could be used as fitting element. In this work the usage of a repository containing experienced data can be used to improve the difference between planned and realized runtimes [16].

At the moment only XOR and AND connections, formal OR definitions, loops and loop breakers as well as a stronger focus on temporal properties of the elements are open tasks and have to be linked to the already stated extensions.

Focused on EPC-modeling only End-Start relationships are available in the current version. The introduction of Start-End or End-End relationships is needed to represent more complex dependencies, like given in production environments (e.g. adhesion processes).

Beneath structural extensions there is much potential by extending the first layer with a bunch of forecasting and optimization approaches. In the area of forecasting especially time-series analysis and prediction is interesting. Facing better optimization one capacity restricted example has been shown, but comparable methods can be used to model and solve problems regarding cost, usage of resources and many more.

5. CONCLUSION

This paper develops an approach to use any business process model in abstracted form in proprietary standard workflow systems. By using a very simplified formulisation it is possible to save complex business processes in list structures and adapt them regarding the system without altering the workflow codes.

Beneath the possibility to switch the models, regarding an appropriate dependency based model, between different systems the loose connected structure enables optimization on a structural level. These optimization methods can be used autonomously to improve and adapt the planed timing of each instance regarding an automatically generated repository of experienced data. Finally the model is extendable with restrictions and boundaries like shown exemplary with limited capacity.

[1] M. Bartusch, **Optimierung von Netzplänen mit Anordnungsbeziehungen bei knappen Betriebsmitteln**, Ph.D. thesis, Universität Passau, 1983.

[2] J. Becker and D. Kahn, Der Prozess im Fokus. **Prozessmanagement,** 5. Edition, Springer, 2005, pp. 3–16.

[3] F. Bernerand, B. Kochendörfer, R. Schach, „Netzplantechnik“, **Grundlagen der Baubetriebslehre 2,** Teubner, 2008, pp. 99–126.

[4] T.H. Davenport, “Process Innovation: reengineering work through information technology”, **Harvard Business School Press**, 1993, p. 5.

[5] R. Dijkman, M. Dumas, and C. Ouyang, “Semantics and Analysis of Business Process Models in BPMN”, **Information and Software Technology 50(12)**, 2008, pp. 1281-1294.

[6] W. Esswein, „Das Rollenmodell der Organisation: Die Berücksichtigung aufbauorganisatorischer Regelungen in Unternehmensmodellen“, **Wirtschaftsinformatik** 35, No.6, 1993, p. 551–561.

[7] O. Grün, S. Kummer and W. Jammernegg, **Grundzüge der Beschaffung, Produktion und Logistik**. Pearson Studium, 2009, p.33.

[8] Herbst, Joachim: **Ein induktiver Ansatz zur Akquisition und Adaption von Workflow-Modellen**. Tenea, 2003.

[9] Hinz, S.; Schmidt, K. and Stahl, C. (2005), Transforming BPEL to Petri Nets,**Business Process Management; Lecture Notes in Computer Science**, pp. 220-235.

[10] J. Lin and E. Keogh, “Clustering of time-series subsequences is meaningless: implications for previous and future research.” **Knowledge and Information Systems**, 2005, pp. 154–177.

[11] J. Lin, E. Keogh, S. Lonardi and P. Patel, “Finding Motifs in Time Series.” **2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, 2002, p. 53-86

[12] L. Litz, **Grundlagen der Automatisierungstechnik: Regelungssysteme - Steuerungssysteme - hybride Systeme**. Oldenbourg, 2005, p. 224.

[13] M. Mühlen, H. Hansmann, „Workflowmanagement“, **Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung**. Berlin Springer, 2002, pp. 373–409.

[14] M. Remco, M. Dijkman, M. Dumas, C. Ouyang, “Semantics and analysis of business process models”, **BPMN, Information and Software Technology**, Volume 50, Issue 12, 2008, pp. 1281-1294.

[15] Verein deutscher Ingeniere (VDI), **Richtlinie VDI 4400 Blatt1, Logistikkennzahlen für die Beschaffung**, 2001.

[16] B. Weber and W. Wild, “Conversational Case-Based Reasoning Support for Business Process Management.”, **Mixed-Initiative Problem-Solving Assistant, held in conjunction with AAAI Fall Symposium**, 2005.