Matthias Böhm, Uwe Wloka, Dirk Habich, Wolfgang Lehner

GCIP: exploiting the generation and optimization of integration processes

SLUB
Wir führen Wissen.

TECHNISCHE
UNIVERSITÄT
DRESDEN

Qucosa
Quality Content of Saxony

GCIP: Exploiting the Generation and Optimization of Integration Processes

Matthias Boehm and Uwe Wloka
Dresden University of Applied Sciences
Database Group
01069 Dresden, Germany
mboehm@informatik.htw-dresden.de
wloka@informatik.htw-dresden.de

Dirk Habich and Wolfgang Lehner
Dresden University of Technology
Database Technology Group
01062 Dresden, Germany
dirk.habich@tu-dresden.de
wolfgang.lehner@tu-dresden.de

ABSTRACT

As a result of the changing scope of data management towards the management of highly distributed systems and applications, integration processes have gained in importance. Such integration processes represent an abstraction of workflow-based integration tasks. In practice, integration processes are pervasive and the performance of complete IT infrastructures strongly depends on the performance of the central integration platform that executes the specified integration processes. In this area, the three major problems are: (1) significant development efforts, (2) low portability, and (3) inefficient execution. To overcome those problems, we follow a model-driven generation approach for integration processes. In this demo proposal, we want to introduce the so-called GCIP Framework (Generation of Complex Integration Processes) which allows the modeling of integration process and the generation of different concrete integration tasks. The model-driven approach opens opportunities for rule-based and workload-based optimization techniques.

1. INTRODUCTION AND MOTIVATION

The scope of data management continuously changes from the management of locally stored data towards the management of highly distributed systems and applications. Here, integration processes—in the sense of an abstraction of workflow-based integration tasks—are increasingly used in EAI (Enterprise Application Integration) servers, ETL (Extraction Transformation Loading) tools or WfMS (Workflow management Systems). Basically, there are the three major problems: (1) the high development effort, (2) the low degree of portability and (3) the inefficient processing. In practice, integration processes are pervasive. Hence, in particular, we need to overcome the problem of inefficiency because the performance of complete IT infrastructures strongly depends on the performance of the central integration platforms.

To overcome the mentioned major problems, we follow a model-driven generation and optimization approach us-

ing our GCIP Framework (Generation of Complex Integration Processes). Here, integration processes are modeled in a platform-independent way using existing process description notations and tools, such as UML (Unified Modeling Language) activity diagrams or BPMN (Business Process Modeling Notation) process specifications. These models are used to generate platform-specific integration tasks for different integration systems. Currently, the GCIP Framework supports the generation of integration processes for the system types of Federated DBMS, ETL tools, and EAI servers. The overall approach [4] and the currently supported products are illustrated in Figure 1. In general, the model-driven development addresses the problems of high development effort and low portability. Additionally, this generation approach opens many optimization opportunities. Here, we try to overcome the problem of inefficiency using rule-based and workload-based optimization techniques.

So far, only little work exists on the generation and optimization of integration processes. From the generation perspective, the Orchid project [6] provides significant contributions. In this project, ETL workflows are generated from declarative mapping specifications. Unfortunately, Orchid exclusively covers IBM ETL products. Further, there is an promising approach [1] that addresses the deployment of ETL processes. However, all relevant generation approaches consider ETL tools only. Hence, the significance of our demo proposal includes the generation of integration processes for different integration system types (e.g., FDBMS, ETL, and EAI servers) with completely different execution models.
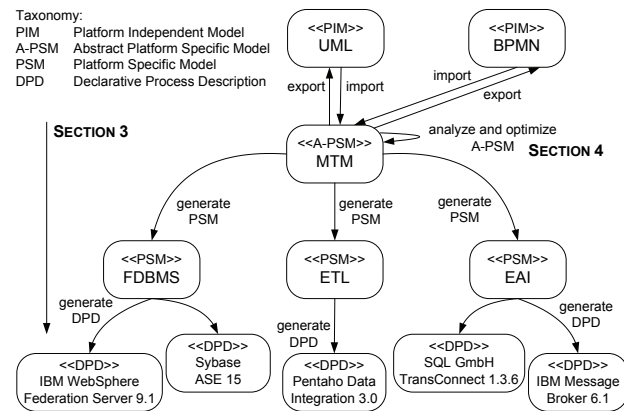
Figure 1: GCIP Generation Approach

Furthermore, there is only little work on the optimization of integration processes. First results have been published for critical path optimization of workflows [8] and the optimization of queries over Web services [10]. These techniques mainly try to use parallelism in awareness of data dependencies. More sophisticated approaches have been published in the fields of ETL process optimization [9], self-optimization of message transformation processes [5], and BPEL process optimization [11]. The major challenge in distributed environments is the missing knowledge of data properties [7]. Further, all of those solutions are platform-specific. Additionally, those solutions use only rule-based optimization techniques, which can lead to inefficiency in the presence of workload shifts. Hence, the significance of our approach includes the cost-based optimization of integration processes, where we use workload characteristics and execution statistics in order to decide on optimality conditions. For integration processes this has not been considered elsewhere.

The contribution of this paper includes the following three aspects that also reflect the structure of the paper:

- We introduce our GCIP Framework and its macro-architecture in Section 2.

- Then, we explain the generation of integration processes for different target integration systems (of different system types) in Section 3.

- Further, we discuss the rule-based and the workload-based optimization of integration processes—including applicable optimization techniques—in Section 4.

Subsequently, in Section 5, we summarize our main research contributions, open challenges and demonstration details that will be up for discussion at the EDBT demonstration desk. Finally, in Section 6, we conclude the paper.

2. GCIP FRAMEWORK ARCHITECTURE

In this section, we describe the GCIP Framework architecture and its main concepts. Figure 2 illustrates the macro-architecture of the GCIP Framework. The `core` implements the defined `API` and uses the subcomponents `Transformer` (realizes model transformations for the supported target integration systems), `Optimizer` (rewrites process plans using rule-based and workload-based optimization techniques), `Dispatcher` (decides on the most efficient target integration system), and `Deployer` (deploys generated process descriptions into the target integration systems). Finally, the framework comprises a `Repository` component that is used as a system dictionary. Hence, it handles configuration management, meta data management, model repository functionalities and persistence. In the following, we focus only on the `Transformer` (generation perspective) and the `Optimizer` (optimization perspective).

The `Transformer` subcomponent is responsible for model-model transformations (e.g., A-PSM to ETL PSM), model-DPD unparsing (generation of product-specific description) and DPD-model parsing (reverse engineering). The model-model transformations are bidirectionally realized with triple graph grammars, where a correspondence graph is modeled between two meta models. The model-DPD transformations are realized using platform-specific and dialect-specific text templates, which contain placeholders for structured and atomic values. In order to generate a DPD, the algorithm iterates over the PSM XML representation, and for
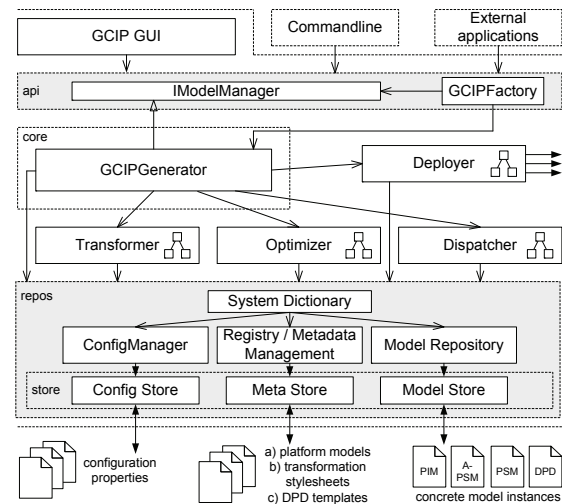

Figure 2: GCIP Framework Macro-Architecture

each element, the correlated templates are loaded, placeholders are recursively replaced, and finally, the root template is stored as DPD. In contrast to this, the platform- and dialect-specific DPD parser has been generated with the *JavaCC* tool (based on supplied grammars). It allows parsing text-based DPDs to the structured models.

Furthermore, the `Optimizer` allows for process plan rewriting on A-PSM level, which can be influenced with hints, policies and given workload and execution statistics. Basically, a process plan P can be rewritten to a more efficient process plan P'. Therefore, the `Optimizer` first executes a preprocessing chain including control flow analysis, data flow analysis and the successive dependency analysis. Here, the A-PSM is transformed into a process plan (specialized node) using the `Processplan Parser`. There, also a dependency graph is generated. Then, the `Core Optimizer` uses the rule-based and workload-based optimizer in order to apply the defined optimization techniques (see Figure 4). Those techniques use the `Processplan Rewriter`—which is aware of the created dependency graph—to rewrite the given process plan. While the rule-based techniques use defined triples (search pattern, anomaly definition, rewrite pattern), the workload-based techniques are more complex. Here, the `Estimator` component predicts the costs of a process plan and its nodes. Therefore, we use our defined cost and workload model. Note that we provide an interface for workload statistics propagation by the target integration system. Further, the `Optimizer` periodically re-optimizes the given process plans, with the aim of adapting to changing workload characteristics. Finally, the `Processplan Unparser` serializes the rewritten process plan in order to deploy it into the target integration system.

In conclusion, the GCIP Framework allows the generation of complex integration processes for the specific platform models of FDBMS, ETL tools and EAI servers. During this generation, integration processes can be optimized with different optimization techniques. Furthermore, the framework can be extended in order to support new integration system types or DPD dialects. However, further research work is required to realize full round-trip engineering capabilities and to ensure the creation of robust integration processes (functional as well as performance aspects).
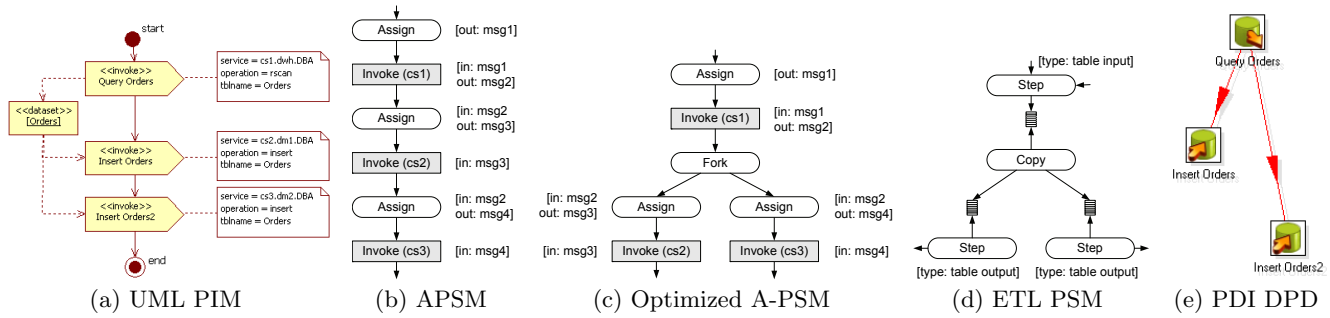
(a) UML PIM   (b) APSM   (c) Optimized A-PSM   (d) ETL PSM   (e) PDI DPD

Figure 3: Generation and Optimization Example (PIM, APSM, Optimized APSM, PSM, DPD)

3. GENERATION APPROACH

The main generation approach comprises four different levels of abstraction. At the highest abstraction level, integration processes are represented with platform-independent models (PIM) specified with graphical process description languages like UML activity diagrams or BPMN diagrams. These models can be imported into the GCIP Framework. During this import, the PIM is transformed into an abstract platform-specific model (A-PSM). This model is the normalized representation for further generation and optimization. The concrete A-PSM models are visualized using the Graphviz DOT language. Note that such a central representation reduces the transformation complexity between different platform-independent representations and different platform-specific representations. From this central representation, several platform-specific models (PSM) can be generated (system-type-specific) using the defined platform models (PM). Currently, the GCIP Framework supports the generation of integration processes for the system types Federated DBMS, ETL tools, and EAI servers. For instance, in the case of the PSM for Federated DBMS, an XML representation of a stored procedure is created. The major challenge when doing so lies in the structural changes (due to different execution models) between the A-PSM and the different PSMs. Based on the generated PSM (for an integration system type category), different declarative process descriptions (DPD) can be generated, which are system-specific representations of an integration process. Here, for instance, SQL-server-specific stored procedures using the specific TSQL dialect are generated. Finally, the model-driven generation overcomes the problems of high development effort and low portability of integration processes.

In order to make this generation approach more understandable let us assume a concrete example.

EXAMPLE 1. *Figure 3 illustrate an example platform-independent integration processes using a UML activity diagram. Here, orders are extracted from a data warehouse and inserted into two physical separated data marts without any schema mapping. Then the PIM is imported into the GCIP Framework, where the central A-PSM (see Figure 3(b)) is created. From this model, we generate a platform-specific model for ETL (see Figure 3(d)). Clearly, this model is platform-specific (ETL, in this case) but not system-specific. Note the structural difference between the instance-based abstract PSM (A-PSM) and the pipes and filter model used for the ETL PSM. Finally, we can generate the declarative process description for the concrete ETL tool Pentaho Data Integration (PDI) (see Figure 3(e) for a screenshot).*

4. OPTIMIZATION APPROACH

Similar to declarative expressions, the model-driven generation opens several opportunities for integration process optimization. Here, we apply rule-based and workload-based optimization techniques on the A-PSM level. This is advantageous due to a single point of normalization and optimization. An overview of these optimization techniques is illustrated in Figure 4. In general, some techniques have been adapted from the areas of database systems and compiler construction and some are specific to the context of integration processes. While the rule-based optimization techniques are only applied during the initial optimization, the numerous workload-based optimization techniques are based on monitored workload and execution statistics (execution times, cardinalities). Hence, those are applied during periodical re-optimization within a feedback loop between our GCIP Framework and the used integration system.

With the aim to make this optimization approach more understandable we revisit our used generation example.

EXAMPLE 2. *The initially created A-PSM P from Figure 3(b) can be optimized to a more efficient A-PSM P' that is illustrated in Figure 3(c). This decision is made based on monitored workload statistics $W(P)$, the defined cost model and estimated workload statistics $W(P')$ with $W(P') = \frac{C(P')}{C(P)} \cdot W(P)$. Here, the techniques* WC2: Rewriting Sequences to Parallel Flows, *and* WC1: Rescheduling Start of Parallel Flows *are used. We can rewrite the sequence of operators to two concurrent subflows because there is no data dependency between subsets of those operators.*

Finally, the GCIP Framework supports the visualization of estimated costs and applied optimization techniques. Thus, our demo will include a detailed presentation of the integration process optimization approach.
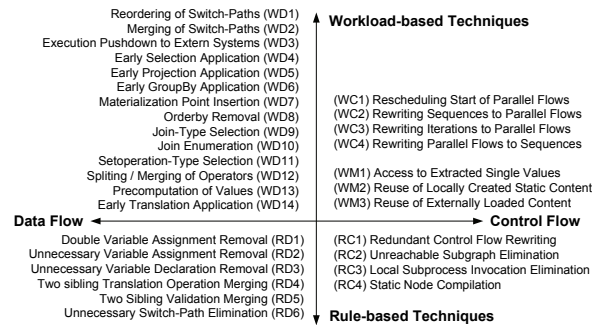

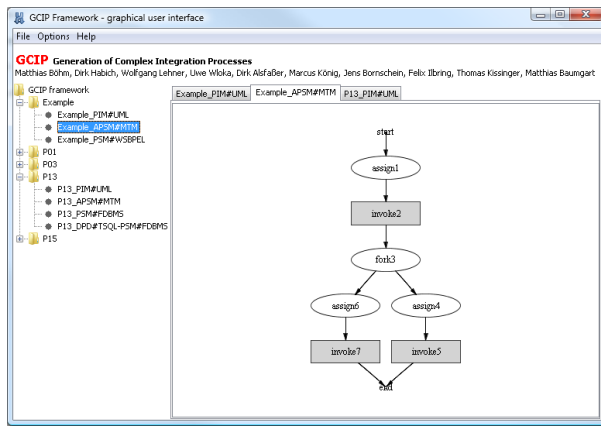**Figure 4: Available Optimization Techniques**

3

Figure 5: GCIP Framework GUI Screenshot

5. DEMONSTRATION DETAILS

Basically, we will demonstrate two perspectives of our GCIP Framework—the generation and the optimization (see Figure 5 for a screenshot of the A-PSM from the previous examples)—using the defined DIPBench process types [2, 3] as example process types. Here, we will use specific integration systems from the categories of Federated DBMS, ETL tools and EAI servers. In general, we want to discuss benefits and problems that arise in the context of the generating and optimizing integration processes. Therefore, our demonstration will include the following issues in the form of an interactive example scenario:

A) Introduction to the research area of integration processes: First, we will briefly discuss the evolving research area of integration processes, including typical process types. There, we will emphasize the specific characteristics and major research challenges of this field with regard to modeling as well as execution.

B) Explanation of platform-independent modeling: Then, we will explain the platform-independent modeling of integration processes with UML activity diagrams (see Figure3(a)) and BPMN process specifications. This will include pre-defined integration processes as well as live modeling from scratch.

C)Explanation of the model-driven generation: Afterwards, we will present our overall model-driven generation approach including our developed GCIP Framework architecture. We will show (similar to Example 1) how to transform an interactively modeled platform independent model (PIM) to an abstract platform-specific model (A-PSM), how to create subsequently the platform-specific models (PSM), and how to finally generate the declarative process descriptions (DPD).

D) Discussion of optimization aspects: Subsequently, we will discuss (1) how traditional techniques of distributed query optimization were adapted to the context of integration processes, (2) which optimization techniques are specific to the characteristics of integration processes (different operators, different execution models, and separation in control flow and data flow), and (3) the application of optimization techniques and the evaluation using the DIPBench toolsuite and its monitoring component [3]. This will demonstrate the benefit of the available optimization techniques.

E) Description of open research challenges: Aside from the demonstration of our approach and the GCIP Frame-

work, we want to point out open research challenges and want to discuss how these could be addressed. From our point of view, open research challenges are (a) the selection of the optimal integration system (invisible deployment), (b) the throughput maximization (vectorization, multi-process optimization), and (c) the multi-dimensional optimization (including dimensions such as performance and quality).

6. SUMMARY AND CONCLUSION

The three major problems within the area of integration processes are high development effort, low portability, and, particularly important, inefficiency. To overcome these problems, we have introduced our GCIP Framework (Generation of Complex Integration Processes), employing the model-driven generation and optimization of integration processes for different integration systems. Moreover, we have introduced our rule-based and workload-based optimization techniques. The significance of the proposed GCIP Framework is based on the fact that it is the first comprehensive solution addressing the generation and optimization of integration processes for different types of integration systems (FDBMS, ETL, EAI). In general, we want to use the EDBT demonstration desk as a forum to discuss the challenging research area of integration processes.

Our vision of *invisible deployment* is based on the hypothesis that a typical IT infrastructure comprises many integration systems with overlapping functionalities. Thus, the goal is to determine the most efficient integration system for a given process plan P. Here, we want to achieve transparency of the used integration systems, providing central deployment capabilities. For that vision, the generation and optimization perspectives are preconditions but many other challenging research issues and problems still exist.

7. REFERENCES

[1] A. Albrecht and F. Naumann. Managing etl processes. In *Workshop New Trends in Information Integration (NTII)*, 2008.

[2] M. Böhm, D. Habich, W. Lehner, and U. Wloka. Dipbench: An independent benchmark for data intensive integration processes. In *IIMAS*, 2008.

[3] M. Böhm, D. Habich, W. Lehner, and U. Wloka. Dipbench toolsuite: A framework for benchmarking integration systems. In *ICDE*, 2008.

[4] M. Böhm, D. Habich, W. Lehner, and U. Wloka. Model-driven generation and optimization of complex integration processes. In *ICEIS*, 2008.

[5] M. Böhm, D. Habich, U. Wloka, J. Bittner, and W. Lehner. Towards self-optimization of message transformation processes. In *ADBIS*, 2007.

[6] S. Dessloch, M. A. Hernandez, R. Wisnesky, A. Radwan, and J. Zhou. Orchid: Integrating schema mapping and etl. In *ICDE*, 2008.

[7] Z. G. Ives, A. Y. Halevy, and D. S. Weld. Adapting to source properties in processing data integration queries. In *SIGMOD*, 2004.

[8] H. Li and D. Zhan. Workflow timed critical path optimization. *Nature and Science*, 3(2), 2005.

[9] A. Simitsis, P. Vassiliadis, and T. Sellis. Optimizing etl processes in data warehouses. In *ICDE*, 2005.

[10] U. Srivastava, K. Munagala, J. Widom, and R. Motwani. Query optimization over web services. In *VLDB*, 2006.

[11] M. Vrhovnik, H. Schwarz, O. Suhre, B. Mitschang, V. Markl, A. Maier, and T. Kraft. An approach to optimize data processing in business processes. In *VLDB*, 2007.