Adaptive Information Integration: Bridging the Semantic Gap between Numerical Simulations

Tobias Meisen[1], Philipp Meisen[2], Daniel Schilberg[1], Sabina Jeschke[1]

[1]Institute of Information Management in Mechanical Engineering,
RWTH Aachen University, Dennewartstraße 27, 52068 Aachen, Germany
{tobias.meisen, daniel.schilberg, sabina.jeschke}@ima-zlw-ifu.rwth-aachen.de

[2]Inform GmbH, Pascalstraße 23, 52076 Aachen, Germany
philipp.meisen@inform-ac.com

Abstract. The increasing complexity and costs of modern production processes make it necessary to plan processes virtually before they are tested and realized in real environments. Therefore, several tools facilitating the simulation of different production techniques and design domains have been developed. On the one hand there are specialized tools simulating specific production techniques with exactness close to the real object of the simulation. On the other hand there are simulations which simulate whole production processes, but in general do not achieve prediction accuracy comparable to such specialized tools. Hence, the interconnection of tools is the only way, because otherwise the achievable prediction accuracy would be insufficient. In this chapter, a framework is presented that helps to interconnect heterogeneous simulation tools, considering their incompatible file formats, different semantics of data and missing data consistency.

Keywords: Application integration, Data integration, Simulation tools, Ontology, Framework

1 Introduction

Within the enterprising environment, the necessity to couple deviating applications being used in a company was recognized early. As a consequence, various concepts were developed that were subsumed under the collective term "Data Integration Techniques" [28]. One of those techniques, "Enterprise Application Integration" (EAI), focuses on integrating business processes based on IT along the value chain of an enterprise without taking into account the platform, the architecture as well as the generation of the applications being used in these processes [4]. Especially in the widely spread field of enterprise resource planning [8] EAI technologies are well established. These technologies are the foundation for such systems concerning data and application integration. In other fields, e.g. Business Intelligence (BI) or Enterprise Performance Management (EPM), other data integration techniques (i.e. ETL, EII) are mainly used to gain information about cross-applicational business processes [19].

The combination of those integration techniques to analyze more complex business processes, like simulation processes, is seldom taken into account [23]. Simulation itself is a well-established field in research and development and different simulations for specific tasks as e.g. casting, welding or cooling and also for whole processes (e.g. transformation or heat-treatment processes) are available. Nevertheless, those simulations have to be seen as isolated applications. They are often specialized for a single purpose (e.g. a specific task) and have neither standardized interfaces nor standardized data formats. Therefore, different results that were received within a simulation can only be integrated into a simulation process if they are checked manually and are adapted to the needs of the subsequent simulations. Current data integration techniques cannot easily be applied to the simulation context because a combination of different techniques and solutions is required. Huge data volumes which are characteristic for simulation processes tend to use ETL techniques, whereby those do not support the important concept of message-oriented transactions. These message-oriented transactions are realized in the field of EAI (e.g. ESB). Although within the field of EAI, huge data volumes cannot be handled satisfactorily. Another problem is the adaption of the data integration process concerning changes within the simulation process (e.g. the integration of a new application, the modification of a simulated object) and the semantics of data that have to be considered by the integration.

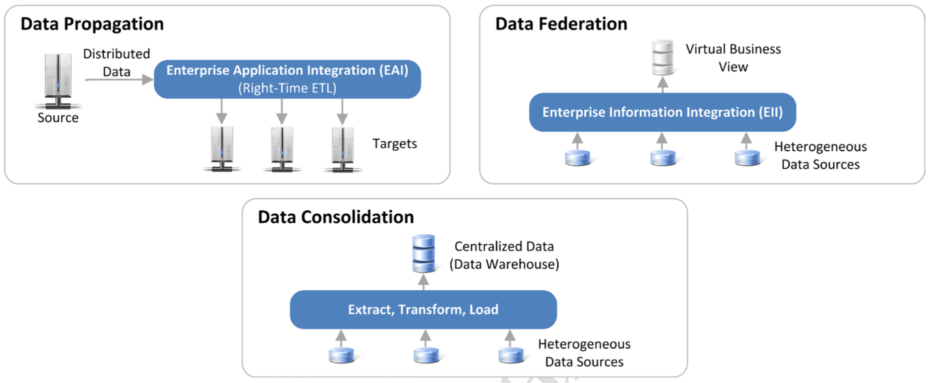
In this chapter, a framework will be described which provides the possibility of simulating a production process by making use of existing isolated applications. The integration is based on ontologies, which describe the domain specific knowledge (e.g. material processing simulations) and planning algorithms used to identify how the data can be transferred between different heterogeneous simulation tools. Thereby, the chapter focuses on the integration of data that was generated during applications' usage, whereas the applications' linkup technique, which can be handled with the help of modern middleware [18], will not be stressed.

The framework has been validated on the foundation of the simulation of three production processes, namely a line-pipe, a gear wheel and a top-box. The framework was developed within the project "Integrated Platform for Distributed Numerical Simulation", which is part of the Cluster of Excellence "Integrative Production Technology for High-Wage Countries".

The chapter is structured as follows: In section 2, the current state of technology will be outlined in order to provide a foundation for section 3, in which one of the simulated production processes is exemplarily presented. Section 4 consists of a description of the architecture of the framework that is completed in section 5 by a specification of the used information integration method. Section 6 points out how the framework needs to be extended with regard to the presented use case. In section 7, a conclusion and outlook will be drawn from the insights generated in this chapter.

2 State of the Art

Since the nineties, data integration belongs to the most frequented topics with reference to finding answers to questions which are raised across application boundaries [9]. Today, a multitude of data integration products can be found which are used in different fields of application, whereby each technology can be assigned to one of three techniques [28] (cf. Fig. 1): data propagation, data federation or data consolidation.



Data Propagation
Distributed Data
Enterprise Application Integration (EAI) (Right-Time ETL)
Source
Targets
Data Federation
Virtual Business View
Enterprise Information Integration (EII)
Heterogeneous Data Sources
Data Consolidation
Centralized Data (Data Warehouse)
Extract, Transform, Load
Heterogeneous Data Sources

Fig. 1. Main areas of data integration [28]

With regard to the operational section, data propagation is applied in order to make use of data on a cross-application basis, which is often realized via EAI. As already presented in [28], EAI mainly focuses on small data volumes like messages and business transactions that are exchanged between different applications. In order to realize EAI, a contemporary architecture concept exists, which was developed in connection with service-based approaches [3] and which will be emphasized within this contribution – the so-called Enterprise Service Bus (ESB). The basic idea of ESB, which can be compared to the usage of integration brokers, comprises the provision of services within a system [25]. Within an ESB different services provide a technical or technological functionality with the help of which business processes are supported. A service can be a transformation or a routing service, whereby all services are connected with each other via an integration bus. Transformation services provide general functions in order to transfer data from one format and/or model into another. In contrast, routing services are used to submit data to other services. Both transformation and routing services are used by adapters in order to transfer data provided by the integration bus into the format and the model of an application. Consequently, transformation services support the reuse of implemented data transformations. The advantage of solutions based on ESB is to be seen in the loose coupling of several services, whereas the missing physical data coupling can be regarded as a disadvantage [20]: If recorded data has to be evaluated subsequently, it has to be read out and to be transformed once again. According to this fact, a historic or at least long-term oriented evaluation of data is unconvertible, even though such an evaluation is often required.

In order to realize such a unified examination on a cross-data basis, other techniques belonging to the field of data integration need to be taken into consideration (cf. Fig. 1). Data federation, which is studied within the field of Enterprise Information Integration (EII), might serve as one possible solution to enable a unified examination. With the help of EII, data from different data sources can be unified in one single view [1]. This single view is used to query for data based on a virtual, unified data schema. The query itself is processed by mediators and divided in several queries fired against the underlying data sources. Because of the fact that most EII do not support advanced data consolidation techniques, the implementation will only be successful if the data of the different data sources can be unified, the data quality is sufficient and if access to the data is granted (e.g. via standard query interfaces).

If a virtual view is not applicable, techniques belonging to the field of data consolidation need to be utilized. Data consolidation comprises the integration of differing data into a common, unified data structure. Extract Transform Load (ETL) can be seen as one example for data consolidation, which is often used in the field of data warehousing [27]. ETL starts with the extraction of data from one or several – mostly operational – data sources. The extracted data is than transformed (e.g. joined, modified, aggregated) and the data model is adapted to a final schema (often a so called star schema). During the last phase the data is loaded into a target database (in general a data warehouse).

The presented techniques of data integration have in common that - independent of the technique - the heterogeneity of data has to be overcome. In literature, different kinds of heterogeneity are distinguished [7, 14, 16]. In this chapter, the well-established kinds of heterogeneity, technical, syntactic, data model, structural and semantic heterogeneity, listed in [16] are considered.

3 Use Case

Within this chapter, the manufacture of a line-pipe will be stressed as example use case. During the manufacture several simulation tools that are specialized for these techniques are used. The goal is the simulation of the whole production process, whereby the results of each specialized tool will be considered across the whole simulation process. The production process which will be used to exemplify the example use case is illustrated in Fig. 2.

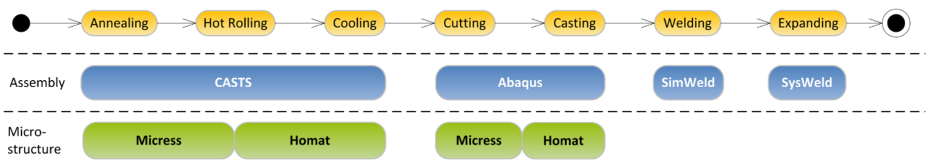


Fig. 2. Production process of a line-pipe (top) and the used simulation tools (middle & bottom)

The use case starts with a simulation of the annealing, the hot rolling as well as the controlled cooling of the components via the simulation tool CASTS (Access). The next step consists in representing the cutting and the casting with the help of Abaqus (Dassault Systems), whereas the welding and the expanding of the line-pipe will be simulated via SimWeld (ISF - RWTH Aachen University), and via SysWeld (ESI-Group). Furthermore, the simulation of modifications in the microstructure of the assembly will be realized by making use of Micress and Homat (Access). All in all, the use case contains six different kinds of tools, each based on different formats and simulation models. Thereby, the required integration solution has to take different requirements into account [22]. Two requirements, which turned out to be central with reference to the framework presented in this chapter, are on the one hand, the possibility of data propagation, focusing the semantic data exchange between the applications, and, on the other hand, the necessity of a process-oriented data consolidation. Both are used to facilitate a subsequent visualization and analysis of data collected within the process.

4 Architecture of the Framework

4.1 System Architecture

The framework's architecture is based on the requirements described in section 3. The architecture is depicted in Fig. 3. As illustrated, the framework follows the architecture concept of ESB, whereby the possibility of data consolidation was realized by implementing a central data storage (CDS) [21].

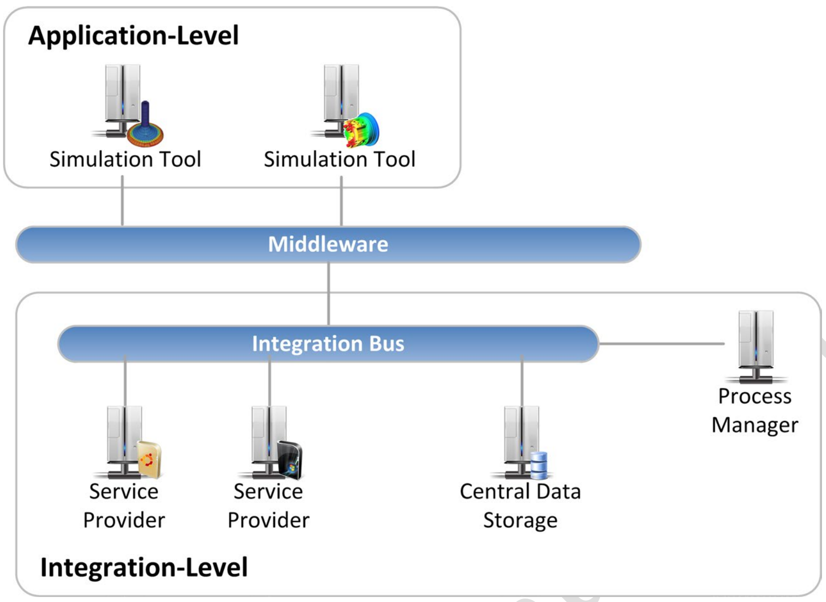


Fig. 3. System-architecture of the framework

In order to realize a communication (e.g. concerning the exchange of files and overcome the technical heterogeneity) between the integration bus and the different simulation tools, a middleware is used that encapsulates the functionality of routing services which are typical of those used in ESB concepts[1]. Hence, routing services are not considered in this framework, as the integration of a standard middleware is straight forward. The framework is employed with the intention of realizing an integration level, at which service providers, which are directly linked to the integration bus, offer different services. With the help of these services, data can be integrated, extracted and transformed. As the connection is realized via a platform independent messaging protocol, it is not bound to the operating system in use. The employment of a process manager as well as of a CDS marks an important difference between the architecture described in this section and the architectural pattern of an ESB. The process manager receives all data transferred by the middleware, analyses it and, subsequently, makes it available for each of the service providers via the integration bus. In turn, the service providers tap the required data in order to process it. After a step of processing is finished, the consistency of the data is checked and the next processing step is determined by the process manager.
Consequently, with regards to the processes of data integration and data extraction, the process manager has the task of a central supervisory authority. The service providers as well as the process manager have access to the central data storage, whereby data consolidation and, as a result, analyses of data collected during the process become possible.

[1] Within the use case mentioned in section 3 the application-oriented middleware Condor [2] is used.

4.2 Software Architecture

The framework comprises three main components: the middleware communication, the process manager and the service provider. In order to guarantee a connection between those components, a codebase component is needed, in which a cross-component functionality is encapsulated. In the following, these components will be described in detail.

Middleware Communication. The Middleware Communication component supports the realization of communication processes between the middleware and the integration bus. It contains adapters, which facilitate the transmission of demands to the integration bus by making use of different communication protocols, such as JMS, RMI or SOAP [13]. As far as a concrete use case is concerned, which is not covered by technologies that were already integrated, the component is modular expandable, which enables the implementation of additional separate adapters (cf. section 5).

Process Manager. The Process Manager comprises the implementation of a management component, which functions as a service provider and a central control unit for integration, extraction and conversion processes. The services provided by this component involve the integration, the extraction and the transformation of data. For each of these services, a service process is stored, which is started and processed as soon as a query is sent to the process manager. A service process describes which services need to be handled with the purpose of providing the requested functionality. The service processes realized within the framework are illustrated in Fig. 4.

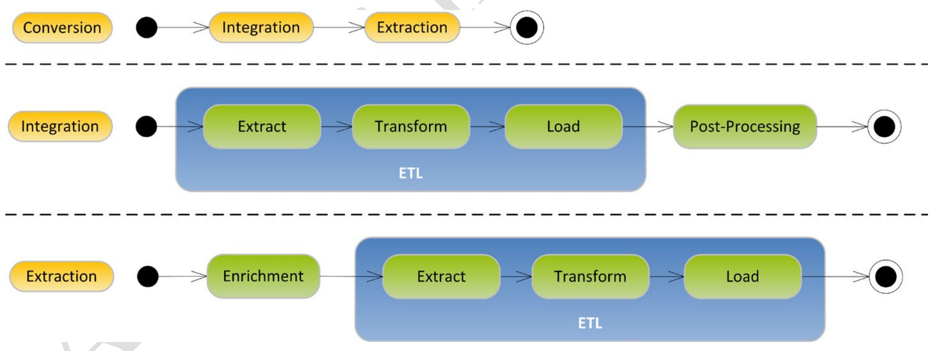


Fig. 4. Service Processes of the framework.

The conversion process is defined by an integration process and an extraction process which are both based upon extended ETL process. Within the integration process, a post-processing of integrated data is succeeded, whereas the extraction process makes use of a data enrichment that is carried out prior to the actual ETL process. Thereby, the process manager is used as a mediator with the help of which data is exchanged to those service providers that feature the postulated functionality and capacity. As a consequence, the algorithms, which are important for the process of information integration and which are depending on the use case in question, are encapsulated within the specified service providers. Additionally, the process manager realizes a process-related integration of data. Thereby, the process manager controls the assignment of data to the process step and transmits the context of the process in question to the service providers.

Service Provider. The functionality provided by a service provider always depends on the provided services and therefore on the concrete use case. For instance, the integration of FE data on the one hand and the integration of data of molecular structures on the other hand are based upon different data schemas, even though these processes of integration consist in the same physical object and deal with comparable physical entities.

The framework offers interfaces to common ETL tools as, for example, the Pentaho Data Integrator (PDI) [15]. Thus, the integration and extraction of data, and therefore the overcoming of the syntactical and data model heterogeneity, can be created on the basis of these tools. Furthermore, additional tools can be implemented in order to realize the processes of integration and extraction in the case that this way of proceeding is convenient and necessary within a concrete use case.

Apart from services which provide an ETL process, the framework supports additional services in order to post-process and enrich data. For instance, the post-processing service allows the implementation of plausibility criteria, which need to be fulfilled by the integrated data without reference to their original source. During the process of enrichment, data transformations are carried out with the purpose of editing data stored within the central data store in such a way that the data is able to meet the requirements demanded with regard to the extraction process. Therefore an adaptive information integration process [17] is used, which is described in the next section.

5 Adaptive Information Integration

5.1 Concept

The main goal of the adaptive information integration is to overcome the problems of structural and semantic heterogeneity considering domain specific knowledge. The adaptive information integration is part of the enrichment process step in the extended ETL process being used during the extraction of data. The goal of the process is to extract data in a defined data format, regarding the data model and structure, as well as the semantics, of this format and the domain. Therefore, the implemented enrichment enables the discovery and exploitation of domain specific knowledge. The concept is based upon ontologies and planning algorithms used in the field of artificial intelligence.

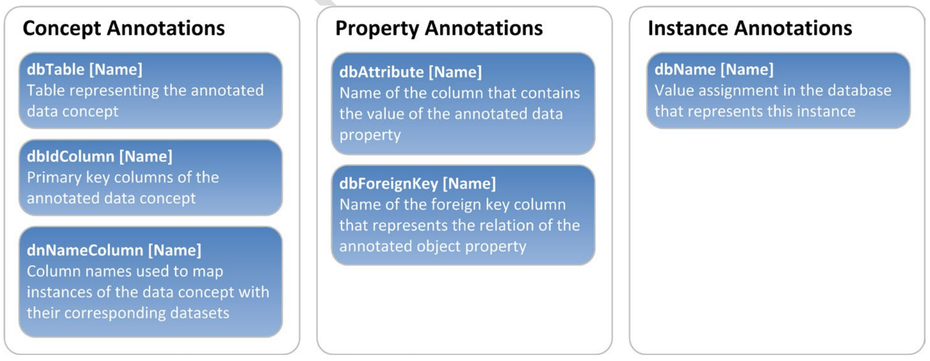
First, the existing data is analyzed. The goal of the analysis is the determination of so called features that are fulfilled by the data. A feature is domain specific and expresses structural or semantic properties that are satisfied by the data. Besides, the analysis step determines features that have to be fulfilled by the data to satisfy the requirements of the specific output format. Following the analysis the planning algorithms are used to find a data translation that transforms and enriches the data, so that the enriched data fulfills the features needed by the output format. After the planning is finished, the found data translation is processed. The data transformation algorithms used for the data transformation are realized as a service. The information about the existing transformations and features is expressed in an ontology. The basic structure of this ontology is described in the following section.

5.2 Ontology

The information used by the enrichment process is subdivided among a *framework ontology* and a *domain ontology*. The *domain ontology* holds information about the concrete transformations, features and applications used in the context of a specific domain. Besides, information about the domain specific data schema is stored. An extract of the domain ontology used to implement the use case is described in section 6, using the Web Ontology Language (OWL).

The *domain ontology* specialize the concepts of the framework ontology in order to specify the conceptualization of the domain. Hence, the framework ontology is a specification of the concepts used in the framework to enable the enrichment process. These main concepts are *data*, *feature*, *application* and *transformation*, which are introduced shortly.

The concept *data* is the generalization of all data concepts used in the domain. More precisely each concept in the domain ontology used to describe the data schema of the domain has to be a specialization of the concept *data*. The mapping between data concepts and the data schema of the domain is realized by using a predefined set of annotations. Because of the single mapping between a well-known ontology and a well-known database schema, automatic schema matching algorithms are not used. Instead this approach follows the concept of annotation-based programming. Fig. 5 gives an overview of the main annotations.



Concept Annotations

dbTable [Name]
Table representing the annotated data concept

dbIdColumn [Name]
Primary key columns of the annotated data concept

dnNameColumn [Name]
Column names used to map instances of the data concept with their corresponding datasets

Property Annotations

dbAttribute [Name]
Name of the column that contains the value of the annotated data property

dbForeignKey [Name]
Name of the foreign key column that represents the relation of the annotated object property

Instance Annotations

dbName [Name]
Value assignment in the database that represents this instance

Fig. 5. Ontology annotations.

Defining domain specific features is done by creating a specialization of the concept *feature*. Such a specialization is a listing of the requirements that have to be satisfied by a set of data, so that the represented feature is fulfilled.

For each definition of applications and their requirements, instances of the concept *application* have to be expressed in the domain ontology. An instance of the concept *application* can have additional object properties to express domain specific information of an application. Similar to an application, a transformation has requirements that have to be satisfied. Otherwise, the transformation cannot be used. Therefore, each instance of the concept *transformation* has to outline the requirements by defining instances of *feature* concepts. In addition, a transformation changes the features of data. This is realized by expressing the effects of the transformation in the ontology. The concept *transformation* and its main relations are depicted in Fig. 6.

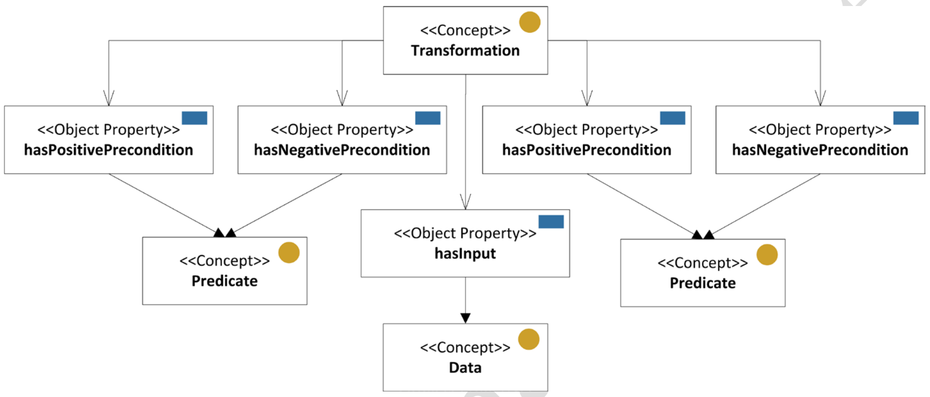


Fig. 6. Fragment of framework ontology - transformation concept

The input is set by an instance of the concept *data*, whereby the requirements are expressed by instances of either *hasPositivePrecondition* or *hasNegativePrecondition*. These object properties realize relations between the concrete transformation and feature instances. The framework ontology provides a set of logical connectives and quantifiers to express complex requirements like *feature1* **or** *feature2*. Similarly, the effects of the transformation are expressed.

5.3 Components

The concept of the adaptive information integration is realized by three services: the data analyzer, the plan generator and the plan processor. Each service implements one of the previously described steps of the enrichment process.

The data analyzer loads the ontology and establishes a connection to the CDS. By using the domain ontology, the features are determined by querying all defined specializations of the concept *feature*. The implementation of this service makes use of the OWL API [12] and the reasoner Pellet [26]. The fulfillment of a feature is checked by querying once again the CDS. The queries are generated by using the annotation based mapping. The result of the query is analyzed according to the feature definition.

The fulfilled features define the initial state of the data. In addition, the goal state is determined by the data analyzer by reasoning. This means that the current context (required output format and domain specific parameters) is used to query the required features by using the information stored in the domain ontology.

Hence, the result of the data analyzer consists of the initial and the goal state. This information is passed to the plan generator to determine the needed data translation. Therefore, the plan generator queries the existing data transformations from the domain ontology and generates a problem description using the Planning Domain Definition Language (PDDL) [5]. The defined planning problem is than solved by a planner component, which generates a solution plan. More detailed, the planner is used to determine a sequence of, so called actions that lead from the initial state to a goal state. The framework supports different planning algorithms like forward, backward and heuristic search, STRIPS algorithm or Planning Graphs [6, 10]. If the planner succeeds a plan is generated that contains the transformations and their parameterization as well as their ordering to transform the data, so that the required *features* are fulfilled by the data after having processed the plan. Finally, the processing of the plan is realized by the plan processor.

6 Application of the Framework

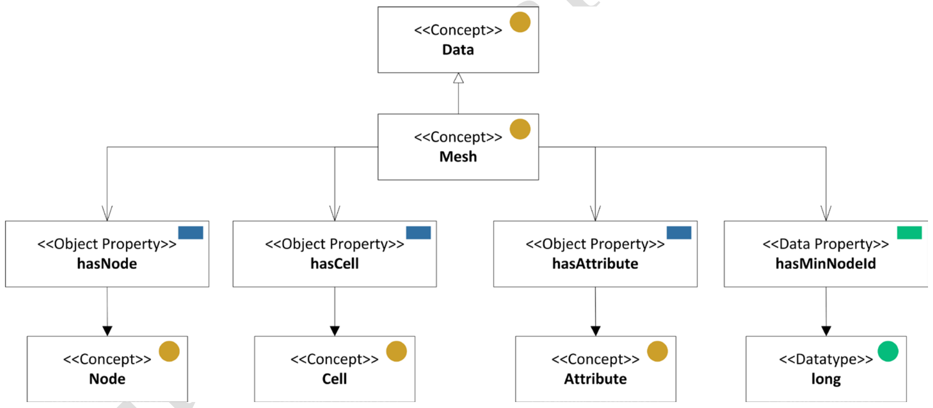
Within the domain of the use case described in section 3 and the requirements resulting from the examination of four additional use cases in the domain of FE-simulations, an integration platform has been implemented in parallel to the implementation of the framework. The integrated applications are simulations based upon the finite-element-method. In order to implement the integration platform a domain specific data schema, adapters for integration and extraction, the transformation library and the domain ontology have been provided. In the following, some selected examples will be presented.

Data schema. The domain specific data schema has been determined by analyzing the different input and output formats of the simulations used in the use case. Within this data schema, a grid structure, representing the abstraction of the assembly that is simulated, is the central entity. It consists of nodes, cells and attributes. The latter ones exhibit attribute values, which are assigned to individual cells or nodes depending on the class of attributes available in the whole mesh. The integration services, which were specified within the use case, read in the mesh data provided by the simulation, transform it into the central data model and store it into the CDS. In contrast, the extraction services proceed as follows: The mesh data is read out from the CDS and transformed into the required format. Finally, the data is saved into the destination file or into the target database. Because of the prior enrichment, all of the structural and semantic data transformations have been performed. Hence, most of the data transformations formerly performed by the adapter services are omitted.

Adapter service. Most of the adapter services have been implemented using the Pentaho Data Integrator (PDI). If more complex data have been given, or binary formats that can only be read by programming interfaces of the manufacturer, either the PDI functionality have been extended using the provided plug-in architecture or the needed functionality has been implemented using Java or C++. For example, the simulation results generated within the simulation tool CASTS are stored in the Visualization Toolkits (VTK) format [24]. Hence, an integration service was implemented, which is based on the programming interface provided by the developers of VTK using the provided functionality of the framework. Furthermore, an extraction service was developed with regard to the Abaqus input format, whereby, in this case, the aforementioned ETL tool PDI was used.

Transformation library. In order to realize the information integration, different sorts of data transformations for FE data were implemented into the application, for example the conversion of attribute units, the deduction of attributes from those ones that are already available, the relocating of the mesh within space, the modification of cell types (e.g. from a hexahedron to a tetrahedron) or the re-indexing of nodes and cells.

Domain Ontology. The domain specific information has been expressed in the domain ontology. As described previously in section 5, the domain ontology uses the concept of the framework ontology to express the data schema, the transformations, the applications and the features of the domain. Fig. 7 sketches a fragment of the concept *Mesh*.

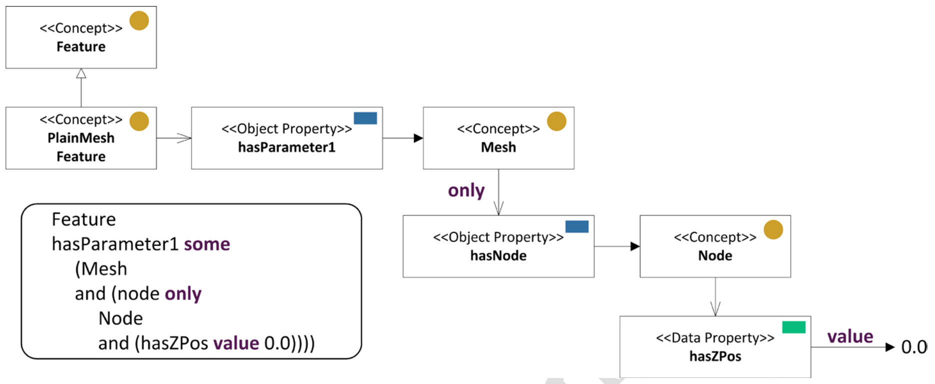


<<Concept>> Data
<<Concept>> Mesh
<<Object Property>> hasNode
<<Object Property>> hasCell
<<Object Property>> hasAttribute
<<Data Property>> hasMinNodeId
<<Concept>> Node
<<Concept>> Cell
<<Concept>> Attribute
<<Datatype>> long

Fig. 7. Fragment of the concept *Mesh* and related concepts

Because of the number of data and object properties, only a few are depicted. Most interesting is the data property *hasMinNodeId*, which is a sub-property of the *hasMinimumValueProperty*. This kind of data property can be used to prompt the data analyzer to use the SQL *MIN* function, whenever a classification requires such information. Analogous data properties for average and maximum exist within the framework ontology. The depicted object properties *hasNode*, *hasCell* and *hasAttribute* represent the relation between the concept *Mesh* and the concept referred to by the object property. Using the previously described annotations the metadata of the relationship like primary and foreign keys are expressed.

The defined data schema is used to point out different data features of the domain. As described, a feature is a kind of classification of existing data. More precisely, if all conditions of a feature are fulfilled, the data belongs to the concept represented by the feature. One feature is the already mentioned *PlainMeshFeature*. It expresses that a mesh belongs to the class of plain meshes if all nodes of the mesh have a z-coordinate of zero. The feature is illustrated in Fig. 8 as well as expressed by the OWL Manchester Syntax [11].



<<Concept>> Feature
<<Concept>> PlainMesh Feature
<<Object Property>> hasParameter1
<<Concept>> Mesh
only
<<Object Property>> hasNode
<<Concept>> Node
<<Data Property>> hasZPos
value
0.0

```
Feature
hasParameter1 some
    (Mesh
    and (node only
        Node
        and (hasZPos value 0.0))))
```

Fig. 8. Expression of the *PlainMeshFeature*

Besides the data schema and the features the ontology also contains information about the available transformations and the used applications. One example of a transformation is *HexaToTetra* that transforms a mesh that is based on hexahedrons into a mesh of tetrahedrons. The transformation searches all occurrences of hexahedrons within the mesh and splits them into tetrahedrons without creating new nodes. Hence, the precondition of the transformation is that at least one hexahedron exists in the mesh. The effect is that all hexahedrons are replaced by tetrahedrons. Preconditions and effects are expressed by using features. The expression of the transformation *HexaToTetra* in the domain ontology is illustrated in Fig. 9.

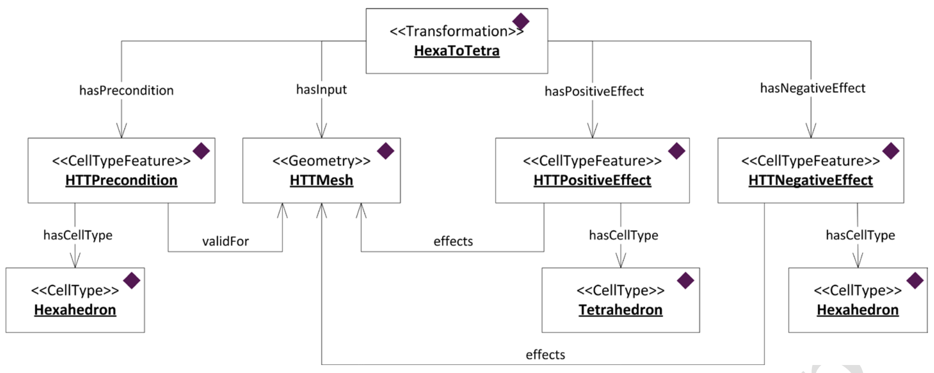


Fig. 9. Expression of the transformation *HexaToTetra*

As described previously a concrete transformation is expressed by an instance of the concept *transformation*, whereby the input, preconditions and effects are expressed by instances of the corresponding concepts. The instance *HTTMesh* of the concept *Mesh* describes that the input of the transformation is some mesh. The precondition is an instance of the concept *CellTypeFeature* expressing that the transformation is only valid if the *HTTMesh* has cells of the cell type hexahedron, which is a concrete instance of the concept *CellType*. Also, the effects are expressed using *CellTypeFeature*. The positive effect is that the resulting mesh contains cells of the type tetrahedron, whereas the negative effect is, that the concrete *CellTypeFeature* representing the hexahedron is forfeited.

Example. Concluding this section, a small example of the data provision of results generated by the simulation CASTS to the simulation Abaqus is presented. The example focuses on the structural changes of the data that are needed, in order to enable the usage of the data in Abaqus. Using the VTK data format, the indexing of nodes and cells begins with zero. Instead, Abaqus requires a sorted indexing starting with one. Additionally, in CASTS, vectors are decomposed into single components and stored as attribute values assigned to nodes, whereas in Abaqus, vectors need to be quoted entirely. Due to the data enrichment, the needed data transformations have been determined autonomously (cf. Fig. 10).

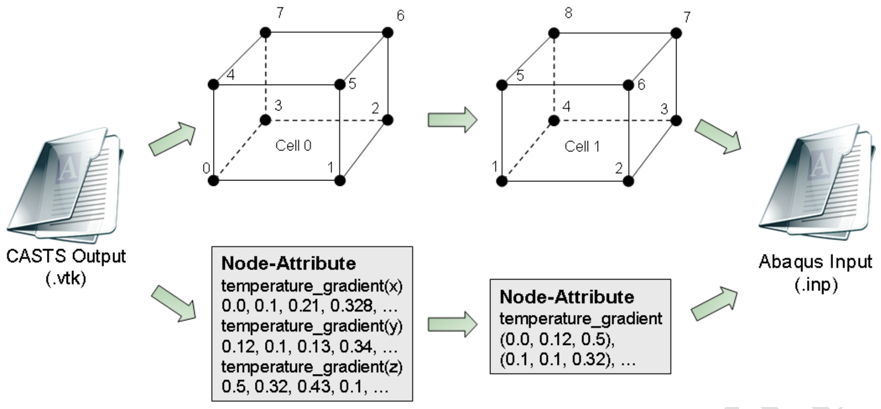


Fig. 10. Simplified illustration of the resulting data translation from CASTS to Abaqus

7 Conclusion

The development of the framework presented in this chapter can be regarded as an important step in the establishment of integrated simulation processes using heterogeneous simulations. Both, data losses as well as manual, time-consuming data transformations from one data format to another are excluded from this approach. The suggested framework facilitates the interconnection of simulation tools, which were - until now - developed independently and which are specialized for certain production processes or methods. Furthermore, the integration of data generated in the course of the simulation is realized in a unified and process-oriented way. Apart from the integration of further simulation tools into an application, which was already established, it is essential to extend the domain of simulations reflected upon with additional simulations covering the fields of machines and production. In this way, a holistic simulation of production processes is provided. Thereby, a major challenge consists in generating a central data model, which provides the possibility of illustrating data uniformly and in consideration of its significance in the overall context, which comprises the levels of process, machines as well as materials. Due to the methodology presented in this chapter, it is not necessary to adapt applications to the data model aforementioned. On the contrary, this step is realized via the integration application, which is to be developed on the basis of the framework. Because of the unified data view and the particular logging of data at the process level, the framework facilitates a comparison between the results of different simulation processes and those of simulation tools. Furthermore, conclusions can be drawn much easier from potential sources of error - a procedure which used to be characterized by an immense expenditure of time and costs. The realization of this procedure requires the identification of Performance Indicators, which are provided subsequently within the application. In this context, the development of essential data exploration techniques on the one hand and of visualization techniques on the other hand turns out to be a further challenge. Concepts and methods focusing this challenge will be developed and summarized under the term *Virtual Production Intelligence*. This term is motivated by the notion of *Business Intelligence*, which refers to computer-based techniques used to handle business data in the aforementioned manner.

Acknowledgments. The approaches presented in this chapter are supported by the German Research Association (DFG) within the Cluster of Excellence “Integrative Production Technology for High-Wage Countries”.

References

1. Bernstein, P., A., Haas, L. M.: Information integration in the enterprise. In: Communications of the ACM - Enterprise information integration and other tools for merging data, vol. 51, no. 9, pp. 72--79. (2008)
2. Cerfontaine, P., Beer, T., Kuhlen, T., Bischof, C.: Towards a Flexible and Distributed Simulation Platform. In: Proceedings of the International Conference on Computational Science and its Applications (ICCSA), Part I, pp. 867--882. Springer, Heidelberg (2008)
3. Chappell, D.: Enterprise Service Bus. Theory in Practice. O'Reilly, Beijing, Cambridge (2004)
4. Conrad, S.: Enterprise Application Integration: Grundlagen, Konzepte, Entwurfsmuster, Praxisbeispiele. Elsevier, Spektrum, Akad. Verl., Heidelberg (2005)
5. Fox, M., Long, D.: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. In: Journal of Artificial Intelligence Research, vol. 20 (1), pp. 61--124 (2003)
6. Ghallab, M., Nau, D. S., Traverso, P.: Automated planning. Theory and practice. Elsevier/Morgan Kaufmann, Amsterdam (2004)
7. Goh, C., H.: Representing and reasoning about semantic conflicts in heterogeneous information systems. PhD thesis, Massachusetts Institute of Technology (1997)
8. Gronau, N.: Enterprise Resource Planning: Architektur, Funktionen und Management von ERP-Systemen, Oldenbourg, München (2010)
9. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: the teenage years. In: Proceedings of the 32nd international conference on Very large data bases (VLDB), pp. 9--16, VLDB Endowment (2006)
10. Hoffmann, J., Nebel, B.: The planning system: Fast plan generation through heuristic search. In: Journal of Artificial Intelligence Research, vol. 14 (1), pp. 253--302 (2001)
11. Horridge, M., Patel-Schneider, P. F.: OWL 2 Web Ontology Language Manchester Syntax, W3C Working Group Note 27 October 2009, W3C, online available http://www.w3.org/TR/owl2-manchester-syntax/ (2009)
12. Horridge, M., Bechhofer, S.: The OWL API: A Java API for Working with OWL 2 Ontologies. In: Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED), vol. 529 of CEUR Workshop Proceedings (2009)
13. Kashyap, V., Bussler, C., Moran, M.: The Semantic Web, Semantics for Data and Services on the Web. Springer, Heidelberg/Berlin (2008)
14. Kim, W., Seo, J.: Classifying schematic and data heterogeneity in multidatabase systems. In: Computer, vol. 24 (12), pp. 12--18 (1991)
15. Lavigne, C.: Advanced ETL with Pentaho Data Integration. Whitepaper, Breadboard BI (2006)
16. Leser, U.: Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. Dpunkt-Verl., Heidelberg (2007)

17. Meisen, T., Schilberg, D., Henning, K.: Planner Based Data Integration for Simulation Chains in Virtual Production. In: Proceedings of the International Conference on Science, Technology and Innovation for Sustainable Well-Being (STISWB), pp. 100--108, Klung NaNa Vithya Press Limited Partnership (2009)
18. Myerson, J. M.: The Complete Book of Middleware. Auerbach Publications, Boston, MA, USA (2002)
19. Panian, Z.: Supply chain intelligence in ebusiness environment. In: Proceedings of the 9th WSEAS International Conference on Computers (ICCOMP), pp. 1--6, World Scientic and Engineering Academy and Society (2005)
20. Rademakers, T., Dirksen, J.: Open-Source ESBs in Action. Manning Publications Co., Greenwich, CT, USA (2008)
21. Schilberg, D., Gramatke, A., Henning, K.: Semantic Interconnection of Distributed Numerical Simulations via SOA. In: Proceedings of the World Congress on Engineering and Computer Science (WCECS), pp. 894--897 (2008)
22. Schilberg, D.: Architektur eines Datenintegrators zur durchgängigen Kopplung von verteilten numerischen Simulationen. PhD thesis, RWTH Aachen University (2010)
23. Schmitz, G., Prahl, U.: Toward a virtual platform for materials processing. In: JOM Journal of the Minerals, Metals and Materials Society, vol. 61, pp. 19--23 (2009)
24. Schroeder, W., Martin, K., Lorensen, B.: The Visualization Toolkit, Kitware Inc. (2004)
25. Schulte, R., W.: Predicts 2003: Enterprise service buses emerge. Technical report, Gartner (2003)
26. Sirin, E.: Pellet: A practical owl-dl reasoner. In: Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5 (2), pp. 51--53 (2007)
27. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual modeling for ETL processes. In: Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP (DOLAP), pp. 14--21. ACM, New York, NY, USA (2002)
28. White, C.: Data Integration: Using ETL, EAI and EII Tools to Create an Integrated Enterprise. Technical report, The Data Warehousing Institute (2005)