SIES: An Approach for a Federated Information System in Manufacturing

Carmen Constantinescu*, Uwe Heinkel*, Ralf Rantzau, Bernhard Mitschang
Institute of Parallel and Distributed High-Performance Systems
Applications of Parallel and Distributed Systems Department
University of Stuttgart, Stuttgart, Germany

Abstract. *Many problems encountered in providing enterprise-wide information are related to the integration of databases and systems that have been independently developed and also to the management of changes and transformations of data from one database (or system) into another. A major requirement is to accommodate heterogeneity and at the same time to preserve the autonomy of the components. This paper presents our approach to a repository-driven federated system based on a propagation mechanism. The Stuttgart Information and Exploration System (SIES), is characterized by its main components: the Federation Manager, the Propagation Manager and the Repository System.*

Keywords: Interoperability, Manufacturing Application System, Federation, Repository, Propagation

1. Introduction

Due to an increasingly turbulent environment, transformability has turned out to be an important success factor for manufacturing enterprises. Transformable, flexible processing systems are characterized by a high degree of manufacturing flexibility and productivity, and, at the same time, by the ability to adapt quickly to a new production situation expressed in terms of, e.g., shop orders and capacity requirements. In this paper, according the results of special research field 467 "Transformable Business Structures for Multiple-Variant Series Production" [15], we approach the envisioned *transformable manufacturing enterprise* as a dynamic network of similar fundamental entities named *Transformable Business Units* (TBUs). A TBU is viewed as a complex of main and support processes, tasks, resources and technology that fit in an appropriate structure and it is mainly characterized by its own potential of transformability. The behaviour of the TBU and its potential of transformability are strongly affected by the external environment, represented by a change, a new and unusual order, and by the internal environment of the enterprise, being here the inputs from the similar TBUs involved in the network.

In Figure 1 we present the transformable manufacturing enterprise by instantiating two of its main TBUs. For example, TBU_1 is the information system for Order Management, and TBU_2 is the information system for Facility Layout Design. An environment change will cause the transformation of TBU_1 from $state_{11}$ to $state_{12}$. Based on the dependencies between these TBUs, by *propagation*, TBU_2 will be transformed from

* This work was partially supported by the German Research Society (DFG/SFB 467).

state$_{21}$ to *state*$_{22}$. In our example a considerable increase in the amount of ordered products requires a greater production capacity, achievable e.g. by adding new equipments, which leads to a new, appropriate facility layout. In order to reach this transformability goal, the development of new concepts and frameworks capable to support the business structure transformability has become high-priority in many research fields. According to these challenges, we have set up as objective for our research the development of a repository-driven federation architecture that is able to coordinate and to integrate the functionality of the application information systems that represent, in our vision, the TBUs. The proposed system, *Stuttgart Information and Exploration System* (SIES), is based on concepts related to the propagation of changes between such federated systems.

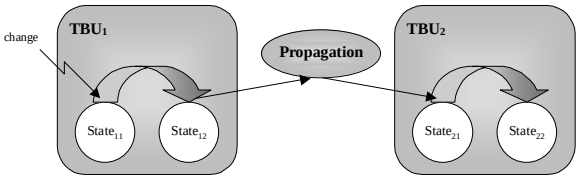

Figure 1. Transformability by propagation

This paper presents SIES by making two main contributions. First, we argue that many of the problems encountered when building applications on top of database systems involve the manipulation of models. We are using the term "model" to describe a discrete structure that represents a design artifact, a UML model, an XML DTD or Schema, a relational schema, or even a database transformation script. The manipulation of models involves managing changes in model structures and dependencies between them as well as transformations of model instances, i.e., data from one model into another. Managing changes requires an explicit representation and manipulation of "mappings" between models. In order to improve flexibility and adaptiveness, we propose to treat "model" and "mapping" as first-class objects with specific operations like create, store and update. Thus, SIES can be regarded as a model management system. The second contribution is the approach of managing changes in models and transformations of data from one model into another by propagation techniques. At the moment, we focus mainly on structure and semantics, whilst implementation being the project's next step.

Since SIES is based on repository technology that offers to our federated system flexibility and extensibility, every model and every dependency between models is explicitly represented and thus available for further usage. Hence, we can imagine that this information is subject to further analysis of enterprise internal processes. An exploration of this information with the help of analysis tools might help to optimise internal processes, to reduce overhead and thus further to enhance flexibility to quickly react to market changes.

The remainder of the paper is organized as follows. Section 2 introduces the federation concepts, our federation framework SIES, its main components and their roles. Section 3 presents the Federation Manager and the Propagation Manager as SIES components and their responsibilities in managing the models, their transformations and data propagation between them. A comprehensive example is used to clarify the concepts introduced. Section 4 concludes the paper with a short summary and gives an outlook on future work.

2. Federation Concepts

Many problems encountered in providing enterprise-wide information are related to the integration of databases and systems that have been developed independently and also to the management of changes and transformations of data from one database (or system) into another. A major requirement is to accommodate heterogeneity and at the same time to preserve the autonomy of each component. The solution, a federated information system, offers access to autonomous heterogeneous databases and systems in an integrated way. Several federated database systems have already been prototyped since the beginning of the 80s [6, 12, 16, 7, 9, 1]. Recently, some research projects have used an object-oriented model [8]. The object-oriented paradigm brings new solutions in several dimensions, including the modelling of local data sources as objects

with a well-defined and published interface, the use of a semantically rich common object model to ease the application integration, the development of standards to interoperate among objects, the use of advanced transaction models, etc. Main contributions in clarifying definitions and terms in the research area of integration of heterogeneous and distributed information systems have been reported by [5] and [4]. The first work have given classification criteria for such systems and particularly defined mediator-based information systems. The definition of terms is accompanied by the identification of relevant concepts and reference architectures. The second research analyse two basic strategies for the development of tightly coupled, federated information systems: top-down and bottom-up and proposes a combined strategy based on the intensive use of object-oriented modelling concepts.

Our research proposes a federation approach based on *propagation techniques*. In developing our framework, we are using *System* and *Partial Model* as main concepts. We consider a System as an *application* or an *information system*, and a *Partial Model* (PM) is considered as its model. Thus, instead of a global schema, we are federating local schemata named Partial Models. In a federation, these models are associated with each other. *Associations* may show up as the partial overlap of PMs or by direct correspondence from one PM with another. All these cases are further on referred to as *dependencies* among PMs. As a consequence, changes to any PM are to be propagated to the dependent PMs. Figure 2 presents the *source system* (System S) and the *dependent systems* (Systems D_i) as inputs and outputs of the change propagation based on the dependencies between the source PM and the dependent systems' PMs. Dashed arrows represent the dependencies not used in the current situation.

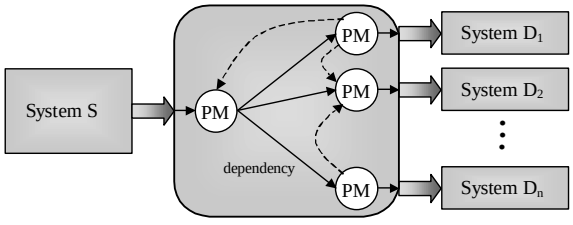

Figure 2. The generic approach of changes propagation in federated systems

Our system has to be easily extended by a new source system, a new destination system and a new dependency between them. In order to create such a flexible and extensible system that is able to efficiently manage the change propagations, we use a *repository* as a central unit of our proposed federation architecture, SIES (Figure 3). SIES is a repository-driven system that manages the PMs of the federated systems, the dependencies between them and the change propagations. It consists of three main components: the Federation Manager, the Propagation Manager, and the Repository.

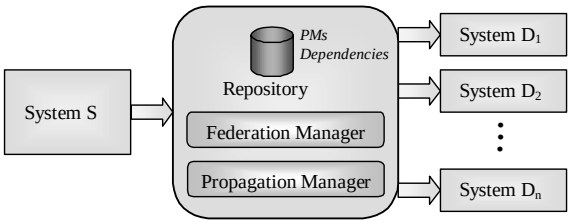

Figure 3. The Stuttgart Information and Exploration System

The role of the *Federation Manager* is to create and to manipulate the PMs and the dependencies between them. The management of change propagations between models, based on the above-mentioned dependencies, determines the functionality of the *Propagation Manager*. The central unit, the *Repository*, stores the models and the dependencies, adding flexibility and extensibility to the entire system.

In order to motivate the benefits of SIES, we present as an example two important information systems of an enterprise, Order Management and Facility Layout Design. The first one is the source system and the second the destination system. Our example assumes that the PMs for Order Management and Facility Layout Design as well as their dependencies have been created and stored in the repository. Based on these dependencies, a change in the source system, for example a higher volume of orders for one product variant, will be propagated to the destination system, the Facility Layout Design. As a result, the facility layout will be adapted according to the new production requirements, for example by reflecting a position of the equipment that better responds to the new situation.

3. Repository-Driven Approach

A main contribution of our approach is the repository-driven SIES. The federation framework includes a repository not only as a shared database of models and mappings. We investigate the usage of a Repository Manager that will support the check-in/check-out, version, configuration management, notification, context management and workflow control for the models and mappings [2, 3].

3.1. Federation

The Federation Manager, a main component of SIES, offers two functions. The first supports the creation, storage and update operations regarding the *models*. The second function is related to the dependencies between models, called *mappings*. The model management presented here is a *meta model* management, using *meta data* in manipulating models and mappings. An immediate effect of this approach is that it does not determine any implementation details, nor specify interoperability semantics, information interchange and so on. All these can be added by additional information as needed.

Since we intend to present SIES as a model management system, it is necessary to start with defining the *model* and the *mapping*. A *model M* is the simplified view of the real world. Figure 4 presents the models of two PMs, here PM_1 and PM_2.

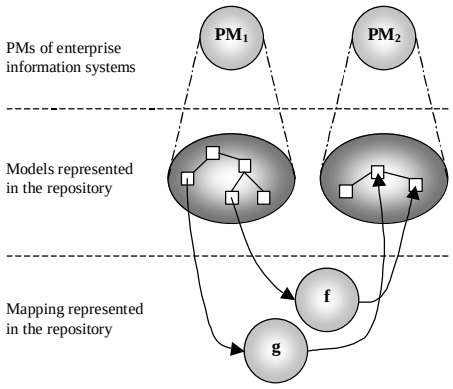

Figure 4. The model and mapping concepts

A *mapping* from a model M_1 to a model M_2 is a model itself that contains only the dependencies between the involved models. Each dependency in a mapping is based on a *function* that is an expression over the objects of M_1 and M_2. The two dependencies between the model objects presented in Figure 4, represented by the functions *f* and *g*, respectively, define the mapping between PM_1 and PM_2. With this approach, we are able to flexibly define also complex mappings that consist of a number of those primitive mappings. This aspect seems to be necessary in view of the various types and complexities of dependencies among PMs.

A fundamental challenge in our work is related to the development of a mechanism for representing models and storing these representations. A key issue here is how much semantic information of a model is necessary to be expressed in its representation. Aspects related to physical operations on models like storing and indexing also raise challenges.

In this section we present a scenario in which the Federation Manager plays a central role as a model management system for SIES. The first activity of Federation Manager is the check-in of PMs as stored repository models (Figure 5a).

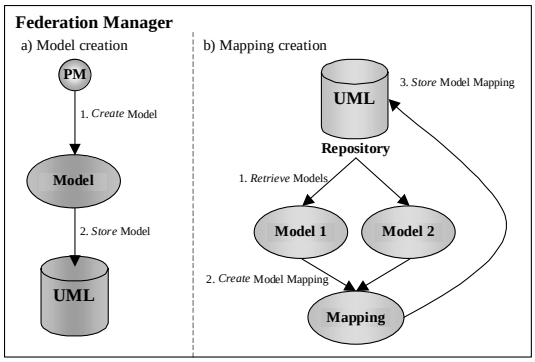

Figure 5. The Federation Manager: the steps of mapping creation

Due to its rich set of modelling concepts and its broad application spectrum, we propose the Unified Modeling Language (UML [13], [11]) to specify the models that describe the PMs of the enterprise information systems that are to be federated via SIES. We intend to use the Unified Modeling Language and supplementing tech-

niques within the object-oriented paradigm. The object-oriented approach supports the design and evolution of federated systems in different directions:

- In (reverse) modelling of relevant parts of information systems as well as in the (forward) analysis and design of new components.
- In relating information structures and their use within business processes by integrating structural and behavioural modelling.
- In relating conceptual and architectural design within a component-based engineering approach – one important aspect in continuous engineering of the system regarding all engineering viewpoints.

Due to its object-orientation, UML offers a large set of structural modelling elements including class structures and several options to define the semantics of relationships, a necessary feature on which we based the creation of the dependencies between models. In this case, for managing the UML models, we have to implement our own UML repository or to use an available system [14]. One of our future research goals is to find out whether or not ORDBMSs provide adequate mechanisms for managing UML models.

The second activity of the Federation Manager is the creation of the dependencies between stored UML models, referred to as the mapping creation (Figure 5b). This functionality consists of the following steps: 1) retrieving the models involved in a dependency from the repository, 2) creating the mapping and 3) storing the mapping in the repository, in a way that is appropriate for the Propagation Manager activity.

3.2. Propagation

To implement the propagation technique in SIES, we propose that the Propagation Manager is responsible of notifying all the systems involved in a dependency on a change in a model. In order to fulfill its role, the Propagation Manager consists of the following main components: the *Router*, the *Mapping Module* and the *Filter Module*. In this paper we assume that the Propagation Manager is built on message communication.

In Figure 6 we present the flow of a change in a Partial Model administrated by the Propagation Manager. The Propagation Manager activity starts (step1) when the source system (System A) puts the change notification into the input queue. This message is read by the Router which, based on its knowledge about stored dependencies relevant to the given change, forwards the change to all dependent systems. It accesses the dependency knowledge from the repository, in which all dependencies are stored. As we presented above, the Federation Manager administrates this repository. Before the update message reaches the target system, it will be transformed according to the model of the destination system.

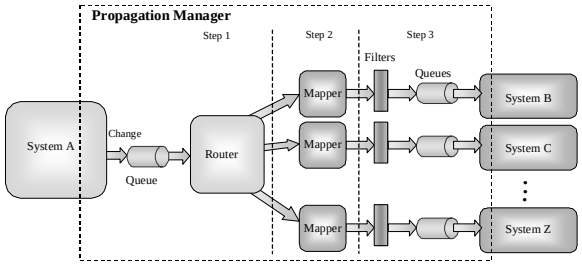

Figure 6. The Propagation Manager: the flow of update propagation

A Mapping Module, using a *mapping function*, achieves the second step, the transformation. A mapping function inputs the notification message in the format of system A and outputs a message relative to a target format. All information regarding the mapping functions is also stored in the repository. In the last step, a Filter can be used to send into the input queue of the dependent system only those messages on which the target system is really interested in. In using queues the receiving systems are decoupled from the source system as well as the Propagation Manager and vice versa. Thus, a system is free to issue a message or to receive messages at any preferred time. Additionally, the receiving system can be unavailable at the time when the message arrives.

In the following paragraph we propose some implementation ideas. For the recognition of modifications in the source system, the system must be extended with a mechanism that detects changes in the Partial Model. If the system is built on the top of a RDBMS, the trigger mecha-

nism in combination with Stored Procedures can be used to detect changes in the database and forward them to the Propagation Manager. We propose the Extensible Markup Language (XML [17]) for describing the messages. This gives us the opportunity to implement the mapping function by using the Extensible Stylesheet Language Transformation (XSLT [18]). XSLT allows the implementation of simple transformation from one XML-document to another. The XSLT-File describes the transformation process that is used by an XSLT-Processor. There is already an extension mechanism in XSLT, that can be used to implement more complex transformations.

3.3. Example Scenario

In order to motivate our work, related to the goal of achieving the transformability and the flexibility of the manufacturing production systems, we present in this section an example which operates with the concepts already explained: federation, repository and propagation.

Our scenario refers to a manufacturing enterprise that receives a new order. To quickly respond to this market change, the enterprise must activate two main systems of the enterprise that process this new demand: Order Management and Facility Layout Design. According to the concepts already introduced, these are viewed here as two TBUs. As a result to this new market situation, Order Management System will process the new information. A possible consequence of this change might be an increase of the enterprise capacity. This output represents the input for Facility Layout Design System, which, based on simulation and optimisation techniques, outputs possible alternatives of facility layout.

Figure 7 presents the integration of Order Management and Facility Layout Design Systems supported by our federation framework SIES. This is a global view that reveals in an integrated way, the responsibilities of Federation Manager, Repository and Propagation Manager.

We assume that the Federation Manager has already created the models of these two systems, the mappings between them, and that it has stored all this information in the repository. Both systems are in the initial state, before receiving the new order. The new situation (1) induces the transition of Order Management System state. This transition between states activates a change notification (2) to SIES. Being informed on this notification, the Propagation Manager triggers the change propagation, based on the stored mappings (3) among systems. The propagated change is then handed over (4) to the destination system, Facility Layout Design System, which, based on this notification, computes a new facility layout. Thus, based on federation and repository techniques, the facility layout is adapted to the new production requirements as defined by the initial "New order" change (1).

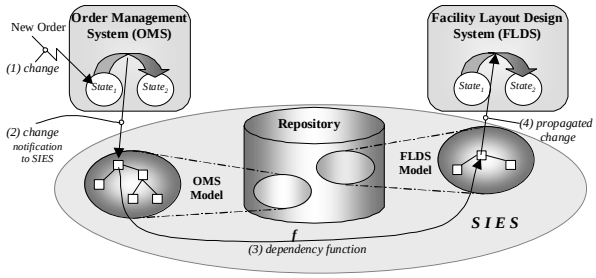

Figure 7. Scenario of change propagation in SIES

4. Conclusions and Future Work

In this paper we presented a concept for integrating information systems in manufacturing by means of a federation system. Our approach, Stuttgart Information and Exploration System (SIES), consists of the following three main components: the Federation Manager, the Propagation Manager and the Repository. We focused in our research on concepts like model and mapping and less on implementation details.

The Federation Manager's role is oriented in two directions: a) to create and manage the models and b) to install the dependencies in between them. All this is supported by a repository approach for extensibility and flexibility reasons. The dependencies are managed by the Propagation Manager, involving components like routers, mappers and filters.

Future work will be focused on implementing the conceptual techniques developed at this stage. As pointed out in this paper, there are many tech-

nical challenges in implementing the SIES managers. In the long term, we expect that a solution for these challenges will result in a valuable system that can be properly used as a federation solution for integrating independent technical databases or systems.

References

[1] W. Benn, Y. Chen and I. Gringer. FMS: A Federated System Manager. In *Technical Report SFB 786*, Technical University of Chemnitz-Zwickau, Germany, 1995.

[2] P. A. Bernstein. Repositories and Object-Oriented Databases. In *SIGMOD Record*, pp. 311-322, 1998.

[3] P. A. Bernstein and U. Dayal. An Overview of Repository Technology. In *Proceedings of 20th VLDB Conference*, pp. 705-715, 1997.

[4] S. Busse, R-D. Kutsche and U. Leser. Strategies for the Conceptual Design of Federated Information Systems. In *Proceedings of the 3rd Workshop EFIS 2000*, pp. 23-32, June 2000.

[5] S. Busse, R-D. Kutsche, U. Leser and H. Weber. Federated Information Systems: Concepts, Terminology and Architectures. In *Technical Report, Nr. 99-9*, Technical University of Berlin, Germany, 1999.

[6] Y. Breitbart, P. Olson and G. Thompsom. Database Integration in a Distributed Heterogeneous Information Environment. In *Proceedings of 2nd IEEE Conference Data Engineering*, pp. 301-310, 1986.

[7] S. Ceri and J. Widom. Managing Semantic Heterogeneity with Production Rules and Persistent Queues. In *Proceedings of 19th VLDB Conference*, Dublin, Ireland, pp. 108-119, 1993.

[8] P. Fankhauser, G. Gardarin and M. Lopez. Experiences in Federated Databases: From IRQ-DB to MIRO-WEB. In *Proceedings of 24th VLDB Conference*, pp. 655-713, 1998.

[9] G. Harhalakis, C. P. Lin, L. Mark and P. R. Muro-Medrano. Implementation of Rule-Based Information Systems for Integrated Manufacturing. In *IEEE Transaction on Knowledge and Data Engineering*, Vol. 6, No. 6, pp. 82-908, 1994.

[10] S. Iyengar, D. Ravi and D. E. Baisley. A Software Architecture for the Design and Implementation of Entire Distributed Object Systems: UML, MOF and XMI. In *XML Journal*, Volume: 1, Issue 3/2000.

[11] C. Kobryn. UML 2001: A Standardization Odyssey. In *Communications of the ACM*, Vol. 42, No. 10, pp. 29-37, October 1999.

[12] W. Litwin and A. Abdellatif. Multidatabase Interoperability. In *IEEE Computing Magazine*, Vol. 19, No. 12, pp. 10-18, 1986.

[13] Object Management Group. The Unified Modeling Language (UML) Specification – Version 1.3, 1999, available at http://www.omg.org/.

[14] N. Ritter and H.-P. Steiert. Enforcing Modeling Guidelines in an ORDBMS-based UML-Repository. In *Research Report of Subproject A3: Supporting Software Engineering Processes by Object-Relational Database Technology, SFB 501*, University of Kaiserslautern, Germany, 2000.

[15] Sonderforschungsbereich 467, Teilprojekt A5, *Modellierung von und Exploration in komplexen Unternehmensinformationen*, 2000, available at http://www.sfb467.uni-stuttgart.de/projekte/a5/ta5.html.

[16] A. P. Sheth and J.A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. In *ACM Computing Survey*, Vol. 22, Nr. 3, pp. 183-236, September 1990.

[17] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition) - W3C Recommendation 6 October 2000, available at http://www.w3.org /TR/2000/REC-xml-20001006.

[18] World Wide Web Consortium. XSL Transformations (XSLT) Version 1.0 - W3C Recommendation 16 November 1999, available at http://www.w3.org/TR/1999/REC-xslt-19991116