Models for Enterprise Computing 2006
International Workshop at EDOC 2006

Organizing Committee
Jean Bézivin, Jeff Gray, Alfonso Pierantonio

The Enterprise Computing Conference
Hong Kong, China
October 16, 2006
Contents

Foreword 5

Program committee 7

Modelling WS-RF based Enterprise Applications,
   Asif Akram, John Kewley, Rob Allan 7

Model Driven Design of Distribution Patterns for Web Service Compositions,
   Ronan Barrett, Claus Pahl 17

Towards Propagation of Changes by Model Approximations,
   Antonio Cicchetti, Davide Di Ruscio, Romina Eramo 25

Performance Improvement of MDD Tools,
   Benoît Langlois, Daniel Exertier, Stéphane Bonnet 35

Structural Patterns for the Transformation of Business Process Models,
   Marion Murzeck, Gerhard Kramler, Elke Michlmayr 43

Business Metadata for the Data Warehouse - Weaving Enterprise Goals and Multidimensional Models,
   Veronika Stefanov, Beate List 53

Modeling for Investment Guidance as a Performance Pre-assessment Method of Enterprise Integration,
   Jian Zhou, Qing Li, DaFeng Xu, TianYuan Xiao 63

A Methodology and Conceptual Framework for Flow-Manufacturing-oriented ERP Systems,
   Jianfu Zhang, Zhijun Wu, Ken Chen, Pingfa Feng, Dingwen Yu 71

Concepts for an Ontology-centric Technology Risk Management Architecture in the Banking Industry,
   Colin Atkinson, Christian Cuske, Tilo Dickopp 79

Index of authors 87
Foreword

Enhancing business performance in diverse domains (e.g., e-commerce, and logistics) requires systems whose size and intricacy challenge most of the current software engineering methods and tools. From early stages in the development of enterprise computing systems to their maintenance and evolution, a wide spectrum of methodologies, models, languages, tools and platforms are adopted. Thus, we decided to promote the International Workshop on Models for Enterprise Computing (IWMEC) as a satellite event of the Tenth IEEE International EDOC Conference (EDOC 2006). Our goal is to bring together researchers and practitioners to share experiences in using modelling as a universal paradigm that assists crosscutting methodologies and techniques to interoperate in a more general setting.

Shifting intellectual property and business logic from source code into models allows organizations to focus on the essential aspects of their systems, which have traditionally been blurred by the usage of standard programming languages and underlying technologies. Model-Driven Engineering (MDE) considers models as first-class entities enabling new possibilities for creating, analyzing, and manipulating systems through various types of tools and languages. Each model usually addresses one concern, and the transformations between models provide a chain that enables the automated implementation of a system initiating from its corresponding models.

The workshop will consider the nature and feature of models and domain-specific metamodels required to capture and measure particular aspects of enterprise computing (e.g., performance, distribution, security, load-balancing, and dependability) and specific business/application domains. Emphasis will be devoted to modelling enterprise legacy systems for integration and evolution, definition of (interoperable) enterprise model repositories, specification of model operations (composition, merging, and difference), model transformation and megamodelling, and the definition of development methodologies that allow all of the benefits of modelling to be realized.

This workshop is part of a more general effort comprising events and initiatives related to models, modelling and model transformations. We believe these topics will have strong impact on the way software is designed and developed. The general aim of the workshop is to provide a forum for those in academia and industry involved in the theory and practice modelling. The theme of the workshop is aligned with the model transformation web site (http://www.model-transformation.org), which promotes and pursues the adoption of models in software design and development.

The proceedings of this first IWMEC workshop contain nine papers that the Program Committee carefully selected out of fourteen submissions. We are indebted to our colleagues in the Program Committee for their willingness to assist us by offering their technical expertise in the refereeing and discussion process, thus making a major contribution to the successful production of the present volume. We wish to thank all those who submitted papers for the workshop and welcome all to the event. Even if your paper was not accepted this time, we express our appreciation for the time and effort you invested. We are also grateful to Antonio Vallecillo, the EDOC 2006 Workshop Chair, who promptly supported and guided us in the organization of the event.

Finally, we hope you will enjoy the meeting and have the opportunity to exchange your ideas and establish new collaborations. We look forward to your active participation in IWMEC 2006, and encourage you and your colleagues to submit your research findings to next year’s technical program. See you in Hong Kong!

Jean Bézivin, Jeff Gray, and Alfonso Pierantonio
IWMEC 2006 Chairs
Program committee

Alfonso Pierantonio, University of L’Aquila, Italy
Jean Bézivin, University of Nantes, France
Jeff Gray, University of Alabama at Birmingham, USA
Ana Moreira, Universidade Nova de Lisboa (P)
Anneke Kleppe, University of Twente (NL)
Arne Berre, Sintef (N)
Brandon Eames, Utah State University (USA)
Daniel Varro, University of Budapest (H)
David Frankel, SAP (USA)
Dirk Weise, Interactive Objects (D)
Dominik Stein, University of Essen (D)
Eelco Visser, Universiteit Utrecht (NL)
Gerti Kappel, Technical University of Vienna (A)
Laurence Tratt, King’s College (UK)
Marten van Sinderen, University of Twente (NL)
Martin Gogolla, University of Bremen (D)
Nora Koch, University of Munich (D)
Pavel Hruby, Microsoft Business Solutions (DK)
Raúl Romero, University of Córdoba (E)
Reiko Heckel, University of Leicester (UK)
Modelling WS-RF based Enterprise Applications

A. Akram, J Kewley and R. Allan

CCLRC e-Science Centre, Daresbury Laboratory, Warrington, WA4 4AD, UK
{a.akram, j.kewley, r.j.allan}@dl.ac.uk

Abstract

The Web Service Resource Framework (WS-RF) specifications originated from the Grid paradigm which has no widespread programming methodology and lacks established design models. The flexibility and richness of WS-RF specifications are ideal for the complex, unpredictable and inter-dependent components in an Enterprise Application. This paper presents a Model-Driven approach for WS-RF to meet the requirements of Enterprise Applications (EAs) spread across multiple domains and institutes. This Model-Driven approach addresses cross-platform interoperability, quality of service, design reuse, systematic development and compliance to user requirements at the design level.

1. Introduction

Modular software is designed to avoid failures in large enterprise systems, especially where there are complex user requirements. A Services Oriented Architecture (SOA) is an architectural style whose goal is to achieve loose coupling among interacting software agents (services and clients). A service is a function that is self-contained and immune to the context or state of other services. These services can communicate with each other, either through explicit messages (which are descriptive rather than instructive), or by a number of ‘master’ services that coordinate or aggregate activities together, typically in a workflow. An SOA can also define a system that allows the binding of resources on demand using resources available in the network as independent services.

In recent years, Web Services have been established as a popular “connection technology” for implementing SOAs. The well-defined interface required for a service is described in a WSDL file (Web Service Description Language [1]). Services exposed as Web Services can be integrated into complex workflows which may span multiple domains and organizations.

There is growing interest in the use of stateless Web Services for scientific, parallel and distributed computing [2]. Web Services Resource Framework (WS-RF) [3] specifications built on top of existing Web Services standards address the limitation of stateless Web Services through the concept of WS-Resources by defining conventions for managing a ‘state’ so that applications can reliably share the information, and discover, inspect and interact with stateful resources in a standard and interoperable way [4]. The lack of a recognized Model Driven strategy, standard patterns and reusable concepts for stateful resources generally results in ad-hoc solutions which are tightly coupled to specific problems, and are not applicable outside the targeted problem domain. This code driven approach is neither reusable nor does it promote dynamic adaptation facilities as it should do in an SOA.

In the Section 2, we discuss the abstract concepts related to WS-RF and WS-Resources and we have envisioned various possible interaction mechanisms for the WS-Resources. In the Section 3, concrete design modelling approaches are presented with respect to the WS-Resource instantiation and Section 4 covers different Notification Models for the WS-Resource state change.

2. Web Services Resource Framework

Web Services lack the notion of state, stateful interactions, resource lifecycle management, notification of state changes, and support for sharing and coordinated use of diverse resources in dynamic ‘virtual organizations’ [5]: issues that are of central concern to the developers of distributed systems. To address these problems, two important sets of specifications: WS-Resource Framework and WS-Notification [7], built on the broadly adopted Web Services architecture [6] and compliant with the WS-Interoperability Basic Profile [13], were proposed.

WS-RF originates from the Grid paradigm which can be described as “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations”. Grid design should ensure cross-
platform interoperability and re-usability of systems in a heterogeneous context; although being a recent computing discipline, Grid Computing lacks established programming practices and methodologies.

WS-RF specifications are based on the Extensible Markup Language (XML) schemas, and Web Services Definition Language (WSDL) interfaces for the properties and ports common to all WS-RF resources. WS-RF comprises four inter-related specifications; which define how to represent, access, manage, and group WS-Resources:

- **WS-ResourceProperties** [8] defines how WS-Resources are described by XML documents that can be queried and modified;
- **WS-ResourceLifetime** [9] defines mechanisms for destroying WS-Resources;
- **WS-ServiceGroup** [10] describes how collections of Web Services can be represented and managed;

### 2.1 WS-Resources

WS-Resources model the state of a Web Service by wrapping atomic/composite data types called WS-Resource Properties. A Resource Property is a piece of information defined as part of the state model, reflecting a part of the WS-Resource’s state, such as its meta-data, manageability information and lifetime.

![Figure 1: A WS-Resource with Resource Properties](image1)

The WS-RF specification supports dynamic insertion and deletion of the Resource Properties of a WS-Resource at run time. Customer details in a Trading System are a single WS-Resource with multiple Resource Properties like name, address, card details and trading history. The address Resource Property can have multiple entries such as billing address and shipping address. Trading history is a dynamic Resource Property, which is added for every new order and may automatically be deleted after a given period. WS-Resource itself is a distributed object, expressed as an association of an XML document with a defined type attached with the Web Service portType in the WSDL. Although WS-Resource itself is not attached to any Uniform Resource Locator (URL), it does provide the URL of the Web Service that manages it. The unique identity of the WS-Resource and the URL of the managing Web Service is called an Endpoint Reference (EPR), which adheres to Web Services Addressing (WSA) [12]. WS-RF avoids the need to describe the identifier explicitly in the WSDL description by instead encapsulating the identifier within its EPR and implicitly including it in all messages addressed through it.

WS-Resources instances have a certain lifetime which can be renewed before they expire; they can also be destroyed pre-maturely as required by the application.

### 2.2 WS-Resource Sharing

WS-Resources are not bound to a single Web Service; in fact multiple Web Services can manage and monitor the same WS-Resource instance with different business logic and from a different perspective. Similarly, WS-Resources are not confined to a single organization and multiple organizations may work together on the same WS-Resource leading to the concept of collaboration. Passing a unique identity of the WS-Resource instance between partner processes and organizations results in minimum network overhead and avoids issues of stale information. The WS-Resource EPRs are generated dynamically and can be discovered, inspected and monitored dynamically via dedicated Web Services. In Figure 2, Resource B is shared between two different Web Services each of them exposing possibly different sets of operations on the same WS-Resource, for instance to provide both an administrators’ and users’ perspective (where an administrator can modify the data and a user can only query the data).

![Figure 2: Web Service managing multiple WS-Resources and two Web Services sharing same WS-Resource.](image2)
WS-Resource sharing is used extensively for load balancing by deploying semantically similar or cloned Web Services for multiple client access. At run time, appropriate EPRs of the WS-Resource are generated with the same unique WS-Resource identity but with different URLs of managing Web Services.

2.3 Managing Multiple WS-Resources

In EAs, WS-Resources related to different entities can be very similar. Seller and Buyer details, for instance, are different WS-Resources in the sample Trading Application, but the majority of operations executed on these WS-Resources are either queries or minor updates. It is more effective to manage these similarly natured operations on different WS-Resources with the single instance Service. In Figure 1, different instances of the same WS-Resource are being managed by a single Web Service; whereas in Figure 2, multiple Web Services are managing different WS-Resources which can have any number of instances. A single Web Service managing multiple WS-Resources could be deployed as a Gatekeeper Service which creates instances of different WS-Resources, returning the corresponding EPR. It could also be deployed as a Monitoring Service which monitors the state of different but inter-dependent WS-Resources. For example, when a particular stock level drops below a threshold value, the WS-Resource related to the order is either created or updated. Monitoring services have different applications like managing Quality of Service (QoS), recording usage for statistical analysis or enforcing WS-Resource dependencies and resolving conflicts: features which can be crucial for Enterprise Applications.

2.4 WS-Resource Referencing

WS-Resources are composed of Resource Properties which reflect their state. These can vary from simple to complex data types and even reference other WS-Resources. Referencing other WS-Resources through Resource Properties is a powerful concept which defines inter-dependency of the WS-Resources at a lower level. This eliminates complicated business logic in a similar way to the mapping of Entity Relationships in a Relational Database through primary and foreign keys. In EAs entities do not exist in isolation but inter-communicate and are dependent on each other’s state. Similarly, WS-Resources are not only dependent on the state of other WS-Resources but can even query and modify them. In a Trading System (see Figure 3), a single User may reference multiple Orders placed by that User on different occasions and each Order references varying numbers of items purchased as a part of a single Order.

3. Modelling the Implied Resource Pattern

The WS-RF specifications recommend the use of the Implied Resource pattern (Figure 4) to describe views on state and to support its management through associated properties. The Implied Resource pattern has a single Factory Service to instantiate the resources and an Instance Service to access and manipulate the information contained in the resources according to the business logic.

During the prototype development we investigated variations of the Implied Resource pattern to model varying enterprise requirements.

3.1 Factory/Instance Pair Model

The Factory/Instance Pair Model (Figure 5) is the simplest model, in which for each resource there is a Factory Service to instantiate the resource and
corresponding Instance Service to manage the resource. In a typical EA different Factory Services are independent of each other and can work in isolation. This is the simplest approach: repeating the similar resource instantiating logic in multiple Factory Services or even the same Factory Service which can be deployed multiple times.

During the development and testing phase it is easier to test a set of Factory/Instance Services and WS-Resources in isolation using a black box testing methodology. This approach requires some fairly complicated logic for the client who must interact with the Factory Services to instantiate different WS-Resources and update the mandatory WS-Resource to reference every optional WS-Resource. The server side implementation of this model is quite simple. The Server-Client implementation is tightly integrated, leaving minimum flexibility in design change without altering the client application.

3.2 Factory/Instance Collection Model

The Factory/Instance Collection Model (Figure 6) is an extension of the Factory/Instance Pair Model. The difference being that a single Factory Service instantiates multiple WS-Resources managed by different Instance Services. Enterprise Applications process various entities which are tightly coupled and due to this inter-dependency all of these WS-Resources must co-exist before a client may interact with them successfully. This model is suitable for core WS-Resources which should be instantiated as early as possible during the application lifecycle. For example, in a Banking System, when a user opens a bank account, three WS-Resources are created immediately: User, Current Account and Saving Account; the User WS-Resource will reference the other two WS-Resources through their EPRs.

The implementation of this Factory Service is likely to be more complicated than the previous, depending on the requirements of the application, and can be implemented in two possible ways:

Returning an Array. This is the simplest scenario where a Factory Service instantiates multiple Resources...
and returns the array of EPRs corresponding to each Resource. The client application parses an array of EPRs and manages each of them accordingly, putting more workload on the client. This solution is “Fragile” as all clients need to know how to handle each Resource and its inter-dependencies.

Returning a Single EPR. In this recommended approach, a single Factory Service implements the business logic and inter-dependency of the different Resources. The Factory Service instantiates all the Resources yet returns only a single EPR which contains references to other Resources; this may or may not be changed by the client. The Banking example quoted above falls in this category: a Client can’t modify his Account since any change would have to be authorized by the bank administration. Returning a single EPR requires a more complicated Factory Service but a much easier to implement client, resulting in a comparatively robust application.

The Factory/Instance Collection Model results in less interaction between the client and the server with minimum network overhead. The Server-Client implementation is quite flexible with minimum inter-dependency, providing the opportunity to update the server business logic without changing the client application. The biggest disadvantage of the Factory/Instance Collection Model approach is the larger initial processing overhead with its correspondingly longer initial latency, especially when the WS-Resources are geographically distributed. Due to the fundamental nature of WS-Resources’ late binding, loose coupling and reusability are fairly limited in this model.

3.3 Master-Slave Model

In a security dominated era with its unpredictable request traffic, different security and load balancing measures are required for an application to run smoothly. These measures should be planned at design time, irrespective of the technologies to be used for implementation. EAs are frequently protected by a firewall. It has to be anticipated that firewall policies will limit direct access from external clients to Resources (i.e. it is most likely that these Resources will be located inside private firewalls, and can only be accessed via known gateway servers). Consequently, an extensible Gateway model is required for accessing these resources. This model mandates that all client requests are sent to an externally visible Gateway Web Service before being routed through the firewall to the actual requested service. In addition, firewall administrators may implement additional security measures such as IP-recognition between gateway server and service endpoint in the form of Web Services handlers.

One approach is to use a Gateway Service to manage multiple Factory Services in the Master-Slave format; the client interacts only with the Master Factory Service without knowing the inner details of the application. The Master Factory Service performs authentication and authorization of the client before invoking respective Factory Services (Slaves) which are behind the firewall and restricted by strict access polices.

![Figure 7: Master-Slave Model](image)

Services expecting higher volumes of user traffic can be cloned and deployed on different nodes in the cluster. The Master Factory service can use any monitoring mechanism to monitor each node related to its service load, quality of service and availability of external resources and thus redirect the clients’ requests to the most appropriate node (ultimate service endpoint). The main advantage of this approach is that if at any time any service is overloaded or even unavailable, then that service can easily be replaced by a compatible or cloned service on another node with minimal effort. The implementation of the corresponding client is quite simple since the client interacts only with the single Master Factory Service and is independent of the location of other protected services. The business model can be modified, updated and refined without affecting the client application as long as the Master Factory Service still provides the same interface.

3.4 Hybrid Model

We propose that the best approach is to combine these variations of the Implied Resource Pattern as follows. The client still interacts with a single Factory
Service which instantiates all mandatory WS-Resources and returns a single EPR. Subsequent client interactions invoke the 'create' operation of the Factory Service with different 'parameters' with the Factory Service instantiating the corresponding WS-Resources according to those parameters. Optional WS-Resources are supported using a Factory/Instance Pair model due to their limited usage. The core WS-Resources which are to be shared among different applications are also instantiated through the Factory/Instance Pair model. The Factory Service is an extension of the Factory/Instance Collection Model with request parsing capabilities, utilising advanced features of XML Schema in its WSDL interface. Since WSDL specifications prohibit the overloading of methods, the Factory Service implements a single “create” operation wrapping all the parsing logic in a few private utility methods. Our experience in using WS-RF for different distributed applications has shown that the “Hybrid Approach” is more manageable and easier to maintain. This is due to having a single Factory Service, shorter response times, inter-dependency logic confined to the server and the facility to upgrade or add WS-Resources.

4. WS-RF and Notification Model

The Event-driven, or Notification-based, interaction model is commonly used for inter-object communications. Different domains provide this support to various degrees: “Publish/Subscribe” systems provided by Message Oriented Middleware vendors; support for the “Observable/Observer” pattern in programming languages; “Remote Eventing” in RMI and CORBA. Due to the stateless nature of Web Services, the Web Service paradigm has no notion of Notifications. This has limited the applicability of Web Services to complicated application development. WS-RF defines conventions for managing 'state' so that applications can reliably share information as well as discover, inspect, and interact with stateful resources in a standard and interoperable way. WS-Notification (WSN) [7] is a set of three separate specifications (WS-BaseNotification, WS-BrokeredNotification, and WS-Topics), but its usefulness beyond WS-RF is limited.

The WSN specification defines the Web Services interfaces for Notification-Producers and Notification-Consumers. It includes standard message exchanges to be implemented by service providers (producers) and clients (consumers) that wish to act in these roles, along with the operational requirements expected of them. Notification Consumers subscribe with Notification Producers to request asynchronous delivery of messages. A subscribe request may contain a set of filters that restrict which notification messages are delivered. The most common filter specifies a message topic using one of the topic expression dialects defined in WS-Topics (e.g., topic names can be specified with simple strings, hierarchical topic trees, or wildcard expressions). Additional filters can be used to examine message content as well as the contents of the Notification Producer’s current Resource Properties. Each subscription is managed by a Subscription Manager Service (which may be the same as the Notification Producer). Clients can request an initial lifetime for subscriptions, and the Subscription Manager Service controls subscription lifetime thereafter. Clients may unsubscribe by deleting their subscription through the Subscription Manager Service. When a Notification Producer generates a message that is sent wrapped in a <Notify> element (though unwrapped “raw” delivery is also possible) to all subscribers whose filters evaluate to ‘true’. WS-BrokeredNotification provides for intermediaries between Notification Producers and Notification Consumers. These intermediaries receive messages from Notification Producers and broadcast them to their own set of subscribers, allowing for architectures in which Notification Producers do not want to, or even cannot, know who is subscribed. In any notification model a Web Service, or other entity, disseminates information to a set of other Web Services or entities, without prior knowledge of them.

4.1 Client as a Notification Consumer

In this approach the client application acts as a Notification Consumer; which is notified of any change in the “state” of the subscribed WS-Resource instance. The client processes the notification messages and updates instance/s of other related WS-Resources through corresponding Instance Services. It is the client’s responsibility to inter-relate dependent Resource instances. The client application exposes a ‘notify’ operation to receive asynchronous notification messages and must implement the complex logic of associating different Resource instances. An EA, on the other hand, is simpler to maintain and independent of notification. There can be many scenarios where the client receives optional notifications which are independent of core functionality of application. Notification processing at the application level can be an overhead due to enormous amount of messages. The notification and subscription at the application level can result in a cyclic notification chain. In a Travel
application, clients subscribe for a particular type of deal (e.g., deals related to South Asia, family packages, multi-city tours or budget deals). With various categories and possible subcategories, managing the notifications can be a significant overhead at the application level since most do not result from a client’s subscription. The client as a notification consumer is only applicable for low priority notifications where immediate action is not required. This notification model does not assume that the client application is continuously executing and any delay in response should not affect the core functionality of the Enterprise Application.

The client can also delegate any other service as a notification consumer on its behalf; provided that the service fulfills the criteria of being a ‘Notification Consumer’ by implementing the appropriate WSN interface. Such delegated services are independent of the business logic of the EA, although they may be provided as utility services. These are provided for the clients who do not want to be notification consumers, are not available all the time, are behind a firewall, or who do not have a reachable notification interface (e.g., desktop clients).

4.2 Service as a Notification Consumer

At the application level, different services managing different WS-Resource instances can have inter-dependencies. These services may have an interest in the state of other WS-Resource instances. It is therefore better to handle notifications of these state changes at the service level without any client interaction. This is the situation where automatic and quick action is required; these actions are not initiated by the client. The client has no role in these decisions and the actions required are related to the core functionality of the application and not with any specific client. This approach is generic where certain types of notifications are processed for a certain clients (e.g., changing the overdraft limit for all international students, a discount offer for all loyal customers, upgrading broadband speed for all customers in central London). These are more or less management and application level policies and updating the appropriate WS-Resource instances should be handled at the application level.

The client application may not even be aware of any such WS-Resource instances and their relationships. The client applications are therefore simplified with most of the processing logic residing on the server side. These server implementations can be extended so that users can subscribe and unsubscribe to certain types of notification; this requires business level support for such filtering.

Overall this approach results in a cleaner design with an easier to manage and maintain EA. Since the notification processing logic is confined to the server, the client application is immune to the ‘state’ changes and there is therefore no need to update the client logic whenever the business logic changes. The main drawback in this approach is that of the loose associations between WS-Resource instances which are of a one-to-many or many-to-many linkage rather than one-to-one.

4.3 Resource as a Notification Consumer

The two notification approaches discussed above (Sections 4.1 and 4.2) have their own limitations and benefits. A third notification model can provide the best of both approaches with an even cleaner design. Applications are still easy to manage and maintain and WS-Resource instances can have one-to-one associations. In this approach WS-Resource itself is a notification consumer, yet may also act as a producer. Each instance of the WS-Resource can subscribe to ‘state’ changes of specific WS-Resource instances whilst broadcasting notification messages related to its own ‘state’. Overall this mechanism gives tighter control on the business logic without interference from the client side (e.g., if the outstanding balance in a customer’s current account is insufficient to pay a bill, funds are either transferred from the customer’s savings account). Implementing a WS-Resource as a notification consumer or a consumer-producer can result in large numbers of messages which can overload the Subscription Manager Service, thus affecting the overall performance of the application. The more inter-related instances, the worse the problem becomes. This model should be applied with caution and WS-Resources serving as notification consumers should obey the following guidelines:

- There is a controlled number of instances of each WS-Resource at any given time;
- Each WS-Resource has limited dependency on other WS-Resources and is not involved in complicated association linkage;
- At least one of each WS-Resource instance is available all the time, Brokered Notification is required for persistent WS-Resources;
- Producer-consumer WS-Resources should be avoided if possible to avoid cyclic notification chains.
4.4 Hybrid Approach

There is no clear rule when to recommend the use of one or another of the Resource-Notification models and when they should be avoided. The application of any specific model depends on the overall role of the WS-Resources in the application; their availability and inter-dependencies; the context in which each WS-Resource is used; and the type of notification to be used (either point-to-point or broker-based notification). Distributed applications spanning multiple domains comprise large number of data entities (WS-Resources), each of them having different roles. This requires mixing different patterns to have the best of both worlds (a clean design with a manageable and maintainable implementation). This suggests a Hybrid Approach; which requires a clear understanding of the limitations and advantages of each approach. A typical EA client or delegated service will act as a notification consumer for optional Resources (which don’t participate in core activities of the application and have low priority in the overall Resource hierarchy). The generic and application level notification messages will be processed by the services, whereas the critical and one-to-one association notification messages will be handled at the WS-Resource level.

5. Conclusions

WS-RF specifications are designed on top of Web Services specifications to address the implementation of heterogeneous and loosely-coupled distributed applications. WS-RF provides the missing concepts of stateful interactions, state change notification, and support for the sharing and coordinated use of diverse resources in an easy and standard way. Being a comparatively new programming paradigm, Grid computing does not have established programming methodologies. In this paper we have investigated a Model-Driven approach to encapsulate different applications and user requirements to design Enterprise Applications. This Model-Driven approach can be used from design to deployment of an application in a standard way. MDE is a useful paradigm to develop reusable, loosely coupled, scaleable and efficient systems. The use of models eases the transformation from design to the implementation of the system in a robust and flexible manner. We have presented different models based on the Implied Resource Pattern to seek a common strategy for the functional and non-functional requirements of the mandatory and optional WS-Resources.

WS-RF based applications can take advantage of many Web Services specifications to implement any level of complexity with flexibility, portability and interoperability. Migration to WS-RF is conceptually easier if the application and user requirements can be modelled earlier in the design phase. The paper further discusses different possible notification models for event-driven systems with their advantages and disadvantages in conjunction with WS-Resources and concluded that the different design models are complementary and can be combined to form a Hybrid Model.

6. References

[1] Web Services Description Language (WSDL) 1.1, Available at http://www.w3.org/TR/wSDL.
Model Driven Design of Distribution Patterns for Web Service Compositions

Ronan Barrett
School of Computing
Dublin City University
Dublin 9, Ireland
Email: rbarrett@computing.dcu.ie

Claus Pahl
School of Computing
Dublin City University
Dublin 9, Ireland
Email: cpahl@computing.dcu.ie

Abstract

Increasingly, distributed systems are being constructed by composing a number of discrete components. This practice, termed composition, is particularly prevalent within the Web service domain. Here, enterprise systems are built from many existing discrete applications, often legacy applications exposed using Web service interfaces. There are a number of architectural configurations or distribution patterns, which express how a composed system is to be deployed. However, the amount of code required to realise these distribution patterns is considerable. In this paper, we propose a novel Model Driven Architecture using UML 2.0, which takes existing Web service interfaces as its input and generates an executable Web service composition, based on a distribution pattern chosen by the software architect.

1. Introduction

The development of composite Web services is often ad-hoc and requires considerable low level coding effort for realisation [1]. This effort is increased in proportion to the number of Web services in a composition or by a requirement for the composition participants to be flexible [5]. We propose a modeling and code generation approach to address this requirement. This approach suggests Web service compositions have three modeling aspects. Two aspects, service modeling and workflow modeling, are considered by [23]. Service modeling expresses interfaces and operations while workflow modeling expresses the control and data flow from one service to another. We consider an additional aspect, distribution pattern modeling [25], which expresses how the composed system is to be deployed. Distribution patterns are an abstraction mechanism useful for modeling. Having the ability to model, and thus alter the distribution pattern, allows an enterprise to configure its systems as they evolve, and to meet varying non-functional requirements.

We base our development approach on the OMG’s Model Driven Architecture (MDA) [12]. MDA considers models as formal specifications of the structure or function of a system, where the modeling language is in fact the programming language. Having rich, well specified, high level models allows for the auto-generation of a fully executable system based entirely on the model. Our models will be generated based on existing Web service interfaces, requiring only limited intervention from a software architect, who defines the distribution pattern, to complete the model.

Modeling of the composed system’s distribution pattern is important, as this novel modeling aspect, provides a more complete picture of the non-functional requirements realised by a distributed system. Our approach provides a high level model which intuitively expresses, and subsequently generates, the system’s distribution pattern using a UML 2.0 based Activity diagram [11]. An associated benefit of our modeling approach is the fast and flexible deployment of compositions. Motivated by these concerns, we have devised an approach, a technique and an implementation, for the model driven design of distribution patterns.

The paper is structured as follows: section two motivates distribution patterns; section three introduces our modeling and transformation approach; section four investigates our model and transformation technique, complimented by a case study; section five considers our tool implementation; section six presents related work; finally, section seven considers future work and concludes the paper.

2. Distribution Patterns

There is a subtle difference between two of the modeling aspects within a Web service composition, namely workflows and distribution patterns [25]. Both aspects refer to the high level cooperation of components, termed a collaboration, to achieve some compound novel task [22]. We consider workflows as compositional orchestrations, whereby the internal and external messages to and from services are modeled. In contrast, distribution patterns are consid-
ered compositional choreographies, where only the external messages flow between services is modeled. Consequently the control flow between services are considered orthogonal. As such, a choreography can express how a system would be deployed. The internal workflows of these services are not modeled here, as there are many approaches to modeling the internals of such services [9, 14]. Merging both modeling efforts has been identified as future work.

Distribution patterns express how a composed system is to be assembled and subsequently deployed. These patterns are a form of platform-independent model (PIM) [12], as the patterns are not tied to any specific implementation language. The patterns identified are architectural patterns, in that they identify reusable architectural artifacts evident in software systems.

To help document the distribution patterns, discussed in this paper, a modeling notation is required. UML is a standards based graphical language for the modeling of software systems [19]. UML documents a system using two categories of diagrams, structural and behavioural.

Different distribution patterns realize different non-functional requirements. Some of these requirements are often grouped under the term, Quality of Service (QoS) [13]. In [2], four categories of QoS which affect systems at runtime are outlined, performance, dependability, safety and security. The first three categories are of particular relevance here. Some specific QoS characteristics applicable to distribution patterns, in addition to some design time issues, are detailed in [18, 21].

The patterns presented here were identified by systematically researching distribution patterns, in existing network based systems. Many of the patterns discussed here are identified by Ding et al.[10], whilst the QoS attributes of the core patterns are documented by [5, 8, 25]. However, their description in an MDA based Web service context is novel. There are three pattern categories, as follows.

- Core patterns
  - Centralised Dedicated-Hub
  - Centralised Shared-Hub
  - Decentralised Dedicated-Peer
  - Decentralised Shared-Peer
- Auxiliary patterns
  - Ring
- Complex patterns
  - Hierarchical
  - Ring + Centralised
  - Centralised + Decentralised
  - Ring + Decentralised

In order to exploit the potential of pattern-driven choreography definition, the consideration of a variety of patterns would be beneficial. To illustrate the principles, we however, focus on the two most prevalent patterns, centralised and decentralised (see Figure 1), before briefly describing the other patterns.

![Figure 1. Examples of distribution patterns](image)

In a centralised shared-hub pattern [5], a composition is managed in a single location by the enterprise initiating the composition. This pattern is the most widespread and is appropriate for compositions that only span a single enterprise. The advantages are ease of implementation and low deployment overhead, as only one controller is required to manage the composition. However, this pattern suffers from a communication bottleneck at the central controller. This represents a considerable scalability and availability issue for larger enterprises. The decentralised shared-peer pattern [25] addresses many of the shortcomings of the centralised shared-hub pattern by distributing the management of the composition amongst its participants. This pattern allows a composed system to span multiple enterprises while providing each enterprise with autonomy [24]. It is most important for security that each business acts upon its private data but only reveals what is necessary to be a compositional partner. In a decentralised pattern, the initiating peer is only privy to the initial input data and final output data of a composition. It is not aware of any of the intermediate participant values, unlike a centralised pattern. The disadvantages of a decentralised pattern are increased development complexity and additional deployment overheads.

We also consider two other core distribution patterns, dedicated hub and dedicated peer, which may be applied to the first two distribution patterns and their complex variants when additional autonomy, scalability and availability is required. The other distribution patterns are the ring pattern, which consists of a cluster of computational resources providing load balancing and high availability, and the hierarchical pattern, which facilitates organisations whose management structure consists of a number of levels, by providing a number of controller hubs. There are also complex variants of these distribution patterns, whereby a mix of two or more patterns are combined. Complex patterns are useful in that the combination of patterns often results in the elimination of a weakness found in a core pattern.

Two collaboration languages, Web Services Business
Process Execution Language (WS-BPEL) and Web Service Choreography Description Language (WS-CDL) [22], can enable the runtime enactment of distribution pattern based compositions. WS-BPEL is an orchestration language whilst WS-CDL is a choreography language. The WS-CDL language provides the most obvious mapping to distribution pattern specification, as only the messages exchanged between collaborators are considered.

3. Modeling and Transformation Approach

The basis of our modeling and transformation approach is illustrated in Figure 2, as outlined by Bézivin in [6], and previously utilised in a web based engineering context by Koch in [17]. We outline the model transformation pattern from UML to our distribution pattern language, and subsequently to a collaboration language. Our relations are defined at the meta-model level using both the recently standardised QVT (Query/View/Transformation) textual and graphical notations [20]. Using QVT relations we have analysed our approach for completeness by verifying the preservation of semantics between related metamodels.

![Figure 2. Model transformation pattern](image)

An example of a QVT relation using the graphical notation can be seen in Figure 3. This relation states that the order value of a Node, from the DPL meta-model, is derived by the order value between two UML CallBehaviorAction elements, from the UML meta-model/DPL Profile.

![Figure 3. Example QVT relation](image)

Our approach to distribution pattern modeling and subsequent Web service composition generation consists of five steps, as illustrated in Figure 4, and subsequently described below.

![Figure 4. Overview of modeling approach](image)

**Step 1 - From Interface To Model:** The initial step takes a number of Web service interfaces as input, and transforms them to the UML 2.0 modeling language standardised by OMG [11], using the UML 2.0 model generator. These interfaces represent the services which are to be composed. The model generated is based on the web services inputted, however, each service is logically separated as no composition has yet been defined.

**Step 2 - Distribution Pattern Definition:** The model produced in step 1 requires limited intervention from a software architect. Guided by a chosen distribution pattern, and restricted by the UML meta-model/DPL Profile (see Figure 5), the architect must manipulate the UML model by defining connections between individual Web services and map the messages from one service to the next. Finally, the architect must set some distribution pattern specific variables on the model, which will be used to generate a distribution pattern instance. Partial automation of this step using semantics is considered in our related paper [3].

**Step 3 - From Model to DPL:** Using the model generated in step 2 as input, the model is transformed to a distribution pattern instance, using the distribution pattern generator. The transformation and resultant pattern instance are restricted by the DPL meta-model. This pattern instance, represented in XML using our novel specification language Distribution Pattern Language (DPL), is called a DPL document instance. The DPL specification, written in XML Schema, has no reliance on UML and so any number of modeling techniques may be used as an input. The use of this new language allows non-MOF compliant description frameworks, such as Architectural Description Languages and the $\pi$ calculus, to be used in place of UML as the transformation source.
**Step 4 - Model Validation:** The DPL document instance, representing the distribution pattern modeled by the software architect, is verified at this step by the distribution pattern validator, to ensure the values entered in step 2 are valid. If incorrect values have been entered, the architect must correct these values, before proceeding to the next step. Validation of the distribution pattern instance is essential to avoid the generation of an invalid system. Although this validation may be considered redundant as the pattern definition has already been restricted by the QVT relations, we envisage supporting non-QVT compliant modeling languages as set out in our future work.

**Step 5 - DPL to Executable System:** Finally, the executable system generator takes the validated DPL document instance and generates all the code and supporting collaboration document instances required for a fully executable system. These documents are restricted by the appropriate platform specific collaboration meta-model. This executable system will realise the Web service composition using the distribution pattern applied by the software architect. All that remains is to deploy the generated artifacts and supporting infrastructure to enable the enactment of the composed system. Dynamic deployment of the executable system is considered in our related paper [4].

### 4. Modeling and Transformation Technique

In this section, we introduce the techniques we have developed for the modeling and transformational approach presented in section 3, before evaluating our technique in the last subsection. There are three specific techniques listed below and elaborated in the five specific steps that follow. As before each step is illustrated in Figure 4.

- **UML activity diagram/Profile extension (step 1,2)**
- **DPL/DPL validator (step 3,4)**
- **Generators (step 1,3,5)**

The technique is accompanied by a small scale case study which motivates our solution. Our case study is an enterprise banking system with three interacting business processes. We choose an enterprise banking system as it is susceptible to changes in organisational structure while requiring stringent controls over data management, two important criteria when choosing a distribution pattern. The scenario involves a bank customer requesting a credit card facility. The customer applies to the bank for a credit card, the bank checks the customer’s credit rating with a risk assessment agency before passing the credit rating on to a credit card agency for processing. The customer’s credit card application is subsequently approved or declined.

#### 4.1. Step 1 - From Interface To Model

As Web services’ WSDL interfaces are constrained by XML Schemas, their structure is well defined. This allows us to transform the interfaces, using the UML 2.0 model generator, into a UML 2.0 activity diagram, an approach also considered by [9]. The UML model generated contains many of the new features of UML 2.0, such as Pins, CallBehaviorActions and ControlFlows.

A UML activity diagram is chosen to model the distribution pattern as it provides a number of features which assist in clearly illustrating the distribution pattern, while providing sufficient information to drive the generation of the executable system. Activity diagrams show the sequential flow of actions, which are the basic unit of behaviour, within a system and are typically used to illustrate workflows.

UML ActivityPartitions, also known as swim-lanes, are used to group a number of actions within an activity diagram. In our model, these actions will represent WSDL...
operations. Any given interface has one or more ports that will have one or more operations, all of which will reside in a single swim-lane. To provide for a rich model, we use a particular type of UML action to model the operations of the WSDL interface. These actions, called CallBehaviorActions, model process invocations and have an additional modeling constructs called pins. There are two types of pins, InputPins and OutputPins, which map directly to the parts of the WSDL messages going into and out of a WSDL operation. For our UML activity diagram to effectively model distribution patterns, we require the model to be more descriptive than the standard UML dialect allows. We use a standard extension mechanism of UML, called a profile [12]. Profiles define stereotypes and subsequently tagged values that extend a number of UML constructs. Each time one of these derived constructs is used in our model we may assign values to its tagged values. An overview of our profile can be seen in Figure 5. The profile extends the Activity, ActivityPartition, CallBehaviorAction, ControlFlow, InputPin and OutputPin UML constructs. This extension allows distribution pattern metadata to be applied to the constructs via the tagged values. For example, the distribution pattern is chosen by selecting a pattern from the DistributionPattern enumeration and assigning it to the distributionPattern tagged value on the DPL-Metadata construct.

The banking case study provides three WSDL interfaces as input to the UML 2.0 model generator. These interfaces represent the bank (CoreBanking), the risk assessment agency (RiskManagement) and the credit card agency (CreditCard). All three are represented in the generated UML activity diagram, albeit without any connections between them. A swim-lane is provided for each interface. Each interface has one operation, represented as a CallBehaviorAction, which is placed in the appropriate swim-lane. The message parts associated with each operation are represented as InputPins and OutputPins. These pins are placed on the appropriate CallBehaviorAction. No model intervention from the software architect is required at this step.

4.2. Step 2 - Distribution Pattern Definition

The UML model produced in step 1, requires additional modeling. First the architect selects a distribution pattern and then assigns appropriate values to the tagged values of the stereotypes. Based on the chosen distribution pattern, the architect defines the sequence of actions by connecting CallBehaviorActions to one another, using UML ControlFlow connectors, each of which is assigned an order value. The architect then connects up the UML InputPins and OutputPins of the model, using UML ObjectFlows connectors, so data is passed through the composition.

Returning to the case study, we must connect up the three Web services to realise a distribution pattern. Before we do this, however, we select a distribution pattern appropriate to the bank’s situation and requirements. The decentralised dedicated peer distribution pattern is appropriate as the bank requires credit rating information from a third party and does not wish to reveal any of the intermediate participant values of the composition. Also, the bank anticipates a high number of credit card applications, so the load must be distributed to avoid availability issues. Other scenarios would demand the use of other distribution patterns. We apply the pattern by connecting the CoreBanking and RiskManagement CallBehaviorActions together and subsequently connect the RiskManagement and CreditCard CallBehaviorActions constructs together, using ControlFlow connectors, as in Figure 6. We do not use a dedicated peer as the entry point to the composition, although this option is available to us. The InputPins and OutputPins of the CallBehaviorActions are connected together using ObjectFlow connectors, to allow the message parts propagate through the distribution pattern. An extra OutputPin, accountName, must be added to the RiskManagement CallBehaviorAction, to provide data for an InputPin, accountName, to the CreditCard CallBehaviorAction. Finally, appropriate values must then
be assigned to the tagged values of the stereotypes.

4.3. Step 3 - From Model to DPL

The UML model completed in step 2 may now be transformed to a DPL document instance by the distribution pattern generator. This document, which is at the same level of abstraction as the UML model, is an internal representation of the distribution pattern which can be validated. The DPL specification, written in XML Schema, and the document instance, an XML file, have no reliance on UML and so provide for interoperability with other modeling techniques. Figure 7 shows the DPL document instance for the case study. The message names and message parts have been truncated for space reasons.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<dpl:pattern-definition xmlns:dpl="http://localhost/dpl"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xsi:schemaLocation="http://localhost/dpl.dxl"/>
<dpl:collaboration_language>WS-BPEL</dpl:collaboration_language>
<dpl:distribution_pattern>decentralised</dpl:distribution_pattern>
<dpl:base_namespace>BankingPeerToPeer</dpl:base_namespace>
<dpl:service_name>BankingPeerToPeer</dpl:service_name>
<dpl:collaboration_language>WS-BPEL</dpl:collaboration_language>
<dpl:base_namespace_prefix>http://foo.com/wsdl/</dpl:base_namespace_prefix>
<dpl:namespace_prefix>http://foo.com/wsdl/</dpl:namespace_prefix>
</dpl:pattern-definition>
```

Figure 7. DPL document instance

With regard to our case study, many of the values in Figure 7 are the same as the values applied by the software architect in step 2, such as distribution_pattern and service_name. The ControlFlow connectors previously defined between the CallBehaviorActions are used to assign an order value to the dpl:nodes, which themselves are derived from the CallBehaviorActions (getAccountName, getRiskAssessment and getCreditCard) in the UML model. The ObjectFlow connectors between the InputPins and OutputPins are used to define the mappings between dpl:nodes. The first dpl:node does not require any explicit ObjectFlow connectors as the initial values passed into the system are used as its input automatically.

4.4. Step 4 - Model Validation

To verify the model, the DPL document instance is verified against the DPL Schema, by the distribution pattern validator. The verification process ensures the distribution pattern selected by the software architect is compatible with the model settings. For example, in our case study, as the decentralised dedicated-hub distribution pattern has been chosen, there must be at least two dpl:nodes having a peer role and there must not be any dpl:nodes with a hub role. If any errors are detected they must be corrected by the software architect by returning to step 2.

4.5. Step 5 - DPL to Executable System

The verified DPL document instance is now used by the executable system generator to generate all the interaction logic documents and interfaces required to realise the distribution pattern. The generator creates interaction logic documents based on the collaboration-language setting. Additional WSDL interfaces are also generated, if necessary. The system is now executable and ready for deployment.

In our case study example, three WS-BPEL interaction logic documents are created to represent each of the three peers in the distribution pattern. Additionally, three WSDL interfaces are created as wrappers to each interaction logic document, enabling the composition to work in a decentralised environment.

4.6. Evaluation

To assess our approach, we use the criteria set out in [23], along with some of our own success criteria.

- **Pattern expression** - We have identified a number of distribution patterns and have shown how patterns can be expressed sufficiently using UML with our DPL-Profile extension and in XML, using DPL.

- **Verification** - We have verified our model transformations using QVT relations between corresponding meta-models.

- **Readability** - Our modeling approach, which visualises the distribution pattern, should be intelligible to software architects. As the model is at the PIM level, clutter from implementation details is avoided.

- **Executable** - Our UML model and associated profile is sufficiently rich to generate a DPL document instance and subsequently all the interaction logic and interface documents needed to create an executable system.

- **Maintenance overhead** - Our MDA approach allows easy manipulation of the system’s distribution pattern.

R. Barrett and C. Pahl
5. Tool Implementation

TOPMAN (TOPOlogy MANager) is our solution to enabling distribution pattern modeling using UML 2.0 and subsequent Web service composition generation. The only technologies required by the tool are the Java runtime and both an XML and XSLT parser. The tool implementation is illustrated in Figure 8.

![Figure 8. Overview of TOPMAN tool](image)

The UML 2.0 model generator uses XSLT to transform the WSDL interfaces of the Web services participants, to a UML 2.0 activity diagram, which generates, using XML DOM, an XMI 2.0 [15] document. XMI is the XML serialisation format for UML models. The model generated includes a reference to our predefined UML profile for distribution patterns, which is also serialised to XMI 2.0.

A number of tools may be used to describe the distribution pattern. IBM’s commercial tool Rational Software Architect (RSA) is compatible with XMI 2.0 and supports many of the UML 2.0 features. The tool has a GUI which allows the software architect to define the distribution pattern. Upon completion, the model can be exported back to XMI for further processing by TOPMAN. An alternative to IBM’s commercial tool is UML2, an open source tool supporting UML 2.0, which allows the model to be viewed and manipulated in an editor. Unfortunately, there are currently no open source GUI based UML tools which support exporting an XMI 2.0 representation of a UML 2.0 model.

The distribution pattern generator uses XSLT to transform the UML 2.0 model to a DPL instance document. The DPL document instance is then verified by an XML validating parser. Finally the DPL document instance is used to drive the executable system generator, resulting in the creation of an executable composition. Within the executable system generator, XSLT and XML DOM are used to generate the interaction logic and interface documents needed by a workflow engine to realise the distribution pattern. Each transformation is written to implement a previously defined QVT relation between source and target meta-models. Ideally, a choreography based specification, such as WS-CDL, should be used. However, there is no enactment engine currently available for WS-CDL. Instead, we choose to use an open source WS-BPEL engine, activeBPEL. Although WS-BPEL is an orchestration engine, we can use it to apply distribution patterns based on the work in [8].

6. Related Work

Two workflow management systems motivate and provide concrete implementations for two of the distribution patterns explored in this paper. However, neither system provides a standards-based modeling solution to drive the realisation of the chosen distribution pattern. The first system DECS [25], from which the distribution pattern term originates, is a workflow management system, which supports both centralised and decentralised distribution patterns, albeit without any code generation element. DECS defines elementary services as tasks whose execution is managed by a coordinator at the same location. The second system SELF-SERV [24], proposes a declarative language for composing services based on UML 1.x statecharts. SELF-SERV provides an environment for visually creating a UML statechart which can subsequently drive the generation of a proprietary XML routing table document. Pre- and post-conditions for successful service execution are generated based on the statechart inputs and outputs. The authors’ more recent work [16] considers the conformance of services with a given conversational specification using a more complete model-driven approach. A mapping from SELF-SERV to WS-BPEL is also investigated.

From the modeling perspective Gronno et al. [23, 9], consider the modeling and building of compositions from existing Web services using MDA, an approach similar to ours. However, they consider only two modeling aspects, service (interface and operations) and workflow models (control and data flow concerns). The system’s distribution pattern is not modeled, resulting in a fixed centralised distribution pattern for all compositions. Their modeling effort begins with the transformation of WSDL documents to UML, followed by the creation of a workflow engine-independent UML 1.4 activity diagram (PIM), which drives the generation of an executable composition. Additional information required to aid the generation of the executable composition is applied to the model using UML profiles.

Another approach of interest is an extension of WebML, which uses the Business Process Modeling Notation (BPMN), instead of UML, for describing Web service processes [7]. The authors consider the assignment of processes to servers, termed process distribution. However, the approach is at a lower conceptual level than that of distribution patterns as communication modes between services are explicitly modeled.
7. Conclusion and Future Work

An engineering approach to the composition of service-based software systems is required. We have introduced techniques based on architectural modeling and pattern-based development, which have already been applied successfully in both object-oriented and component-based systems. We have also applied patterns, which have been found useful in a networking context, to the Web service domain. Our contribution is a modeling and transformation approach, technique and implementation for expressing the distribution pattern of a Web service composition. Our novel modeling aspect, distribution patterns, expresses how a composed system is to be deployed, providing for improved maintainability and comprehensibility. Any of the distribution patterns discussed may be used to guide the generation of an executable system, based on the enterprises requirements. Three modeling and transformation techniques were introduced, along with a tool (TOPMAN) which assists in the generation of an executable system guided by the chosen pattern.

We intend considering alternatives to our UML modeling language approach, based on \( \pi \) calculus and Architecture Description Languages. In addition, quantitative analysis of the reduction in coding effort due to our modeling approach would provide additional motivation for our work. Finally, documentation of the QoS attributes of the complex patterns is also an important future effort.

8. Acknowledgment

The authors would like to thank the Irish Research Council for Science, Engineering and Technology IRCSET.

References

Towards Propagation of Changes by Model Approximations

Antonio Cicchetti, Davide Di Ruscio, Romina Eramo
Dipartimento di Informatica
Università degli Studi di L’Aquila
I–67100 L’Aquila, Italy
Email: {cicchetti, diruscio, eramo}@di.univaq.it

Abstract

A number of model transformation approaches have been proposed both from academia and industry since automated manipulation of models plays a central role in model driven development. Ideally, a model transformation technique should also be compatible with manual changes that might be performed by designers on the generated models in order to resolve unforeseen requirements or limited expressiveness of the involved metamodels.

This paper proposes an approach to model transformation based on answer set programming. Starting from target models that have been manually modified (and possibly not belong to the co-domain of the transformation being used), the approach is able to deduce a collection of models that approximate the ideal one from which it is possible to generate the previously modified target.

1 Introduction

A number of approaches to model transformation has been proposed both from academia and industry. Different classifications [5, 14] of languages and systems devoted to model transformation have been proposed and all of them distinguish at least among hybrid and declarative solutions. The former offers declarative specifications and imperative implementations, like ATL [10], which wraps imperative bodies inside declarative statements. In the latter category, graph transformation approaches (like AGG [20], PROGRESS [17], GreAT [19] and VIATRA [4]) play a key role as inspired by the theoretical investigation in the area of graph transformations. Therefore, at the moment none of the existing approaches have been universally accepted (in the same way EBNF is commonplace for syntax specification, for instance). As depending on specific needs, designers tend to adopt the proper transformation platform and technology. Nevertheless, increasingly models are manipulated by means of automated transformations in order to produce collections of documents which range from different and more abstract system views to code packages which implement the application on given technological assets. Ideally, the adopted transformation technology should also be compatible with manual changes that might be performed by designers on the generated models in order to resolve unforeseen requirements or limited expressiveness of the involved metamodels.

In this paper, we illustrate an approach to support model transformations with limited bidirectionality even in the case the generated model has been manually modified and thus are not reachable anymore by any transformation. With respect to the changes, the proposed approach is able to deduce from the modified targets a set of models that approximate the exact source; i.e., the set of possible models which are the closest to the ideal one from which to generate the previously modified model. In this way, the designer can choose among the approximated models the result of the reverse application of the given transformation. The approach is implemented by means of the Answer Set Programming (ASP) [8] which is essentially based on the concept of stable models used for the representation of the approximated models. In other words, the approach enables the support to specific forms of evolution where a model can be manipulated both manually or in an automatic manner.

The structure of the paper is as follows. Sec. 2 presents some preliminaries about model transformation and change propagation. Sec. 3 briefly presents the Answer Set Programming on which the model transformation approach described in Sec. 4 is based. The change propagation and the approximations of models are discussed in Sec. 5. Section 6 relates the work presented in this paper with other approaches. Finally, Sec. 7 draws the conclusions and presents some future work.

2 Model Transformation

Increasingly, the use of models has become more widespread in several application domains denoting a
global trend called Model Driven Development (MDD). Models are recognized as a way to shift the focus of software development from the source code to the business logic, making it possible to gain several advantages, such as better comprehension and reuse opportunity. However, models reach their fundamental relevance when starting from them it is possible to automatically obtain the corresponding implementation. In fact, since models become first-class entities, transformations play the critical role of the glue between the several levels of abstraction and enable the automated generation of the implementation.

Model transformation is intrinsically difficult and requires dedicated support by means of adequate languages and systems. In general, model transformations can be characterized by different orthogonal concerns (see [5] for a detailed classification). Model–to–model techniques encompass, for example, imperative specifications of target element creation by means of rules [7]. Typically each rule works on elements representing a partition of the source which has to be transformed to the same target typology, and represents a starting point from which all the remaining necessary operations are performed. The languages have an object–oriented fashion with model traversing capabilities, often written in OCL–like style. Another kind of specification is the declarative description of mathematical relations which must hold between source elements and target ones (e.g. xMOF [3]); the relations have the form of pre–post–condition pairs (e.g., expressed in xMOF by OCL constraints) and are solved by constraint solving systems. Since the order of the relationships is not relevant, the resolution procedure can be non-deterministic and/or concurrent. The interpretation of source models as graphs and the subsequent specification of graph transformations to obtain the target (e.g. VIATRA) is a further transformation description solution. Each rule is made up of a subgraph representing a source pattern and the related subgraph identifying the mapping toward the target; eventually some refinement can be linked with the mapping relations. The order is not relevant in this case. Finally, hybrid approaches can be found, which combine several techniques (e.g. ATL, TRL [16]). Usually they are a declarative and imperative combination, i.e. they embody imperative transformation algorithms in declarative rules.

One of the properties of model transformations is bidirectionality, i.e. the possibility to apply the transformation rules in both ways, from source to target and viceversa. All declarative approaches can satisfy the bidirectionality requirement, but some of them allow only unidirectional use of definitions, while it is possible to obtain bidirectionality with operational techniques too, by describing manipulations in both ways [5].

Change propagation is another fundamental property of model transformation which provides the possibility to reflect changes built on the target models toward the source ones without the need to entirely rebuild the models. Change propagation is not obvious to obtain and usually is realized by incremental update mechanisms and consistency checking mechanisms. It is strictly related with traceability information which stores the links between the elements of the target which have been created from a certain source element. This information is critical, especially if the back propagation is supported, because the relations between source and target are not bijective in general [21]. Thus, even if a declarative approach is considered, each time a change occurs without traces there is no guarantee to obtain a unique reverse result.

According to the above classification, in this paper a declarative approach to model transformation is proposed. The approach is able to define bidirectional transformations and provides support to change propagation particularly when manual changes on the target models occur and do not belong to the co-domain of the transformation being used. In this case the approach is able to deduce a collection of models that approximate the exact source.

In the following, before describing the details about the proposed approach, some preliminaries about ASP are given.

## 3 Answer Set Programming (ASP)

Answer Set Programming (ASP) [8] is a paradigm of logic programming [13] that is, in turn, a well-known declarative method that uses logic and proof procedures to define and resolve problems.

The main idea of logic programming is that the language of first-order logic is well-suited for both representing data and describing desired outputs [12]. A program consists of a set of axioms and rules, whereas a logic programming system computes the consequences of the axioms and rules in order to answer a query.

For compactness and convenience of notation we limit ourselves to an overview of traditional ASP. The syntax of ASP is based on the first-order language [1] according to the following definitions.

**Definition 1** A first-order language is based on an alphabet \( \Sigma \) consisting of: (1) a set of predicate symbols, (2) a set of function and constant symbols and (3) a set of variable symbols;

**Definition 2** A term is recursively defined as follows: (1) a variable is a term, (2) a constant is a term, (3) if \( f \) is an \( n \)-ary function and \( t_1, ..., t_n \) are terms then \( f(t_1, ..., t_n) \) is a term.

**Definition 3** If \( p \) is an \( n \)-ary predicate symbol and \( t_1, ..., t_n \) are terms then \( p(t_1, ..., t_n) \) is an atom.
Contrarily to the classical logic and standard logic programming, in ASP no function symbols are allowed.

**Definition 4** An ASP program is defined as a collection of rules and constraints.

- A rule is an expression of the form: \( \rho \leftarrow A_0, A_1, \ldots, A_n, \text{not } A_{n+1}, \ldots, \text{not } A_m \) where \( A_0, \ldots, A_n \) are atoms and not is a logical connective called negation as failure. For every rule \( \rho \), (a) head(\( \rho \)) = \( A_0 \), (b) pos(\( \rho \)) = \( A_1, \ldots, A_m \), (c) neg(\( \rho \)) = \( A_{m+1}, \ldots, A_{n} \) (d) body(\( \rho \)) = pos(\( \rho \)) \( \cup \) neg(\( \rho \)).

- The head of rules is never empty, while if body(\( \rho \)) = \( \emptyset \) we refer to \( \rho \) as a fact.

- Queries and constraints are expressions with the same structure of rules but with empty head.

The semantics of an ASP program are defined in terms of answer sets. ASP is based on the answer set semantics (or equivalently stable model semantics) [8] that is each solution to a problem is represented by an answer set (also called a stable model) of a deductive function-free logic program encoding the problem itself. Informally, these are sets of atoms that are consistent with the rules of a program and supported by a deductive process. If \( P \) is an ASP program, then ground(\( P \)) is the set of all ground instantiations of rules with variables, obtained by substitution of the variables with the element. The stable models of \( P \) are in the subset of minimal models of ground(\( P \)), that is the minimal sets of atoms that are interpreted as true (\( B_P \)).

Positive programs are unambiguous and they have a unique stable model, while programs with negative literals can have multiple stable models like in the following example:

\[
\begin{align*}
p(x) & : = r(x, y), \text{ not } q(x), \\
q(x) & : = \text{ not } p(x), r(x, y).
\end{align*}
\]

This program has two answer sets: \{\( r(x, y), p(x) \)\} and \{\( r(x, y), q(x) \)\}. In the first one, \( q(x) \) is true whereas \( p(x) \) is false; in the second one, \( p(x) \) is true whereas \( q(x) \) is false.

In ASP we are able to manage cyclic negative dependencies that represent incomplete knowledge or denote the possibility of different alternatives, by representing each consistent choice by means of an answer set. Consequently, a program in ASP may have none, one, or several answer sets. Solving a problem by means of ASP induces the specification of a logic program whose execution, by means of an answer set solver, gives place to the answer sets corresponding to the solutions. In this paper the Smodels solver [22] will be used for executing the provided ASP code.

4 ASP for Model Transformations

This section describes how the ASP can be used for specifying and executing declarative model transformations. The approach is relational, which is based on the specification of mathematical relations among source and target element types. Logic programming is used as a formal foundation where predicates are used to define relations.

The approach supports the reverse transformation of manually changed models. In particular, once a model has been generated, the designer could have the need to manually modify it in order to resolve unforeseen requirements or limited expressiveness of the source metamodels. The approach is able to generate a set of approximations representing the source models from which then it could be possible to obtain, by means of the previously applied transformation with eventual extensions, the manually changed model. The deduction of this set of models can be also performed when trace links for the involved elements are not available (i.e. the case of manual addition of new elements). In this case the approach generates the set of models closest to the ideal source.

The proposed ASP-based approach for performing model transformations is depicted in Fig. 1. The ASP program consists of rules defining the relations between the elements of the source and target metamodels. The execution of the programs is based on predicate symbols and terms obtained by encoding the involved metamodels (\( MM_S \) and \( MM_T \)) and the source model (\( M_S \)). Once the ASP program has been executed, a pretty printing operation on the generated encoding can be executed for obtaining the target model (\( M_T \)).

In the following, the approach is illustrated by introducing in Sec. 4.1 the model and metamodel encoding and then in the next section how to manipulate them by means of ASP programs.
4.1 Model and Metamodel encoding

The simplified UML metamodel depicted in Fig. 2 gives place to the predicate symbols element, package, class, association, attribute and primitivedatatype corresponding to the classes in the metamodel. The predicate symbol parents, to, from and name, isPersistent, isPrimary are also introduced with respect to the association roles and class attributes.

The predicate symbols induced by metamodels are used for encoding corresponding models. For example, in Fig. 3 a model conforming to the metamodel of Fig. 2 is depicted and a fragment of its encoding might look as follows:

```java
package pl.

package(p1).
name(p1.sales).

package(p2).
name(p2.company).

parents(p1.p2).

primitiveDataType(int).

//Customer class encoding
class(c1).
nome(c1.customer).
isPersistent(c1).

attribute(at1).
nome(at1.customer_id).
type(at1.int).
isPrimary(at1).

attribute(at2).
nome(at2.customer_name).
type(at2.string).

attrs(c1.at1).
attrs(c1.at2).

elements(p1.c1).
elements(p1.at1).
elements(p1.at2).
```

The fragment of the ASP specification encodes the packages sales and company (line 1-5), and the predicate symbol parents is used for expressing the inheritance relation between them (line 7). The encoding of the UML classes contained in the packages is performed by means of a number of predicate symbols like class, isPersistent and name as in the lines 11-23. The predicate symbol elements is used for setting the belonging of the elements to the proper package like in the lines 28-30 where the content of the package sales is given.

4.2 ASP programs

After the encoding phase, the deduction of the facts representing the target model is performed according to the rules defined in the ASP program. In order to better describe this phase, a fragment of an UML2RDBMS model transformation is described by considering the metamodels depicted in Fig. 2 and Fig. 4 which are inspired by the case study given in [2]. The transformation should map each persistent class to table and all its attributes or associations to columns in this table. If the type of an attribute or association is another persistent class, a foreign key to the corresponding table is established.

The ASP program implementing the required transformation, should specify a number of relations among the elements belonging to the involved source and the target metamodels. In particular, Listing 1 contains the code implementing the following relations:

- Class2Table_Key: each persistent class in a UML model induces a table in the corresponding RDBMS model (line 2) and a key (line 3). The name of the table will be the same of the source class (line 4) and all the keys of a table are maintained by means of the predicate keys as specified in line 5. In order to support the inverse, the ASP rules in lines 6-8 have to be specified. In this way, if the same relation is applied to a RDBMS model, each table induces a corresponding persistent class having the same name of the table;

- Attribute2Column: each attribute has to be mapped to a column (line 11). The name and the type of the new column will be the same as the name of the related attribute (line 12-13). The inverse of the rule is specified in the remainder of the relation description (line 14-16);

1The apex is used to distinguish the predicates encoding the target model from the source.
- **ClassAttribute2TableColumn**: this merges together the above two relations. All the attributes of a class are mapped to the columns of the corresponding table (line 19). Primary attributes induce the generation of key columns (line 20). By considering the inverse rule, for each column in a table an attribute has to be generated in the related class (line 21). If the column is a table key, the generated attribute has to be set as primary (line 22);

- **Association2ForeignKey**: if an associated class is persistent, a foreign key in the corresponding table has to be generated;

- **AttributeAssociation2ColumnForeignKey**: if the type of an attribute is a persistent class, the attribute gives place to a column which is also a foreign key in the table to which the column belongs.

Note that the specification order of the relations is not relevant as their execution is bottom-up; i.e., the final answer is always deduced starting from the more nested facts.

Due to space limitation, without compromising the readability of the paper, the implementation of the relations **Association2ForeignKey** and **AttributeAssociation2ColumnForeignKey** are not presented here. The interested reader can download the complete implementation from [6].

```
// Relation Class2Table_Key
1 table(A) :- class(A). isPersistent(A).
2 key(A) :- class(A), isPersistent(A).
3 name(A,B) :- table(A), key(A), name(A,B).
4 name(A,B) :- table(A), key(A), class(A).
5 isPersistent(A) :- table(A).
6 name(A,B) :- class(A), name(A,B).
7 class(A) :- table(A).
8 name(A,B) :- class(A), name(A,B).

// Relation Attribute2Column
9 column(A) :- attribute(A).
10 name(A,B) :- attribute(A), name(A,B).
11 type(A,B) :- type(A,B), attribute(A).
12 primitiveDataType(A,B) :- type(A,B), primitiveDataType(A).
13 primitiveDataType(A,B) :- type(A,B), not primitiveDataType(A).

// Relation ClassAttribute2TableColumn
14 cols(A,B) :- attr(A,B), cols(A,B), isPrimary(B).
15 cols(A,B) :- attr(A,B), cols(A,B), isPrimary(A).
16 cols(A,B) :- attr(A,B), cols(A,B).
17 cols(A,B) :- attr(A,B), cols(A,B), isPrimary(B).
18 cols(A,B) :- attr(A,B), cols(A,B), isPrimary(A).
19 cols(A,B) :- attr(A,B), cols(A,B), isPrimary(B).
20 cols(A,B) :- attr(A,B), cols(A,B), isPrimary(A).
21 cols(A,B) :- attr(A,B), cols(A,B), isPrimary(B).
22 cols(A,B) :- attr(A,B), cols(A,B), isPrimary(A).
```

**Listing 1. Fragment of the ASP UML2RDBMS transformation**

If the content of the package Sales is given as source to the **UML2RDBMS** transformation, the model depicted in Fig. 5 can be obtained. In particular, the atoms specified in Listing 2 can be automatically deduced by applying the Smodels [22] solver to the ASP code in Listing 1.

```
// table(c1). name(c1, customer).
1 // column(atl). name(atl, customer_id).
2 // type(atl, int).
3 // column(at2). name(at2, customer_name).
4 // type(at2, string).
5 // cols(c1, atl).
6 // cols(c1, at2).
7 // col_key(c1, at1).
8 // col_key(c1, at2).
9 // foreignKey(as1).
10 // name(as1, customer_order).
11 // refers_to(as1, c12).
12 // f_key(as1, c11).
```

**Listing 2. Fragment of the generated RDBMS model encoding**

The proposed approach is declarative and it is well suited for performing bidirectional transformations. For example, the transformation in Listing 1 can be applied both ways and a UML model conforming to the metamodel in Fig 2 can be obtained by applying the described **UML2RDBMS** transformation to a RDBMS model. However, not always the inverse of a rule generates unique elements. In fact, a same target could be generated from different sources.

**Figure 4. Sample RDBMS Metamodel**

The invertibility of transformation rules may depend on a number of factors, such as the different expressive power of the involved metamodels. This situation can be a problem in performing model transformation based reverse engineering. A possible solution can be based on trace information...
that connect source and target elements between the source and the target domain. In this way, once a model transformation has been performed, all the target elements are linked to the source.

Bidirectional transformations and trace links can play a key role in supporting change propagation; e.g. enterprise contexts where models incrementally change during all the phases of the development process. The next section describes how ASP can be used for supporting change propagation and to approximate source models when manual changes occur on the generated target.

5 Change Propagation and ASP

Change propagation management tends to reflect modifications of the target model to the source. Supporting this feature for model transformation is not obvious and it is strictly related to traceability information which stores the links between the generated elements in the target model with the corresponding elements in the source.

This section describes how answer set programming can be used for supporting change propagation, especially when manual changes produce elements outside the image of the considered transformation. The explanation will use the UMLMetamodel1_TO_UMLMetamodel2 model transformation from the metamodel depicted in Fig. 2 to the one in Fig. 6. In the latter, hierarchies of packages are not allowed and the transformation T, given a model A conforming to the UML Metamodel 1, generates a target where the element of the parent packages are brought into the child. The transformation uses predicates for maintaining trace links that are necessary to obtain the model A from the model B which are depicted in Fig. 8. Without trace links it should be impossible to perform the reverse of UMLMetamodel1_TO_UMLMetamodel2 because of a number of information are missed, such as how many packages have to be generated and in which package a given class has to be filled. Due to space limitation, the transformation UMLMetamodel1_TO_UMLMetamodel2 is not given here but it can be downloaded from [6].

The scenarios that will be considered in this section are depicted in Fig. 7. In particular, once a model B is automatically obtained from a model A by means of a transformation T, the designer may need to manually modify the generated model to accommodate unforeseen requirements or limited expressiveness of the source metamodel. Considering the transformation as a function, the obtained model after the manual changes can be in the image of T (like B' in the dashed part of the lower-side of Fig. 7) or outside (like B'' in the gray part which represents the target metamodel). Being that T is a bidirectional transformation, it can be applied on B' and, in general, a number of models can be obtained. If we consider T^{-1} as the reverse application of T, the models that can be generated by means of T^{-1}(B') are in the domain of T (see the dashed part on the upper-side of the figure). For example, if we apply the UMLMetamodel1_TO_UMLMetamodel2 transformation to the model A depicted in Fig. 8, the model B is generated. Adding the class Transfer the model B'' is obtained. In the proposed approach, the manual change (∆1) can be codified by means of the predicates

```prolog
class(addeClass1).
name(addeClass1,transfer).
```

that can be added to the encoding of the generated model B in order to automatically obtain the corresponding B'. As previously said, UMLMetamodel1_TO_UMLMetamodel2 is bidirectional and so it could be applied to B'. In the example, the application of the reverse transformation on B' could generate three models. In fact, B' can be obtained independently from the belonging of the class Transfer that can be contained in each of the three packages without changing the result of the transformation.

The application of the ASP program encoding the transformation UMLMetamodel1_TO_UMLMetamodel2 deduces the following three answer sets corresponding to the different A''

```prolog
//Answer Set 1,
class('x4') name('x4,manufacturer')
class('x1') name('x1,item')
class('x2') name('x2,order')
class('x1') name('x1,manufacturer')
class('x5') name('x5,transfer')
package(p1) name('p1,company')
package(p2) name('p2,sales')
package(p3) name('p3,stock')
parents(p1,p2)
parents(p1,p3)
```
Listing 3. Generated Answer Sets

In this case, the user can choose which generated model has to be considered as a result of the reverse transformation.

The designer can change the generated models introducing elements that are outside the image of the used transformation (like in the case of $B''$ in Fig. 7). This could happen when a model transformation has to be extended as the generated elements do not completely satisfy new requirements. In this case, the transformation has to be improved in order to generate elements that still conform to the target metamodel but that have not been considered before. Furthermore, the improvement of the transformation could induce also a refinement of the source metamodel. In Fig. 7 the improvement of the transformation $T$ is encoded in $T_{\Delta_2}$ and the application of $T^{-1} \circ T_{\Delta_2}$ on $B''$ will generate a model that conforms to the extended metamodel. The application of $T^{-1}$ will generate models that are in the domain of $T$ only as $T^{-1}$ is not aware of the elements outside its domain.

Figure 7. Change Propagation Scenarios

For example, the model $B''$ depicted in Fig. 8 is obtained by adding a dependency relationship between two classes with respect to the metamodel of Fig. 6. Nevertheless, this concept is not considered in the $UMLMeta-model1_TO_UMLMeta-model2$ transformation and the reverse application of the transformation on $B''$ will generate $A$. In order to maintain this dependency also in the source model, the source metamodel has to be extended with this new construct and the $UMLMeta-model1_TO_UMLMeta-model2_{\Delta}$ has to be defined. In the example, this extension consists of the following rules

that relates the dependencies contained in the source model with the target. Furthermore, the rules are based on the predicate symbol dependency that has been added in the source metamodel encoding.

6 Related Work

Model transformation is intrinsically difficult and requires dedicated support by means of adequate languages and systems. Hence, since 2002, when the OMG issued a Request For Proposal (RFP) on Query/Views/Transformations (QVT) [15], a lot of initiatives have been presented, each of which trying to meet the requirements raising from transformations usage like declarativeness, bidirectionality and change propagation.

To the best of our knowledge a number of declarative approaches are available but one of the closest to what has been presented here is xMOF [3]. It is based on a constraint solving system which enables the specification of
model transformations by means of OCL constraints on involved model elements, aiming to provide bidirectionality and change propagation. Mercury [18] and F-Logic [11] are logic languages that have been used for exploring the application of logic programming to model transformation [9]. The main difference with what has been presented here is that the above approaches are not based on the answer set programming that in our opinion seems well suitable for the approximation of models for change propagation purposes.

An attempt to support change propagation is presented in [21], where an imperative paradigm capable of backtracking and a language used to handle declarative aspects build an hybrid method trying to deal with tool integration mechanisms. The main difference with the approach presented here could be observed when manual changes of the target models consist of the addition of new elements for which trace links are not available. The proposed approach tries always to approximate the source models with respect to the available transformations and the involved metamodels.

7 Conclusions and Future Work

This paper presented an approach to model transformation based on the logical setting of answer set programming. The work pursued a twofold objective: on one hand we have been investigating how to take advantage of a declarative paradigm in order to achieve certain forms of bidirectionality; on the other hand we intended to have some machinery to assist the evolution of those models which have been subject to both manual intervention and automated manipulation.

Models and transformations are encoded as facts in an ASP program which consist of some generic rules representing the transformation engine, i.e. an abstract machine able to interpret the transformation rules, manipulate the source model facts and automatically generate the facts encoding the target model. Another set of generic rules is also given to perform a reverse transformation starting from the encoding of a generated model.

One of the most interesting scenarios arises when the designer is forced to manually modify the generated model due to the limited expressiveness of the source metamodel or of the specified transformation. In other words, the desired behavior is outside the image of the transformation, i.e. there does not exist a source model from which to generate the manually modified one. The ASP can offer the possibility to generate a set of possible models which are the closest to the ideal model from which to generate the modified model, enabling the approach to support specific forms of evolution where model can be manipulated both manually or in an automatic manner.

We believe that the approach proposed here is also suitable for dealing with constrained metamodels even though
the handling of metamodel constraints during the propagation of changes and model approximation is still under study. The main limitation of the approach is due to the limited expressiveness of the ASP implementation, in fact the system does not have any kind of complex types (for instance, associative arrays) and which limited the approach only to those manipulations which can be considered structural. However, in principle the approach does not present intrinsic difficulties and just require a more adequate implementation of the ASP paradigm.

Acknowledgements

We thank Andrea Formisano and Alfonso Pierantonio for the enlightening discussions and for the insightful comments.

References


Towards Propagation of Changes by Model Approximations
Abstract

From first to mature versions of Model-Driven Development (MDD) tools, there is a gap, as for any other software applications. All functional requirements must be met, including qualities of services, at the risk of seeing MDD tools rejected by users. In this paper, we focus on performance, especially for large-scale developments. After an overview of methodological elements, we give a list of reusable practices on performance. We conclude by a set of observations and stakes in order to understand where efforts must be applied during the development process.

1. Introduction

Generally speaking, requirements express functional expectations. However, software qualities of service are underestimated or completely forgotten. In the context of MDD tools for which the paradigm shift already jeopardizes user acceptance, bad performance results become a motive of reject by users. This is the reason for which performance must be tackled at its right value. In this paper, we present a set of reusable lessons learned on the subject.

The Architecture and Engineering Department (AED) of the Thales Software Research Group is working on putting the MDA® vision at work. In this perspective, activities include the development and deployment of a MDE tool suite, an integrated set of Model Driven Engineering (MDE) workbenches. The main purpose is to improve productivity and quality of system development by realizing a paradigm shift from handcrafted to industrialized development processes. As depicted in Figure 1, this MDE tool suite includes three workbenches: i) MDSysE [6][12], a tool for modeling systems, such as an aircraft, ii) MDSoftE, a tool for modeling software subsystems, such as a software device of an aircraft, iii) MDSoFa [10], a tool for generating MDE tools, such as MDSysE and MDSoftE. From the highest down to the lowest levels of a system, a MDSysE to MDSysE bridge ensures the consistency of system decomposition and a MDSysE to MDSoftE bridge ensures the consistency between system and software levels. Regarding the reusability aspect, MDSoFa capitalizes reusable assets (core models, patterns, platform description…) stored in a database called MDCorE. MDSysE is used today by three Thales business units and by two major programmes, in air traffic management and naval domains. MDSoFa is also used by a Thales business unit contributing actively to the development of MDSoftE. However, while these tools are gaining more strength and are providing most of the expected functionality, the acceptance of MDSoFa and MDSysE by users has recently been strongly jeopardized by performance problems. Therefore, we needed to focus on theses issues, just as we would have for any non-MDD tool when scalability and complexity of the expected functions are growing.
second class of questions is technical. The main one, treated in the second part of this paper, is: Are there good practices improving performance that can be reused by projects building MDD tools?

Section 2 provides further details about MDD tools and elaborates on the deployment of these tools in operational conditions. Section 3, about methodology, stresses both on performance requirements and measure. Section 4 gives reusable practices for performance improvement. Those are neither revolutionary nor exclusively MDD-specific, but they constitute a kind of check list of what is often and unfortunately forgotten. This section crosses these practices with MDD techniques, such as query or model-to-text techniques, and formulates a set of observations and stakes. Section 5 presents further work and section 6 concludes.

2. MDD tools in an industrial context

At first, in order to understand our directions for performance improvement, it is necessary to precise the context our MDD tools.

From the generation viewpoint, MDSoFa produces about 80% of code of MDSysE and MDSoftE. From a bootstrap, MDSoFa is produced by itself.

About the scalability of MDSoFa: MDSysE represents 20 metamodels, 200 domain classes, 400 associations, 50 constraints, 6 aspects, and 20,000 generated methods, while MDSoftE is about 10% bigger than MDSysE. All classes and associations of the metamodels are mapped onto UML, for a domain to UML language mapping. Typically during the generation of the infrastructure of MDSysE with MDSoFa, about 500 patterns are applied: i) 150 core patterns from MDCore, shared in 8 pattern families (model introspection, model management, model structuring, model checking, model transformation, modeling wizard, serialization), and ii) 300 generated domain-patterns.

About the scalability of the MDD tools: MDSysE manages models of 15,000 model elements (excluding notes and UML profile elements). The volume is not yet known for MDSoftE. Kinds of functions of MDSysE and MDSoftE are mainly modeling wizards, traceability facilities, model checking and refinement transformations.

About maturity, MDSysE is already being used in operational environments and is in the process of being industrialized. MDSoftE is being developed with the lessons learned from the MDSysE experience. At this stage, MDSoFa remains a prototype.

Where it is deployed, MDSysE is often the first experience of system engineers with model driven development. At most, some of them were previously using UML for documentation purposes. Despite coaching sessions and intensive online support help dealing with the paradigm shift, the learning curve remains important and it takes even more time before end users feel comfortable with both the tool and the methodology.

In the context of its transition from a prototype to an operational tool, the maintenance of MDSysE has mainly been driven lately by the requirements and milestones the business units had to meet in their respective programmes. The focus has thus often been put on development of new features and bugfixes, to the detriment of non functional aspects. Typically, MDSysE suffered from important performance problems because of the introduction of too many new features in a too short period of time. As trust is hard to gain but easy to lose, system engineers started to reject both the tool and the methodology for wrong reasons, using performance as a pretext. Since then, the performance issues have been tackled and the feedback of system designers about the MDD approach has become much better.

MDSysE, MDSoftE, and MDSoFa deal with three communities of users: system engineers, software engineers, and developers of MDD tools. But whatever the type of tool, meta- or MDD tool, whatever the way of building MDD tools (e.g. with / without the product-line technique or the architectural patterns used), whatever the functions developed (e.g. wizard or model transformations), whatever the level of maturity, prototype or industrial, and the platform (e.g. tool-vendor vs. open source tool), code compiled or not: level of performance must be acceptable by users.

3. Methodology

Having a minimal set of methodological elements is necessary before tackling the technical aspect of performance. Two points are central: requirements and measure. Requirements define the objectives to be satisfied. Measure, used through scenarios, shows objectively and precisely i) how far the level of software performance is from the requirements and ii) the improvement ratios observed during the intermediary steps.

3.1. Performance requirements

Performance is an indicator of how well a software system or component meets its requirements for
timeliness, in terms of response time or throughput [15]. In relationship, scalability is the ability to continue to meet response time or throughput objectives as the demand for software functions increases. More precisely, a performance requirement quantifies with accuracy the level of performance expected from functions and devices implied. For instance, the requirement “Response time < 1 second for the function X” can be decomposed as follows:

- “Response time” = nature of the expected quality,
- “<” = function of comparison,
- “1” = value,
- “second” = dimension of comparison,
- “for the function X” = the identified function.

A performance requirement allocated at first for instance to use cases is refined all along the development process from analysis down to design and code. Traceability links check every requirement is allocated and verified at different model levels.

Performance might be straightly solved, for instance through an algorithm improvement. Performance might also be indirectly solved. For instance, performance can be solved by mitigation means, such as by latency, by concurrent computations, or, in another way, with more powerful devices.

Finally, performance requirements must take into account evolutions. Scenarios can prevent worsening performance over the coming years.

3.2. Performance measure

An essential property of qualities of service, including performance, is that they can be valued. The interest is software assessment. Benchmarks are the means of measure to accept or reject with objectivity improvement decisions and to evaluate the distance to meet requirements. Then, some questions emerge: what, where, and how to measure?, what are the best conditions for assessment?, how to compare when performance strategies or platform change?, who assess?, when assessing during the development process? Answers depend on process, development team, and MDD tools. But, whatever the environment and its evolutions, scenarios are always the means to measure, in the same way, performance improvement, and then to have fixed comparison points.

The backbone of the scenario steps can be defined as follows:

1. Defining scenarios. All performance requirements are covered by scenarios.

2. Preparing scenario execution, e.g. model initialization.
3. Executing scenarios and measuring performance.
4. Analyzing performance results and comparing them with previous ones.
5. Deciding how to improve performance and measure quality when needed (granularity, instrumentation, monitoring).
6. Iterating in step 1 or 2 until performance requirements are met.

Different strategies can be applied, for instance top-down / bottom-up analyzing, by function / component / data, by view / wizard / model transformations. But quality of improvement depends on i) the pertinence of the scenarios and the way to continue investigations in order to reduce the problem space of performance, ii) the measure accuracy in order to have reliable measures and to have objective conclusions.

In the meantime, at the development process level, one general constraint must be respected during the successive iterations of improvement: keeping up the same quality level of functional capabilities and the same assessment level of the other non-functional aspects. For instance, no software failure, additional maintenance cost, or less usability must be introduced. All is in balance where improvement of one aspect must not degrade the others.

4. Reusable practices

Requirements and measure are the elements used for assessing objectively performance improvement and rationalizing the way to meet software objectives. Now, the interest of this section it to give an open set of best practices which can populate technical materials of development processes. These practices are deduced from our experience with MDSoFa, MDSysE, and MDSoftE. After their presentation, they are crossed with the most used modeling techniques. At last, we give a set of observations and stakes.

4.1. Best practice catalog

The starting point is classification of best practices for performance. We exclude from this paper idioms, i.e. platform-dependent best practices, simply because they are not reusable enough. We have chosen to divide up those presented here into seven categories: scheduling, resource consumption, learning, distribution, transaction, exchange, and architecture. This classification is not strict.

AVOIDING REDUNDANT TREATMENTS
Problem: A treatment is applied many times while a data analysis prevents this redundancy of treatment. The grain of treatment can be either small, e.g. a query, a function, or large, e.g. model checking, model refinement.
Resolution: Getting the minimal data set according to the condition of redundancy and applying the considered treatment.
Example: 1/ A list contains several times the same model element; creating a new list where every model element, for the given condition, is unique. 2/ When reasoning at the metamodel level, flattening inheritance levels of metamodels can be a strategy avoiding a treatment applied both at the leaf metaclass level and uselessly at the super-metaclass levels.
Comment: 1/ Reduction is simple when it is based on model element values. Conditions become more complex when multiple dependencies exist between model elements or when model element analysis is applied at the metamodel level, 2/ The data analysis can be realized during the execution. In this case, a precondition of redundancy must be applied on each element.
Related Solutions: Optimizing queries, Storing in cache.

POSTPONING COMPUTATION RESULTS UNTILL NEEDED
Problem: Time-consuming computations are realized first while their results can be available later. This concerns particularly model queries.
Resolution: Applying computations only when their expected results are needed.
Example: For wizard performance, do not apply all queries when opening the wizard but delegate them on actions which really need them, for instance on button activation.
Comment: 1/ The evaluation context must be available what compel sometimes to manage an evaluation context. 2/ The same result must not be computed many times. Then, using a cache for storing computation results.
Related Solutions: Storing in cache.

OPTIMIZING QUERY
Problem: The response time of a query is deplorable.
Resolution: Changing the order of the query conditions in order to reduce as soon as possible the size of the object list. On the other hand, the same computation is realized just once.
Example: A complex query on large models is split up into sub-queries. Most restrictive selection conditions are placed forward and variables store reusable computations.
Comment: This issue is well known in the database area. An optimization analysis requires to know very well how data, here model elements, are organized and their volume in order to find out the best query organization.
Related Solutions: Optimizing navigation, Storing in cache, Applying pipeline and filter architecture.

REDUCING NUMBER OF NAVIGATION IN MODEL
Problem: A navigation through the model is applied several times or is very long.
Resolution: Delegating the navigation task to a path finder which optimizes and avoids redundant navigations.
Example: Between two model elements in large models, several paths can exist and the exploration path can be long. A path finder can optimize and store navigations already done.
Comment: 1/ A path finder avoids spaghetti navigations and returns exhaustive results, while programmer can miss navigation paths. 2/ For improving performance during execution, a generation phase can find out the shortest path between two model elements, and detect cycle.
Related Solutions: Computing during generation.

4.1.2. Resource consumption. Way of improvement: With the same resources, consuming less, i.e. realizing economies.

REDUCING ALGORITHM COMPLEXITY
Problem: An algorithm decreases dramatically performance when the model size increases.
Resolution: Reducing algorithm complexity, and as far as possible, into n.log (n)
Example: See algorithmic examples for reducing complexity.
Comment: -
Related Solutions: -

GETTING INFORMATION DIRECTLY
Problem: Getting information needs search and computation, and this many times with any added value.
Resolution: Having key/value associations where values contain searched information or computation results.
Example: For getting super(sub)-classes of a class, the key is the class object, and the value its super(sub)-classes.
Comment: 1/ A path finder avoids spaghetti navigations and returns exhaustive results, while programmer can miss navigation paths. 2/ For improving performance during execution, a generation phase can find out the shortest path between two model elements, and detect cycle.
Related Solutions: Storing in cache, Computing during generation.

REDUCING USE OF TIME-CONSUMING FUNCTIONS
Problem: Performance is getting worse due to time-consuming functions.
Resolution: Detecting time-consuming functions and finding mitigation functions.
Example: Sorting is time-consuming; remove it when useless else reduce the number of sort criteria. For information display, a verbose mode can be proposed, with print out on console or in file, which is faster.
Comment: Detection of these functions are essentially realized by profiling.
Related Solutions: -
**OPTIMIZING MOST USED FUNCTIONS**

_Problem_: Performance is getting worse due to bad performance of core functions.

_Resolution_: The most used functions must have the lowest level of time consumption. All possible optimization must be applied on.

_Example_: In a IF condition, put forward the most discriminating conditions. Using caches.

_Comment_: This optimization concerns functions at the lowest level of the application, in bottleneck situation, or an execution framework which is often solicited.

_Related Solutions_: -

**REMOVING EXTRA FUNCTIONS**

_Problem_: Some functions are never or rarely useful but often applied.

_Resolution_: Any useless function must be removed or isolated in order to be called intentionally.

_Example_: Exotic behaviors or historic code are inlaid and pollute conditions or navigations.

_Comment_: -

_Related Solutions_: -

4.1.3. Learning. Way of improvement: Reusing results of previous treatments.

**STORING IN CACHE**

_Problem_: Computations or searches need data often used.

_Resolution_: Storing data in cache, that is in a temporary memory.

_Example_: Sets of primitive types, bottleneck model elements, results of time-consuming navigation in model or computations can be stored in cache. Their life time can be limited for instance to a model transformation.

_Comment_: 1/ A cache can be split into homogeneous sub-caches. 2/ The mechanism for storing data in cache is completely encapsulated. The objective is to retrieve efficiently and safely information often used. 3/ Contrary to a blackboard, the lifecycle of an element in cache is determined by the cache manager and not according to a strategy for building a solution. For instance, the same element can be several times, and transparently, in and out of the cache.

_Related Solutions_: Getting information directly.

4.1.4. Distribution. Way of improvement: Distributing tasks between resources or in time.

**DISTRIBUTING COMPUTATION**

_Problem_: Sequences of computation have long execution time.

_Resolution_: Isolating independent computation parts and distributing them.

_Example_: 1/ Using threads for queries on model. 2/ For model to text generations, independent generation parts can be stored in files, and, when needed, joint in a second time.

_Comment_: 1/ Model transformation in parallel can create inconsistency; for this reason, modification impacts must be evaluated and tested. However, model view or generation working independently are good candidates. 2/ Avoiding overhead situations.

_Related Solutions_: -

**COMPUTING WHILE GENERATING**

_Problem_: Computations with exactly the same result are regularly executed.

_Resolution_: Generating static computations instead of finding their results during execution.

_Example_: Generation of static information on metamodel for model reflection.

_Comment_: -

_Related Solutions_: -

4.1.5. Transaction. Way of improvement: Benefiting from transactional possibilities.

**USING NON-PERSISTENT TRANSACTION**

_Problem_: At the end of a transaction on model, actions on model elements imply computations, for instance model checking, while the initial state of the model, i.e. before starting the transaction, can be kept.

_Resolution_: Using non-persistent transaction with a transaction rollback.

_Example_: For model checking or documentation generation, when model elements created for computations become afterward obsolete, instead of deleting them before the end of the transaction or saving them uselessly after the transaction, using the rollback mechanism is the best efficient solution regarding the execution time and for model consistency.

_Comment_: For model transformations, which need persistence, lifecycle of intermediary model elements must be managed by the model transformations themselves.

_Related Solutions_: -


**ACCESSING TO DATA BY BLOCK**

_Problem_: An application needs to access to groups of data, at different moments and in a scattering way, multiplying queries.

_Resolution_: Instead of multiplying queries, reading information in homogeneous way by block and selecting or completing it contextually in order to get, with opportunity, the needed information.

_Example_: Successive wizards need to display common model elements but in a different way. Model elements read are stored in a common workspace where wizards apply on queries in order to get the expected data.

_Comment_: -

_Related Solutions_: Storing in cache.

**MINIMIZING MASS DATA TRANSFER**

_Problem_: An application faces to read and compute high volume of data; in the meantime, performance is decreasing.
Resolution: Reading information by segment.
Example: An XML file contains 10,000 data records; with a mass data transfer, managing large sets worsens computation time, and generally ends with a crash. Reading by segment, for instance segments of 100 items, allows keeping a constant computation time for the same functionality.
Comment: -
Related Solutions: -


APPLYING PIPELINE AND FILTER ARCHITECTURE
Problem: Functions are successively applied but separately.
Resolution: Creating a continuous data flow where output become input of the next function.
Example: Creating successions of model queries, model query / model transformation, and model query / model view.
Comment: 1/ Filters must be applied as soon as possible in order to reduce the data size. Pipes avoid creating local contexts with intermediary results. 2/ Parallel processing is not necessary more efficient due to cost for transferring data between filters.
Related Solutions: -

APPLYING BLACKBOARD ARCHITECTURE
Problem: A solution built in a deterministic way makes complex service calls to reach the expected solution.
Resolution: The expected result is built by cooperation of contributors sharing their results in a common workspace.
Example: For large models and complex transformations, different model transformation contributors can invoke the same transformation but from different interests. The solution is to build the model transformation incrementally by cooperation. A planner monitors and optimizes transformations. The result of this cooperation can be the target model or a temporary workspace. A temporary workspace is semantically richer than a cache because elements are stored for building a final solution.
Comment: 1/ Cooperation avoids redundant work. 2/ In accordance with the Blackboard pattern, should this solution introduce either too many control tasks, for invoking the best interesting contributors at a given time, or wrong intermediary results, which are rejected later, then this practice is not advised.
Related Solutions: Storing in cache.

DELEGATING TO MOST EFFICIENT RESOURCES
Problem: Code of common functions or techniques have bad execution time.
Resolution: Using native functions or delegating functions to optimized libraries, frameworks or tools.
Example: Languages, such as Python or PERL, for regular expression are more optimized than home-made solutions; for interpreted languages, native functions are less time-consuming.
Comment: For external tools, the complete communication and treatment time must be lesser than the solution developed on your own.
Related Solutions: -

INTEGRATING PARTS
Problem: With scalability, communication time between components increases due to multiple dependencies.
Resolution: Isolating and integrating components by cohesion of functional and communication dependencies.
Example: For a M / V / C pattern, join the V and C components.
Comment: Two main weaknesses to be prevented: loose of semantics and creation of monolithic applications.
Related Solutions: -

ENSURING DATA CONSISTENCY FOR EXCHANGE
Problem: Components apply several times consistency checks on the same pool of data.
Resolution: Checking data consistency at the lowest levels.
Example: Model transformations and wizard share the same data. A component ensures at one time data consistency, avoiding multiple model checking.
Comment: Intermediary blocks of data, for instance for a semantic bridge, can be prepared for a set of components. In this case, a data consistency management is required due to data evolutions.
Related Solutions: -

These practices are not isolated from each other. On the contrary, they are generally used with synergy. They can also be combined with modeling patterns. This is the case in the “Avoiding redundant treatment” using a “flattening class hierarchies” pattern. The key, as for design patterns, is to identify the problem we face and select the appropriate practice.

4.2. Crossing with modeling techniques

The previous sub-section has introduced a set of practices for performance improvement. The interest is now to match them with major modeling techniques of MDD tools. Those identified here are: 1) Repository, 2) Query / View / Model Transformation, 3) Model-to-Text Transformation [14], 4) Data integration [16], 5) AOP (Aspect-Oriented Programming) [9], 6) DSL (Domain-Specific Language) [5], 7) Product-Line [4]. The interest of crossing performance practices and modeling techniques is to determine where efforts for performance improvement have to be focused.

General observations about techniques.
- Regarding the techniques used by the different kinds of tools: i) MDD tools for final users, such as MDSysE or MDSofE, use more of the Repository, Query / Model Transformation, Data integration, and DSL techniques; ii) meta-MDD tools, such as MDSofa, use all techniques but more intensively the model-to-text transformation technique.
All practices for performance improvement are candidate for all techniques, and one technique can benefit to another one. For instance, the model-to-text transformation technique uses queries and navigations; then, query and navigation performance improvement benefits to model-to-text transformations. AOP can use model reflection and then queries as well.

Repository is a bottleneck. At this level, performance must be irreproachable. On the contrary, DSLs for presentation are at the highest level and then are dependent of all improvements.

From our measures, and mainly about resource consumption, if optimization of core functions is needed, we observe that algorithm refactoring is essential. Gains are substantial, typically for model checking or long model transformations.

Use of cache and computed data, typically by MDSoFa for MDSysE and MDSofE, help having more fluid applications, a condition for mature tools.

Observations by category of tools.

- Meta-tools manage high volume of data during generation. Then, all the lowest levels must be optimized: queries, model transformations, and model-to-text transformations.
- Capabilities of tools can be intentionally reduced. For instance, our AOP tool has a limited weaver for ensuring fast answers for core API services.
- During the generation phase, MDSoFa plays a role of compiler about model reflection: some metamodel information is translated into symbolic data for quicker answers.

Observations about reusing performance improvements.

- Integrated tools, such as software factories tools, are dependent on all sub-techniques, such as query improvement.
- With software factories, and typically with shared patterns, performance improvements benefit to all produced MDD tools. For instance, improvement of patterns in MDCorE benefit simultaneously to MDSoFa, MDSysE, and MDSofE.

Observations about sticking to performance requirements.

- Non-regression tests are costly but they are necessary to ensure that functions evolutions do not decrease performance.
- Performance improvements are interrelated. This implies that when one improvement is realized for one modeling technique, other improvements must be assessed by non-regression testing.

4.3. Stakes

Meeting the performance criteria is necessary for MDD tools to become and remain mature. Refactoring of existing tools is costly and delicate regarding the risk of functional regressions, especially when MDD tools are already widely deployed. Therefore, for prevention, it is mandatory to express and take into account performance requirements as soon as possible in the development process. This means clearly expressing requirements in order to avoid any misconceptions and underestimations. During the development process, these requirements must be allocated on different model views (specification, architecture, design, implementation) with a constant follow-up.

Regarding capitalization, during the development process, any knowledge on the issue becomes useful and must be saved either in textual form of best practice or with productive assets, such as with patterns. This is the condition to improve the general maturity of tools, and teams as well.

Regarding quality, performance improvement has a tendency to make code less readable. Explanations in patterns or in code are necessary for maintainability. This clearness is also needed for the follow-up of architecture decisions.

Regarding the development process, everybody is concerned: analysts for problem expression and requirement allocations, designers for detailed decisions, developers for implementation and benchmarking, integrators with benchmarks for validating or not requirements, and architects all along the process. For development with patterns or COTS, if solutions and evolutions are encapsulated, integration and testing objectives remain the same. When several model techniques and functional spaces imply different specialists, they have to work with cohesion, with compatible strategies; architects and integrators become central to meet requirements with long-term solutions.

5. Further Work

Considering MDSoFa, we need to carry on the efforts on performance to face scalability of metamodels which can become as large as any model application composed of several thousands of classes. For tools such as MDSysE, the need is now to introduce performance into testing plan for non-regression
6. Conclusion

Putting MDD in practice in concrete industrial contexts represents a significant paradigm shift. An important deployment program of MDD technologies is ongoing among Thales Business Units. Whereas repetitive training sessions have been ensured to disseminate the methodology, the success of the deployment has been jeopardized by non functional issues such as maturity and performance problems. Since these latter have been solved, the user acceptance rate has significantly increased. Therefore, we have focused this paper on performance improvement.

From a methodological point of view, we have stressed on the need of performance requirements for validating that developed tools effectively meet requirements. We have also stressed on measure, the objective means of assessment, used during scenarios for appreciating progress and distance to objectives. Furthermore, we have listed a set of practices from improving performance. These have been crossed with most used MDD techniques for identifying where efforts must focus on. At last, we have identified stakes on the risk and cost to tackle performance lately, about capitalization and quality, and impact on the development process.

As a conclusion, meeting non functional requirements is mandatory when trying to deploy MDD tools, but certainly not sufficient. Engineers facing the paradigm shift will remain reluctant as long as they do not perceive the added value of MDD, which is not always immediate. It is our role to build ever simpler and more efficient tools, for example through an improvement of our software factories.

Acknowledgements

We thank Serge Salicki, head of the Architecture and Engineering Department of the Thales Software Research Group, the members of the Architecture and Engineering Department, and especially Jean Barata, and Catherine Lucas.

References


B. Langlois, D. Exertier, and S. Bonnet
Structural Patterns for the Transformation of Business Process Models

Marion Murzek  
Women's Postgraduate  
College for Internet  
Technologies (WIT),  
Institute for Software  
Technology and Interactive  
Systems, Vienna University  
of Technology, Austria  
murzek@wit.tuwien.ac.at

Gerhard Kramler  
Business Informatics  
Group (BIG),  
Institute for Software  
Technology and Interactive  
Systems, Vienna University  
of Technology, Austria  
kramler@big.tuwien.ac.at

Elke Michlmayr  
Women's Postgraduate  
College for Internet  
Technologies (WIT),  
Institute for Software  
Technology and Interactive  
Systems, Vienna University  
of Technology, Austria  
michlmayr@wit.tuwien.ac.at

Abstract

Due to company mergers and business to business interoperability, there is a need for model transformations in the area of business process modeling to facilitate scenarios like model integration and model synchronization. General model transformation approaches do not consider the special properties of business process models and horizontal transformation scenarios. Therefore, we propose a model transformation approach based on domain-specific patterns which are applied for analyzing business process models in a precise way. This approach facilitates the definition of business process model transformations, which can be easily adapted to different business process modeling languages and specific transformation problems. At the same time it supports the intuitive understanding of the domain-experts in business process modeling.

1. Introduction

As companies discovered the benefits of Business Process Modeling (BPM), the use of Business Process (BP) models moved from a "luxury article" to an "everyday necessity" in the last years. Meanwhile many companies own thousands of models which describe their business. Since business changes over the years, e.g., business to business interoperability came up with new inventions in communication and companies merge with others, there arises a need to keep existing business models up-to-date and to synchronize or translate them into a contemporary BPM language. To facilitate these scenarios, a model transformation technique for BP models is needed.

Figure 1. Different Model Transformation Scenarios

According to these scenarios, model transformation in the area of BPM can be classified into three different kinds as illustrated in Fig. 1: model integration, model translation and model synchronisation. In the case of (a) Model Integration, two or more models which conform to different BPM languages are merged into one new model conforming to another BPM language. In case of (b) Model Translation, all models conforming to one BPM language are translated into models conforming to another BPM language once. In case of (c) Model Synchronisation, the aim is to keep pairs of models after user changes up-to-date. A suitable model transformation approach for BP models should support all of these three scenarios.

∗This research has partly been funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.
Inspired by the vision of the model driven architecture (MDA) [1], model transformations are currently a hot research topic. The main research interest lies on vertical transformations, i.e., transformations between models of different levels of abstraction such as platform independent models (PIMs), platform specific models (PSMs) and code. Another important aspect are horizontal transformations, where models are translated into other models on the same level of abstraction, e.g. on the PIM level. For models in the area of BPM, this aspect is vital for integrating, synchronizing or transforming models of different BPM languages.

Current techniques or specifications used for defining model transformations, such as ATL [2] or QVT [3], operate at the level of metamodel elements. Therefore, transformation definitions can become quite difficult when direct 1:1 mappings between metamodel elements are not sufficient for specifying a transformation.

For instance, consider transforming the source business process model depicted in Fig. 2 into the target business process model in Fig. 3. In the BPM language of the source model, there is no explicit model element to denote the join of alternative threads, whereas in the target modeling language there is. Moreover, the semantics of multiple arcs depends on the context, i.e., on the origin of these arcs. In the given example, the arcs belong to two different threads, originating in a Split element. However, if they had been originating from a Decision element, their semantics and hence the desired target model would be different.

To address these problems, we are currently developing a framework for domain-specific model transformations in the area of BPM [6]. It comprehends a method to define a model transformation definition in well structured steps based on business aspects [7, 8]. Furthermore, the framework will offer BPM model patterns to facilitate the definition of BPM model transformation scenarios, on the one hand to solve non-trivial transformation requirements, and on the other hand to support the intuitive understanding of the BPM domain-experts. In this work we concentrate on the definition and detection of the BPM model patterns. We focus on the structural patterns that can be found within the control flow of a BPM model. Moreover, we show how the defined patterns assist the transformation definition for BPM models. The contribution of this paper is twofold. First, we thoroughly investigate the control flow aspect of BPM models as described by the workflow patterns [9]. As we will show, workflow patterns are not sufficient for certain model transformations. Therefore, we define higher-level patterns based on the existing workflow patterns in order to detect structural queries are error-prone and difficult to maintain and re-use.
transformation patterns in business process models. Second, we present a case study that shows how the patterns are used in the context of BP model transformations. The case study illustrates the intuitive use of the patterns and their advantages compared to general model transformation approaches.

The remainder of the paper is structured as follows. Section 2 gives an overview of different transformation definitions, business aspects and workflow patterns. In Section 3 the transformation patterns for the control aspect are introduced. Section 4 provides a case study which illustrates the use of the patterns for the transformation of BP models. In Section 5 related work is discussed. Finally Section 6 provides the conclusion and gives an outlook on further research directions.

2. Pattern-oriented Transformation of Business Process Models

In this section we discuss the problems of metamodel-oriented transformation definitions and semantic abstraction to solve these problems. Furthermore, the applicability of the workflow patterns which are the basis of our work is discussed.

2.1. Problems of Metamodel-oriented Transformation Definition in defining model transformations for BP models

Established model transformation techniques, such as QVT [3] or ATL [2] are metamodel-oriented. That means, that the definition of model transformation scenarios is based on the correspondences of the elements in two or sometimes more different metamodels. This concept covers the local correspondences of each element which is used for copying, splitting and merging elements.

2.2. Patterns to Simplify Transformation Definition

As opposed to ad-hoc and metamodel-oriented transformation definitions, we propose to use patterns as basis for transformation definitions. Patterns abstract from individual metamodel elements to represent semantic concepts. The intention of using these higher-level abstractions is to simplify transformation definition and increase reusability.

Patterns allow for a divide and conquer approach to model transformation. The transformation definition can be divided into pattern matching, i.e., analyzing the source model and identifying pattern occurrences, and pattern instantiation, i.e., synthesizing the target model such that the same patterns as in the source model appear. Defining a model transformation in two steps promises simplification of each step and re-use of steps when transformations between multiple BPM languages need to be defined.

We presume that this approach copes with the fine grained heterogeneities of modeling languages, because differences in particular details can be abstracted as higher-level patterns. Of course, a transformation can only be defined based on patterns that can be identified and generated using the source and target modeling languages, respectively. Furthermore, compared to metamodel-oriented transformations using patterns is more of a top-down approach. This will help to quickly identify cases that can be transformed (i.e., identified and generated), and cases that cannot.

2.3. Workflow Patterns

The behavior aspect is the main aspect of process modeling. It is used to describe how the business process proceeds from its beginning to its end, and the order in which its tasks should be executed. This aspect is generally seen as an integrating aspect, meaning that in BP models all other aspects, such as
informational and organization aspect, are integrated by the behavior aspect [7].

Kindler et al. [8] made a more precise differentiation and divided the behavior aspect into two parts. The integral aspect covering only the tasks and the control flow aspect which consists of the concepts used to model the succession of the tasks. Only the tasks are used to integrate other aspects.

Since the integral and control flow aspects are the "heart of a process", we focus on them and leave other business aspects to future work. Of particular interest is the control flow aspect, because it is very rich in concepts and variability, as will be discussed in the following. There is a huge body of work relevant to the control flow aspect, most IThroughout this paper, we use the terms task and activity interchangeably concepts used to model the succession of the tasks.

Only the tasks are used to integrate other aspects.

Throughout this paper, we use the terms task and activity interchangeably. The integral aspect covering only the tasks and the control flow aspect which consists of the concepts used to model the succession of the tasks.

Throughout this paper, we use the terms task and activity interchangeably. The integral aspect covering only the tasks and the control flow aspect which consists of the concepts used to model the succession of the tasks.

3. Structural Transformation Patterns

In this section we propose a set of transformation patterns for the control flow aspect that cover the majority of cases occurring in BP models. The transformation patterns have been developed based on the WF patterns and on the analysis of existing BP models of different BPM languages.

Figure 5. Block-structured Process Model

The transformation patterns are introduced by means of block-structured and graph-structured models. For the initial definition of the patterns, a block-structured model (see Fig. 5) is used. This kind of representation seems to be most suitable for an intuitive understanding of the patterns. As BPM languages also support the creation of graph-structured models, i.e., models that are not restricted to a block-structure, the difficulties which arise in pattern matching are discussed by means of graph-structured models (see Fig.7 and Fig.8).

3.1. Patterns in block-structured models

As the name implies, block-structured models are composed of blocks of elements. Blocks could be nested within other blocks but it is not allowed that blocks partly overlap or that blocks include an end element. This eases the identification and definition of the transformation patterns, because each kind of block
can be identified as a pattern. In the following, seven transformation patterns are proposed (see Fig. 6).

The Start Pattern consists of a single element found in the Model in Fig.5 which marks the begin of a process model.

The End Pattern consists of a single element which marks the end of a process model.

The Sequential-Path Pattern is composed of one or more directly successive activities. For this pattern, WF pattern 1 (Sequence) has been extended. To enable uniform transformation definitions, a single activity is also recognized as Sequential-Path Pattern.

The Branching Pattern is a part of a process model between the point where the control flow is split into multiple flows, and the point where the control flow is merged from multiple into one successive flow (see Fig. 6). In contrast to the patterns described so far, a Branching Pattern contains nested patterns (Sequential-Path and/or further Branching Patterns and/or Loop Patterns). The Branching Pattern is further categorized according to the kinds of branchings that are used quite often in BP models. These are:

- Parallel Branches. The Parallel Pattern starts with a parallel split and ends with a parallel join. This pattern combines WF pattern 2 (Parallel Split) and 3 (Synchronization).
- Alternative Branches. The Alternative Pattern starts with a alternative split and ends with a alternative join. This pattern combines WF pattern 4 (Exclusive Choice) and 5 (Simple Merge).
- Multiple Alternative Branches. The Multiple Alternative Pattern starts with a multiple alternative split and ends with a multiple alternative join. This pattern combines WF pattern 6 (Multi-choice) and 7 (Synchronizing Merge).

A special kind of Branching Pattern is the Loop Pattern. It is a part of a BPM where a flow leads from a split element to a merge element which is a predecessor of the split element (see Fig. 6). This pattern includes all elements on the branch leading backwards. It is based on WF pattern 10 (Arbitrary Cycles).

Each block-structured process model consists of a combination of different patterns, and the above defined patterns are sufficient to describe every block-structured process model. The different patterns can be found in sequence or nested. While finding transformation patterns in block-structured models is straightforward, the next section deals with finding transformation patterns in graph-structured models.

3.2. Patterns in graph-structured models

Compared to block-structured models, patterns in graph-structured models can be overlapping (see Fig. 7) and branching patterns are not necessarily merged by a common merge element (see Fig. 8). The pattern definitions given in Section 3.1 need to be extended in adequate order to enable matching these patterns in graph-structured models.

The definitions for the Start Pattern, the End Pattern and the Sequential-Path Pattern remain unchanged.

A special kind of Branching Pattern is the Loop Pattern. It is a part of a BPM where a flow leads from a split element to a merge element which is a predecessor of the split element (see Fig. 6). This pattern includes all elements on the branch leading backwards. It is based on WF pattern 10 (Arbitrary Cycles).

Each block-structured process model consists of a combination of different patterns, and the above defined patterns are sufficient to describe every block-structured process model. The different patterns can be found in sequence or nested. While finding transformation patterns in block-structured models is straightforward, the next section deals with finding transformation patterns in graph-structured models.

3.2. Patterns in graph-structured models

Compared to block-structured models, patterns in graph-structured models can be overlapping (see Fig. 7) and branching patterns are not necessarily merged by a common merge element (see Fig. 8). The pattern definitions given in Section 3.1 need to be extended in adequate order to enable matching these patterns in graph-structured models.

The definitions for the Start Pattern, the End Pattern and the Sequential-Path Pattern remain unchanged.

For the Branching Pattern the end-condition needs to be refined: All of the outgoing branches (including the branches of subsequent split elements) of a split
element lead either (a) to a common merge element (see Fig. 9a) or (b) to an end element that can not be reached from the common merge element (see Fig. 10b) or (c) to an predecessor element of the split element (see Fig. 10c).

Figure 9. Nested Branching Patterns

The definition for the Loop Pattern itself remains unchanged but a Loop Pattern starting in a Branching Pattern must be considered in the definition of the Branching Pattern as follows: In addition to leading to an end element which is not the common merge element a branch could also lead (c) to a predecessor element of the common split element (see Fig. 10).

Figure 10. Nested Branching Patterns and Loop Pattern

By adding these extensions, it is possible to identify the transformation patterns in each model of a generic graphstructured language. In the next section, a case study demonstrates the use of transformation patterns.

4. Case Study - Using Structural Patterns for BP model transformation

In the following we shortly introduce the involved BPM languages and their main elements. Then the use of our structural patterns is demonstrated by means of a case study, which treats the translation from a ADONIS® BP model into the according EPC BP model. Finally we compare our approach with a supposed definition of the same scenario in QVT and summarize the advantages of our approach.

4.1. ADONIS® Standard Modeling Language

The ADONIS® Standard Modeling Language [4] provides different kinds of model types which cover different business aspects. The BP model is used to model the business processes itself, i.e., its behavior and control flow aspect. Furthermore it is used to integrate the organizational and the information aspect. Since our work focuses on the control flow aspect, we concentrate on the BP model. ADONIS® is a graph-structured BPM language.

Figure 11. Main elements of ADONIS®

The integral model element is the activity (see Fig. 11a). A sequence of activities is modeled by means of the successor which represents the control flow in the BP model. The control flow elements (see Fig. 11b) are used to model the control flow.

The ADONIS® BP model provides no special element for modeling merges of alternative control flows. Furthermore, the decision element does not distinguish between alternative split and multiple alternative split.

4.2. Event Driven Process Chains

Event Driven Process Chains (EPCs) [5] have been introduced by Keller, Nuetgens and Scheer in 1992. EPCs are basically used to model processes.

Figure 12. Main symbols of EPCs

We focus on the main elements which are used to model the integral and control flow aspect of a BPM, the elements Function, Event, AND, OR and XOR (see Fig. 12).
The Function describes an activity. It creates and changes Information Objects within a certain time. The event represents a BP state and is related to a point in time, it could be seen as passive element compared to the function as an active element (compare [15]). The remaining control flow elements (see Fig. 12b) are used to structure the proceed of the BP model. Different from ADONIS®, EPCs do not provide a specific element to indicate the begin and the end of a BP model. Event elements are used instead. Event elements are not allowed to be in front of an OR and XOR element. Function and event elements must alternate in the proceed of the BP model.

4.3. Translation of an ADONIS® model into an EPC model

For the case study we have chosen a model translation scenario in which a BP model defined in the ADONIS® Standard Modeling Language (see Fig. 13) is transformed to a BP model defined in EPC (see Fig. 17). The aim is to leave the given semantic of the BP model unchanged and adapt the syntax of the model according to the target BPM language (EPC).

The case study consists of four parts, definition of the patterns in each language, decomposition of the source BP model (see Fig. 13) according the defined patterns, transformation of each pattern, gluing of the target patterns to form the target BP model (see Fig. 17).

The source BP model describes a simplified reviewing process (see Fig. 13). When a paper is assigned the first activity is to download and print it. After that, the flow is split into two flows indicating that the update of the own CV and the Reviewing can be executed in any order. The intensity of the reviewing is dependent on the celebrity of the author. In case of a well-known author the paper could be assumed as good and according to this be reviewed fast. Otherwise the paper must be reviewed carefully. After that the control flows merge in a parallel join element and the process ends. The BP model consists of one start, one end, four activity, one parallel split, one parallel join and one decision element. The merge of the decision "Well-known author?" is modeled implicitly. For the transformation we suppose an algorithm that contains the general configuration which is equal for all inspected BPM languages, e.g. the start pattern has no predecessor and the end pattern no successor elements and that the loop pattern ends in a predecessor element etc. Based on this, specific configuration parameters have to be defined for each BPM language (cf. definitions in Table 1).

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Begin element</th>
<th>Explicit end/merge element</th>
<th>End/merge element</th>
<th>Other elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Start</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>End</td>
<td>End</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Sequential-Path</td>
<td>Activity</td>
<td>n/a</td>
<td>Last activity in sequence</td>
<td>n/a</td>
</tr>
<tr>
<td>Parallel Branch</td>
<td>Parallel Split</td>
<td>yes</td>
<td>Parallel Join</td>
<td>End</td>
</tr>
<tr>
<td>Alternative Branch</td>
<td>Decision</td>
<td>no</td>
<td>Any element</td>
<td>End</td>
</tr>
<tr>
<td>Loop</td>
<td>Decision</td>
<td>no</td>
<td>Any element</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table 1. Pattern defined in ADONIS®

At the moment we assume four different parameters, the begin element, the explicit end/merge element (indicating whether the end/merge is explicitly modeled or not), the merge/end element and other end elements (if branching patterns are not merged - see graph-structured models).

The second step in our transformation scenario is to walk through the source BP model and decompose it into our defined patterns. Since the example includes nested patterns, multiple decomposition cycles are used:

- In the first decomposition cycle we found a start pattern (see Fig. 14a), a Sequential-Path pattern (see Fig. 14b), a Parallel Branch pattern (see Fig. 14c) and an end pattern (see Fig. 14d).
- The second decomposition cycle analyzes the Parallel Branch pattern. It is further decomposed in an Alternative Branch pattern (see Fig. 14e) and a sequential-path pattern (see Fig. 14f).
- In the last cycle the sequential-path pattern is decomposed into two Sequential-Path patterns as depicted in Fig. 14g).

Now we have decomposed our source BP model into adequate patterns which will in the following be transformed into patterns of the EPC language as defined in Table 2.
The transformation of each pattern has to be done according to the information captured in the tables 1 and 2. The transformation of the start and end patterns is straightforward (see Fig. 15a and c).

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Begin element</th>
<th>Explicit end/merge element</th>
<th>End/merge element</th>
<th>Other end elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Event</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>End</td>
<td>Event</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Sequential-Path</td>
<td>Event or Function</td>
<td>n/a</td>
<td>Last event or function in sequence</td>
<td>n/a</td>
</tr>
<tr>
<td>Parallel Branch</td>
<td>AND</td>
<td>yes</td>
<td>AND</td>
<td>End Event</td>
</tr>
<tr>
<td>Alternative Branch</td>
<td>XOR</td>
<td>yes</td>
<td>XOR</td>
<td>End Event</td>
</tr>
<tr>
<td>Loop</td>
<td>XOR</td>
<td>yes</td>
<td>XOR</td>
<td>n/a</td>
</tr>
</tbody>
</table>

**Table 2. Pattern defined in EPCs**

In case of transforming the Sequential-Path patterns from ADONIS® to EPC we have to take care, that the EPC conditions, such as alternate Events and Functions and no Event followed by an OR or XOR element, are fulfilled. Therefore, we transform each ADONIS® Sequential-Path pattern into a EPC Sequential-Path pattern that starts with an Event. The information contained in the Events is derived from the preceding model element and the control flow edge that connects them. In case that the control flow edge contains no information, the postfix "-done" is added (see Fig. 15b, d and e). The transformation of the Branching pattern consists of a translation and a composition.

**Figure 14. Decomposition of the model**

**Figure 15. Transformation of Start, End and Sequential-Path pattern**

Figure 16a illustrates how the Alternative Pattern is transformed into EPC. Here the knowledge of the explicit merge element in EPC is used to create this element at the end of the Alternative Pattern. After that, the Parallel Pattern is translated and composed as shown in Fig. 16b.

**Figure 16. Transformation and Composition**

Finally we have to glue all the parts together to form the target EPC model as depicted in Fig. 17.

**Figure 17. Translated target model**

Attention must be given to the sequential-path pattern after the start pattern. In this case the event of the target pattern has to be deleted because of the preceding start event (see Fig. 15b) to avoid consecutive events. In case of defining the above BP model transformation scenario with a metamodel-based language such as ATL or QVT, there are similarities concerning the Start, End and Sequential-
Path pattern. These patterns could also be transformed by using simple metamodel correspondences. But in case of creating the explicit merge element in the target language we have to write a lot of imperative code. This code is in most cases not reusable for other model transformation scenarios, whereas the pattern-definition for a given BPM language is reusable when transforming its models into another BPM language. Additionally pattern-based definitions can be easily adopted for translating the models in the other direction. This is useful in case of the synchronization of BP models as depicted in Fig. 1c. In case of merging BP models (see Fig. 1a) the patterns fulfill two different functions. First the detection of the equal structures in the source models and second, their correspondence to the target model.

A general observation that we made is that in case of a transformation from a BPM language with few elements for expressing control flow structures, such as ADONIS®, to a BPM language which provides more elements to express control flow structures, such as EPC, it is preferable to take a model transformation approach that additionally supports pattern-based definitions.

5. Related work

There is a formal method for decomposing flowgraphs [16] which at first sight looks very similar to our approach of defining patterns for BP models. Basically this approach aims to improve the measurability of imperative source code. Based on the fact, that imperative programming languages could be illustrated as directed graphs the authors show that specific programming constructs could be modeled as sub graphs, so-called primes. Decomposition of flowgraphs into primes assumes block-structured models. In case of graph-structured models which occur in BPM languages, occurring primes can become arbitrarily complex. Furthermore the number of different primes is not limited in case of graph-structured BP models. Thus it is not possible to define the number of primes in advance. So structural patterns for analyzing BP models have to be more abstract as the introduced primes. Defining the begin and the possible ends of a distinct pattern reduces the number of different patterns to be defined.

In defining model transformations, patterns have already been used for various purposes.

Design patterns in Model Driven Engineering (MDE), as initially discussed in [17], capture recurring problems in the design of metamodels and model transformations. In [17], examples of design patterns relating to transformation implementation are given. Different to our work, these patterns are domain-independent and thus the patterns address issues with MDE technologies rather than a particular modeling domain.

Pattern-based model refactoring [18] is a concept for controlled model evolution and development employing model transformations. Model refactoring focuses on "forward transformation", i.e., transformations that modify a model but do not change the modeling language, as opposed to horizontal transformations.

The MDA tool OptimalJ combines the MDA with pattern-based development [19]. It uses two kinds of patterns. First, so-called transformation patterns perform vertical transformations, i.e., PIM to PSM and PSM to code; these patterns embody design knowledge. Second, so-called functional patterns reside on a particular layer, i.e., PIM, PSM or code, and represent recurring solutions to problems of application development. The well-known GoF design patterns are an example of functional patterns. While our patterns are also situated within a particular layer, they differ in that they do not address application development but rather recurring problems of horizontal transformations.

The final submissions for the QVT language [3] includes patterns as core concept of a model transformation language, to be used for queries on source models and as templates on target models. The QVT submission uses a declarative formalism for defining patterns. The patterns and the transformation rules using the patterns are designed to support aspect-driven transformations, as they allow construction of a target model element by application of several rules. While QVT defines a language to define patterns independent of particular domains, our work focuses on identifying particular patterns in the BPM domain. Another difference is that the QVT language requires distinct patterns for source and target, whereas our approach proposes generic patterns that are used as a bridge between source and target models, thus exploiting the specifics of horizontal transformations.

Using higher-level abstractions to design horizontal transformations is also proposed by [20]. In particular, [20] defines horizontal model transformations based on semantic concepts as defined in ontologies rather than on syntactic concepts as defined in metamodels. Our patterns are not focused on particular semantics but rather are motivated by their usefulness in transformation definition. Furthermore, the use of patterns, i.e., complex structural abstractions, is not elaborated in [20].
6. Conclusion and Outlook

In this paper a general approach to BP model transformation based on patterns has been introduced. We focus on the control flow aspect of BP models, which is considered as one of the catchiest of the aspects with respect to BPM transformations.

The objective was on the one hand to introduce a transformation approach which abstracts from the single metamodel elements usually used. On the other hand, we have provided a detailed description of the transformation patterns which helps to comprehend the structure of BP models. Furthermore, a case study about the translation of a model from ADONIS R to EPC has been discussed. An apparent advantage of covering the structure of BP models with transformation patterns is that it makes the differences between the various BPM languages visible. This knowledge helps to focus the transformation development effort especially on the differences between BPM languages.

This work represents the first step in realizing the proposed domain-specific transformation approach for BP models. We are currently working on an algorithm for detecting the defined patterns automatically. The next steps include the definition of the transformation of the individual patterns between different BPM languages, and it is planned to evaluate this approach in the scope of an industry project where BP models have to be merged.

References


Abstract

Enterprise organizations use Data Warehouses (DWHs) to analyze their performance. Performance is judged regarding the achievement of goals.

DWH data models are well established. There exist numerous domain-specific modeling approaches. Enterprises also often model their goals in terms of formal or semiformal goal models.

The problem is that these two aspects - the Data Warehouse and the Enterprise Goals - are described separately and not related to each other. We identify a need for combining these two aspects. If their relationship is made explicit, it can be used to enhance the way users access and interpret data in the DWH.

To address this limitation, in this paper we introduce a weaving model between enterprise goals and DWH data. Thus we present a domain-specific application of model weaving to an aspect of enterprise computing. We describe metamodels for both aspects as well as the weaving links between them, which allows to show the aspects separately but also in combination. We furthermore illustrate how to use the weaving links to create business metadata. Business metadata can be used in the DWH to describe the business context and implications of the data to the users, but is usually not available in today’s DWHs. We apply our approach to a sample situation, which is used as a running example in the paper.

1 Introduction

Data Warehouse (DWH) systems represent a single source of information for analyzing the status, the behavior and the development of an organization [17]. Analysts and decision makers analyze this information with regard to the goals and the strategy of the organization and use it to recognize warning signs or trends and to decide on future investments.

Today’s DWHs provide their users with very powerful tools for analyzing large amounts of data, but supporting the actual data interpretation by decision makers, i.e. with business metadata, has not yet been a primary concern in research. Business metadata describes the business context of the data, its purpose, relevance, and potential use [26], thus supporting data interpretation.

The context information needed for meaningful data interpretation, e.g. about the goals of the enterprise and the metrics used to evaluate them, is usually available in enterprise models or other repositories within the organization. But the link between the organizational information and the DWH is not easily accessible to DWH users and remains mostly implicit knowledge. If knowledge about the business context is left to chance, data analysis is bound to miss important points, and large parts of the potential of the DWH are wasted.

Models for DWH data structures are well researched and established (see Sect. 2). There exist numerous domain-specific modeling approaches to describe the data structures on the conceptual, logical and physical level. Enterprise organizations using a DWH are aware of these models and rely on them to design and describe their DWH.

To define and analyze their objectives and goals, enterprise organizations use different types of goals models (see Sect. 3). For the goals, metrics are derived to measure the achievement of the goals. There exist many modeling approaches for enterprise goals as well as for metrics.

The problem that we address here in this paper is that these two aspects - the Data Warehouse and the Enterprise Goals - are described separately and not related to each other. We identify a need for combining these two aspects.
If their relationship is made explicit, it can be used to enhance the way users access and interpret data in the data warehouse. The goal of this paper is to address this limitation by:

- making the relationship between the DWH data and the goals of the organization visible and accessible by showing how the enterprise goals and metrics are mirrored in the DWH data model
- using this information to support and improve data interpretation
- enriching the DWH with business metadata that explains the relevance and business context of the data

To achieve these goals, we employ modeling techniques. We propose a weaving model (see Sect. 4) to link an enterprise goal model with the DWH data model. The weaving links then can be used to provide business metadata to the DWH users. Business metadata informs users about the organizational context and implications of what they are analyzing [26].

Thus we apply model weaving to the domain of data warehousing. Our approach offers the following contributions:

- It makes the implicit relationships between the data in the DWH and the business goals visible and accessible.
- By relating the measures in the DWH to the overall organizational strategy and enterprise goals, decision makers can better interpret the enterprise performance, and understand the implications.
- Creating the weaving links is a comparatively small investment for valuable metadata that gives meaning to DWH measures.
- The enterprise goals serve as a “single source of information” to avoid DWH requirements analysis and (re-)design are notoriously challenging tasks, because the business context of a DWH is difficult to extract from user interviews and practically impossible to store directly in the multidimensional data structures. Weaving an enterprise goal model with the data model makes context information accessible, and does so without disrupting the involved models.
- The weaving model can be used for model validation, as it identifies missing or superfluous tables and measures as well as omissions in the goal model.
- The link to enterprise goals and business strategy can help to evaluate DWH investments, and to justify them.

The following sections describe how the DWH data (Sect. 2) and the enterprise context (Sect. 3) are modeled. Section 4 introduces the weaving model that connects the two models, and illustrates the resulting business metadata with an example. Related work is discussed in Sect. 5.

2 Multidimensional Data Models

The main data model in Data Warehousing is the multidimensional model, also called star schema [8]. It is meant to provide intuitive and high performance data analysis [17].

DWH applications involve complex queries on large amounts of data, which are difficult to manage for human analysts. Relational data models “are a disaster for querying because they cannot be understood by users and they cannot be navigated usefully by DBMS software” [17]. In Data Warehousing, data is often organized according to the multidimensional paradigm, which allows data access in a way that comes more natural to human analysts. The data is located in n-dimensional space, with the dimensions representing the different ways the data can be viewed and sorted (e.g. according to time, store, customer, product, etc.).

A multidimensional model, also called star schema or fact schema, is basically a relational model in the shape of a star (see Fig. 1 for an example). At the center of the star there is the fact table. It contains data on the subject of analysis (e.g. sales, transactions, repairs, admissions, expenses, etc.). The attributes of the fact table (e.g. cost, revenue, amount, duration, etc.) are called measures. The spokes/points of the star represent the dimensions according to which the data will be analyzed. The dimensions can be organized in hierarchies that are useful for aggregating data (e.g. store, city, region, country). Stars can share dimensions, thus creating a web of interconnected schemas that makes drill-across operations possible.

There are many approaches to modeling the multidimensional data structures of data warehouses [1, 6, 30], some of which are object-oriented models or based on the Unified Modeling Language (UML) [1, 21, 28].

For weaving enterprise goals (as part of the enterprise model, see Sect. 3) with the DWH, we need a data model that supports weaving. We choose the object-oriented approach first presented in [28] and then further developed to a UML profile in [19]. A UML Profile is a domain-specific extension to the UML modeling language. This profile adapts the UML class diagram for multi-dimensional modeling, i.e. the base class of the stereotypes is Class. It allows to create detailed models of the conceptual characteristics of multidimensional data models. Figure 2 shows the main elements of the UML Profile and their relationships as a metamodel, and Fig. 1 shows an Expenses fact from modeled with the profile.

The Expenses fact has four dimensions: Time, Account (e.g., IT, Marketing, etc.), Scenario (e.g., Actual, Forecast) and Store. The levels of the dimensions are only shown for the store dimension. Each entry in the fact table contains information about a single expense incurred. In this example there is only one measure that can be analyzed for each expense: the amount. Aggregations such as “total value of...
an account in all stores in one year” become possible by selecting aggregation levels from the dimensions. Several such facts can be connected by sharing the same dimensions, creating a more complex multi-cube model.

The elements shown in Fig. 2 allow to model any number of Fact tables. Each Fact table can have any number of optional Measures and must have at least two Dimensions connected to it, at least one of which is usually a Time dimensions. Dimensions may be shared between facts and have one or more Aggregation Levels, which form the aggregation hierarchy.

See [19] for the additional, more detailed elements not shown here, such as the attributes of the dimensions, as well as a number of logical constraints on the model elements [19]. They are defined in the Object Constraint Language (OCL) [22] and cover aspects of multidimensional data models such as ensuring that the classification hierarchies have the form of directed acyclic graphs, as well as additivity rules, derived measures, and degenerate dimensions.

3 Enterprise Models

3.1 Modeling the structure, behavior and goals of an enterprise organization

An enterprise model formally represents the basic building blocks of an organization, its structure, behavior and goals. It is usually organized into several aspects [31] that can be modeled individually but also related to each other. The Architecture of Integrated Information Systems (ARIS) [27] is a typical example for such an enterprise model.

Figure 3 shows the outline of a generic enterprise model, organized into five aspects: The enterprise strives to achieve goals, acts through processes, has an organizational structure, produces products and uses applications. In the enterprise model, an organization chart can be used to describe the organizational structure, i.e. the dependencies between the departments, groups and roles that exist within the organization. Similarly, a business process model describes the structure of business processes with control flow, inputs and outputs. The products, applications, and strategic goals can also be modeled separately. An overview model would then connect these model to show for example how processes fulfill goals, are performed by organizational roles, fall into the responsibility of departments, and use applications to produce products for other departments.
### 3.2 Enterprise Goals

A core part of every enterprise model is the goal model. “Increase market share” or “reduce operating costs” are typical enterprise goals. Goals form the basis for business decisions and the way a company does business. What is relevant and important for business performance measurement can be read directly from the enterprise goal model. They govern the design of business processes and the way the organization behaves. Nevertheless, a goal model is basically very simple, and enterprise goals are long term goals that should remain stable a lot longer than business processes, role definitions, and operating rules. Therefore, they provide excellent metadata for a DWH.

Based on the description of the goals, the enterprise derives metrics that measure the level of achievement of the goals and indicate the performance of the enterprise. These metrics are not identical but closely related to the measures in the DWH. In the early 1990s, business goals and strategy experienced a revival in theory and practice with approaches like the Balanced Scorecard (BSC) [13]. The BSC’s focus is on vision and strategy. Companies define goals from various perspectives, which are then translated into measures. The BSC does not mention the behavior which will lead to the fulfillment of a goal. Rather, people are assumed to be guided by the measures they have to fulfill. Measures and not the desired operations are communicated to the employees. The goals and measures give the long-term focus, while the conditions in which people operate are constantly changing.

In the Goal Question Metric Approach [3], originally aimed at software quality improvement, measurement and evaluation is based on a three-level hierarchy in which the goals (of an organization) form the first, the conceptual level. Goals are the starting point of all measurement activities and provide the organizational context according to which the measurement data can be interpreted.

Different kinds of goals, including enterprise goals, are often used in software engineering for requirements elicitation. For example, the i* Methodology [32] provides an agent- and intention-oriented way to analyse requirements for software (and other) systems. The focus of i* is on interaction between autonomous agents, their actions and strategies.

For enterprise goals in particular, there is often a distinction between three levels of goals: strategic, tactical and operational. In order to be able to transform high level enterprise goals of the strategic level via tactical level goals to every-day operational goals, a goal is decomposed via a causal transformation or operationalization into one or more subgoals, which in turn can be further decomposed, thus creating a hierarchy (cf. [18]).

### 3.3 Enterprise Goal Metamodel (EGM)

The Enterprise Goal Metamodel presented here incorporates features from a number of existing goal modeling approaches (cf.[13–15, 18, 32]) It is aimed at providing a sufficiently detailed and comprehensive, yet concise model of the main concepts that are needed to model the context of a DWH.

Figure 4 shows the Enterprise Goal Metamodel (EGM). For sake of clarity and readability, it is shown as two separate graphics: Fig. 4(a) explains goal decomposition hierarchies and relationships between goals and Fig. 4(b) shows the other elements of the metamodel related to goals. The model uses the notation of the UML 2.0 class diagram [23].

Figure 5 shows example goals for Fig. 4(a). In the EGM, a Goal may participate in a goal hierarchy via a Goal Decomposition. The goal decomposition connects a higher-level goal with a number of lower-level subgoals. A goal may have only one set of subgoals but may participate itself as a subgoal in more than one goal hierarchy. Therefore it can be related to only one goal decomposition in the role of a satisfied goal but to many in the role of a satisfier goal. The goals “reduce out-of-stock” and “increase freshness” in Fig. 5 are subgoals of “satisfy customers”. From the viewpoint of the “AND” goal decomposition in Fig. 5, the four lower-level goals are satisfier goals, and “satisfy customers” is the satisfied goal. The type of a goal decomposition is either AND or OR, depending on whether all or only some of the subgoals have to be satisfied to satisfy the upper level goal.

Orthogonally to the goal hierarchy, goals can be seen to influence each other in various ways. The fulfillment of one goal might be detrimental to another goal, or the goals may be related to each other in such a way that if one of them is satisfied, this also supports the other goal. Therefore, there are two influencing relationships between goals: support and conflict. Both may occur between any number of goals, e.g. a goal can support several goals and conflict with others at the same time.

Figure 4(b) (illustrated with values in Table 1) shows that for each Metric assigned to a goal it is necessary to define a Target Value. Because target values usually change over time, a Time Frame for each combination of metric and target value is necessary. Goals can be relative goals (“increase the value by x”), or absolute goals (“the value should be x”), indicated by the attribute isRelative (shown in Fig. 4(a)). This influences the semantics of the timeframe: For a relative goal and its metric, it means that the change is to be achieved during this time, whereas for an absolute goal it indicates the validity period of the target value. A metric can be constrained with Parameters, which define the scope: The goal “reduce inventory cost” has none, “reduce inventory cost of top-20 products” has one and “reduce
Figure 4. Metamodel for describing (a) goal decomposition hierarchies and relationships between goals and (b) the details related to enterprise goals

Figure 5. Sample goal hierarchy model corresponding to the metamodel in Fig. 4(a)
Table 1. Details regarding Fig. 4(b) for the example goals from Fig. 5

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal name</td>
<td>reduce “out of stock”</td>
</tr>
<tr>
<td>Goal perspective</td>
<td>Customer</td>
</tr>
<tr>
<td>Reported to + contact info</td>
<td>Sales Dept., Ms. Baker, Tel. 5800193</td>
</tr>
<tr>
<td>Metric name</td>
<td>number of days in month with no items on stock</td>
</tr>
<tr>
<td>Metric target value + unit</td>
<td>&lt; 1%</td>
</tr>
<tr>
<td>Metric responsible + contact info</td>
<td>Mr. Jones, Tel. 5800234</td>
</tr>
<tr>
<td>Conflicting/supporting goals</td>
<td>conflicts with “increase freshness” and “reduce IT cost”</td>
</tr>
<tr>
<td>Goal this goal satisfies</td>
<td>“satisfy customers”</td>
</tr>
<tr>
<td>Goal name</td>
<td>increase freshness</td>
</tr>
<tr>
<td>Goal perspective</td>
<td>Customer</td>
</tr>
<tr>
<td>Reported to + contact info</td>
<td>Mr. Groop, <a href="mailto:groop@dpt.company.com">groop@dpt.company.com</a>, Ms. Fitt, Tel. 5800156</td>
</tr>
<tr>
<td>Metric</td>
<td>avg. time in warehouse for product group “fresh goods”</td>
</tr>
<tr>
<td>Metric target value + unit</td>
<td>&lt; 8 hours</td>
</tr>
<tr>
<td>Metric parameter(s)</td>
<td>Warehouse, product group</td>
</tr>
<tr>
<td>Metric responsible + contact info</td>
<td>Mr. Stephens, Tel. 5800655</td>
</tr>
<tr>
<td>Conflicting/supporting goals</td>
<td>conflicts with “reduce out of stock”, supports “reduce IT cost”</td>
</tr>
<tr>
<td>Goal this goal satisfies</td>
<td>“satisfy customers”</td>
</tr>
</tbody>
</table>

In order to gain business metadata, we need to link the enterprise goals to the DWH data. For this task, we have chosen to employ the technique of model weaving [5]: An additional model is created between the two metamodels that are to be linked. This so-called weaving model consists of typed weaving links that describe the relationships between two or more model elements from the respective metamodels. Advanced modeling tools such as the ATLAS Model Weaver [2, 9] (available as an Eclipse [11] plug-in) support model weaving. The models concerned have to be based on a common high-level modeling approach, e.g., be based on MOF [24] as their common meta-meta-model.

Model weaving superficially resembles techniques used in ETL or EAI, but the intention behind it is different. A weaving link does not necessarily imply that the two elements it connects should be transformed from one into the other. Rather, it simply indicates that the two elements share some semantic link, e.g. “lies in the responsibility of”, “is measured by”, “affects”, etc. Still, weaving can of course be used as a preliminary step to transformation, by indicating transformation sources and targets and then using the weaving model as an input for a transformation language. In this paper it is employed for annotating the DWH data with business metadata, and therefore does not imply transformation.

We choose model weaving because it offers a number of advantages: By adhering the “Everything is a model” principle [4], we can capture practically all information in terms of models, also the relationships and correspondences between models. This makes it possible to store, analyze and edit the links with modeling tools. Weaving avoids having one large meta-model “for everything”, but instead keeps the individual meta-models separate, easy to handle and focused on their domains, while at the same time they are interconnected into a “lattice of metamodels” [4].

Figure 6 shows the weaving model linking the data (right) and enterprise goal (left) metamodels presented in Sect. 2 and 3. It consists of three links: two binary and one ternary link.

The first link is between the Parameter describing the focus or scope of the metric (e.g., Region is the parameter when a value is given by by region) in the EGM and an Aggregation Level (e.g. per Month on the Time Dimension or per Region on the Store dimension). These are similar concepts, which can be easily mapped. The corresponding dimension to a level is provided by the data model.

Weaving links can connect more than just one element on either side. The second, most complicated link in this weaving model connects a Metric with a Measure and optional Aggregation Levels. A metric roughly corresponds to
Figure 6. Weaving Model connecting the Goal Model (left) with the Data Model (right). Only a part of the Enterprise Model is shown here for readability. It has more than the two aspects, and the organizational aspects has more than the two elements shown here.

Figure 7. Displaying business metadata connected to fact data
a fact attribute. The fact attribute itself is not aggregated, but the metric can be restricted by parameters: This has to be indicated on the data model side by adding the corresponding aggregation base(s) to the link. Additionally, because the fact attribute itself contains only the absolute value of a variable, while the metric related to it might contain an average, a percentage, or a rate of change, this weaving link can contain a formula (e.g., \((IT\cost/total\cost)*100\) for the percentage of IT costs).

Finally, the third link allows us to handle the relationships between the Timeframe of a metric’s target value and a Dimension containing time values in the data model\(^1\). A timeframe is a time period, indicated by start and end point, whereas a time dimension contains single points in time. Therefore the weaving link connects one timeframe with two points on the time dimension. Or if the timeframe has the format of “until x”, with one point.

Analysis tools and applications can use the business metadata derived through the weaving links, similarly to technical metadata, to enhance the way users access and interpret data. Where before there were basically only numbers, there now is context and explanation. Through knowing which goal it measures, it becomes clearer what a certain value means and why it is important. The actual values of the measures can be compared to the target values of the metrics. This business metadata can be also incorporated directly into the user interface of analysis tools.

Figure 7 shows how the business metadata can be displayed for an example cube, based on the prototype we are developing. The organizational knowledge captured in the enterprise goal model becomes available to the user. Providing this information to the user directly within the analysis tool helps to improve data interpretation. The business metadata thus increases the usefulness of the data.

In the example in Fig. 7, business metadata is derived from the link between metrics and measures. Combined with the two other links (concerning dimensional aggregation and temporal values), the links can also provide insights for DWH (re)design and maintenance, or requirements analysis. The knowledge captured by the weaving model can be exploited by analysis tools (e.g., to offer better navigation, or hints).

5 Related Work

The term “weaving” is also used in aspect-oriented programming, where it denotes the integration of aspects into the base program [16]. See the AOSD Ontology [29] for more general definitions that apply not only to the programming level, but also to modeling.

\(^1\)This can be ensured by an OCL [22] constraint, e.g.

\[
\text{Context TimeFrame}
\]

\[
\text{inv: self.dimensions-> forAll(d|d.isTime = true)}
\]

In [7], Breton and Bézivin apply model weaving to the area of workflow and process modeling. The build-time and the run-time workflow definitions are weaved together to create a binding between definition and execution of the process.

The alignment of models in the Data Warehousing research field is quite young, although there are some very good examples available.

Mazon et al. defined in [20] the application of the MDA framework to DWH repository development, and aligned multidimensional conceptual data models with code. In MDA terms, they aligned a Platform Independent Model (PIM) with a Platform Specific Model (PSM) and defined a transformation between them. Starting with a conceptual model, Mazon et. al developed a logical model using a relational approach to build a star schema. Then, they derive the necessary code to create data structures for the DWH.

Our approach can be seen on top of this work targeting the Computation Independent Level (CIM) level, as we align enterprise goals, representing the business requirements as well as context, with the DWH conceptual data model.

Giorgini et al. focus on DWH requirement analysis based on goals in [12]. They derive the data model from the goals, which represent a rather narrow software engineering type of goals. In contrast, we integrate enterprise goals and align the DWH directly with business strategy.

Sarda linked DWH business metadata with technical metadata in [26], in order to provide a better context for decision support. Several business metadata categories like goals, organizational elements, processes, events, measures, etc., and a number of desirable characteristics such as evolution of navigation between metadata and data, are defined. The business metadata is described with UML classes and associations and then linked directly to the technical metadata within the same model. The approach only covers metadata and does not use separate conceptual models of the business context. Additionally, our weaving model is focused on the details of enterprise goals and their measures, rather than on all aspects of an enterprise.

6 Conclusion and Future Work

In this paper we have presented an approach to business metadata that is based on the relationship between the DWH data and the goals of an organization. The enterprise goals and information related to them such as metrics and target values as well as the people and departments involved, are taken from an enterprise model. The business metadata is created by linking this knowledge about the organization to the DWH by means of a weaving model.

The business metadata is then created directly from the weaving model. It improves data interpretation by explaining the relevance and context of the data, whereas the
weaving model itself supports DWH requirements analysis, (re)design and evolution by making context visible and accessible. The approach is applied to an example.

Regarding future work, we are currently working on a prototype of a toolkit that supports users in choosing and integrating the enterprise models available to them, through creating the weaving model that links them to the DWH, to finally creating and displaying the business metadata. We are also investigating the use of other aspects in the enterprise model (apart from the enterprise goals) as suitable business metadata for the DWH. The prototype is based on the Eclipse [11] platform. It will make use of the open-source DWH platform Pentaho [25], which is built on Eclipse and combines well-known DWH components such as Mondrian, BIRT, and JFreeReport, on the one hand, and on the other hand use the the Eclipse Modeling Framework (EMF) [10] and the ATLAS Model Weaver [2]. It forms the basis for future case studies.

References

Modeling for Investment Guidance as a Performance Pre-assessment Method of Enterprise Integration

Jian Zhou¹,², Qing Li¹, DaFeng Xu¹, TianYuan Xiao¹
¹Department of Automation, Tsinghua University, Beijing, 100084, P. R. China
²National University of Defense Technology, Changsha, 410073, P. R. China
zhoujiandragon@gmail.com

Abstract

Current enterprise performance research focuses on the post-assessment and is always lack of approaches about the necessary pre-assessment. In this paper a modeling method for investment guidance of enterprise integration is proposed, which is a significant pre-assessment method. Because of the numerous dynamic and subjective indicators of performance, modeling method presented in this paper is mainly based on the fuzzy rule for its natural advantages in this area. Firstly, considering the research requirements discussed in the literature review, the performance modeling framework for investment guidance is put forward grounded on the process model. Secondly, the original motivation to employ the fuzzy logic method is discussed and a theorem with logical criteria is suggested for conflict and redundancy validation of fuzzy rules which is necessary for performance modeling. Finally, the model for investment guidance is constructed and then a non-linear programming model is formulated to optimize the investment of enterprise integration.

1. Introduction

Years ago, enterprise integration became a popular concept and it was considered as the most useful approach for enterprise to improve their competitive ability and effectiveness. In long-term industrial practice and the implementation of a large number of enterprise integration projects in China, It has been found that entrepreneurs or business managers can scarcely implement enterprise integration based on their confidence in advanced technology. They expect that the return on investment can be known in the designing phase of a new system. As a farseeing business-person, he/she should take into his/her consideration not only tangible benefit, but also, more importantly, intangible benefit. However, how to evaluate this return is a difficult but ‘must-be-solved’ problem. Similarly, when it is required to evaluate different alternatives, the models in those existing enterprise integration architectures are not able to connect functionality with economic consequences so that design trade-off can be made.

Actually in a system integration project, demands of the project target are reflected by demands of the economic characteristics, and influences on the system are realized by the integration strategy and the technology project. Therefore our research team addressed Economic View [1] to complement the Enterprise integration architecture and it is widely accepted in the related circles as a measurement method for integrated system performance. Furthermore, since the last year, it has become the annex C of ISO 15704. However the work on Economic View issued before mainly focused on the performance framework and it is lack of performance modeling methods that support it from concept to reality. To partially fulfill the purpose of economic view, namely to optimize the necessary investment according to possible benefits before implementation, in this paper a performance pre-assessment modeling method of enterprise integration is proposed.

The rest of this paper is organized as follows: The second section is about the literature review. In the third section, the performance modeling framework for investment guidance is presented. In the fourth section, a theorem is suggested for conflict and redundancy validation of fuzzy rules. In the fifth section, based on process model and fuzzy rules, the modeling process and optimization method for investment guidance is proposed. The final section is about conclusions.
2. Related Work of Performance Modeling

Related research of performance measurement, from the modeling perspective, can be depicted in four categories as follow [2]:

(1) Performance model consists of indicators, which happened in the stage of early years, such as the pilot process proposed by Neely et al. [11].

(2) Frameworks are developed as the Performance model, which have undoubtedly made the largest impact on the performance circle, for instance the structural integrated performance measurement framework developed by Rouse and Putterill [12].

(3) The most mature model is the performance measurement system and it is basically composed of frameworks and necessary performance management tools. The business process reengineering performance measurement system [13] is a typical example.

(4) Besides, there are a large number of other researches related to performance of enterprises, for example applying the dynamic system theory and genetic algorithm to resolve problems in this area.

With the efforts of many researchers, research in this field is significantly developed however still has not fully stepped up to meet the actual requirements. There still remains following insufficiencies and obstacles of present study in this research circle:

(1) Current research mainly focuses on the measurement method for the performance of implemented integrated system and is always lack of the approach about the necessary analysis for investment before implementation. Thus, it is difficult to analyze the necessity for project implementation as well as to discover the deviation of investment.

(2) Much of the performance information is inherently uncertain--lack of precision, vague concepts, more or less reliable information, etc. The corresponding indicators are numerous and a large number of them are intangible and qualitative.

(3) Performance measurement is still hard to be computer processible and automatic to make the advantage of IT applied in enterprise.

(4) The multidisciplinary character of the research is hindering developments in the field of Performance measurement [9] [2], which will result in the fact that various research conclusions may be isolated, duplicated and/or contradictory.

Intending to partially solve the problems discussed before, the performance modeling method here is a significant complement for Economic View. In the following sections it will be detailed step by step.

3. Performance modeling framework for investment guidance

As discussed in the above, the former research of Economic View is largely about the conceptual framework and is lack of concrete modeling method. In the dissertation [7] of one fellow in our team this year, as a further work, why the performance model is constructed as a structure of multi-indicators hierarchy is discussed from the view point of the depth/span of performance indicator and the information delivery distortion. In his opinion, among the multi-indicators, those with high depth form the bottom layer and are employed to gather professional and special information from real system, while those with wide span compose the top layer and provide the universal performance information of model. To establish the performance model, it is believed that the key problem is to decide what kinds of indicators should be selected and how to bridge the relationships between these indicators. From industrial practice and literature, the selection always depends on experiences, while the research on relationships between them are always configured directly by some experts or related people. Actually, without the special context, the dispersed information in the complex integrated system, denoted by the indicators, is hard even to describe and understand. In this case, the relationships between them are more difficult to analyze and form. So performance modeling is undoubtedly supposed to make full use of all sorts of related knowledge and information.

In this paper, the performance model will be constructed with the support of process model, which is actually ‘to be’ process model. Reasons to align with process model are convincing. Firstly the hierarchy of performance model can be constructed associated with the process hierarchy. Secondly process model is map or image of the logical and temporal order of business activities performed on a process object and also contains both behavior and structure of all objects that may be used during process execution, so a large sum of diverse information is embedded in it, which is a great help for model establishment and performance analysis. Thirdly the research on process model is relatively mature and abundant. Furthermore some experts also argued that the measurement system should be focused on processes, not on whole organizations or organizational units [6].

What should also be mentioned about the method proposed in this paper is the application of fuzzy logic. It is widely accepted that fuzzy set theory provides a prominent paradigm in modeling and reasoning with
uncertainty [3]. Fuzzy logic is based on fuzzy set theory and it has two major industrial applications in the history: in the field of fuzzy controllers and information systems. In the integrated system, there are an enormous amount of electronic accessible information and knowledge which is inherently uncertain—lack of precision, vague concepts, etc [5]. And it is supposed to make full use of all of them for the performance measurement despite its uncertainties and imperfection. In accordance, fuzzy logic can not only be used to represent this knowledge exactly, but also to utilize it to its full extent. And the complex knowledge presented by fuzzy logic still can be computer processible and allow fast mining and processing even there are a large bodies of them. In view of these facts, the performance modeling in this paper is chiefly based on fuzzy rules.

4. The preparative research for the application of fuzzy rules

In this paper, the performance modeling is based on and begins with fuzzy rules. In general such rules are obtained from human experts and are formulated as a linguistic IF THEN paradigm. However because of the insufficient and shortcoming of this kind of approach for knowledge acquisition, many methods have been presented later for automatic extraction from numerical data [4] [8]. To make the fuzzy rules more sufficient and reliable, no matter those extracted by certain automatic algorithm or those obtained from human experts, etc., all of them will be adopted to support the performance modeling. In this case, there must exist many unexpected and undesired conflicts and redundancies between those rules [10]. Thus before applying these fuzzy rules, a theorem with logical criteria is suggested for partial conflict and redundancy validation of them that are necessary for our method.

The form of the fuzzy rule \( f_i \) is always shown as:

\[
\text{IF} \ (x_1 \in A_{i1}) \ \text{AND} \ (x_2 \in A_{i2}) \ \text{AND} \ \ldots \ \text{AND} \ (x_n \in A_{in}), \ \text{THEN} \ (y_1 \in B_{1}), \ y_2 \in B_2, \ldots \ y_j \in B_j
\]

\[1 \leq i \leq n, \ i, j \in N
\]

Where \( A_{ij} \) is one of the possible intervals of the i-th premise that is corresponding with certain performance indicator \( p_{ij} \) of fuzzy rule \( f_i \). Let us suppose \( PI_i \) is the set of indicators associated with premises of \( f_i \), then \( PI_i = \{p_{i1}, p_{i2}, \ldots, p_{in} \} \). \( B_i \) denotes one of possible intervals of the relevant conclusion that is associated with certain performance indicator \( PI_{1x} \).

The alphabet \( n \) represents the total number of premises and \( N \) denotes the natural number. \( x_{ij} \) is the value of the i-th premise of \( f_i \) and \( y_{ij} \) is the value of associated conclusion.

Let’s suppose F is the set of fuzzy rules. \( \forall f_{i1}, f_{i2} \in F \), if \( PI_{i1} \) is the same with \( PI_{i2} \), or \( PI_{i1} = PI_{i2} \), then there will exist the potential conflict and redundancy between \( f_{i1} \) and \( f_{i2} \). For every fuzzy rule, the premise is the sufficient but not necessary condition for the conclusion, so the reasoning from the
premise to the conclusion can be implemented to validate the conflict and redundancy if PI_j, is the same with PI_i, which is impossible. If A_i \cap A_j \neq \phi, because, for each concrete fuzzy rule f_j, A_i is the determinant interval of the i-th premise corresponding with PI_j, so it is impossible for f_i and f_j to have the same value of the premise if A_i \cap A_j = \phi, which means that there is no conflict and redundancy between them from the aspect of the premise. On the other hand it is imperative to construct the corresponding logical value, actually two-value logic here, for every premise and the conclusion of fuzzy rule f_j, which is denoted as Logic(f_j) = (L_{f_{i1}}, L_{f_{i2}}, ..., L_{f_{im}}, L_{f_{j1}}, L_{f_{j2}}, ..., L_{f_{jn}}). If x_{i} \in \text{or} \in A_i and x_{j} \in \text{or} \in B_j, then Logic(f_j) = (L_{f_{i1}}, L_{f_{i2}}, ..., L_{f_{im}}, L_{f_{j1}}, L_{f_{j2}}, ..., L_{f_{jn}}). Where if x_{i} \in A_i \text{and} x_{j} \in B_j, then L_{f_{ij}} = 1 \text{or} 0; and if x_{i} \in B_j \text{and} x_{j} \in A_i, then L_{f_{ij}} = 1 \text{or} 0. So if x_{i} \in A_{j} \text{and} x_{j} \in A_{j}, then L_{f_{ij}} = 0. If x_{i} \in A_{i} \text{and} x_{j} \in A_{j}, then the relation between L_{f_{ij}} and L_{f_{ij}} is the same. If B_{i} \cap B_{j} = \phi, then it is definite that L_{f_{ij}} = L_{f_{ij}}. Since there must not exist any exception if there is not any conflict and redundancy between different fuzzy rules, based on the discussion before, if only to validate the conflict and redundancy and B_{i} \cap B_{j} = \phi, the following assumption is reasonable and acceptable:

\begin{align*}
\text{Logic}(f_i) & = (L_{f_{i1}}, L_{f_{i2}}, ..., L_{f_{im}}, L_{f_{j1}}, L_{f_{j2}}, ..., L_{f_{jn}}) = \text{Logic}(f_j), \quad (2)
\end{align*}

It is easy to understand that we have the similar conclusion about f_j. Thus if we want to compare Logic(f_i) with Logic(f_j), it is just required to compare Logic(f_i) with Logic(f_j). Then the theorem with logical criteria to validate the conflict and redundancy of fuzzy rules are proposed as below:

**Theorem 1** If PI_j is the same with PI_i,

PI_fe = PI_fe, A_i \cap A_j = \phi, and

Logic(f_i) = (L_{f_{i1}}, L_{f_{i2}}, ..., L_{f_{im}}, L_{f_{j1}}, L_{f_{j2}}, ..., L_{f_{jn}}),

Logic(f_j) = (L_{f_{i1}}, L_{f_{i2}}, ..., L_{f_{im}}, L_{f_{j1}}, L_{f_{j2}}, ..., L_{f_{jn}}),

and if B_{f_{i}} \cap B_{f_{j}} = \phi, then L_{f_{ij}} \oplus L_{f_{ij}} \neq 0 , then:

**Criterion 1:** if L_{f_{i1}} \oplus L_{f_{i1}} + L_{f_{i2}} \oplus L_{f_{i2}} + ... + L_{f_{im}} \oplus L_{f_{im}} = 0

and L_{f_{ij}} \oplus L_{f_{ij}} = 0,

then f_i and f_j are redundant.

**Criterion 2:** if L_{f_{i1}} \oplus L_{f_{i1}} + L_{f_{i2}} \oplus L_{f_{i2}} + ... + L_{f_{im}} \oplus L_{f_{im}} = 0

and L_{f_{ij}} \oplus L_{f_{ij}} \neq 0,

then f_i and f_j are conflicting.

To expand the range of its application, two lemmas are proposed as below, which are easy to be proved.

**Lemma 1** If the intervals of the premises corresponding with the same indicator ‘pi’ involved in different fuzzy rules are the same and actually every indicator ‘pi’ is correlated with only one possible premise interval. It is also the same about the associated intervals of the conclusions. Namely if \( p_{i}^{f_{j}} = p_{i}^{f_{j}} \) and \( p_{i}^{f_{j}} = p_{i}^{f_{j}} \), then \( A_{i} = A_{j} \) and \( B_{i} = B_{j} \). And if the value of the premise and the conclusion has been logical value already such as two-value logic or multi-value logic, then the conclusion of the theorem, namely the logic criteria, can be used directly to validate the conflict and redundancy.

**Lemma 2** If \( PI_{j} \subset PI_{j} \), with the reasoning method mentioned above, the conflict and redundancy can be validated only based on the analysis of those premises associated with \( (p_{i}^{f_{j}} = p_{i}^{f_{j}}) \in (PI_{j} \cap PI_{j} = PI_{j}) \) and the conclusions corresponding with \( PI_{j} \) and \( PI_{j} \), because, in this case, fuzzy rule \( f \) may be contained inside fuzzy rule \( f \).

5. Performance modeling for investment guidance

5.1 Construction of model for investment guidance

Based on the theorem for conflict and redundancy validation of fuzzy rules presented above, the selection of performance indicators with the support of process model will be depicted in this sub-section.

As discussed in the third section, the structure of performance model is always a multi-indicators hierarchy and both the selection of indicators and the configuration of relationship between indicators involving the hierarchy are hard to analyze and determine. To solve these problems and take the related advantages of process model described before, multi-indicators hierarchy of performance model in this paper will be constructed principally aligned with ‘to be’ process model, and, what is important, the
hierarchy of performance model will be matched to the hierarchy of relevant process model.

Before the selection of performance indicators, it should be mentioned that the indicators base and the fuzzy rules base, which serve as the most important basis, are supposed to be built. The indicators base consists of two major parts, basic indicators base that comprises a large number of generic performance indicators and application indicators base that contains those popular indicators as well as the information of their usual locations in the process model. The fuzzy rules base has the same structure with indicators base, one part is about the generic fuzzy rules for choice and the other is about those frequently used fuzzy rules and relevant relationship between them and the process activities. These two bases are useful and powerful to manage the relevant knowledge for performance modeling and they will be getting more abundant and effective with the time going on.

As shown in Fig 2, the core of the selection of indicators is the related process model and actually what we do is to determine the right location for performance indicators in the process model, namely to put those adopted indicators into the appropriate process activities. In this case performance indicators associated with every process activity in different layers of the hierarchy of process model are selected and configured into the corresponding process activity. Because of the present weaknesses and deficiencies to establish the hierarchy of performance model independently without the support of other related models, in this paper the hierarchy of the process model is applied as the basic structure of related hierarchy of performance model. To make this selection process more reliable and dependable, it is encouraged to scratch indicators from various possible aspects, for there is not generally accepted list of performance indicators and fuzzy rules, and also the ones selected in those two knowledge bases are not definitely sufficient. Firstly it will be based on the indicators base and fuzzy rules base, especially the application indicators base and the application fuzzy rules base that can provide not only the popularly applied indicators and fuzzy rules but also the relationship between them and the process model, to make the full use of history experience and knowledge. Secondly, according to the concrete enterprise characteristics and requirements, fuzzy rules mining and simulation from referenced enterprise integration system and business process is getting more convincing and dependable with the rapid development of related research and technology in this area. It can provide sufficient and necessary fuzzy rules for special performance modeling and actually this step is the most important one to obtain discriminative performance model for different measurement and application objects. At last other corresponding aspects such as research and outcomes of the performance indicators and framework should also taken into account. By the way the enterprise requirements and expert knowledge are used respectively as the guideline and reference in the whole process of selection of indicators. With these steps, which indicator to be selected and which process activity they will be located in is confirmed. About the indicators adopted, it is important that only those favorable and reasonable indicators are supposed to be selected and each indicator can be put into different process activities in the hierarchy if it is necessary. In accordance with the discussion in the third section, indicators in the top layer of the process model mainly represent much width information, which are always qualitative, while those in the bottom layer depict much depth information, which are largely quantitative. In the performance modeling process, both qualitative and quantitative indicators are concerned and qualitative ones are decomposed into measurable and quantifiable quantitative ones from top to bottom layer until the total indicators in the bottom layer guaranteed to be quantitative with the associated decomposition of the process model.

About the selection of indicators, several things should be mentioned. Firstly, with the fuzzy rules and the relationship between them and process activities, it is easy to correlate the process activity with the performance indicators associated with fuzzy rules and then it is also easy to locate these indicators in appropriate process activity. What is more, the relationship between indicators of performance model hierarchy in this paper will be primarily correlated.
through a set of fuzzy rules. Secondly, the conflict and redundancy validation of fuzzy rules is indispensable and the theorem proposed above can be applied for this necessary step. Thirdly, those performance indicators and fuzzy rules adopted in the performance model at the last stage of our modeling process will be fed back respectively to indicators base and fuzzy rules base for the reuse in the future as in Fig 2.

Except the selection of indicators, to establish the relationship between indicators is another important part of the modeling process.

Fig. 3. Performance model for investment guidance

With reference to process context, related research, enterprise experience and expert knowledge, performance indicators in every process activity are respectively categorized into different indicators subsets according to the premises of fuzzy rules and also taking the aggregate characteristics and functions into account, as shown in the right part of Fig 3. With the associated context of process model, performance indicator subsets are adjusted appropriately if it is necessary. Then the relationship between the indicator subsets and fuzzy rules is checked up and if some indicator subsets are not related to any fuzzy rule, it need mine new corresponding fuzzy rules. To be clarified, the subsets can be formed in terms of any significative and meaningful combination. There are no completely identical subsets in the same process activity however different subsets can comprise several same indicators. Then considering the relationship between the premises and the conclusion of fuzzy rules and also taking into account those relevant knowledge mentioned before, the relationship between every subset in different child process activity and their parent indicators in the parent process activity is established respectively.

With the above steps, the original performance model is formed and it can be used to measure the fuzzy performance of implemented enterprise integration project based on certain fuzzy functions involving the fuzzy rules. The detailed method, which is considered as a post-assessment method, is omitted, because method proposed in this paper is about pre-assessment. However, the presented original performance model, which can reflect the effects of the implemented integration projects, can correspondingly be employed as an important support for the analysis of investment benefits. Detailely, the model for investment guidance will be constructed based on this original performance model with the following steps. At first, with the consideration of the intenseness of the casual relationship of fuzzy rules, every subset of each child process activity is assigned with a weight (W) through some appropriate methods (e.g. AHP) to show its relative importance in comparison with other subsets associated with the same parent indicator and in the same child process activity. Then, in the similar way, in each child process activity, the combination of subsets corresponding with the same parent indicator is also designated with a weight ($W'$) to show the relative importance of the subsets’ combination of this child process activity comparing with other associated subsets’ combination of other child process activity, which has the same parent indicator with the former one.

5.2 Non-linear programming model for investment optimization

As discussed in the introduction of this paper, there is not any entrepreneur and business manager to implement advanced IT technology and system without considering the return on investment, namely to make full use of the limited investment. In this subsection, based on the proposed model for investment guidance a non-linear programming model is formulated to optimize the allocation of investment and the related method is also proposed to prune those relatively unnecessary performance indicators.

First of all, a definition is proposed as below:

**Definition 1** ‘Informatization Pertinence (P)’ shows the degree of the pertinent relationship between the individual purpose of enterprise informatization and each performance indicator in the hierarchy. Namely it denotes that how much effect and benefits it can possibly get to realize the individual target of enterprise informatization, through the adjustment and improvement of each indicator in the related process.

The limited investment (R) here is mainly about investment budget for informatization; however other resources can also be taken into account. Let’s suppose that this limited investment is totally allocated to

J. Zhou, Q. Li, D. Xu, and T.
improve those indicators in the bottom layer. As discussed before, those indicators in higher layer are composed of and also can be reflected by the indicators in the bottom layer of the hierarchy. Technically the improvement of the indicators in the bottom layer will directly result in the relevant change of indicators in the higher layer without any additional actions and investment. So the use of the investment will actually be reflected totally by the indicators in the bottom layer and the assumption presented before is reasonable and acceptable.

With the above discussion, the methods to gather data are apparently required for those indicators in the bottom layer. According to Sinclair (1995), the methods for gathering data can usually be divided into three categories which are always applied together: observational methods; database methods; and subjective methods, such as questionnaire and interviews. Because each indicator in the bottom layer of the hierarchy is quantitative, the maximum investment requirement of the indicators will be relatively easy to be designated based on the methods to gather data and expert knowledge, etc. And the relationship between the ‘Informatization Pertinence (P)’ value of each indicator in the bottom layer and relevant allocated investment will be formulated as a non-linear function \( \phi \) as below. If any one of these indicators is allocated with enough investment \( R_j \), the maximum \( R_j \), in the possible interval of ‘P’ value, will be achieved, which will also be assigned with the similar methods to designate the \( R_j \).

\[
P_{j \times} = \frac{P_{j \times \times}}{R_{j \times}}, \quad 0 \leq R_{j \times} \leq R_{j \times \times}
\]

if \( R_{j \times} = 0 \), then \( \phi(\frac{R_{j \times}}{R_{j \times \times}}) = 0 \)

if \( R_{j \times} = R_{j \times \times} \), then \( \phi(\frac{R_{j \times}}{R_{j \times \times}}) = 1 \) \( \text{(3)} \)

Where ‘I’ denotes the bottom layer of the hierarchy, and if \( P_{j \times} \) is the j-th indicator of the bottom layer ‘I’, then \( P_{j \times \times} \) is the ‘P’ value of \( P_{j \times} \) and \( R_{j \times \times} \) is the possible maximum of the \( P_{j \times} \). Similarly \( R_{j \times} \) means the investment allocated to the indicator \( P_{j \times} \), and \( R_{j \times \times} \) is the maximum investment requirement of this indicator, namely the enough investment to realize its possible effect.

Afterwards, based on the ‘P’ values of indicators in the bottom layer, those relevant ‘P’ values of other indicators in the higher hierarchy can be calculated accordingly. The related formulas are shown in the following. In this process some appropriate adjustment of related value and cost is encouraged with the aid of experienced operator and expert. For example, if the ‘P’ value of an indicator in the top layer is unreasonable, the adjustment happens.

\[
\text{num}(S_{\text{ji}}) = \prod_{P_{\text{ij} \times \times}} P_{\text{ij} \times \times}
\]

This formula denotes the ‘P’ value, calculated by geometric mean, of the \( l \)-th subset \( S_{\text{ji}} \) of the \( C_{\text{ij}} \) and \( C_{\text{ij}} \) is the \( k \)-th combination of subsets of the indicator \( P_{\text{ij} \times} \). \( P_{\text{ij} \times} \) is the j-th indicator in the layer ‘I’.

\[
\text{num}(S_{\text{ji}}) \text{ implies the number of indicators in the subset } S_{\text{ji}}.
\]

Actually, as in Fig 3, each indicator \( P_{\text{ij} \times} \) in the parent process activity ‘PA’ is associated with several combinations of subsets \( C_{\text{ij}} \). Each \( C_{\text{ij}} \) is corresponding with a child process activity of ‘PA’ and every \( C_{\text{ij}} \) consists of several subsets of indicators in a relevant child process activity.

\[
\sum_j (W_{\text{ji}} \times \text{num}(S_{\text{ji}})) \prod_{P_{\text{ij} \times \times}} P_{\text{ij} \times \times}
\]

\[
\sum_j W_{\text{ji}} = 1
\]

The above formula is about the ‘P’ value of \( C_{\text{ij}} \) and \( W_{\text{ji}} \) is the relevant importance weight subset \( S_{\text{ji}} \).

\[
\sum_j (W_{\text{ji}} \times \text{num}(S_{\text{ji}})) \prod_{P_{\text{ij} \times \times}} P_{\text{ij} \times \times}
\]

\[
\sum_j W_{\text{ji}} = 1
\]

This formula denotes the ‘P’ value of the indicator \( P_{\text{ij} \times} \) and \( W_{\text{ji}} \) is the relevant importance weight of the \( C_{\text{ij}} \).

Because the selected performance indicators are usually numerous, it always make the model complex and every \( C_{\text{ij}} \) consists of several subsets of indicators in a relevant child process activity. Through the above discussion, the methods to gather data are apparently required for those indicators in the bottom layer. According to Sinclair (1995), the methods for gathering data can usually be divided into three categories which are always applied together: observational methods; database methods; and subjective methods, such as questionnaire and interviews. Because each indicator in the bottom layer of the hierarchy is quantitative, the maximum investment requirement \( R_j \) of those indicators will be relatively easy to be designated based on the methods to gather data and expert knowledge, etc. And the relationship between the ‘Informatization Pertinence (P)’ value of each indicator in the bottom layer and relevant allocated investment will be formulated as a non-linear function \( \phi \) as below. If any one of these indicators is allocated with enough investment \( R_j \), the maximum \( R_j \), in the possible interval of ‘P’ value, will be achieved, which will also be assigned with the similar methods to designate the \( R_j \).

\[
P_{j \times} = \frac{P_{j \times \times}}{R_{j \times}}, \quad 0 \leq R_{j \times} \leq R_{j \times \times}
\]

if \( R_{j \times} = 0 \), then \( \phi(\frac{R_{j \times}}{R_{j \times \times}}) = 0 \)

if \( R_{j \times} = R_{j \times \times} \), then \( \phi(\frac{R_{j \times}}{R_{j \times \times}}) = 1 \) \( \text{(3)} \)

Where ‘I’ denotes the bottom layer of the hierarchy, and if \( P_{j \times} \) is the j-th indicator of the bottom layer ‘I’, then \( P_{j \times \times} \) is the ‘P’ value of \( P_{j \times} \) and \( R_{j \times \times} \) is the possible maximum of the \( P_{j \times} \). Similarly \( R_{j \times} \) means the investment allocated to the indicator \( P_{j \times} \), and \( R_{j \times \times} \) is the maximum investment requirement of this indicator, namely the enough investment to realize its possible effect.

Afterwards, based on the ‘P’ values of indicators in the bottom layer, those relevant ‘P’ values of other indicators in the higher hierarchy can be calculated accordingly. The related formulas are shown in the following. In this process some appropriate adjustment of related value and cost is encouraged with the aid of experienced operator and expert. For example, if the ‘P’ value of an indicator in the top layer is unreasonable, the adjustment happens.
influence of every subset to the ‘P’ value of $C_{k,j}$ as well as the influence of every combination of subsets to the ‘P’ value of indicator $P_{i,j}$ can be classified. Taking into account the expert knowledge and the requirements of enterprises, the relatively unimportant subsets or combinations of subsets can be pruned from the hierarchy. Then those related indicators are cut down correspondingly. The pruning will consider the parent indicators one by one in the same layer and it will execute from top to bottom, which makes it easier and faster.

At last the non-linear programming model is formulated to optimize the allocation of investment of indicators in the bottom layer with the limitation of investment and the purpose of maximizing total ‘P’ value of the model for investment optimization.

\[
\begin{align*}
\max & \sum_{j} \left( R'_{i,j} \times \phi \left( \frac{R_{i,j}}{R'_{i,j}} \right) \right) + \\
\sum_{j} & \sum_{i} \sum_{\phi} \left( W_{C_{i,j}} + \sum_{j} \left( W_{S_{i,j}} \times \sum_{S_{i,j}} \left( \prod_{P_{i,j} \in S_{i,j}} \right) \right) \right) \\
\sum_{j} R_{i,j} & \leq R_{\text{max}}
\end{align*}
\]

(9)

Here $R_{\text{max}}$ denotes the ceiling of the investment possibly provided. And if every indicator in the bottom layer is allocated with either enough investment or nothing, this non-linear programming will be actually the 0-1 integer programming. Applying a typical method for this mathematic problem, we can then get the desirable result.

6. Conclusions

To complement the research of economic view, the performance modeling method for investment guidance is put forward, which is actually a pre-assessment method. In this proposed method, amounts of related knowledge, information and data are employed and researches and outcomes in various corresponding realms are also made full use of. Because of the natural advantages of fuzzy rules, the numerous dynamic and subjective indicators of performance can be considered effectively. In addition, based on knowledge mining and management, our research actually intends to make performance measurement computer processible and the presented method won’t be too complex to apply.

Grounded on the advantages of our method, every concrete performance model for investment guidance will be constructed with the requirements and the characteristics of measured individual object. It can be used to optimize the necessary investment based on the possible influences and benefits of enterprise integration as well as to improve the correctness of investment decisions. Therefore it is a more convincing support approach to carry out the strategy of enterprise effectively.

Because of the limitations of research in this area, there are still many problems to be solved and some of them are close and subservient to our work with good research prospects, such as the further computer aided performance analysis as well as the knowledge mining and management to take more advantages of IT.

7. References

A Methodology and Conceptual Framework for Flow-Manufacturing-Oriented ERP Systems

Jianfu Zhang\textsuperscript{1,2}, Zhijun Wu\textsuperscript{1}, Ken Chen\textsuperscript{1}, Pingfa Feng\textsuperscript{1}, and Dingwen Yu\textsuperscript{1}
\textsuperscript{1}Department of Precision Instruments and Mechanology, Tsinghua University, Beijing, 100084, P.R. China.
\textsuperscript{2}School of Mechanical Engineering, Shenyang Ligong University, Shenyang, 110068, P.R. China.
E-mail: zhangjf05@mails.tsinghua.edu.cn

Abstract

This paper proposes an activity-based modeling methodology and conceptual framework for an ERP system that oriented to flow manufacturing enterprises. The proposal that activity theory might serve as a model or theory for business process modeling remains an open question. Its purpose is to explore the use of the Activity Theory as a modeling technology to illustrate the proposed methodology. The advantage of this modeling methodology is easy to integrate and extend the especial requirement of the implementation of the middle-sized flow-based enterprises. In this paper, by analyzing the components of an activity and the relationship of the business process modeling, several basic concepts are given and the method of the process modeling is also expressed. Then, a conceptual framework for a flexible integration and extensible ERP system is studied through communicating among different activity units. As a case study, the paper also discusses the process of the information exchange and interface working between the different modules.

1. Introduction

Over the past few years, flow manufacturing enterprises have been undergoing a thorough transformation in reaction to challenges such as globalization, unstable demand, and mass customization [1]. Scientific and technological progress has become a decisive factor for the development of the enterprises. There are desperate needs to efficiently improve the flow speeds, shorten delivery times, and reduce costs to deal with shortened product life cycles and the requirements by the international market place. Hence, a well-founded modeling technique and an efficient information management tool are required to help enterprises to cope with rapid changes to ensure survival and growth.

As process is usually defined as “a set of activities arranged in a specific order”, many different techniques can be used for modeling business processes in order to give an understanding of possible scenarios for Improvement (Eatock et. al, 2000). These models, usually created by graphical modeling tools, including Petri Net, EPC, RAD, UML and IDEF3 etc., have significant advantages on simplicity and descriptive ability [2]. However, business process modeling is not limited to process diagrams. They fall short in analyzing capability to assist enterprise users with process designing.

This paper first analyses the characteristics and requirements of the flow manufacturing enterprises. Then, based on the Activity theory, through analyzing the components of an activity and the mediating mode between subjects and objects, several basic concepts are given and the method of processes modeling is also expressed. The proposal that activity theory might serve as a model or theory for business process modeling remains an open question. Its purpose is to explore the use of Activity Theory as a modeling technology to illustrate the proposed methodology. At last, as a case study the research also discusses the process of the information exchange and interface working between the different activity units.

2. Characteristics and requirements of the flow manufacturing enterprises

In the current e-business environment, with the evolvement of industrial structure, the organization and management mode of enterprises would certainly
be changed accordingly [3]. And different enterprises have big differences in production and management practice. The implementing plan must consider these differences and have adaptable solutions to fit the firm's conditions [4]. Especially, for the implementation in the flow manufacturing industries, which has its own special characteristics, much commercial ERP software such as SAP, Oracle, UFERP and several other ERP systems are also hard to fulfill its especial requirement. The characteristics of the flow and process in the flow manufacturing enterprises are:

- Straight and short product flow patterns
- Make to order
- Plans are simple and relatively stable
- Production capacity of the equipment is fixed
- Single-breed production
- Just-In-Time/pull/dependent demand scheduling
- Highly flexible and responsive processes
- Continuous flow work cells
- Collocated machines, equipment, tools and people
- Empowered employees

Although ERP has specific features, its application modules vary with system suppliers. The key factors to determine the success of ERP implementation are the use of proper modules, the performance evaluation, and the improvement during the process [5]. The flow manufacturing is in direct opposition with traditional mass production approaches characterized by use of economic order quantities, high capacity utilization, and high inventory. It is fundamentally different from traditional "push" style manufacturing. The demand-based "pull" of material through production contrasts with the traditional "push" production process, driven by a schedule that pushes inventory into stocking locations may not reflect customers' requirements. Basically, products are made to customer demands and production is based on daily schedules to identify the product mix for that day. TQM is implemented as part of each process in the line.

In this study, an investigation aimed at the middle-sized flow manufacturing enterprises has conducted to collect information on the performance and action of implementing ERP. And with the evolvement of the industrial structure, the organization and management mode of enterprises are changing accordingly. The flow-based industries are now in the evolvement process of changing from traditional industries to modern high-tech industries. Just based on the feature of the flow manufacturing enterprises proposed above, this paper applies itself to explore the use of Activity Theory as a modeling technology and propose a new conceptual framework to fulfill the especial requirement of the systems implementation.

3. Modeling methodology

3.1. Introduce of activity theory

Activity theory developed from the ideas of the Russian psychologist Vygotsky (1896–1934) and his successors [6]. More recently, Engestrom [7], has extended these ideas to include a model of human activity and methods for analyzing activity and bringing about change. The flavour of activity theory employed here draws primarily upon the contemporary work of Engestrom, which has been adopted and elaborated by many Scandinavian [8].

Activity theory begins with the notion of activity. An activity is seen as a system of human “doing” whereby a subject works on an object in order to obtain a desired outcome. In order to achieve this objective, the subject employs tools that may be external or internal [9]. Central to activity theory is the concept of mediation. The relationships between subject, object and community are mediated by tools, rules and distribution of work. These artifacts are used by a community to achieve a desired set of transformations on an object. Artifacts can range from physical tools, such as hammer, to psychological ones, such as languages, procedures, methods and general experience. The basic model of an activity is illustrated by Engestrom in Fig.1 [10].

![Fig.1. The components of an activity](image)

Although, the proposal that activity theory might serve as a model or theory for business process modeling remains an open question. Our motivation stems from the observation that in recent years a number of prominent researchers have suggested that activity theory might serve as a framework or a methodology for information systems. Reflecting this, Engestrom has formulated three basic principles, building on the work of earlier activity theorists, which are widely used and cited within the activity theory community. These are, in no particular order,

1) Activities are the smallest meaningful unit of analysis;
2) The principle of self-organizing activity systems driven by contradictions;
3) Changes in activities (and by extension the organization hosting them) as instantiations of cycles of expansive learning.

3.2. Advantages of the activity-based modeling method

Business processes are a set of logically related tasks performed to achieve a defined business result. A set of processes acting on data forms a business system - the method in which an organization carries out its mission. Although we rich in technologies to solve the isolated business problems, such as product scheduling, quality control, production planning, accounting, marketing, many other problems still reside within enterprise. To solve these problems require an integrated approach. On the other hand, most process modeling work has focused on the evolutionary behaviour of artifacts through the various developmental stages rather than on the behaviour of those creating the artifacts. This restricted approach ignores the fact that the ever expanding range of computing and communications technologies plays only an enabling role in the development of the systems eventually used to achieve a user has defined business objectives. Moreover, it was readily apparent that many exiting approaches provided excellent facilities for capturing “hard” data associated with a processes, while the “softer” factors are not recorded. Most underlying models do not have a place for the motives or goals that people might bring to bear in carry out a process.[11].

Activity theory provides not only a rich descriptive tool that is useful analyzing and understanding modeling work in general, but also a number of useful concepts that can be used to address the lacuna in process modeling discussed above, especially the lack of expression for “softer” factors. One such concept is the internal plane of actions. Activity theory recognizes that each activity takes place in two planes: the external plane and the internal plane. The external plane represents the objective components of the action while the internal plane represents the subjective components of the action. In addition, the concepts of motives, goals and conditions discussed above also contribute to the modeling of soft factors. The basic unit of the analysis in activity theory is a human activity. An activity, as defined by activity theory, provides enough contextual information to make an analysis meaningful, while avoiding either a narrow focus on an individual or a too broad focus on a whole social system.

Activities transform objects via a process that typically has several steps or phases. Chains of actions guided by a subject’s conscious goals carry out an activity over a small team, resulting in objective results. Actions themselves are realized through series of operations which are performed automatically in non-conscious fashion. There is no a strict one-to-one relationship between activities, action, and operations, one activity is achieved through many actions and one single action may be part of multiple activities. The situation is analogous for operations: an action may be carried through one more operations, and a single operation may be part of multiple actions.

Additionally, coordination, cooperation and co-construction are three different interactive levels of subject-object-subject relations in business processes. The “subject-object-subject” combines object-subject and subject-subject aspects of an activity. The former refers to the instrumental aspects of an activity, while the latter is known as the communicative aspects of an activity [12]. These three levels represent an instantaneous status of an activity.

3.3. A modeling methodology

Basic to process modeling is the informational component and the interaction between process and data. The process component specifies what processes are to be performed to carry out the business function using the information from the data component. The data component (i.e. documented by a data model) specifies the informational needs of the business. Together, the interaction provides a context in which clarity, completeness and quality can be analyzed, leading to an effective design which meets business objectives.

A Process is a sequence of actions or operations to achieve a particular goal. Comparing with the cooperativity of the process, activities emphasize their independence. And activities are used to describe an independent object, which may be a certain business, task or procedure. Similar to the process, activities also can be seen as the conversion from input to output [13].

To describe the method of the process modeling more clearly, the basic definition about the activity, activity unit and process meta-model are as follows:

**Define 1:** *Activity* is the basic element to form the meta-model. Based on the model illustrated in Section 3.1, the definition is as follows:

\[ A=\{T, S, 0, Ru, C, D, 0u\} \]

Where: \( A=\text{Activity} \),
\( \text{T= Tools()} \),
\( \text{S= Subject()} \),
\( \text{O= Object()} \),
\( \text{Ru=Rules()} \),
\( \text{C=Community()} \),
D=Division-of-labour(),
Ou=Outcome().

Define 2: Activity unit consists of four parts: input of unit, output of unit, attributes space of unit and constraint rules, the definition is as follows:

\[ Au = \{ I, Op, AuS, Cr \} \]

Where: \( Au \) = Activity unit,
\( I \) = Input,
\( Op \) = Output,
\( AuS \) = Attributes space,
\( Cr \) = Constraint rules.

The components of the activity unit are shown in Fig.2.

The components of the activity unit

Define 3: The process meta-model consists two parts, the relation represent as follows:

\[ P = \sum Au_i \cup R(); i=1,2,3... \]

Where: \( P \) = Process meta-model,
\( Au_i = \{ A[i] \} i=1,2,3... n,n \in N,n>1 \), \( n \) is the number of the activity in a process.

And \( R() \) named process attribute structure is the relation set among the \( Au_i \) in a process, it is an abstract of the relationship between the different \( Au_i \) that is based on the different attributes, such as time, cost, resource, etc. Here, we define it as attribute gene and use \( x \) to denote it.

In the given environment, based on one of attributes, if \( Au_i \) and \( Au_j \) has relationship: \( R(x)(1 \leq i \leq n, 1 \leq j \leq n, i \neq j, n>1) \), The process attribute structure is defined as follows:

\[ R(x) = \{ R(x)(1 \leq i \leq n, 1 \leq j \leq n, i \neq j, n>1) \} \]

Based on analysis above, an activity unit is a process that is composed of a sequence of different hierarchical activities. The structure of the activity unit is shown in Fig.3.

It is a mechanism for structuring design activities so that they can be reviewed and analyzed at appropriate abstract levels. It does not map directly to a single low-level design activity in the real world.

An activity object is an object at the bottom of the process decomposition hierarchy. Each Activity object represents a single “low level” design activity in the real world. Constraints represent a restriction or a boundary condition that governs the behaviour of an activity in a modeling process. To support the refinement of an exiting process model for a particular application domain without modifying the original process model, processes can be linked to each other through a specialized relation. A process object is also directly connected to other process objects through the precedence relations. The precedence relations define and control the sequencing of the process. The structure of the process meta-model is shown in Fig. 4.

The structure of the process meta-model

The system modeling process is carried out through communication among different module. Each module itself is knowledge database and it can not only solve the problems of its own area independently, but also exchange knowledge with other module in order to solve the problems of other areas by three different interactive levels. Module is an active groupware with the ability of processing information, whose attributes and methods can be described by using Semantic Web. The relationship among different modules is shown in Fig.5.
The modeling method is a tool for storage and manipulation of process models that facilitate the creation of a software system and act as artifacts for the software development activity. Such artifacts can represent, to a limited degree, the procedures and rules that mediate a development effort, as well as some aspects of the division of labor. By analyzing the modeling features according to a framework based on Activity Theory and their corresponding levels of activity, we are interested in characterizing the enterprise modeling process according to their ability to cope with the individual demands of the flow manufacturing enterprises.

4. An architecture and conceptual framework

Based on the analysis above, we can classify the module as seven important properties: coordination, cooperation, co-construction, autonomy, extensibility, social ability and reactivity. These properties in fact also are inherited by an activity unit or a single activity. They allow activity, activity unit or module to interact and communicate with other activities, activity unit or module via various activity communication languages. And different function modules can be formed through the method discussed above.

In this research, aimed at the characteristics and the specific requirement of the flow manufacturing enterprises, the proposed conceptual framework is composed of five basic modules: (1) interact module; (2) interface module; (3) user interface module; (4) business module; and (5) process manager module, and it also can be seen a general architecture for the continuous flow manufacturing industries. We propose that the last three system modules (the user interface module, the business module and the process manager module) within each functional area interact through the interaction module or the interface module. In the conceptual framework of the ERP system, each business module can use the interaction module managed by the other business module. The user interface module serves as a communication tool between the user and the ERP system, and there are several business modules and the process manager module that perform specific assignments related the actual department over the enterprise’s network. The business module may be marketing, distribution, materiel, stock, manufacture, equipment, quality, and accounting. The conceptual framework of the ERP system is shown in Fig.6.

Generally, the system development team consists of multiple remote designers and managers. Each designer works on more than one project while each manager typically manages multiple projects. The methodology we propose above is recognized as a key enabler of design process improvement and cycle time reduction. With advices in wide-area networks, distributed object management methods, client-server architectures, and Web-based user interface, design process management within a wide-area heterogeneous computing environments has become entirely feasible.

4.1 Process manager module

The process manager module provides the major capabilities for the conceptual framework system. These capabilities include dynamic product-process instantiation, process execution, project coordination, progress tracking, process monitoring, performance measurement, as well as value computation and process history-related data collection. It is the responsibility of the process manager module to create and execute a project based on a customized, dynamic product-process model instance. It serves the various business modules for exchanging information through the web-based environment. At the same time, the process manager module controls the access to the project database. The project database provides a repository for information about individual business project. This information includes customized product and process descriptions, process and progress management information, and process history data.

On the other hand, the process manager module is also charged with the duty to retrieve information requested by its own interaction module; query specific database within the department; perform data
warehousing and prepare data set upon request from interface module; and identifies invalid data and missing values so that the data is complete and applicable when being returned to the interface module.

Thirdly, this module provides the application logic for modeling, customizing, and tailoring processes and products. It manages access to component process/product models that are stored in the project database, and supports concurrent manipulation of these component product-process models by a term of users.

The key capabilities and characteristics of the process manager module includes: (1) Design process, definition and verification. (2) Just-in-time availability of roles and tools. (3) Flexible entry into the design process, for example, goals, plan, tools and data. (4) Communication of design progress on an ongoing basis to the project manager. (5) Online design progress status reports. (6) Replay of design process history.

4.2 The interaction module

The interaction module is the bridge of different business modules and is the controller of the other modules within the department and the user interface module. The interaction module possess three properties when communicating with other business modules, they can be represent as coordination, cooperation, co-construction. Depending on the nature of business complexity, it can have one or all of the three properties. With their domain knowledge, the interaction modules have the ability to monitor, communicate, and collaborate with other modules, react to various requests, as well as assign tasks to proper business module.

The other major responsibility is used to supervise and control the situation of the other business modules. According to the need of each module, it examines the business module whether to be placed in work appearance or not; receiving instructions and reporting to the human user through an interface module; assigning tasks to and receiving feedback from business modules; and communicating with and providing requested information to other modules. At the same time, it is also used to detect the possible emergence's interference and unexpected exceptions, ultimately getting the solution on time.

4.3 The business module

The business modules usually possess flexibility and can act autonomously within their own domain knowledge without the intervention of the interaction module. For example, a business module that is assigned to monitor price change would go to and stay in a supplier’s site to monitor the supplier’s price and report any price change crosses given threshold values with or without the instruction from its business module. The number of business modules varies by the number and complexity of tasks within a department. The functions of a business module may also vary from department to department depending on what needs to be accomplished.

In general, the responsibilities of a business module include: receiving data or information from the interaction module; performing data and information analysis by running specific program or algorithm; and then reporting the results back to the interaction module.

4.4 The interface module

The interface module provides the interface function for exchange data or information between the business module and the process manager module. It involves dynamic instantiation of process flows and subject decomposition, tracks and controls the process progress. The interface module possesses the ability to learn and store preferences of users and the ability to monitor and inform users when tasks have been completed without the inquiry of users. With enhancement, the interface module may observe and record the user’s disposition to follow the recommendations of interaction module and invoke machine learning to determine many of the user’s preference.

The primary responsibilities of an interface module include: communicating between process manager module and business module; interpreting results; following the guidance of the enterprise tasks; continually expanding and refining the tasks; and invoking system design tools. It provides logical and collaborative workspaces for system user to perform business role by following the instantiated process model.

4.5 The user interface module

The user interface module is used to manage different roles who participate in the process of system operation. The roles include: principle engineer, chief, engineer, designer, drafter, market researcher, warehouse manager and system manager, etc. These individuals have the responsibility to define/create process descriptions that are compatible with their respective tasks, relevant technologies, and requisite levels of specificity.
On the other hand, the user interface module provides the user interface to define, customize, and tailor component process reference modules, which can then be instantiated to support a particular enterprise process instance. It supports java-enabled, form-based model definition and customization that does not require the user to learn any particular notation or modeling construct.

5. An illustration of the proposed conceptual framework

The process is carried out through communication among different basic modules composed by activity units. The business processing retrieval requires more interactions to specify user’s requirements and give satisfied response. According to the various modules, different modalities should be integrated to fulfill the business task. The main problems here include:

1) How to differentiate and analyze the user’s requirements?
2) How to transform the user’s input into business processing?
3) How to give effective response efficiently?
4) How to demonstrate the output information or data to the user friendly?

In order to illustrate the proposed conceptual framework, the process of the information exchange and interface working between the different modules will be explained as follows. Fig.7 presents a typical operation workflow of a flow-based enterprise.

**Fig.7. Workflow of the flow-based enterprise**

For simplicity, we only consider the main operation workflow and assume that there are five departments: the marketing, the production, the stocking, the inventory, and the consignment. Each department corresponding to one business module and we named them as Mm, Pm, Sm, Im and Dm. And we also assume that one interaction module between the marketing module and the production module is named M-P Iam; the others respectively are P-S Iam, S-I Iam, I-D Iam and D-M Iam. Due to the complexity of business processing, for each business module, there is one interface module connect to the process manager module. At the same time, we farther assume that one interface module between the marketing module and the process manager module is named M IIfm; the others respectively are P IIfm, S IIfm, I IIfm and D IIfm. And the information server, database, and data architecture of each of the business module are integrated into the process manager module. Fig. 8 illustrates the information exchange and interface working process between the different modules.

**Fig.8. The process of the information exchange and interface working**

Now, we assume that a district chief in a marketing department needs to determine if the company can accept an order from product agent A of city C for N units of product M by Friday? At first, the Mm will communicate the above question through the M IIfm to coordinate the process manager module. By exercising server, process manager module will organize three activities concurrently: a) communicate with the Im by I IIfm to obtain the current inventory information of product M; b) communicate with Pm by P IIfm to obtain the status of the production power; c) communicate with Sm by S IIfm to query the information of the material storage. Then, the process manager module will differentiate and analyze the requested information from various modules, and return the result of the process to the Mm.

Upon receiving instructions from the process manager module, the following activity is executed simultaneously in different departments: a) in the marketing management department, the Mm will form the requirement planning based on the feedback, by the M-P Iam, the information of the requirement planning will transform to the Pm; b) the Pm then will inquiries...
the inventory information by the interaction module P-I IAM and make the producing planning; c) based on the principle of the lowest storage, the Im, then, will calculate the required material to fulfill the N units of product M of the order, at last, form the material requirements planning and transform the data to the Sm by the I-S IAM; d) if the current material cannot satisfy the requirement, the stocking department will communicate to the supplier, otherwise, provide the material for the production department and wait for the medicine to carry through batch packing; e) in the end, before distribution, the consignment department will wait for the instruction coming from the marketing, stocking and the process manager module, if the sum of the produce time, packing time and the delivering time is less than the required time, that is to say, satisfy the dead time Friday, then the marketing department can accept the order and the consignment department will deliver the medicine M on time. If all of above activities are fulfilled, the process will be terminated.

6. Conclusion and future work

Increasing the exchange of information and communication between upstream and downstream department in the flow manufacturing enterprises are now crucial to expedite the flow, increase value, reduce flow cost and enhance overall response capability and competitiveness. Hence, an efficient information management tool is required to help enterprises to cope with rapid changes to ensure survival and growth. This paper first analyses the characteristics and requirements of the flow-based industries. Then, based on the Activity theory, by analyzing the components of an activity and the mediating mode between subjects and objects, several basic concepts are given and the method of processes modeling is also expressed. And the application of the activity-based modeling technology in business process modeling is also reviewed. An architecture and conceptual framework is proposed, and it is feasible, extensible and can be used in the flow manufacturing enterprises. At last, we demonstrate that the proposed ERP system is the solution by enhancing efficiently at a low cost and increasing overall performance.

Further research is needed to extend the current work and to demonstrate its practicability. Secondly, a complete set of ERP countermeasures and a performance analysis model are required to enable high-level management to support the implementation of the ERP system. Thirdly, based on the above research work, a flow-oriented system implementation will be developed to validate the correctness of the methodology.

References


J. Zhang, Z. Wu, K. Chen, P. Feng, and D. Yu
Concepts for an Ontology-centric Technology Risk Management Architecture in the Banking Industry

Colin Atkinson  
Chair of Software Engineering  
University of Mannheim  
68163 Mannheim  
Germany  
atkinson@informatik.uni-mannheim.de

Christian Cuske, Tilo Dickopp  
Chair of Information Systems  
University of Mannheim  
68163 Mannheim  
Germany  
cuske@wifo.uni-mannheim.de  
dickopp@wifo.uni-mannheim.de

Abstract

Due to the increasing impact of technology on the banking industry, the resulting risks play a vital role in bank-wide risk management. Additional gain in attention can be ascribed to the recent Basel II regulatory framework covering operational risk. To deliver adequate support for the process of technology risk management, flexible and seamlessly integrated software architectures are required. Based upon ontology-driven information systems the concept of an ontology-centric technology risk management architecture is explored. As a basis, the conceptual role of ontologies in information systems is portrayed and the technology risk environment in the banking domain summarized. The proposed architecture then is constructed around two essential concepts: The modeling of domain knowledge within an ontology as well as the integration of multiple, special-purpose software components into one architecture.

1. Introduction

The dependence of financial service companies on information technology is constantly rising [35]. Striking examples are the spreading of Internet-based services or the automation of trading and settlement activities [25]. Moreover, a reliable and scalable IT-infrastructure is increasingly seen as a competitive advantage for banks operating in a cost-aware environment. Naturally, the greater the influence of IT on an enterprise's economic success the greater the associated risks [3]. The understanding and handling of operational technology risks has thus become a vital part of the bank-wide risk management process, following market and credit risk. As a consequence the 2004 Basel capital adequacy framework integrates technology risk as part of operational risk into

the regulatory requirements [5]. These risks comprise system failures, technology-related security issues, systems integration aspects or system availability concerns [4].

As a result, the need for tailor-made information systems that fully support the management of operational technology risk has become more evident [9]. Two major challenges therefore have to be met: Firstly, the system will need to model operational technology risks from within every relevant activity of the enterprise. This requires an appreciation of the concept of technology risk itself as well as its relation to the entire IT-environment. Secondly, the system will be required to circumvent the vast variety of existing risk management methodologies in order to be fully accepted. Most existing risk management architectures in use today lack at least one of these fundamental requirements [9]. It can also be seen as problematic, that existing systems might be built around models but do not share the same understanding of the domain [17].

The idea of applying formal ontologies has been proposed as one way of fostering the understanding of technology-related risk in the banking industry [13]. Consequently we follow this approach to develop the functional and technical fundamentals for an ontology-centric architecture satisfying the relevant requirements. In the following sections we first give the conceptual background of ontologies in information systems. Subsequently, the business-environment of technology risk management in banking is clarified and the most outstanding challenges outlined. The proposed architecture is then introduced, especially the aspects of model building and component integration.
2. Ontologies in Information Systems

The primary use of formal ontologies in information systems is seen in the support for knowledge acquisition and modeling, leading to so-called knowledge-based systems. One common definition of formal ontologies in this sense is the following: “An ontology is an explicit specification of a conceptualization” [18]. With the vision of the semantic web [6], the application of ontologies has been extended to build the conceptualization of a domain and technical model at the same time. This leads to the more pragmatic interpretation captured by the following definition: “Ontologies have been developed to provide machine-processable semantics of information sources that can be communicated between different agents” [15].

Formal ontologies are able to deliver several advantages in the context of information systems [32]. Their support for enhanced communication relates to the capability of ontologies to reduce the terminological confusion within or between organizations. They not only provide unambiguous definitions for the terms used but also allow for the integration of individual perspectives. Their support for interoperability of software tools is achieved through the translation of differing representations and paradigms into an integrated model. A third advantage can be found in the engineering of software. Ontologies can be used to help identify and specify requirements. Furthermore, they can enhance the reliability and reusability of software systems.

For this paper the focus lies on the communication and interoperability characteristics of ontologies. The advantages concerning the process of software engineering are thus neglected here. Summarized, the following aspects of ontologies are seen as the most relevant enablers for the proposed ontology-centric risk management architecture:

- Their support for Communication subsumes a shared understanding of the problem domain. This includes an accepted definition of all relevant terms as well as important coherences with the enterprise environment.
- Their support for Interoperability covers the facets of integrating multiple software components relying on individual models into one architecture. This means coupling the different views into a shared model.

3. Technology Risk in Banking

To tie in with the conceptual background presented in the last section we now turn our attention to the application environment. Based on the advantages of ontologies in the context of information systems, the technology risk specific requirements for the banking industry are enumerated. This covers the need for an abstract definition of the risk as well as the procedural particularities.

3.1. Risk Definition

The general understanding of technology risk is embedded in the Basel definition of operational risk: Operational risk is defined as the risk of loss resulting from inadequate or failed internal processes, people and systems or from external events [5]. The term technology risk is commonly [34] used instead of Basel’s “system risk”. This indicates the broader meaning and additionally captures risks triggered by the technical infrastructure or IT-related projects.

The term is refined further in various approaches. Possible refinements include enumerations of potential threats [4], categorizations into further sub-elements [34] or standardizations of security-relevant characteristics [20]. A detailed analysis of the different understandings is beyond the scope of this paper. However, it is possible to identify a common set of core elements that form the basic conceptualization of operational technology risk. Commonly the risk affected assets are further sub-divided into categories such as basic IT-entities, complex systems or dynamic aspects which depend on each other. Furthermore basic properties which guarantee the security of information systems are included. It is important to take into account the degree to which the concept of risk is widened however, because overlapping with the other operational risk categories must be avoided. Another important aspect is the Basel classification of potential losses from operational risks. Here one can identify three areas related to technology risks [5]:

- External Fraud – Systems Security
- Business disruption and system failures
- Execution, delivery and process management – Transaction capture, execution & maintenance – Model / system misoperation

As briefly discussed one challenge for the technology risk management architecture clearly lies in the construction of a shared model. This needs to cover the understanding of technology risk as well as integrate the regulatory approach for operational risk.
3.2. Risk Management Process

To structure the potentially wide area of operational risk management many recommendations can be found in the literature. What most approaches have in common is a process-like character of risk management [35]. This is also proposed by the Basel framework [4]. In the sense of this paper, risk management is organized as a feedback control system divided into different phases. The separation of the phases is not distinct throughout the literature, though as a minimum scope three sub-steps can be identified as depicted in Figure 1. The identification serves as the basis for a sound management of risks. It covers the revelation and analysis of the relevant risks. Furthermore a classification might be necessary to structure the gathered information. The quantification of the identified risks is subsumed in the measurement phase. This targets the frequency and extent of the potential loss either from a qualitative or a quantitative perspective. Possible counter-measures to the quantified risks are defined in the controlling phase. This comprises the mitigation, reduction or transfer of risk. Additionally back testing is seen as part of this phase, to continuously validate the risk management process itself.

In the implementations of the above process many different methods are to be found [10]. On the one hand this is due to the varying challenges presented by each phase in the above process. On the other hand it is due to the history of operational risk management inside banks which has lead to individual preferences for certain approaches. In general single methods are usually best suited for specific phases, but do not cover the entire process [35]. Some of the most accepted methods today are self-assessment [27], model building [30] and loss databases [2]. Additionally the regulatory approaches must be included to fulfill the Basel requirements.

The most common methodology for targeting risk identification is self-assessment [27]. This enables the bank to focus on the insights of risk experts. This approach can be criticized with respect to subjectivity [29], especially concerning quantitative aspects such as likelihood. For the measurement of operational risk, procedural models, factor or actuarial approaches are preferred [30]. Loss databases [2] and modelling approaches [23] seem the most promising for technology risk. The controlling phase covers the mitigation or insurance of risks [25] as well as back testing the measurement results. Furthermore, compliance with the Basel II requirements must be assured through the application of the regulatory approaches. Figure 2 subsumes the most relevant approaches and lays out their applicability throughout the different phases of the risk management process. Without comprehensively comparing the approaches, one can say that none of the above methodologies fully covers the entire process. Therefore an information system must integrate a hybrid combination of the above approaches to support the complete risk management process.

4. Risk Management Architecture

Constructing a technology risk management system presents two major challenges: The risks have to be fully understood and the collaboration of the components implementing the different methodologies must be enabled. Those targets can also be understood as an integral part of the Basel II requirements, where a deep and bank-wide risk understanding [5] as well as a process-oriented management [4] is postulated. In the following sections the basic concept for an ontology-centric architecture is proposed to fulfill both demands.

In the following sections the architecture of a technology risk management system is described. The discussion covers relevant aspects of the model, basic principles of the applied methods, and implementation details of the prototype: At the centre of the proposed architecture lies the enterprise model, which in our case is resembled by the technology risk ontology. The different
methods used throughout the risk management process are centrally bound to the enterprise model. The realization in the object-oriented prototype is customized to at the same time meet the requirements of the individual methods and to also ensure compatibility with the enterprise model.

4.1. Technology Risk Ontology

To foster the bank-wide technology risk management an ontology is proposed. Its focus lies on the concept of technology risk, though it must still embody a general understanding of business and operational risk. In the remainder of the paper this is referred to as the technology risk ontology. The most important elements of this ontology are exemplary depicted in figure 3. The skeleton of the ontology constitutes the risk affected core activities of the value chain. Examples are the trading or the credit activities or the accounting activity. Essential resources enable the enactment of the activities. In accordance with the theory of operational risk, the ontology is built around a cause (property with random behaviour) effect (gains or losses) model leading to financial deficits, such as depreciation, direct payments, or missed gains. Following the Basel II accord, the internal resources are then divided into process, technology (system) and human. Finally, the ontology subsumes the theory of technology risk: the technology resources can be organized in terms of basic IT-components, complex systems and dynamic projects. This includes interdependencies as well as relevant properties of security. A more detailed discussion of this ontology can be found in [11].

4.2. Process Integration

So far it has been shown how technology risk can be modelled and expressed with ontologies. However, as described in section III.B each phase in the risk management process puts the focus on different aspects and requires separate methods. When the risk management process is implemented this often means that a number of tools is used, which are not or just poorly integrated. The main reason why it is difficult to combine those different methods into one integrated architecture is that each requires a different perspective on the technology risk ontology. Some examples illustrate the difficulties which herefrom arise:

- During the identification phase questionnaires are used to carry out a self-assessment. Since the understanding of risk depends on individual backgrounds, the answers which are collected during the survey will bring up ideas quite different from the predefined conceptualization. Several iterations can hereby lead to a unified understanding of the risks. Therefore, the ontology which is used for the
The questionnaire must be flexible and extensible towards new concepts. This flexibility, which is a prerequisite for the identification phase, comes at the price of diffuse and difficult to reuse models.

- During the measurement phase Monte Carlo simulations can be used to generate approximations of the risk figures. While the investigated objects can still be categorized as for example systems or projects, such a distinction is unnecessary and even cumbersome for the mathematical simulation. More likely the simulation algorithm will deal with entities such as inputs, outputs, states, etc. In addition to this, a weight will have to be assigned to the relations between the ontology elements.

- During the controlling phase back-testing regularly involves a comparison of the calculated risk figures with values derived from other sources, such as loss databases. This is not only important to assure the quality of the entire process, but is also carried out to meet regulatory demands. Whatever source is used though, it will have its own underlying model, which can be quite different from the ontology presented in this article. For example loss data will usually refer to the regulatory categories rather than to individuals, like systems.

Despite the different requirements of the models used during each phase, we suggest an architecture which puts the described risk ontology in its centre (cf. figure 4). The concept of ontology views [33] can be used to map the risk ontology to the models needed in the different phases. As the term suggests, the centrally stored ontology is mapped to ontologies which fit the specific needs of the components used within the risk management process. Ontology views allow the necessary transformations to be carried out instantly by means of reasoning, e.g. logical deduction. This way each component can use an adjusted model while it is actually working on the central ontology. For example, an alteration of the ontology by the measurement component is instantly available in the controlling component and can quickly be validated.

Only a simple set of operations is necessary to achieve this. The following examples of transformations clarify how an application with just one risk ontology operates. We use the same scenarios as described above:

- For the questionnaire component it is sufficient to rename the constructs defined in the central ontology. For example a user might use the term “Hardware” instead of the term “IT” which we used in Figure 3. Despite the apparent simplicity of this case, it is actually carrying the concept of ontology views to its limits: The view itself cannot be fixed at development time, i.e. the mapping has to be modifiable at runtime. This is the case, because the application cannot anticipate the terms which will be identified during the survey.

- The simulation component only needs a small number of concepts [12]. This means that several classes from the central technology risk ontology collapse into one class in the simulation-specific ontology. For example a distinction of different properties like “Timeliness” or “Security” is no longer necessary. Instead, properties appear to the simulation algorithm as entities of the type “Input”. Furthermore, some additional attributes (like text properties for the formulas) have to be added.

- In the back-testing component many ontological individuals have to be aggregated in a way which also carries out simple mathematical operations on its properties. For example the loss of an aggregated resource is the sum of the individual losses.

### 4.3. Implementation considerations

We didn’t limit ourselves to a theoretical discussion, but also implemented a prototype of the risk management application outlined so far. The prototype consists of a separate (i.e. stand-alone) applications for every phase of the risk management process; all of which use a view of the centrally stored ontology as the main artifact for the user to work with. Figure 5 shows the prototype for the measurement phase which consequently displays the adequate ontology mapping.

In this section we discuss how the ontology can be represented in Java. After the concrete risk ontology had been specified, this technical representation is critical for the application design. As the process descriptions, presented above, indicated, the platform has characteristics of both knowledge management systems and classical enterprise resource planning (ERP) systems. For the precise definition and the sharing of knowledge the advantages of ontologies are applied. For the processing of numeric data traditional methods like the ones presented in [24] are the best choice. Combining these two approaches is especially beneficial in typical banking domains such as finance where large amounts of numeric – e.g. monetary – data must be structured, processed, and made available to decision makers by means of aggregation and conceptualization.
Thus, the design of our prototype needed to be carefully balanced between the requirements which are imposed by the knowledge management nature of the system on the one hand, and those which have their origin in the ERP-nature of the system on the other hand. The difficulties arising from the presence of two – in some points conflicting – groups of requirements can be very well illustrated by focusing on the representation and the processing of enterprise data, while other requirements, such as usability, can be left out of this discussion, although they had of course an equally large impact on the final application.

From the point of view of the knowledge management part of the application all data is part of an enterprise-wide ontology, or at least it is structured conforming to the conceptualization of such an ontology. Therefore the main requirements concerning the data representation are:

(R1.1) Open to changes: Being able to work with the data in ways which take full advantage of the benefits which are typically accredited to ontologies. This includes but is not limited to the ability of improving and refining the data and even its structure at any time.

(R1.2) Reasoning: Having the possibility to infer new data (i.e. knowledge) from the existing data by means of automatic reasoning. A direct consequence of this requirement is that inferable knowledge must not necessarily be part of the data pool.

In addition to this, the ERP-part of the application requires a data representation which conforms to the algorithms, formulas, and processes that are applied to the data:

(R2.1) Reliability: The correct treatment of certain data entities by the business process must be (formally) verifiable.

(R2.2) Performance: The business processes must perform and terminate in an acceptable time.

(R2.3) Standardizable: Data should be easily exchangeable with legacy systems.

Obviously some requirements of the first group (R1.x) are in conflict with some requirements of the second group (R2.x). Three conflicts are especially hard to resolve:

The requirement of a data structure which is open to changes (R1.1) is contradicting with (R2.1), which requires the data structures to be immutable after it has been verified that they can be processed correctly by certain algorithms. This problem cannot be resolved even when only the slightest changes to the structure of the data are allowed.

In the same way (R1.1) and (R2.3) call for different assumptions about which parts of the data are static and which are dynamic. Static data structures can easily participate in standardized communication processes among applications while dynamic data structures require the participants of a communication to agree on the changes before every interaction. This either makes the communication much more complex or even impossible.

Inferring new knowledge as demanded by (R1.2) is a time consuming task and stands in direct contrast to (R2.2).

When deciding upon a way of implementing ontologies in an object-oriented target language, there are generally speaking, three options at hand:

(O1) Ontologies are usually serialized in an XML language called Web Ontology Language (OWL) [37]. The option of preserving the XML properties of this language and using the parsed XML tree as the data representation for an ontology has some advantages

C. Atkinson, C. Cuske, and T. Dickopp
As has been discussed in section IV.B our prototype relies on ontology views. This allows for a modular architecture which minimizes the need for interaction between the ERP and the knowledge management parts of the system. Different components make use of different options: The central knowledge management component implements the ontology using (O2) and can therefore take full advantage of the ontology, e.g. refinements of its structure are legal. After the ontology has been transformed to a specific view by means of reasoning, it is transferred to one of the three components which support the three steps of the technology risk management process. These components still wrap the (numerical) data which they process in an ontology and are therefore compatible with the central risk knowledge base, but they are able to use a completely different run-time representation. Namely (O3) is used so that these components are in conformance with the requirements of the (R2.x) group, which allows fast calculations.

Much work has been done which is related to our discussion: Especially in papers describing the implementation of tools and APIs which support ontologies there are usually some hints on how ontologies are represented technically. For example in [7], [21], and [26] an approach is suggested which resembles the one that we call (O2). Besides the object-oriented representation, representing ontologies in the relational database model is very important. This is indirectly done in [28], [1], and [8]: Although these papers focus on RDF [36], they also apply for ontologies, since RDF is generic enough to store ontologies as well. A pure C representation of RDF (and hence of ontologies) is described in [14].

5. Conclusion

In this paper we have proposed an architecture for information systems which supports the management of technology risk. Two major functional challenges were faced in this endeavor: the coverage of a fuzzy and constantly changing knowledge space and the integration of different risk management methodologies. We addressed these challenges by designing an ontology-based technology risk model. Whereas the decision to rely on ontologies helps to overcome the knowledge-related problems, a new set of technical problems emerges. While...
we consider the introduction of ontologies to support knowledge-driven information systems a conceptual success, there has been little research on the application of ontologies in risk management nor on the technical problems which occur when it comes to the integration of ontologies in an object-oriented application.

Both aspects have been assumed by our architecture: Firstly, we presented an ontology with which risk management related knowledge can be classified. Secondly, we presented our idea of a framework which allows multiple clients to introduce their interpretation of the risk knowledge into an enterprise ontology. Applications, which apply our ideas by using ontologies as the central run-time artifact and not only as a development aid, can be beneficially used in many enterprise computing scenarios.

6. References

C. Atkinson, C. Cuske, and T. Dickopp
Index of authors

Akram, Asif, 7
Allan, Rob, 7
Atkinson, Colin, 79
Barrett, Ronan, 17
Bonnet, Stéphane, 35
Chen, Ken, 71
Cicchetti, Antonio, 25
Cuske, Christian, 79
Di Ruscio, Davide, 25
Dickopp, Tilo, 79
Eramo, Romina, 25
Exertier, Daniel, 35
Feng, Pingfa, 71
Kewley, John, 7
Kramler, Gerhard, 43
Langlois, Benoit, 35
Li, Qing, 63
List, Beate, 53
Michlmayr, Elke, 43
Murzek, Marion, 43
Pahl, Claus, 17
Stefanov, Veronika, 53
Wu, Zhijun, 71
Xiao, TianYuan, 63
Xu, DaFeng, 63
Yu, Dingwen, 71
Zhang, Jianfu, 71
Zhou, Jian, 63