An adaptive modeling approach in collaborative data and process-aware management systems

Ion Lungu, Andrei Mihalache

***Abstract*—**Informational systems are used to reflect the business they are supposed to assist. This is the reason why each informational system needs a representation of the business objects that are involved in the processes and also the business rules that are applied to the business objects. Every object is simpler than we can think and also more complex than we can imagine. Objects should be represented as simple as they are, not simpler and not more complicated. Anytime people need to communicate or record information, in any context, it is very useful to create a model. Once the model is implemented into a business application, most of these software platforms are too inflexible to keep pace with the business processes they support and take place in a changing business context. This paper introduces an adaptive approach for enterprise data and process flow modeling in informational systems.

***Keywords*—**data model, business process management, service oriented architecture, event-driven architecture, adaptable applications.

I. Introduction

Developing and keeping up-to-date an informational system involves time, resources with high-costs and specialists to ensure the success of the system.

Most business applications are too inflexible to keep pace with the business processes they support. The business context where the organization interacts with other economic agents is characterized by a continuous change that must be reflected in the business application in order to maintain the effectiveness and efficiency of the business.

In the process of software development, the main objective of the analysis process is to discover, understand and document the business needs and requirements of a certain system [1].

A ***model*** is an abstract representation of a specification, design or system, from a particular perspective. The development of a model during the design of a software application has the purpose to express the essentials, most of the times visually, by one or more diagrams that have a clearly defined and well understood meaning.

The ***Unified Modeling Language*** (***UML***) is the most used language for visual notations that enable people to identify, think about and discuss problems and find direct solutions.

UML allows a model to be constructed, viewed, developed, and manipulated in a standard way at analysis and design time. UML models allow an application design to be evaluated and critiqued before it is coded, when changes are easier and less expensive to make [2].

The ***Model-Driven Architecture*** (***MDA***) defines an approach to modeling that separates the specification of system functionality from the specification of its implementation on a specific technology platform [3].

Based on models, the Model-Driven Architecture is major evolutionary step in the development of the software field. MDA starts with the idea of separating the specification of the operation of a system from the details of the way that system uses the capabilities of its platform. The three primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns [4].

The ***Business Process Management*** (***BPM***) discipline focuses on actual process design, automation, integration and monitoring to identify and make process improvements at the execution level. All the processes that take place in the enterprise exist whether we acknowledge them or not. They have a profound effect on the decisions we make and the actions we take.

Business Process Improvement (***BPI***) is a technique that seeks to alter the nature of the work done in a business function, with the objective of radically improving performance. Process improvement is a strategy for reducing overall costs, shortening cycle times, and improving quality and user satisfaction. This need for improvement is triggered by the change in the environment in which the enterprise evolves.

The change aims to increase enterprise effectiveness towards a compliant state with the changed conditions, and determines the system to adapt to a new stable state.

The approach of reality using models and events creates a simplified and holistic view of any type of system.

This paper aims to synchronize the business applications with the dynamic environment by introducing an adaptive approach for enterprise data and process flow modeling in information systems.

Manuscript received June 10, 2010: Revised version received July 15, 2010. This work is a part of the publications dedicated to documenting research and experiences of Andrei Mihalache's PhD in the area of "Adaptable applications" – Professor supervisor: Ion Lungu.

I. L. is with the Academy of Economic Studies, Bucharest, Economic Informatics Department, Calea Dorobanților 15-17, Sector 1, București, Postal Code 010572, ROMÂNIA (e-mail: ion.lungu@ie.ase.ro).

A. M. is with [2]Virtual Research Cluster at Edata SRL, Menuetului 12, Bucharest Business Park, Building D, Bucharest, Postal Code 013713, ROMÂNIA (phone: +40 (752) 172 909, email: andrei@edata.ro, vrc@edata.ro).

II. The Necessity of Low Coupled Systems

In enterprise software development, the simplification of the development model represents the major trend. The target is to increase developer productivity and software flexibility.

This can be reached by minimizing coupling between application business logic and the runtime containers that host that business logic.

Coupling is the degree to which each program component relies on each one of the other component. A measure of coupling is comparable to a level of dependency.

Depending on the cause that determines the coupling [5], types of coupling can be ordered from highest to lowest coupling as follows:

- *Content coupling* is caused by a component which surreptitiously modifies internal data of another component; it is the tightest type of coupling.
- *Common coupling* appears when components share data using a global variable and thus become dependent on each other.
- *Type use coupling* is similar to common coupling; instead of data being shared, only globally-defined data types are shared.
- *External coupling* is determined by a software component that has a dependency to software written by a third party, or to a particular type of hardware
- *Inclusion coupling* is a form of coupling in which one component includes the source code of another component. All the modules that include a component are coupled to each other and also to the included one.
- *Import coupling* is weaker than the inclusion coupling and it is present when one component declares that it imports (makes use of the definitions in) another.
- *Control coupling* occurs when one procedure calls another using a command or flag that explicitly controls what the second procedure does; the major disadvantage is that the called procedure will have to change whenever any of its callers adds a new command.
- *Stamp* (*data structured*) *coupling* is concretized by at least two components modify or access parts of a composite data structure in the same object; if the shared class changes, then all the classes that access its variables will likely have to change.
- *Data coupling* is when modules share data through, for example: arguments. Each argument is an elementary piece, and these are the only data which are passed.
- *Message coupling* is defined by modules are not dependent on each other; instead they use a public interface to exchange parameter-less messages. This type is endorsed by the service oriented architecture and ensures the loosest type of coupling.

Components or modules that do not communicate at all with one another have no coupling.

Loose coupling is achieved by:

- minimizing the number of necessary relationships by eliminating those that are unnecessary,
- reducing the complexity of necessary relationships.

Low coupling is often a sign of a well-structured computer system and a good design, and when combined with high cohesion, it supports the general goals of high readability and maintainability.

III. Information Systems Improvement

One major improvement in the software applications development is the use of the base elements employed in the business architecture as dynamical linked containers to represent the "as is" model and design the optimized "to be" model.

This step towards models also determined the decrease of coupling and encouraged the use of services to integrate different platforms and systems. The main change that IT is currently undergoing is the shift to Service Orientation which is completely based on open standards-based computing. The Event-Driven Architecture consolidates the Service Oriented Architecture by creating decoupling points (events) that create links between loosely coupled systems.

A. Enterprise Architecture

The Enterprise Architecture (EA) discipline is responsible for understanding the current state of the organization and defining a better future state, while strategy defines the main objectives to be reached using internal or external resources.

The ***Business Motivation Model*** (***BMM***) allows a business plan to be developed, communicated and managed in an organized fashion. Business strategy is modeled in terms of the Vision, Goals, Objectives, Mission, Strategies and Tactics, and the Influences, internal and external (Fig. 1).

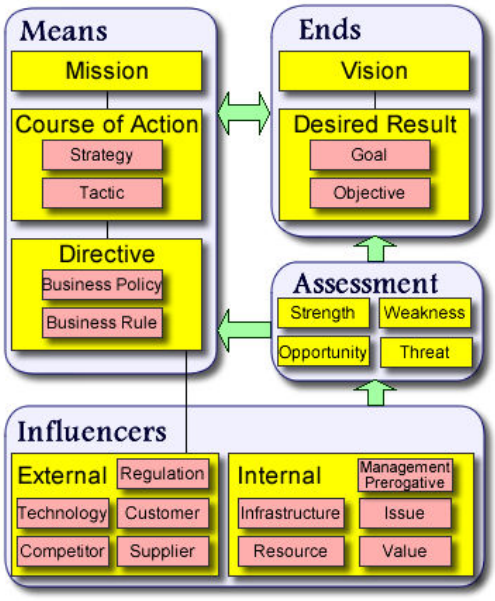

Fig. 1 overview of the Business Motivational Model

In 2005, BMM became an adopted standard of the OMG (Object Management Group). The basic idea is to develop a business model for the elements of the business plan before system design or technical development is begun. In this manner, the business strategy can become the foundation for such activity, connecting system solutions firmly to their business intent.

BMM is sufficiently generic that it can be used to capture business requirements about any domain. That is, BMM can easily be used to capture business requirements, governance requirements, IT application development requirements, nonfunctional requirements, and so forth across the application lifecycle.

BMM captures business requirements across different dimensions to rigorously capture and justify **why** the business wants to do something, **what** it is aiming to achieve, **how** it plans to get there, and how it assesses the result:

- **Ends:** What the business wants to accomplish, not how
- **Means:** How the business intends to accomplish its stated ends
- **Directives:** The rules and policies that constrain and/or govern the available means
- **Assessment:** Who and how the means are assessed against the ends, and the resulting potential impact
- **Influencers:** Who or what judges or otherwise influences the assessment.

B. Data-Oriented Design

The design of the right data structures and the well modeling of the main flows determine algorithms almost always to become self-evident. Data structures are central to programming.

Data modeling is a method used to define and represent data requirements needed to assist the business processes that take place in the enterprise.

Currently used methodologies and training for system software design have their roots in the principles of the object-oriented design. These principles worked extremely well for small scales systems, but they seem to render ineffective as their operational limits are revealed. The business processes they support need frequent and unintended redesign in programs.

For an information system, the ***data model*** is the logical representation of the data structures corresponding to the objects that present an interest for the software application. Furthermore, this model includes also constraints and relationships established between the real world objects.

Most applications require that some form of data be stored for future use. When it comes time to save this data, the most common choice is a relational database due to its performance and accessibility.

The databases contain homogeneous data collections and interrelated and that they ensure the logical and physical data independence towards applications via a third level of data description called database schema [6].

The development of a common consistent view and understanding of data elements and their relationships across the enterprise is referred to as ***enterprise data modeling***. This type of data modeling provides access to information scattered throughout an enterprise under the control of different divisions or departments with different databases and data models.

C. Process Representation

All the processes that take place in the enterprise exist whether we acknowledge them or not. They have a profound effect and also depend on the decisions we make and the actions we take.

The workflow representation facilitates the analysis of existing or planned business processes to understand the whole process and improve those stages that have a major impact on the overall.

The most frequently used graphical notation to represent these business process models are BPMN (Business Process Management Notation) or YAWL (Yet Another Workflow Language). The core concepts [7] used by the business architecture are:

- ***activities*** – that can be executed by any type of *actor*
- ***actors*** – a human or automated service that interacts with the process using graphical interfaces or web-services
- ***events*** – the occurrence of a happening during a business process
- ***business rules*** – represented by gateways where some choice is made by an *actor*
- ***messages*** – that interconnect the sequence of *activities*

Using these five main concepts as containers any business reality can be easily staged. For any enterprise, processes represent vital intellectual property. Keeping processes updated, operational and effective, constitute the premise for a continuous improvement and ensures the efficient and efficacy of each operating process.

The BPM Group has long espoused a view that Business Process Management is "…a natural and holistic management approach to operating business that produces a highly efficient, agile, innovative, and adaptive organization that far exceeds that achievable through traditional management approaches." BPM recognizes that change is as fundamental to business as the law of gravity is to physics, and that agility is therefore a fundamental requirement of enterprise architecture [8]. Therefore, any application software that assists a business process must dynamically link business objects and represent procedures and rules in a flexible manner.

The life cycle of a BPM step can be broken down as follows:

- **Study** of the enterprise BMM, analyzing its objectives and its organization in order to be able to break down its entire activity into business processes.
- **Modeling of business processes**, by representing the model which comes as close to the reality as possible

in a BPM Notation tool.

- **Implementation of the solution** in a BPM Suite, integrated with information systems of the company (applications and databases)
- **Execution** refers to the operational phase during which the BPMS is implemented for each instance of the process.
- **Monitoring**, consisting in analyzing the status of the processes by defining Key Performance Indicators for evaluation of the process performances.
- **Optimization**, consisting in proposal of solutions which make it possible to improve the performances of the business processes.

In order to obtain the most obvious results, stake holders decide during the study step to implement the BPM approach on those processes that are vital for the enterprise and represent it.

D. Service Oriented Architecture

Performing in a continuous evolution of business environment, organizations tend to change their structure frequently.

The evolving focus on service oriented architectures creates the premises for network oriented business structures with independent service providers and service consumers.

Most composite business applications are too inflexible to keep pace with the business processes they support. Built on three layers architecture (Fig. 2), these applications have two major problems: "component sharing" and "application integration".

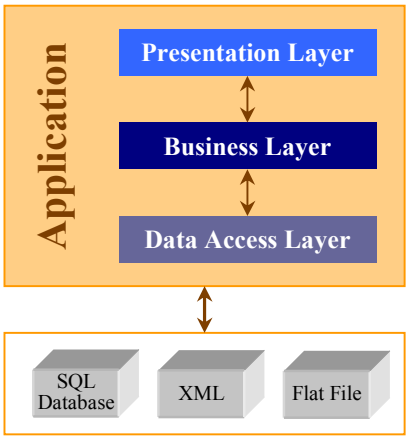

Fig. 2 component-based architecture

Sharing components across heterogeneous platforms is difficult, if not impossible. When two component-based applications need to exchange data, a user needs to manually get the data from one application and enter it to the other application.

The code integration creates a tightly-coupled application pair, difficult to manage, erratic in its behavior and any change to either application results in nothing working.

Service Oriented Architecture (***SOA***) is an architectural discipline that may be used to build an infrastructure enabling those with needs (consumers) and those with capabilities (providers) to interact via services across disparate domains of technology and ownership.

The "Service-Oriented Architecture" makes it possible to share business functions across all types of platforms. Unlike component-based applications, SOA exposes on the third layer services. This layer describes its interface in terms of SOAP and WSDL and exposes a universal interface that can be used by any user interface (presentation layer) on any platform (Fig. 3).

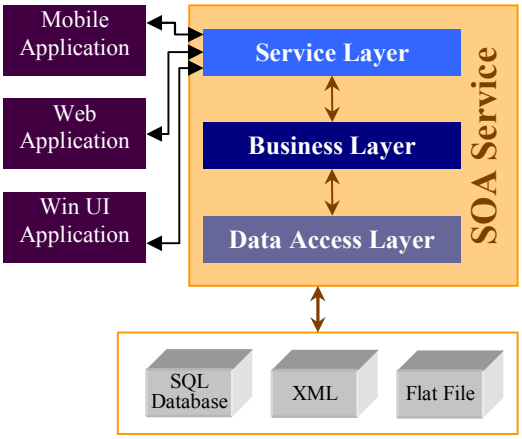

Fig. 3 service-centric architecture with multiple presentation layers (user interfaces)

The purpose of SOA services is decoupling the presentation layer from the business layer so that the presentation layer could be detached and completely eliminated.

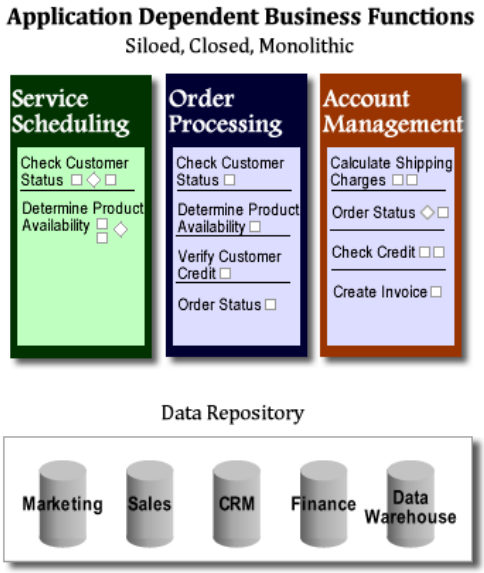

Fig. 4 Application Dependent Business Functions

Let us consider the applications involved in the Sales Orders process. Fig. 4 illustrates the three silos applications that assist this process.

In this traditional IT architecture, business process activities, applications, and data are locked in independent, often-incompatible "silos". Users have to navigate separate applications, networks, databases or even Excel files to conduct the chain of activities that complete the Sales Order business process. Independent silos absorb an inordinate amount of IT budget and staff time to maintain.

Fig. 5 represents the SOA approach of the same process that manages sales orders. The three silos applications are now integrated by reusable business services.

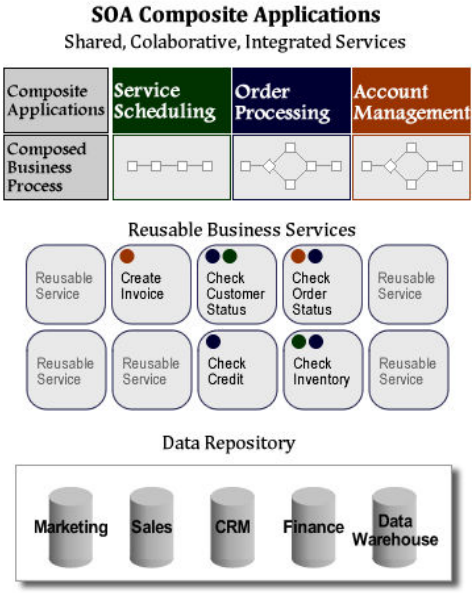

Fig. 5 SOA Composite Applications

SOA leads to reuse. A new service is frequently a composition of existing services. This reduces the overall development time, simplifies maintenance, and gives more flexibility.

SOA is now generally recognized as a significant development, particularly for business application systems. Perhaps the key reason for the success of service-oriented architectures is the fact that there has been an active standardization process working alongside technical developments and all the major hardware and software companies are committed to these standards.

E. Event-Driven Architecture

Event-Driven Architecture (***EDA***) is a term that describes an evolved state of enterprise software characterized by creation, detection, consumption of, and reaction to real time events.

Fig. 6 illustrates how SOA requests link processes using a command and control approach.

In the context of EDA, an event is essentially a significant or meaningful change in state, which triggers another process (Fig. 7).

In this perspective, an event takes a higher level semantic form where it is more granular and abstracted into business concepts and activities [9]. Moreover, the Complex Event Processing (CEP) examines the relationships of multiple business events with the goal of identifying and correlating meaningful events.

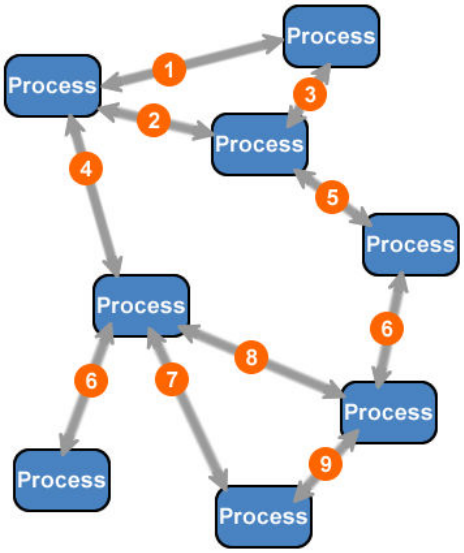

Fig. 6 logical view of request-driven architecture

A CEP engine monitors the event composition that defines a new type of event and triggers its dependents when it occurs, therefore it must be a real-time agent.

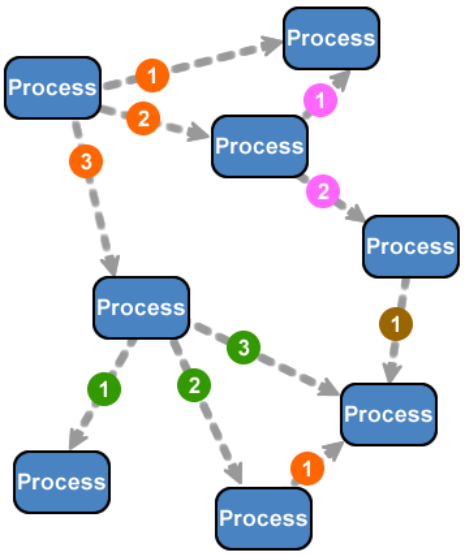

Fig. 7 Logical view of event-driven architecture

In a very simplistic statement, a CEP agent can be further characterized by two functions: a guarding function and an action function. The former determines whether an event should trigger a response, and the latter specifies the responses

to be taken if the guard is satisfied.

EDA provides a better way of loose coupling comparing to SOA, as it is not a synchronous command-and-control approach like SOA does. EDA is an asynchronous publish-and-subscribe approach, as the publisher is a completely unaware of the subscriber and vice versa. Components are loosely coupled in the sense that they only share the semantics of the message.

IV. A NEW SIMPLE MODEL

The way that current computer applications are developed consists of well defined steps and procedures, which consume a considerable amount of time. The adaptive approach of the business context begins with the following idea: resources (objects) have services (capabilities). Both resources and services are determined by objective information.

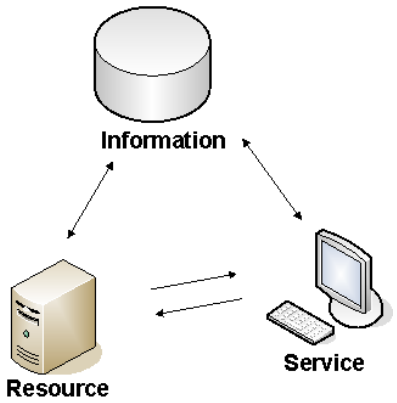

Fig. 8 The Resource – Service – Information Model

A business process has input resources, services that transform resources and the output resources. Each resource exposes services that interconnect it with other resources. This flow can be represented using the workflow diagram in the business architecture.

The business environment consists of all the *resources* the enterprise is aware of and is able to use in its processes. Resources are described and determined by objective information and capabilities called *services*, which consist in algorithms or business rules applied to resources. Fig. 9 illustrates how SOA puts forward resources and their capabilities to any type of consumer.

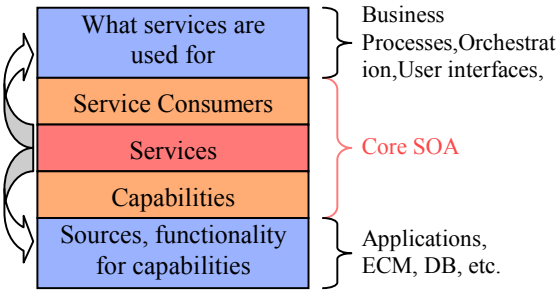

Fig. 9 Standard SOA stack

The dynamic perspective of the business context can be represented by a sequence of transformations. A business process has input resources, services that transform resources and the output resources.

V. A SOA AND EDA APPROACH

There are three basic levels to model data: conceptual, logical and physical. Conceptual data models take an abstract perspective, identifying the fundamental things. The logical data model is the actual implementation of the conceptual model and it has a major insufficiency: the cabled structure with non-reusable objects.

The objects used in software applications developed right now are *highly coupled*. In order to be able to reuse objects, they must first be exposed as services. Moreover, there is an imperative need of a mechanism that interconnects objects dynamically.

A. Service-Oriented Approach

A service centered approach makes possible the achievement of reuse objects and dynamically interconnect them. Using an access point service it is possible to make use of the data, relationships, methods, visibility or events provided by a database or other systems that expose them through services [10]. The software agility is made possible by the interconnection of objects at runtime.

The dynamic perspective of the business context can be represented by a sequence of transformations. A business process has input resources, services that transform resources and the output resources.

Business processes are represented using the workflow diagram, after analyzing existing or future planned business processes. Their purpose is to understand the different stages of these processes, then representing the process in a workflow design notation.

Once the high level analysis and design is done, service engineers can define loosely coupled, reusable software components that encapsulate discrete functionality and expose data, events and methods using standard Internet and XML based protocols.

B. Setting borders between subsystems

In order to improve performance, and also value the advantages of SOA and EDA, it is important to find points of decoupling, by looking for parts of the business process of which always will stay together for sure in one organizational unit. They usualy are atomic business functions with a strong coesion.

Loose coupled modules can be obtained by designing independent modules and the accent is focused on the way that each module can be reused by any other module. Most of the times, loose coupled modules have the biggest values for the reuse indicators.

Loose coupling means also independency of the subsystems. Loosely coupled components do not rely on each other, as

system failures can always occur.

C. Event-Driven Advantages

Maintaining redundancy across the decoupling borders makes the loose coupling more robust, as components do not rely on each others stored data. Each loosely coupled environment maintains its own copy of the data and services, as in loosely coupled environments redundancy must be seen as strong.

As Fig. 10 shows, EDA is perfectly suited by its nature to support automatic data synchronization mechanisms [11] in redundant environments while maintaining overall loose coupling.

Each module uses only the necessary data structures and subsets of records used in the assisted business processes.

The command and control SOA approach inside each module links dynamically the necessary actions, rules and data structures. This architecture enables an ongoing change in the software systems and a mechanism that interconnects objects dynamically.

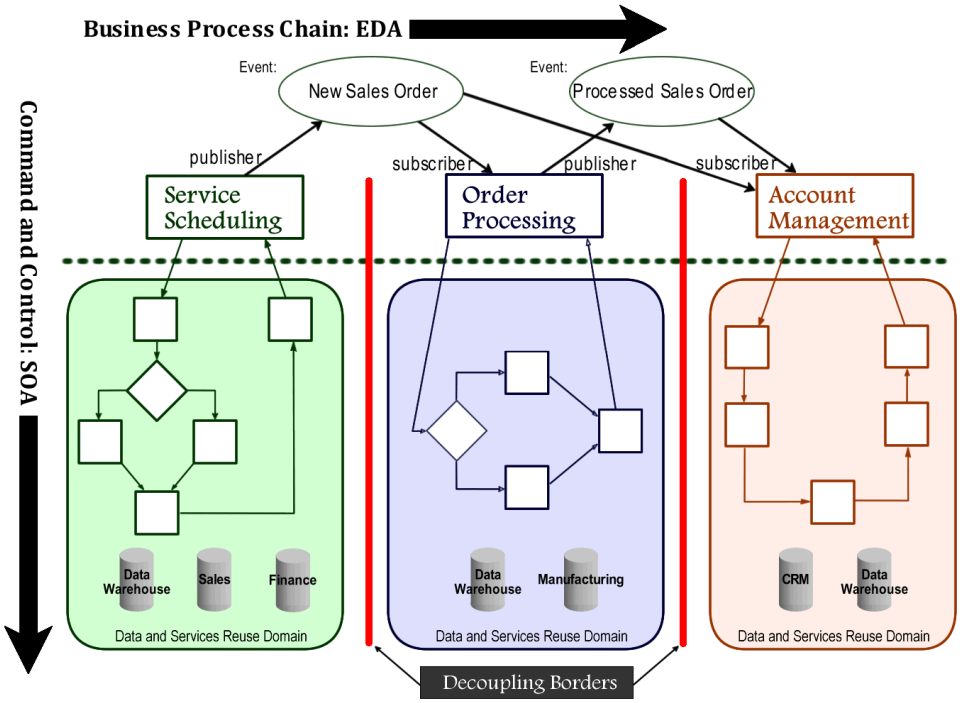

Fig. 10 SOA and EDA in action

VI. Conclusion

The purpose of this paper is to reveal the importance of adaptable applications in the definition of data and process models from the perspective of strategy, analysis and execution.

Systems and technology tend to borrow more of the characteristics from the business architecture objects and processes they assist. New software for enterprise management must adapt to the business and its work and evolve with it.

Acknowledgment

Andrei Mihalache thanks professor Ion Lungu, his supervisor during the PhD studies for all the trust invested in me throughout the PhD training program and development of thesis. A. Mihalache also thanks Cornel Vintila as he owes a good part of the achievements of this paper, and the initial point to start the route that formed his vision on business process management applications, resources and service oriented architecture.

References

[1] J. Satzinger, R. Jackson, S. Burd, *System analysis and design*, Fourth Edition, Thomson Technology 2007, ISBN: 1-4188-3766-0

[2] R. Soley, *MDA Foundation Model*, OMG 2005, http://www.omg.org/cgi-bin/doc?ormsc/05-04-01

[3] S. Ambler, *Architecture and Architecture Modeling Techniques*, http://www.agiledata.org/essays/ enterpriseArchitectureTechniques.html

[4] A. Watson, *MDA Guide Version 1.0.1*, 2003, http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

[5] University of Ottawa, School of Information Technology and Engineering, *Object Oriented Software Engineering*, http://www.site.uottawa.ca:4321/oose/index.html#coupling

[6] I. Lungu, M. Velicanu, I. Botha, "Database Systems – Present and Future", *Informatica Economica Journal*, Ed. Inforec, ISSN 1453-1305, EISSN 1842-8088

[7] A. Mihalache, C. Vintilă, A. Cornescu, "A New Model Approach for Business Applications in Enterprise 2.0 Environment," in *Proceedings of the World Multiconference on APPLIED ECONOMICS, BUSINESS AND DEVELOPMENT (AEBD '09)*, WSEAS Press, Tenerife, ISBN 978-960-474-091-8, pp 192-195, July 2009, Available: http://dynamicapplications.org/documents/AEBD-29.pdf

[8] H. Smith, P. Fingar, *Business Process Management: The Third Wave*, Meghan-Kiffer Press, 2002, ISBN: 0-929652-33-9

[9] H. Taylor, A. Yochem, Les Phillips, F. Martinez, *Event-Driven Architecture: How SOA Enables the Real-Time Enterprise*, Addison-Wesley, 2009, ISBN 978-0-32-132211-1
[10] I. Lungu, A. Mihalache, "Adaptable Enterprise Modeling - a new challenge for collaborative data and process-aware management systems," in *Proceedings of the Recent Advances in Mathematics and Computers in Business, Economics, Biology & Chemistry (MCBE '10)*, WSEAS Press, Iaşi, ISBN 978-960-474-194-6, pp 261-266, June 2010.
[11] J. van Hoof, *How EDA extends SOA and why it is important*, 2006, http://soa-eda.blogspot.com/2006/11/how-eda-extends-soa-and-why-it-is.html
[12] A. Mihalache, "Mobile applications design for Project and Knowledge Management," *Informatică Economică Journal*, submitted for publication.
[13] P. Pocatilu, F. Alecu, M. Vetrici "Measuring the Efficiency of Cloud Computing for E-learning Systems," *WSEAS TRANSACTIONS on COMPUTERS*, Volume 9, Issue 1, January 2010, ISSN: 1109-2750 pp. 42-51
[14] W. Khlif, N. Zaaboub, H. Ben-Abdallah, "Coupling Metrics for Business Process Modeling," *WSEAS TRANSACTIONS on COMPUTERS*, Volume 9, Issue 1, January 2010, ISSN: 1109-2750 pp. 31-41
[15] A. Bara, I. Lungu, S. Oprea, "Public Institutions' Investments with Data Mining Techniques," *WSEAS TRANSACTIONS on COMPUTERS*, Volume 8, Issue 4, April 2009, ISSN: 1109-2750 pp. 589-598

Ion Lungu is a Professor at the Economic Informatics Department at the Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies of Bucharest. He has graduated the Faculty of Economic Cybernetics in 1974, holds a PhD diploma in Economics from 1983 and, starting with 1999 is a PhD coordinator in the field of Economic Informatics.

He participated (as director or as team member) in more than 20 research projects that have been financed from national research programs. He is a CNCSIS expert evaluator and member of the scientific board for the ISI indexed journal Economic Computation and Economic Cybernetics Studies and Research. He is also a member of INFOREC professional association and honorific member of Economic Independence academic association. In 2005 he founded the master program Databases for Business Support (classic and online), who's manager he is.

His fields of interest include: Databases, Design of Economic Information Systems, Database Management Systems, Decision Support Systems, and Executive Information Systems.

Andrei Mihalache has a background in computer science and is interested in database related issues and enterprise modeling techniques.

Currently, he is a PhD candidate in the field of Economic Informatics at the Academy of Economic Studies. His research interests include: enterprise data model, adaptable information systems, business process improvement, and web collaborative technologies.