12

SOA and supporting software processes integrated with self-organizing business networks

Francesco Rago
M3 Comp. Llc
USA

1. Introduction

Actual business leverages on software services to improve general firm performance. Software is always more a strategic asset to sustain business, but in meantime business is changing continuously: there is a transition from centralized to distributed and cooperative organizations. For this reason Business Models of firms are changing: there is an evolution toward distributed models better suited for integration into a global economy. This scenario implies a strategic capacity to manage new emerging values, strategies, technologies and products. The enterprise organizational models have to consider a network of internal and external agents (partners) with fuzzy boundaries and continuous exceptions in processes. The Business Model approach help a firm or a network of firms to structure its organization in a way to become more efficient, more flexible and responsive to customer demand, to forecast possible future scenarios and therefore to stay competitive.
This environment is particularly challenging for software maintenance and new "smart" approaches have to be defined. We describe a software development and maintenance process strictly integrated with enterprise evolution. The innovative aspect derives from an agile approach integrated with self-organizing and changing business. It is also based on the basic hypothesis that the service model of software maintenance has to be integrated in the process of define or improve business models at run time and not in a separate step. The shift from centralized to distributed and cooperative organizations needs software with Service Oriented Architecture (SOA) dynamically integrated with business architecture. Software services become the fundamental platform to support products delivery and services management. Their maintenance is becoming a major challenge to guarantee software aligned to business processes.

2. Inter-Organizational Systems (IOS)

The consequence of actual economic and technological scenario is the blurring of boundaries of companies in the field of knowledge, locations, information. This process is amplified by

the so called "ubiquitous computing" fundamentally characterized by the "connection of things" in the world with computation.
Partnering is becoming an alternative to mergers and acquisitions particularly for Subject Matter Experts, as it is cheaper and less capital intensive. The formation of Inter-Organizational Networks has environmental drivers such as technological changes, changing customer behavior and increasing importance of information and knowledge. Motives of the co-operating parties derive from the need to enhance services or products portfolio, to improve the efficiency of processes and to share of risks and development costs.
The network strategy is described by an inter-firm strategic planning and a technical configuring networking initiative. ITC is an enabler making collaboration feasible and it supports IOS to coordinate interdependencies or to pool information with the emergence of networked organizations with core business ICT-based (e.g. virtual organizations, value webs, value nets etc.). The different partners offer specialization and focus on core competencies. Firms need clear responsibilities and clear views of contribution of each element of the net defined essentially as groups often belonging to more companies. Forming of groups is Project-dependent and Integration of groups means an Organizational context that facilitates the cohesion.
This scenario is feasible if technical trends are in right focus. The ICT infrastructures become global supported by Standards and Protocols like EDIFACT, XML and Three-tier architecture.
We have different types of interdependencies between firms and their activities: first of all a configuration can be called Pooled. Pooled means the existence of some firms that create pools having a basic set of capabilities to share with others firms. In this case there are infrastructures coordinated by Standards & Rules. Type of Inter-Organizational Systems Pooled are information and resource IOS.
A second type of interdependence is Sequential: it is a chain of companies creating a connection to satisfy specific goals. Value Chain stages are coordinated by Standards & Rules, Schedules & Plans. Types of Inter-Organizational Systems are EDI, SCM Applications and Workflow Systems.
Last but not least, Reciprocal is a "small-world", based on Collaborative project management to produce customer specific product. The Co-ordination mechanism is more complex requiring Standards & Rules, Schedules & Plans and Mutual Adjustments. Inter-Organizational Systems are Groupware and Distributed Project Management. All these organization style requires SOA as a common base of services: Inter-Organizational Systems fit well organizations if services and not application oriented. This generates flexibility and adaptability.
With the extending enterprise, organizational networks arise and information systems cross the firm's boundaries. There is a complex interaction between technology and the Organization generated by IOS driven linkages and networks.

3. Firms and ITC Strategies

"Business strategies specify how a business model can be applied to the market to differentiate the firm from its competitors" (Elliot, 2002).
In the book "Competitive Advantage", Michael Porter introduces the value chain as a tool for developing a competitive advantage. The tool has the goal to define a strategy for

interrelationships between value chains of different firms segments. Value chain analysis describes the activities within and around an organization, and relates them to an analysis of the competitive strength of the organization, starting with the generic value chain and then identifying the relevant firm-specific activities. Process flows are defined and used to isolate the individual value-creating activities. Linkages between activities describe the affects of performance or cost of one activity on another. The linkages are flows of information, goods and services, as well as systems and processes for adjusting activities. These linkages are crucial for corporate success and are the basic point of introduction of appropriate SOA. Competitive advantage may be obtained by optimizing and coordinating linked activities. A result of his approach was the understanding of the role of technology in competitive advantage.

Integration between business planning and ITC planning is one important enabler of business-IT alignment (Teo and King 1997). Teo and King found a significant relationship between the level of business and IT planning integration and the extent of information systems contribution to organizational performance. The role of ITC function has an evolution from technically oriented and non-strategic at first, until resource to support and influence business strategy. ITC becomes critical to long-term survival of organization in a fully inter-operational strategy. In mean time the different stage of business integration have to happen. The purpose of integration begins with administrative and non-strategic as first stage arriving to full integration with joint development of business and ITC strategies. Thompson (1967) describes three types of technology, which are important to the understanding of strategy in organizations: long-linked technology, mediating technology, and intensive technology.

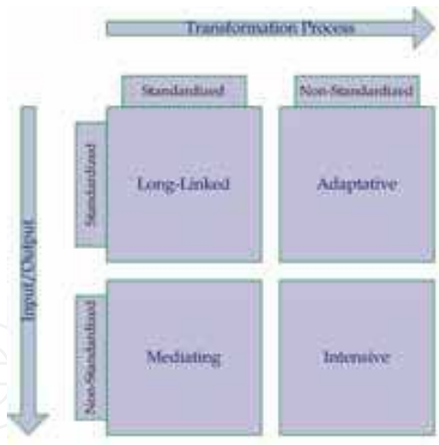

Fig. 1. IT and Business Integration

The classification has importance in the integration approach and in the platforms choice. As the three technology types have distinctive value creation logic, this has an impact in strategic planning.

Long-linked technologies focus on transformation of inputs into outputs having a production process consisting of a fixed sequence of steps to transform standardized inputs into standardized outputs.

Mediating technologies are characterized by a standardized transformation process and unique inputs and outputs. Service providers deliver to unique clients based on a pre-defined delivery process. These technologies often link partners in an exchange that helps them locate appropriate parties to conduct their transactions. Of course this type of technological approach has a meaning when precise and unique chain of firms is defined. Intensive technologies combined a non-standard transformation process with unique inputs and outputs. This technology focused on the coordination of experts and pooling their expertise to create a unique outcome.
We have had a forth technology: Adaptive. It normalizes the status of transformation process to standard inputs and outputs with the scope to value what it is happening when environmental changes of the net happen.
Having in mind a general strategy, the organization structure and the type(s) of technology the integration of general strategy and IT Planning can be represented using a business model that all Stakeholders share and agree.
Rappa (Rappa 2005) gave the following definition of Business Model:

"*In the most basic sense, a business model is the method of doing business by which a company can sustain itself -- that is, generate revenue. The business model spells-out how a company makes money by specifying where it is positioned in the value chain.*"

A Business Model is a blueprint that describes how a network of cooperating agents intends to create and capture value. In a Business Model one of the main points is the Customer Value: this describes how the firm offering its customers something distinctive or lower cost than its competitors. Another basic issue was the Scope which describes to which customers is the firm offering his value and the range of products/services offered that embody this value. Capabilities are the firm's capabilities and capabilities gaps that need to be filled to sustain the Scope. Something distinctive characterizes capabilities to allow the firm to offer the value better than other firms and that makes them difficult to imitate. The sources of these capabilities can be many and exploited using appropriate SOA and specific ITC infrastructure. For this reason we have to clarify what is the fit between them and how SOA do to the strategy, structure, systems, people and environment of firm. The components of a business value are here summarized:

- Model Ontology Concepts: purpose of the ontology is to improve communication, inter-company interoperability, intra-company interoperability, achieving reliability, enhance business model maintenance, knowledge acquisition, fundament for enabling support tools. The Ontology allows to accurately describe the business model of a firm
- Service domain: it is a description of the service offering, its added value, and the market segment at which the offering is targeted.
- Technology domain: a description of the SOA required to realize the service offering.
- Organization domain: a description of the structure of the multi-agents value network required to create and distribute the service offering (organizational arrangements)
- Finance domain: a description of how risks, investments and revenues are divided over the different actors of a value network (financial arrangements).

Of course Business Models have their dynamics depending on external and internal issue. Any change has a direct impact on business models to continue to give to a firm a competitive advantage. The Model is subject of Technological changes because the natural evolution of IT and of its application in competitors.

4. Business Model Framework

We propose a specific Business Model that we consider well suited to improve IOS and SOA interrelationship. The framework is divided into six principal components:

(1) The agents that agrees the alliance.

(2) The products and services a firm offers represent a substantial value to a target customer (value proposition), and for which he is willing to pay.

(3) The relationship capital the firm creates and maintains with the customer, in order to satisfy him and to generate sustainable revenues.

(4) The infrastructure.

(5) The network of partners that are necessary in order to create value and to maintain a good customer relationship.

(6) The financial aspects such as cost and revenue structures.

There are no interfaces listing Agents and their all processes in a common and agreed open space. The Business Model can be enforced deriving its components from shared value propositions, available capabilities in each forms and business constraints. In an organization based on the described Business Model the SOA permits more easily to organize and use distributed capabilities that may be under the control of different ownership domains.

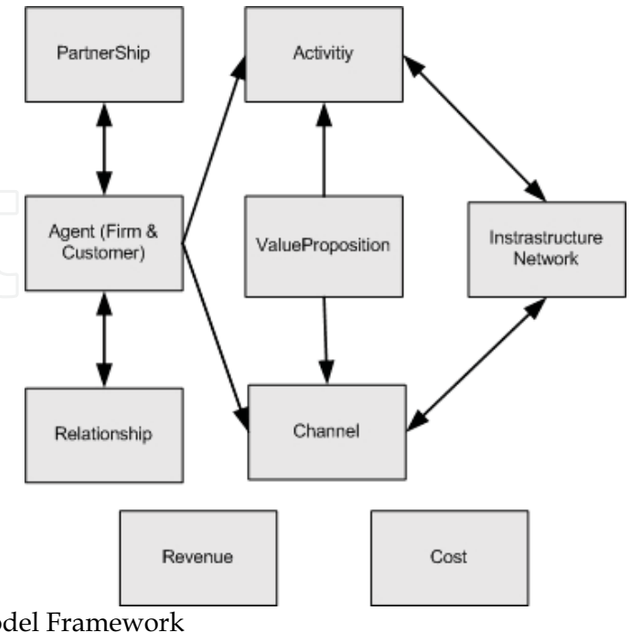

Fig. 2. Business Model Framework

The adaptability is improved if a Business Model is able to provide a systems of values, a shared structure and a SOA input to ITC. ITC loses the role of infrastructure and becomes a component of business organization.
In general, agents create capabilities to solve or support a solution for the problems they face in the course of their business. The needs of a web of firms are met by capabilities offered by net members.
The value of SOA is that it provides a powerful framework for matching needs and capabilities and for combining capabilities to address firms nets. Services are the mechanism by which needs and capabilities are brought together. SOA is a mean of organizing solutions that promotes reuse, growth and interoperability.
Visibility, interaction, and effect are key concepts of the SOA paradigm. In our approach visibility refers to the capacity for those with needs and those with capabilities to be able to see each other. The descriptions need to be in a form (or can be transformed to a form) in which their syntax and semantics are widely accessible and understandable.
The Business Model declares activities based on visible services and interactions are the way of using a capability. At its core, an interaction is "an act" and the result of an interaction is a set/series of effects. This effect may be the return of information or the change in the state of entities that are involved in the interaction. Effects are couched in terms of changes to shared states.
Visibility is promoted through the service description which contains the information necessary to interact with the service and describes this in such terms as the service inputs, outputs, and associated semantics. The service description also conveys what is accomplished when the service is invoked and the conditions for using the service.
Agents offer capabilities and act as service providers. Those with needs who make use of services are referred to as service consumers. The service description allows prospective consumers to decide if the service is suitable for their current needs and establishes whether a consumer satisfies any requirements of the service provider.
A software service specification is the definition of a set of capabilities that fulfil a defined purpose. In model-driven systems development, a service specification can be made using Systems Modelling Language like SysML or UML or any other Specification tool.

5. Software Maintenance of Systems

This environment is particularly challenging for software maintenance and new "smart" approaches have to be defined. Maintenance is becoming a major challenge to guarantee software aligned to the business processes of the Business Model.
The maintenance process designed to satisfy the described scenario has an agile approach integrated with self-organizing and changing business.
The shift from centralized to distributed and cooperative organizations needs software with SOA dynamically integrated with business architecture. Software services become the fundamental platform to support products delivery and services management. There is a paradigmatic changes: from top down development and maintenance to a bottom-up evolutionarily life cycle where software assets maintenance is integrated with organizational assets maintenance. The benefits of the approach are measured reduction of the number of defects on high level requirements and the incremental commitment nature of the process: expenditures tend to be balanced with certainty level. Another advantage is a reduced time to upgrade software assets.

Once a business model is defined and agreed by all agents, we used a method to support software maintenance through advanced iterative and incremental approach. We adopted a process similar to SCRUM because it is not a step-by-step cookbook approach and requires active, thoughtful development and management. The method starts with the premise that maintenance environment is complicated and unpredictable. You can predict or definitively plan what you will deliver, when you will deliver it, and what the quality and cost will be, but you have to negotiate continuously them according to various risks and needs as you proceed. SCRUM method is integrated with the Business Model maintenance to improve the associated SOA model and the software architecture.

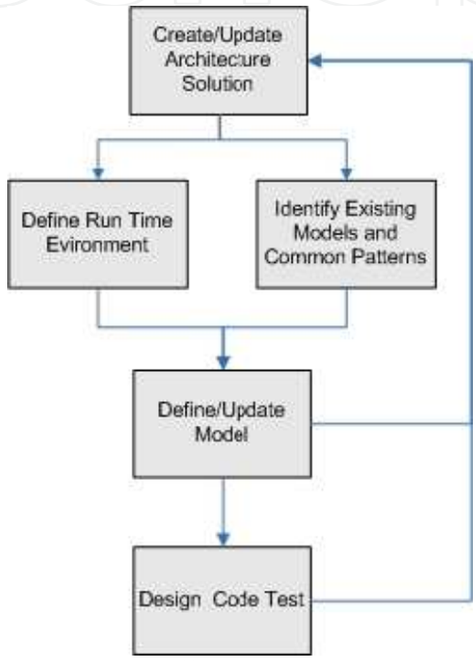

Fig. 3. Self-Organization Business Model Ontology

The fundamental steps of the method follow:

Stage I. Concept

The purpose of the concept stage is to better define exactly how the business model and its context was changed or improved, who is aimed at, how it will be positioned in market segments and how Information Technology assets has to be changed/improved to support business focus on target.

Stage II. Backlog update

The software maintenance begins in earnest. It is used a streamlined, flexible approach to requirements change management reflecting both Extreme Programming (XP)'s planning game and the SCRUM methodology.

Stage III. Technical Cycle

This is an iterative stage where technical operations are accomplished. The structure of the service application is created/maintained. The stage has the following steps:

III.1 Create/Update Solution architecture Definition of Model Technology and its use. Define/Update the conceptual structure of the SOA.

III.2 Define/Update runtime environments
Define the runtime environments in which the Service application should run. This covers all test environments, including unit test and final production environments.

III.3 Identify Existing Models and Common Patterns
The repeating patterns are identified within the service application.

III.4 Define/Update design model

III.5 Design, code, test

Stage IV. Trial

Stage V: Launch

6. Stage I. Concept

The purpose of the concept stage is to better define exactly how the business model and its context was changed or improved, who is aimed at, how it will be positioned in market segments and how Information Technology assets has to be changed or improved to support business focus on target. The Business Model can be analyzed with a process approach using Statistical Process Control results (where applicable) to value the actual status of processes. Processes are at the core of Business Model and the starting point of any innovation, adaptation or integration of firms (Dubray, 2007). There is a clear distinction between business processes and services orchestrations that can be defined in another Stage. Business processes are defined using Business Process Modelling Notation (BPMN) and are modelled from a user point of view. These process models describes user's view of the BM. We now introduce some details on BPMN so that it is more easy to understand how to integrate it with software maintenance process.
BPMN is focused on creating a standard look and feel for process diagrams that is more "business-friendly". By standardizing the semantics of things like tasks, sub-processes, and events, and linking each to specific graphical shapes, icons, and line styles in the process diagram, BPMN allows Agents to understand a process diagram regardless of which vendor's modelling tool created it, and conversely to create process diagrams that other analysts can immediately understand without special training. While the BPMN specification suggests mappings of certain shapes and patterns to specific BPEL code, it's been reduced a bit from the original vision of a standard business-friendly front end for executable process models.

BPMN Diagrams are essentially flowcharts. Process participants are defined by pools, which may be subdivided into swim-lanes, as in many other modelling notations. From the perspective of the executable model, each pool represents a BPEL process. In addition, blank but named pools can also be used to represent other business processes for which the internals are opaque, a black box, as decided in the contract phase. A single business process diagram can be composed of multiple pools, meaning multiple executable processes.
The basic units of a process are tasks, sub-processes, and events. Sub-processes and tasks are represented by rectangles, events by circles. Various icons within those shapes indicate the particular type of event or sub-process. Sub-processes may either be embedded in the calling process or launched as an independent process, running in parallel with the calling process, and synchronizing with it via messages. Solid lines group sequence flows and interconnect the tasks, sub-processes, and events within a single pool. Each sequence flow may be conditional or unconditional. In addition, BPMN offers a diamond Gateway shape that, depending on its icon, can be used for branching, splits, conditional splits, merges, or synchronizing joins.

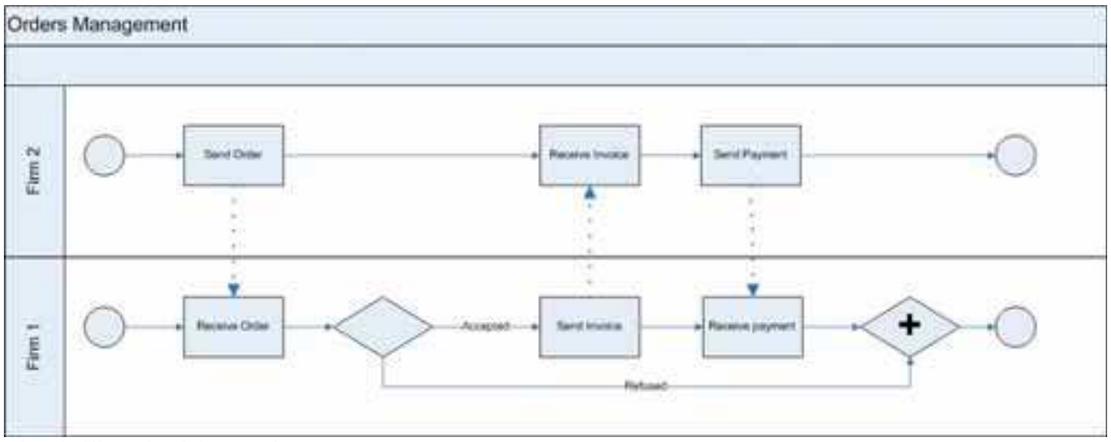

Fig. 4. Simple BPMN chart.

In addition to sequence flows within a pool, BPMN shows message flows exchanged between pools. These indicate the signals the process uses to communicate with invoked services and partner processes. BPMN explicitly shows events, actions triggered by a signal of some sort, such as receipt of a message, expiration of a timer, detection of an error, etc. Each event has a trigger and a resulting action. The various types of triggers and resulting actions are indicated by the placement of the event in the diagram along with its internal icon. Events shown with an incoming sequence flow mean that the process issues the event (i.e., sends the message, waits for a time delay, or throws the error), and those with no incoming sequence flow mean that its outgoing sequence flow is triggered by the event (e.g., receipt of a message, timeout, or exception). Events of the second type placed on the border of a task, sub-process, or pool indicate that the normal flow within that task, sub-process, or pool is to be interrupted and the exception flow connected to the event is to be triggered. Already completed tasks within a process interrupted in this way are reversed by compensation actions defined by a compensation event linked to the task. Thus with BPMN events, exception-handling behaviour and inter-process communications critical to the

executable implementation are shown explicitly in the diagram, without requiring the modeller to specify the underlying technical details.

The analysis of a BM permits to decide the implementation or update of SOA. A service will be implemented or updated if it adheres to the following principles (Erl, 2007):

- *Standardized Service Contracts:* services within the same service inventory are in compliance with the same contract design standards.
- *Service Discoverability:* services are supplemented with communicative meta data by which they can be effectively discovered and interpreted.
- *Service Abstraction:* service contracts only contain essential information and information about services is limited to what is published in service contracts.
- • *Service Loose Coupling:* service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment.
- *Service Autonomy:* services exercise a high level of control over their underlying runtime execution environment.
- *Service Composability:* services are effective composition participants, regardless of the size and complexity of the composition.
- *Service Statelessness:* services minimize resource consumption by deferring the management of state information when necessary.

Standardized service contracts, discoverability and abstraction are all about the definition of the interface (or contract) of a service. These general requirements are accomplished in compliance to the chosen BM that different Agents have agreed. The other principles are more technical and are needed to enable the principles of reusability and loose coupling. Loose coupling calls for a messaging system supporting the communication between services. Service composability asks for the concept of assemblies to be introduced as a programming concept. The principle about service statelessness is good to strive for, but not always possible. Therefore the notion of state alignment is crucial to loosely coupled solutions (Dubray, Composite Software Construction, 2007).
From a technical perspective the presented principles can be applied only if an infrastructure is defined and available.
A distinction can be made between two main types of services (Cohen, 2007): those that are infrastructural in nature and provide common facilities that would not be considered part of the application, and those that are part of the application and provide the application's building blocks. Infrastructure services are common facilities that are part of the required infrastructure for the implementation of any business process in a SOA. Infrastructure services can be further divided into (Cohen, 2007): Communication services, which are mainly used for message transportation, and Utility services, which deliver generic (non-application-specific) infrastructural functionality. Application services, on the other hand, are services that take part in the implementation of a business process providing explicit business value. Cohen (Cohen, 2007) divides these services into entity, capability, activity and process services. Dubray (Dubray, Composite Software Construction, 2007) adds two important types of services: delivery and decision services. These classifications are described as follows:

- Entity services, exposing and managing business entities;
- Activity services, implementing a specific application-level business capability;
- Capability services, implementing a generic value-add business capability;
- Process services, implementing a business process by orchestrating other services;
- Decision services, supporting the externalization and reuse of complex and critical decision points within a task, process or business object;
- Delivery services, enabling user interactions with the system, which are always performed within a task.

The main difference between Capability services and Activity services is the scope in which they are used. While Capability services are an organizational resources, Activity services are used in a much smaller scope, such as a single composite application (Cohen, 2007).
The common and shared Business Model generates decision about common SOA. At this point a service contract is agreed on comprehensive of description of the business functionality provided by the service, but also the high-level requirements of the service along with specifications governing its usage service level declarations and key performance indicators. A service contract description develops and catalogues a *uniform* yet multi-dimensional, *enterprise-wide* understanding of the purpose, scope and representation of what this service entails and involves.

Starting from BPMN model the following characteristics will be collected:

Responsibility of the service: This provides the basic description of why the service should exist. In other words, this states the basic business scope of the service. A service "does" something in terms of business significance (in other words, encapsulates business functionality). This also identifies the software component/s which implement(s) the service.

Pre-conditions and post-conditions: Describe the factors that must be in place for this service to be used and the limitations and constraints of the service.

Synchronous or asynchronous conditions: This refers to the call semantics of the service - synchronous or asynchronous.

Identify the consumers of this service: Define the Agents and their roles that need to represent when calling this service.

Other documented requirements: Security, Data, availability and Service Level Agreement governing the service with costs.

7. Stage II. Backlog update

The software maintenance begins in earnest. It is used a streamlined, flexible approach to requirements change management reflecting both Extreme Programming (XP)'s planning game and the SCRUM methodology.
Scrum is a "lean" approach to software development. Scrum is a simple framework used to organize teams and get work done more productively with higher quality. Designed to

adapt to changing requirements during the development process at short, regular intervals, Scrum allows teams to prioritize customer requirements and adapts the work product in time to customer needs.
The output of periodic Concept stage is an input to a plan which includes a Product Backlog. The Product Backlog is a list of functional and non-functional requirements that will deliver the SOA updates when turned into functionality. The Product Backlog is prioritized so that the more value items are top priority. The Product Backlog is divided into proposed releases and the release are reported and validated in periodic Concept Meetings. Changes in the Product Backlog reflect changing requirements and how quickly or slowly the Team can transform the Product Backlog into functionality.
All work is done in Sprints. Each Sprint is an iteration of one month. Each Sprint is initiated with a Sprint Planning meeting, where the Product Owner and Team get together to collaborate about.

8. Stage III. Technical Cycle

This is an iterative stage where technical operations are accomplished. The structure of the service application is created or maintained. The Stage transforms requirements in a technical feasible set of Services and Applications. We have the problem to integrate different realities and technical environments and this cannot be coped with traditional approaches.
According to the Object Management Group (OMG): "Given the continued, and growing, diversity of systems, this will never be achieved by forcing all software development to be based on a single operating system, programming language, instruction set architecture, application server framework or any other choice. There are simply too many platforms in existence, and too many conflicting implementation requirements, to ever agree on a single choice in any of these fields." (OMG, 2003). The solution of the OMG is Model-Driven Architecture (MDA).
OMG generated MDA as the difficulty to manage Enterprise Architecture able to cover distributed multi firms organizations. Organizations are complex and as consequences the design of architecture is difficult even if SOA approach is used. We give some definition to introduce SOA/MDA:
A model of a system is a description or specification of that system and its environment for some certain purpose. The text may be in a modelling language or in a natural language (OMG,2003).

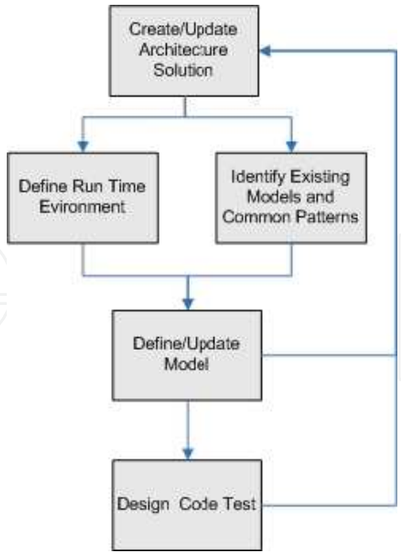

Fig. 5. Technical Cycle

A viewpoint on a system is a technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within that system. The word abstraction is used to mean the process of suppressing selected detail to establish a simplified model (OMG, 2003).
A platform is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented (OMG, 2003).
MDA is an approach using models in software development. The MDA prescribes certain kinds of models to be used, how those models may be prepared and the relationships of the different kinds of models. The basic concept of the Model-Driven Architecture is the separation of the operations of a system from capabilities details of its platform. The MDA provides an approach in which systems are specified independently of the platform that supports it. It also provides an approach for specifying platforms, for choosing a particular platform for the system and for transforming the system specification into one for a particular platform. The three primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns (OMG, 2003).

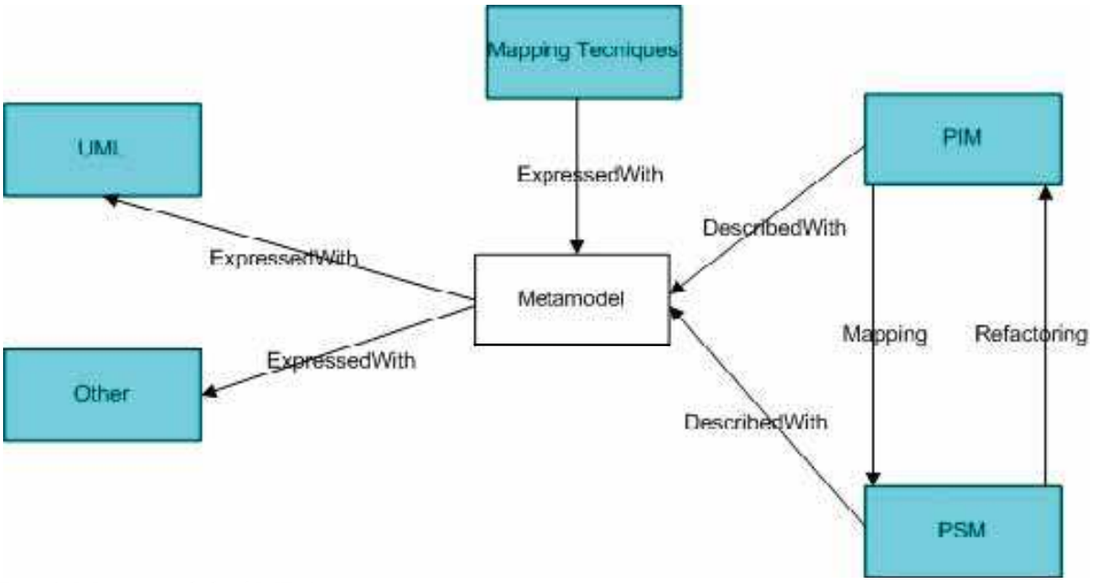

Fig. 6. Meta-model of MDA

MDA specifies three default models of a system:

- The Computation Independent Model (CIM).
- Platform Independent Model (PIM).
- Platform Specific Model (PSM)

The Computation Independent Model (CIM). A computation independent model is a view of a system from the computation independent viewpoint. A CIM does not show details of the structure of systems. A CIM is sometimes called a domain model and a vocabulary that is familiar to the practitioners of the domain in question is used in its specification (OMG, 2003). It is assumed that the primary user of the CIM is the domain practitioner or business consultant. The user of a CIM doesn't have to have knowledge about the models or artefacts used to realize the construction of the application complying to the requirements defined in the CIM. The CIM specifies the function (or external behaviour) of a system without showing constructional details.
An instantiation of a CIM can be an UML or BPMN chart or business functional characteristics (inner and external objects, functional features) by means of noun and verb analysis in the informal problem description. Cause-effect relations form causal chains that are functioning cycles. All the cycles and sub-cycles should be carefully analyzed in order to completely identify existing functionality of the system.
In case of studying a complex system, a CIM can be separated into a series of subsystems according to identified.
A platform independent model is a view of a system from the platform independent viewpoint. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type (OMG, 2003). A PIM describes the construction of a system on an ontological level, meaning that the construction of the system is specified without implementation details.

A platform specific model is a view of a system from the platform specific viewpoint. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform (OMG, 2003). In other words: the PSM is a more detailed version of a PIM. Platform specific elements are added. When defining a PSM a target Platform Model has to be available.

8.1 Create/Update Solution architecture

Transition from an initial problem domain model to a CIM "output" model, i.e. a use case model, goes as follows:
1) Acquire the list of Agents and their goals belonging to BPNM charts. Identification of goals is the identification of the set of functional features necessary for the satisfaction of the goal.
2) Identification and refinement of system's use cases that include discovering functional features specified by requirements that are needed to achieve a business goal. An executor of the goal is transformed into an (UML) actor. Identified use cases can be represented in an UML activity diagram by transforming functional features into activities, and cause-effect relations into control flows.
The last step is identification of a conceptual class model. In order to obtain a conceptual class model each functional feature is detailed to the level where it only uses one type objects. This is an architecture solution that can also be retrieved melting or updating existing CIM's in organisation repository.

8.2 Define/Update runtime environments

This step define the runtime environments in which the service application should run. A runtime environments is a platform defined as a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns. Any application supported by that platform can use without concern for the details of how the functionality provided by the platform.
As known, the CIM will be transformed into a Platform Independent Model. The resulting PIM has to be targeted to a platform to complete the build process. Therefore a detailed model of the platform is needed.
Of course all test environments, including unit test and final production, will be defined inside the running environments.

8.3 Identify Existing Models and Common Patterns

The main challenge of the design is the transformation between the different models. In traditional approaches these transformation are mostly very inefficient because no formal models are used. Without formal models it isn't possible to define a formal transformation which can be (partly) automated. In practice the process from CIM to PSM might be a lot more complex. Gaps can exist between models not small enough to perform a direct transformation. In that case many interrelated models may consist on different layers of abstraction. Within this global set of models horizontal transformations may occur within a single layer of abstraction. This happens if models already exists and are identified for reuse and adapted to new requirements.

The repeating patterns are identified within the models. These patterns often occur either because of the consistent use of an architectural style or because of the requirements of the runtime platforms. Common patterns are compared with existing models, making any necessary small adjustment to their architecture to exploit what is already available.

8.4 Define/Update design model

The transformation of a PIM to a PSM will be done by a technical specialist. The resulting PSM can be an implementation if it provides all the information needed to. Between the models gaps can exist not small enough to perform a direct transformation. A transformation is the manual, semi-automatic or automatic generation of a target model from a source model, in accordance with a definition of transformation. It is a collection of transformation rules that describe how to transform a model specified in a source language into another model specified in a target language. Approaches that transform models can be: marking, transformation meta-model, model transformation, and application of patterns merger models (model merging).

Within a set of models horizontal transformations may occur within a single layer of abstraction. By example, a PIM is transformed in a more detailed PIM several times. These horizontal transformations are an addition to the vertical transformation of models across the layers. At the current level of MDA, the model of entry is often the PIM and the target model is often PSM. However, a transformation can have a PSM as a source and a target PSM.

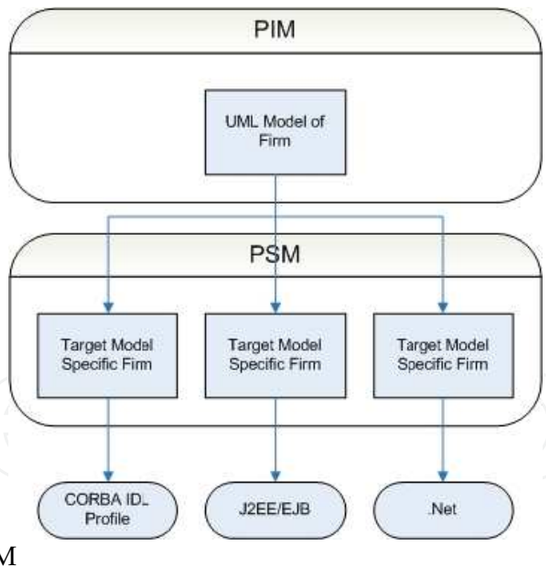

Fig. 7. From PIM to PSM

8.5 Design refinement, code, test

The resulting PSM can be an implementation if it provides all the information needed to construct a system and to put it into operation. The design artefact has the final refinements, if necessary.

Service orchestrations are modelled from a system's point of view and are used to implement new composite services based on other services, if appropriate. The coding and testing activities are executed.

9. Stage IV. Trial

The Trial stage is a validation of the product's design and features in use. Software prototypes are tested within the firm to determine that no technical flaws exist. In parallel, an agent test of the product is conducted. The object is to identify design defects, and, in particular, modifications needed to improve business agents' acceptance. The trial stage represents a "dry run" of all commercial facets of the software. The agents' tests provide the inputs to finalize the business model if issues appear in the new enterprise architecture. This means the identification of needed adjustments to the business model. A final estimate of market share and expected sales are two results of the test market.

10. Stage V. Launch

The launch stage involves startup of full or commercial production and the implementation of infrastructure assets of the business model. Post launch evaluation or control points at pre-designated times after launch provide benchmarks to gauge whether a software product is "on target." Such benchmarks include market share, sales volume, production costs, etc. Post-launch evaluations are essential to control the software product and to signal the implementation of corrective schemes to move the software product back on course.

11. Conclusion

People culture is still not aligned to service approach. Managers think in term of functional black box, with strictly defined boundary following a typical top-down engineering approach. This is a problem in the beginning of the innovation process. It is difficult to reason in term of knowledge and services sharing on common goals in a network. This has a heavy impact on the start-up of software process with an over-cost of 30-40% of effort. The real improvement was measured in the number of defects on high level requirements, giving a positive answer to the basic hypothesis that the software service model has to be integrated in the process of define/improve business model at run time and not after in a separate step.

New software product/service maintenance will never be risk free. Much can be learned about effective new software management from a review of the experiences in past projects. Many of these insights have been incorporated into the method presented. The benefits of the model are many. One result is that the process becomes more multidisciplinary. The balance between the internal versus external orientation becomes obvious. A second payoff is that interaction between agents is encouraged: many evaluation nodes demand diverse inputs from different groups in the company. A third benefit is the incremental commitment nature of the process: expenditures tend to be balanced with certainty level; each stage involves progressively better information and concurrently entails progressively higher expenditures; and risk is managed. Further, decision nodes and bail-out points are provided at each stage. Finally, the process is market oriented, providing for ample market

information and marketing planning, not only towards the launch phase, but throughout the entire process.

12. References

Cohen, S, 2007, Ontology and Taxonomy of Services in a *Service-Oriented Architecture,* The Achitecture Journal. MSDN. Journal 11. May 2007.

Elliot. (2002) Electronic Commerce: B2C Strategies and Models. Chichester: Prentice-Hall

TSH Teo, WR King. (1997). Integration between business planning and information systems planning: an evolutionary-contingency, Journal of Management Information Systems I Summer

Erl, T. (2007). SOA Principles. Retrieved June 18, 2007, from *SOA Principles, an introduction to the Service-Orientation paradigm:* http://www.soaprinciples.com, Dubray, J.-J. (2007). *Composite Software Construction.* InfoQ.com: C4Media.

Dubray, J.-J. (2007, December 04). The Seven Fallacies of Business Process Execution. Retrieved April 24, 2008, from InfoQ: http://www.infoq.com/articles/seven-fallacies-of-bpm, Dubray, J. J. (2007). *Composite Software Construction,* Lulu.com

Erl, T. (2007). SOA Principles. Retrieved June 18, 2007, from *SOA Principles, an introduction to the Service-Orientation* paradigm: http://www.soaprinciples.com/

Dubray, J.-J. (n.d.). Automata, State, Actions, and Interactions. Retrieved April 25, 2008, from eBPML: http://www.ebpml.org/pi-calculus.htm

Küster, J. M., Ryndina, K., & Gall, H. (2007). Generation of Business Process Models for Object Life Cycle Compliance. In G. Alonso, P. Dadam, & M. Rosemann (Ed.), *BPM 2007,* LNCS 4714 (pp. 165-181). Berlin Heidelberg: Springer-Verlag OMG. (2003, 06 12). MDA Guide Version 1.0.1. Retrieved from Object Management Group: http://www.omg.org/mda

Rappa, M. (2005). *Managing the Digital Enterprise,* North Carolina State University Thompson, J. D. (1967). *Organizations in action,* McGraw-Hill.

King, W.R., and Teo, T.S.H. (1997a), *Integration between business planning and information systems* planning: *validating a stage hypothesis,* Decision Sciences, Vol 28, Number 2, p 279-308.

King, W.R., and Teo, T.S.H. (1997b), *Integration between business planning and information systems planning: An evolutionary-contingency perspective, Journal of Management Information Systems,* Vol 14, Number 1, p 185-214

Engineering the Computer Science and IT

Edited by Safeeullah Soomro

ISBN 978-953-307-012-4

Hard cover, 506 pages

Publisher InTech

Published online 01, October, 2009

Published in print edition October, 2009

It has been many decades, since Computer Science has been able to achieve tremendous recognition and has been applied in various fields, mainly computer programming and software engineering. Many efforts have been taken to improve knowledge of researchers, educationists and others in the field of computer science and engineering. This book provides a further insight in this direction. It provides innovative ideas in the field of computer science and engineering with a view to face new challenges of the current and future centuries. This book comprises of 25 chapters focusing on the basic and applied research in the field of computer science and information technology. It increases knowledge in the topics such as web programming, logic programming, software debugging, real-time systems, statistical modeling, networking, program analysis, mathematical models and natural language processing.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen