Arun Kumar Chaturvedi, Kavita Verma / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 2,Mar-Apr 2012, pp.1308-1315

Modeling Abstract BPMN from Business Driven Model

Arun Kumar Chaturvedi*, KAVITA VERMA**
*(Department of Computer Science, MD University, Haryana)
** (Department of Computer Science, MD University,Haryana)

ABSTRACT
We address the alignment [1], [2], [4] problem following the Business Driven Development model. A representation system for the first stage of BDD, namely, model is proposed. It is argued that the representation system should be a pure conceptualization of the business process, should abstract out important constructs of business processes, and should be able to represent both intra and inter organization processes. We use the notion of dependency graphs developed in the generic method model as our representation system and show that it meets our requirements. We consider three intra-organization processes and then put them together in an inter organization system, the supply chain system, to illustrate that the proposed representation can handle both kinds of systems. Through this example, we also show that the last stage of BDD, analyze and adapt can also be facilitated by the proposed representation.
Key Words—Business Process, Model, Alignment, Adaptation.

I. INTRODUCTION
In today's increasingly tight economy, business processes undergo constant change and the enterprise needs to swiftly adapt its strategies to reflect these changes. The inherent problem with the enterprise business process is that it suffers from a lack of agility to match the pace at which the business needs to change in order to keep up with the market trends and competition. In order for enterprise information systems to survive and adapt to a controlled environment and to react to the fast-paced change in business processes, IS needs to enhance its capability and maturity to align itself with the business demands. IS must move away from creating IT-centric solutions and move toward creating solutions that realize one or more business process. Business-driven development (BDD) is a methodology for developing IT solutions that directly satisfy business requirements and needs.

I.1 THE NEED FOR BDD
Traditional applications and architectures are not able to keep up with business innovation, primarily because the processes are not adaptable to on demand business needs. Business requirements often get transformed into siloed IT projects that cannot work together; reusability between artifacts created for different IS is often very low. Creating applications that are flexible enough to react to the unknown requires a more systematic approach toward application development. With the business not able to create IS functionality that is capable of reacting to the unknown; it has traditionally been very difficult to justify the deeper budgetary requirement to create flexible IT applications like COTS [5], [6], [7], [8], [9]. The traditional inflexibility of application architectures makes even small improvements so expensive that they become virtually impossible to justify.

A mechanism[3] needs to be devised by which IT efforts[12] are interlocked with business strategy and requirements through an execution framework that is standardized, well understood, and can be executed repeatedly and successfully. The enterprise might achieve business flexibility through IT by modeling the business processes that collectively define the way the business executes. The first thing to do is model a business process through its constituent process steps. By measuring a business process or a key use case through return on investments (ROIs), key performance indicators (KPI), or other metrics, the enterprise can use these business process models (BPMs) as an essential mechanism to communicate the business needs to the IS realm. The business and IS can significantly bridge the communication chasm by using well-articulated BPMs that create a link between what the business needs and what IS implements and delivers.

While the starting step for BDD is the creation of BPMs, the IT solution structure also needs to adapt to using the BPMs as input artifacts to the design and development phases of the software development life cycle. The IT architecture needs to be able to design and implement the process activities as software components or services.

By using BDD, the enterprise models provide new business processes (when conceptualized) to the IS. Analysis of the new process might reveal either that software services might already exist to address the need and the only work effort required is to wire the existing software services to realize the new business process, or it might reveal that the enterprise needs to implement

new software services and add them to the IT service portfolio. Similarly, if changes are needed to an existing process, the BPM is revamped to reflect the change and delivered to IT for subsequent technical revision based on which services might need to be enhanced or modified.

A BDD approach helps increase the agility of the business and also helps prioritize and align IT initiatives with business imperatives. It also indirectly helps in simplifying the process of cost justification for IT budgets within an enterprise.

II THE EXECUTION MODEL

Enterprise IS should strive to bridge the gap between business needs and IT solutions and also be agile and responsive in creating IT solutions. This need has led to the development of a Services-Oriented Architecture (SOA), which provides an IT framework along with a set of principles and guidelines to create IT solutions as a set of reusable, composable, and configurable *services* that are independent of applications and runtime platforms. Transitioning an enterprise to SOA requires a BDD approach that uses business goals and requirements to drive downstream design, development, and testing. This promises to create composite business applications by reusing existing or newly created services, which helps to create adaptable and flexible business solutions. It also brings a much needed flexibility in enterprise IS and helps to align IT solutions with business needs.

. Figure 1 depicts the flow of activities that define the high-level steps of BBD.

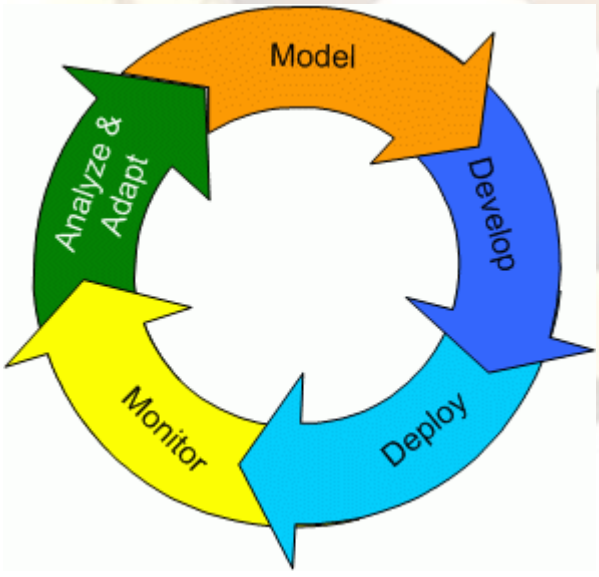

Figure 1. The execution model

The first step is to model the business processes that need IT enablement. It is advisable to start by modeling the key business processes and, using the outputs of the modeling activity, to communicate the business requirements to the IT domain

Once the processes are modeled [10], [11], the outputs of the models can be used as inputs to the requirement gathering phase of an initiative. The activities or process steps that make up a given business process model can be analyzed to form the basis of use case modeling. Developing use cases is a significant step in the requirement gathering phase of a project. Based on the use cases, the application architecture is structured and the enterprise services are identified, designed, developed, and subsequently wired together as service composites that realize the business processes. After development, the project moves to the deployment stage during which the developed components are exposed as publishable, location-transparent, and discoverable services. These software services are deployed to an execution runtime, such as an application server.

In post deployment, the project enters the monitoring or management phase. Once they are up and running, business processes can be monitored for real-time performance and data capture, reporting, and analysis.

IT solution meets the needs of the business as defined by a service level agreement (SLA).

Data obtained from the run-time monitoring is analyzed against the expected SLA or other benchmark performance metrics and criteria. The captured information is provided to the architects, designers, and developers who analyze the data and find out innovative ways of optimizing or improving the process through enhancements and performance tuning of implementation code. Sometimes the changes might also be made by the business users by changing business rules using external interfaces, which requires no code changes. If the analysis suggests changes to be made in the business process, the corresponding process models can be modified and the same steps (that is, develop-deploy-monitor) repeated to enhance the implementation. This completes the execution loop with analysis and process adaptation techniques feeding back to the modeling step. This mechanism helps both the business and IS to adapt to the changing business needs with quick turnaround time.

III BUSINESS REQUIREMENTS ANALYSIS

The first and foremost step during any IS initiative is to understand the business requirements [15]. The high-level business requirements reside in the minds of the key business stakeholders and inside existing legacy systems. Unless they are documented and signed off, the project itself might have quite a few unknowns at the very onset. Getting the time of the business executives is often a challenging endeavor, but it is a necessity and needs to be well planned and executed. The following (at a minimum) need to be documented:

1. Business vision
2. Business goals (long- and short-term) that realize the enterprise vision
3. High-level business requirements that help attain the goals
4. Problems with existing business processes (such as customer pain points, high costs, schedule issues, and so on)

It is also very important to have an understanding of the high-level business functions that a given business domain is expected to provide. Having a business domain matrix with its associated high-level business functions is a good way of concluding the business analysis activities.

IV BUSINESS PROCESS MODELING

BPM is the technique used to visually model a business process through a sequence of activities, use cases, and decision points. The purpose of BPM is to create fully executable models that an engineering group can implement as technical services. BPM involves the activities of modeling the *as-is* and *to-be* business processes and allocating resources to implement each process. Process optimizations are performed through simulations that help in attaining the *ideal* business process state. The to-be state is finalized as a first feasible step -- both from budget and timeline -- toward the ideal process state.

BPM techniques are used to model the aspects of behavior, organizational structure, and business domain objects. Each task is assigned to a role. A role is an entity (a person, computer, or any other type of actor) or group of entities that have the same rights and obligations with respect to performing a task or a group of tasks. A role might be assigned to any number of tasks and an entity might act in any number of roles.

Each business process, when analyzed, can be represented as a sequence of activities or tasks. A task is the smallest unit of work that makes sense to a user.

V NEED OF BPMN

The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts who create the initial drafts of the processes, to the

technical developers responsible for implementing the technology that will perform those

processes, and, finally, to the business people who will manage and monitor those processes.

BPMN will also be supported with an internal model that will enable the generation of executable

BPEL4WS. Thus, BPMN creates a standardized bridge for the gap between the business processes design and process implementation. BPMN defines a Business Process Diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations. A Business Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance.

VI RESEARCH PROPOSALS

Traditional systems development life cycles assume that alignment shall be looked after in upstream activities. A major role is played by the requirements engineering stage. However, the alignment issue is not explicitly brought out in these life cycles. In contrast, the life cycle proposed in Business Driven Development, BDD attempts to address this issue. BDD considers alignment as a top down activity. It proposes a life cycle that consists of five steps, (a) model, (b) develop, (c) deploy, (d) monitor, (e) analyze and adapt.

The modeling step consists of identification of business goals and modeling of business processes. The development and deployment steps convert the model into real implementations which are then monitored. Finally, change and adaptation is carried out in the last step. Thus BDD proposes that business and implementations must stay aligned at all times.

Business process models can also be represented in BPMN which also contains guidelines on their transformation into BPEL. Nevertheless, *the question is as to what constitutes a good representation system for representing business process models at the model level. We formulate the following requirements of such a system:*

1. *The most basic requirement is that the representation system should be independent of any implementation details. It should be a pure conceptualization of the business process.*

2. It must be at a high enough level to form a bridge between business and system analysts. A broad high level view should be represented that abstracts out the important features of processes. Thus, features like deadlines, and other standard features like long and instantaneous processes, parallelism, choice etc. should be represented but in a simple, abstract form.

3. The representation system should be able to represent both intra-organization and inter organizational processes.

4. It should facilitate movement from the analyze and adapt stage of BDD to its model stage.

We propose a representation system that attempts to meet these requirements.

Our proposal is summarized in Fig 1 that shows the analysis model in two stages, a high level stage, which we call the Analysis Model Representation System, AMRS, stage followed by the BPMN stage. Thus, there

is to be a transformation from the former to the latter in the Model stage of BDD.

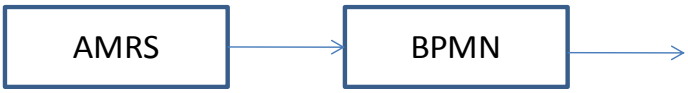

Fig. 1: The Position of Abstract Model Representation System

Now, for AMRS, we choose the notion of dependency graph developed as part of the generic method model [16]. This dependency graph establishes a successor-predecessor relationship between nodes. This is represented as an edge between a pair of nodes. Additionally, it labels edges with the properties of urgency and necessity. We will show here that this is sufficient to provide to us a high level conceptualization of a business process. It has facilities for an abstract representation of deadlines etc. of requirement (2) above. By labeling entire dependency graphs, we can extend these graphs to allow for inter organization process models.

VII THE GENERIC METHOD MODEL

The generic method model [16] builds a dependency graph that can be used to represent process models. Nodes of this graph are method blocks. Reference to Fig. 2 shows that there are three kinds of method blocks:

a. Method primitives are the simplest kind of method blocks. They are atomic. Loosely speaking, a method primitive has two parts, an argument part and an action part. The action part acts upon the argument part to produce the product. These two parts correspond to product primitive and process primitive respectively of Fig. 2. The product primitive is found in the product model. The process primitives correspond to the operations allowed. This gives to us the basic notion of an activity or task of a process model.

b. Complex method blocks are built out of simpler ones. These correspond to the notion of activities/tasks and their sub-activities or sub tasks.

c. Abstract method blocks establish a generalization/specialization relationship between method blocks. They allow us to sub classify tasks/activities into those that display similar properties.

Fig. 2 shows that there is a 'depends on' relationship between method blocks. A method block, MB1, that is dependent upon another, MB2, can only be enacted after MB2 has been enacted.

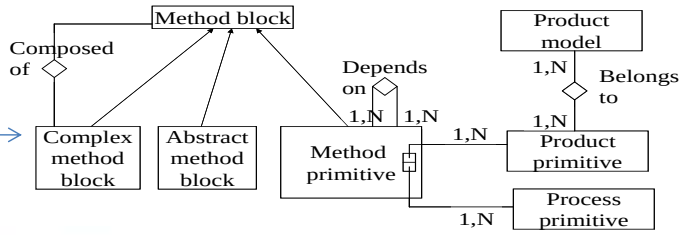

Fig. 2: The Generic Method Model

The generic model associates two main properties with a dependency, namely, urgency and necessity. Urgency refers to the **time** at which the dependent method block, MB_2, is to be enacted. If MB_2 is to be enacted **immediately** after MB_1 is enacted then this attribute takes on the value *Immediate*. If MB_2 can be enacted any time, **immediately or at any later moment**, after MB_1 has been enacted, then urgency takes on the value *Deferred*. Necessity refers to whether or not the dependent method block MB_2 is necessarily to be enacted after MB_1 has been enacted. If it is **necessary** to enact MB_2, then this attribute takes the value *Must* otherwise it has the value *Can*. Combining these two properties together, we get the four possibilities shown in Table I.

Table I: The Four Dependency Properties

Abbreviation	Urgency	Necessity
IM	Immediate	Must
IC	Immediate	Can
DM	Deferred	Must
DC	Deferred	Can

Given a set of method blocks and dependencies between them, the entire process model can be represented as a dependency graph. For example consider the dependency graph of Fig. 3. The IC dependencies are shown. Assume that the rest are IM dependencies.
The graph shows that method blocks O_{10}, O_{11}, and O_{14} are to be enacted in parallel after O_9 has been enacted. Similarly, once O_6 is enacted, O_7 and O_8 are enacted in parallel whereas a choice between enactment of O_{13} and O_{14} is to be made.

Enactment Initiation and Termination

A dependency graph has a set of nodes, called START, that have no edges entering them. This implies that enactment can begin from any of the nodes in this set. For example, for Fig. 3, START contains exactly one node, O, and enactment begins from this node.

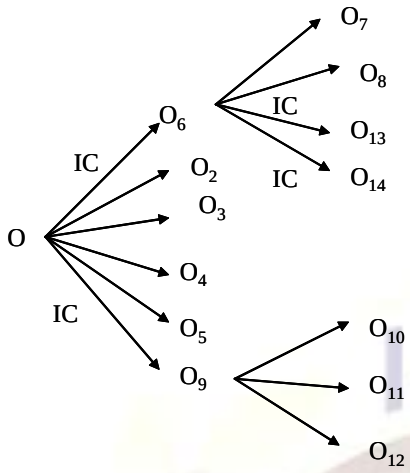

Fig. 3: A Dependency Graph with its properties

Now consider termination of enactment. We define a set STOP that contains nodes at which enactment can terminate. The following nodes belong to this set:

1. Nodes that have no edges coming out of them. For example, in Fig 3, O_7, O_8 and O_{10} to O_{14} shall be members of STOP.
2. Nodes that have edges leaving them but **all** these edges have Necessity = Can. Since the edges identify nodes which are optional and may not be enacted, it is possible for enactment to terminate. Notice that even if one of the edges has Necessity=Must then termination cannot occur since the node determined by such an edge is to be necessarily enacted. Again for Fig. 3, only two out of the six edges leaving O have Necessity=Can and the rest Must be enacted. Therefore, O is not a member of STOP. All edges coming out of O_9 have Necessity=Must. Therefore, it is not a member of STOP. Similarly, O_6 is not a member of STOP because only two of its four edges have Necessity=Can. Therefore, we have O_{11}, O_{12}, and O_{13} as members of STOP.

VII.1 SOME PROPERTIES

In this section we consider the abstraction capabilities of the generic method model. We point out a number of features of lower level notations that can be captured using dependency graphs. There are three interesting aspects here

- Dependency Types for representing parallelism, choice and deadlines
- Dependency graphs for representing business rules and process models

We consider each of these in turn.

Dependency Types

Let there be two edges emanating from an application chunk AC1 and leading to two others, AC2 and AC3 respectively. Then the dependency types allow us to represent:-

Parallelism*:* Let both the edges have the property, IM. The two method blocks must both be immediately enacted. Evidently, this is the situation of parallel enactment.

Choice: Let both the edges have the property IC. Any of the two application chunks can be enacted but the selection must be exercised immediately. A similar argument can be made if both edges had the property DC.

Iteration: This can be represented by introducing cycles in the dependency graph.

Hard Deadline: Let the edge to AC2 be DM. This says that AC2 must be performed perhaps, after a time delay. We see this as an initial recognition of a deadline that can be made exact, in the design stage of BDD, by specifying the time delay before which AC2 must be performed.

Soft Deadline: Now, let the edge to AC2 be DC. This says that performing AC2 is an option that can be exercised perhaps, after a time delay. Since the possibility of not performing the action is left open, in contrast to the DM case, we refer to this as a 'soft' deadline. As before, we see this as being made exact by specifying the time delay before which the choice is to be exercised. After this time, the choice cannot be exercised.

It is possible to mix these dependency properties in a dependency graph to express the appropriate execution property.

VII.2 MODEL STAGE OF BDD

Let there be an organization with its own procurement process that asks for quotation enquiries for items meeting specifications, evaluates the received quotations, issues purchase orders to selected vendors, takes delivery of items and finally, makes payment. There is no choice at any step and there can be a time delay between each step. Thus, the type of dependency between these steps is Deferred- Must, DM, as shown in Fig. 6.

Fig. 6: A Procurement Process Model

It can be seen that the procurement process has been represented at a high level of abstraction. There is no reference to messages, timers, signals etc. In other words the 'hows' of the process are de-emphasized. Similarly, the 'normal' flow of the process is depicted without any error paths or compensation paths.

Now consider another organization that supplies computer systems. It receives requests for system configurations, determines that the configuration asked for is indeed realizable and responds with a quotation. If any additional information/clarification is required then it obtains it and verifies it once again. Since it may happen that it has to order system parts that are missing, such missing parts are ordered and the system is assembled together, software installed and tested, if required, and the system is delivered. This process is shown in Fig. 7.

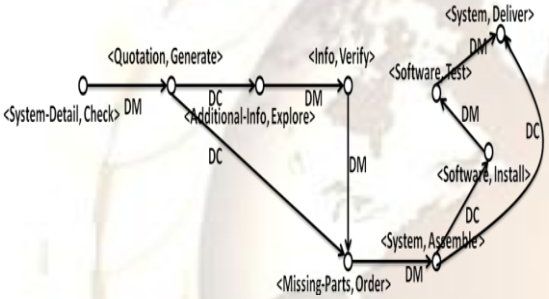

Fig. 7: A System Supplier

Notice that after <quotation, generate> there are two possibilities, either additional information is to be handled or missing parts are to be ordered. Both these actions can be done after a time delay. Thus, we get Deferred- Can, DC, as the type of dependency. A similar situation exists after the system has been assembled, i.e. at <system, assemble>. To handle the case where no software is to be loaded and a bare system is to be supplied, the dependency types are DC as shown in Fig. 7.Finally, consider a simple supplier process that sends out a quotation and upon receipt of an order delivers parts. The type of dependency is again DC. This is shown in Fig. 8.

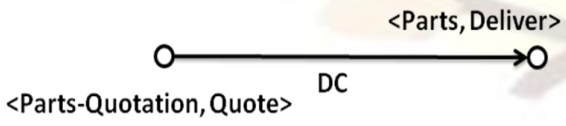

Fig. 8: The Supplier Process

The three process models considered here are all initial, first cut models that aim to broadly describe the process. No attempt is made to look into the details of the processes for example, the manner of initiation of activities by message, timer etc. is not considered. In our scheme, these details are left for the next stage where a full BPMN representation shall be made. This broad definition of the process model can be discussed among business and system analysts, agreed upon, and then taken to subsequent stages of development.

VII.3 ANALYZE AND ADAPT STAGE OF BDD

Now, consider the 'analyze and adapt' stage [13], [14] of BDD. Our attempt is to show that the basic process models developed above can be adapted to a different situation. Again, the new situation is described at a high level to lay a basis for subsequent development.

Let it happen that the three process models above are to be put together to form a supply chain. We label the three graphs presented earlier with the names of the roles of the organizations. We shall refer to the first as End User, the second as Systems Integrator and the third as Parts Vendor.

The first point of variation is at <specs, enquiry> of the End User. Whereas earlier this was self contained in the End User process, now it is possible to invoke the Systems Integrator as well, for the purchase of systems. In this case, the DM edge in Fig. 6 to <quotation, evaluate>, which was originally DM, shall be changed to DC. The edge from <spec, enquiry> to the System Integrator shall now be introduced and shall also be DC. As a result, this allows a choice between the two courses of action. A soft deadline is also imposed that limits the time before which enquiries should be received. This is shown in Fig. 9.

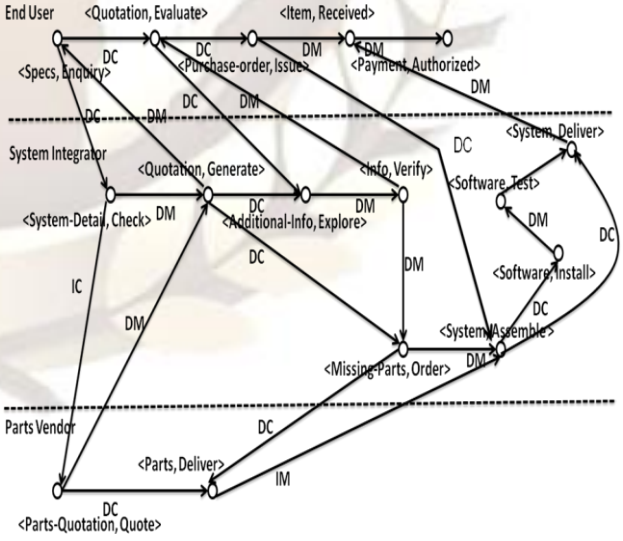

Fig. 9: The Supply Chain Process

Now, one moves to the next node in the End user process, <quotation, Evaluate>. Again the dependency

type to <Purchase-order, Issue> is changed to DC. The End user may ask for additional information from the System Integrator (DC) which is supplied after it is explored and verified. The evaluation activity can now be completed by the End User. This is shown by the dependency from <info, verify> to <quotation, Evaluate> which has the dependency type DM. This means that if the additional information is not supplied by a hard deadline then the quotation evaluation activity of the End User can ignore this quotation and proceed on. The next node is <purchase-order, issue> on which the System Integrator activity of <system, assemble> is dependent. The dependency type is DC. It is assumed that the System Integrator has already ordered all the missing parts in anticipation of the order being received. The next node of the End User process is dependent upon the system being received from the System Integrator, see the <system, deliver> activity of the System Integrator. The dependency type shows that delivery must be made within a hard deadline. The compensation activity, in the event of failure to meet the deadline is again not of interest at this abstraction level. Finally, the End user authorizes payment within a stipulated deadline.

The foregoing shows the interaction between the End user and System Integrator from the End User point of view. A similar exercise is carried out in the System Integrator process model. The interesting interaction now is that with the Parts Vendor. As soon as the <system detail, check> activity of the System Integrator is initiated, the Parts Vendor is asked to quote for the parts comprising the system. The quotation received is used to generate the quotation, <quotation, generate>, which is sent to the End User. Again, the dependency of the Parts Vendor and System Integrator interaction is DM, showing a hard deadline. Further interaction between the two takes place upon the System Integrator placing an order for the missing parts, <missing parts, Order>. This dependency is IM; the missing parts must be ordered and the next activity is assumed to be instantaneous. The <parts. Delivery> activity of the Parts Vendor is to be done within a specified hard deadline as shown by the IM dependency type between <parts, delivery> and <system, assemble>.

In this manner, cooperation between the three process models is set up at a high level. The points of change are identified by a walk through the three process models to set up the appropriate interaction.

VIII RELATED WORKS

According to BPMN, process modeling is flow oriented. Nodes of these are events, activities, and gateways whereas edges are sequences or messages and may carry information as associations. In our case, the basic nature of the process as a flow is maintained. However, all our edges are of the same type. To handle intra process and inter process communication, we propose that all edges in AMRS are messages. Thus, any node sends a message to the other. In this sense our proposal resembles the message passing approach of object orientation. Additionally, each node resembles the signature of object orientation. It defines externally visible data and the operation that shall be performed on it. The difference is that there is no notion of a return type. As we develop our proposals further, we intend to examine whether or not polymorphism would be useful in our context as well.

Requirements engineering seeks to lay down a basis for system design. It abstracts away from the ‘hows’ of systems to showing why the system is like what it is. We consider our proposal as similar. We do not express the ‘hows’ of the system. Instead we aim to discover through an elicitation process (not yet developed by us) the nature of the process model and express it in the model stage of BDD. This model, as our supply chain example shows, can be the As-Is model that can be adapted to yield the To-Be model in the analyze and adapt stage of BDD.

Change management in requirements engineering corresponds to the last step, namely analyze and adapt, of BDD. Elicitation of changed requirements has been reported in[14] . A full proposal centered on the notion of gaps has been presented. The idea is to elicit gaps and then fill them in. A different perspective to change is presented in [13] where the notion of variability in requirements is introduced. As a result, enough variations are available from which the pertinent ones could be picked up for describing the new system.

In our proposal, change management is done by business and systems analysts performing a walk through the process model(s), identifying the points of change and changing nodes, edges, the action performed at a node, and dependency properties. The issue of variability is for us a lower level issue when developing a detailed representation of the process model. It is at this stage that variability is to be built into the process model.

IX CONCLUSIONS

We have shown that the dependency graph provides us a representation system that is a high level, abstract conceptualization of intra and inter organization systems. Since it is devoid of the ‘hows’ of the process, we believe that it forms a good interface between business and systems analysts. It acts as the ‘why’ of a process model that may be represented in BPMN. Indeed, we

believe that in the model stage of BDD itself, it should be transformed into BPMN before it is passed on to the develop stage of BDD.

We are working on developing guidelines for transforming AMRS into BPMN. Also we are currently exploring the implications of our object oriented view on process modeling. Finally, we intend to look into the elicitation process for arriving at the representation of a process model.

X REFERENCES

[1] Chan Y.E., Sabherwal R., and Thatcher J.B., Antecedents and Outcomes of Strategic IS Alignment: An Empirical Investigation, IEEETEM, 53, 1, 27-47, 2006

[2] Malik K., and Goyal D.P., IS Alignment and IS Effectiveness: Experiences *from* Indian Industry, IEEE, 96-100, 2003

[3] Henningsson S., Svensson C., and Vallén L., Mastering the Integration Chaos Following Frequent M&As: IS Integration with SOA Technology, Proc. 40th HICSS, 2007

[4] Lee J, Siau K, Hong S, Enterprise Integration with ERP and EAI, CACM , 46, 2, 54 – 60, 2003

[5] Navarrete F., Botella P., and Franch X., Reconciling Agility and Discipline in COTS Selection Processes, Proc. Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, 2007

[6] Fox G., Lantner K., and Marcom S., A Software Development Process for COTS-based Information System Infrastructure, IEEE, 137-142, 2008

[7] Keil M, Tiwana A, Beyond Costs: The Drivers of COTS applciaiton value, IEEE Software, 22, 3, 64-69, 2005

[8] Horowitz BM, Lambert JH, Assembling Off-the-Shelf Components: "Learn as you Go" Systems Engineering, IEEE Transactions of Systems, Man, and Cybernetics - Part A Systems and Humans, 36, 2, 286 - 297

[9] Fox G, and Lantner K, A Software Development Process for COTS Based Information System Infrastructure, Proc. 5th Intl. Sym. On Assessment of Software Tools and technologies, 133-142, 1997

[10] Castro Valeria de, Mesa J.M.V., Herrmann E., and Marcos E., A Model driven Approach for the Alignment of Business and Information Systems Model, Mexican Intl. Conf. on Computer Science, 33-43. 2008

[11] Henkel M., and Zdravkovic J., Supporting Development and Evolution of Service-based Processes Proceedings of the 2005 IEEE International Conference on e-Business Engineering, 2005

[12] Stein S., Kuhna S, Ivanov K., Business to IT Transformations Revisited, First Intl Workshop on Model driven Engineering for Business Process Management, Pautasso C. ad Koehler J (eds.), 1-12, 2008

[13] Umapathy K, Towards Co-design of Business Process and Information Systems Using Web Services, Proc. 40th HICSS, 172a – 172a, 2007

[14] Shah R., Goldstein S. M., and Ward P.T.,Aligning Supply Chain Management Characteristics and Interorganizational Information System Types: An Exploratory Study, IEEETEM, 49,3, 282-292, 2002

[15] Daneva M. and Wieringa R., A Coordination Complexity Model to Support Requirements Engineering for Cross-organizational ERP, Proc. 14th IEEE International Requirements Engineering Conference, 2006

[16] Fang K, Wu ACH, Tung Yang C, A Study of Information Systems Integration with the Structuration Model of Technology as Foundation, Portland International Centre for Mangament of Engienering and Technology, 1556-1563, 2007