INTEGRATION OF A SIMULATION PLATFORM INTO A BUSINESS PROCESS MANAGEMENT TOOL

İBRAHİM HALİL KANALICI

BOĞAZİÇİ UNIVERSITY

2008

INTEGRATION OF A SIMULATION PLATFORM INTO A BUSINESS PROCESS MANAGEMENT TOOL

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Management Information Systems

by

İbrahim Halil Kanalıcı

Boğaziçi University

2008

Thesis Abstract

İbrahim Halil Kanalıcı, “Integration of a Simulation Patlform into a Business Process Management Tool”

Information technology has already gone beyond being a supportive tool for businesses. Strong competition and increased costs force businesses act more efficiently and rapidly, so almost in every business, the operations are carried out in IT platforms where the processes are automated. These platforms are referred to as Business Process Management (BPM) Suites. With this motivation, most companies analyze and reengineer their workflows. At this point, it is vital to be able to monitor current or reengineered business processes with key performance indicators and keep the process flows under control. This introduces a challenge for the management to evaluate how any particular modification to any of the process workflows affects the performance measures. Therefore simulation becomes an essence in the contemporary management of businesses. In practice, major business process management softwares include some kind of simulation environment either as an embedded engine within the modeling component or through third-party integration. In this study a general purpose simulation platform is analyzed and a mapping algorithm is developed to integrate modeling capabilities of a BPM tool with this simulation environment.

Tez Özeti

İbrahim Halil Kanalıcı, “Integration of a Simulation Platform into a Business Process Management Tool”

Günümüzde bilgi teknolojilerinin kullanımı iş süreçlerine yardımcı olmaktan öteye gitmiş, süreçlerin tamamiyle otomasyonunu sağlamakta kullanılmaya başlanmıştır. Her geçen gün artan rekabet ve maliyetler, şirketleri süreçlerinin en az maliyetle ve en izlenebilir şekilde yönetilmesine zorlamaktadır. Bu ihtiyacı karşılamak için İş Süreç Yönetimi yazılımları geliştirilmiştir. Bu yazılımlar üzerinde şirketler iş süreçlerini modellemekte, bu süreçler içerisinde takip etmek istedikleri anahtar performs göstergelerini tanımlayabilmekte, organizasyonel yapılarını süreçlere entegre ederek manuel işlerimlerin daha hızlı bitirilmesini sağlamakta ve anlık ya da dönemsel raporlar alabilmektedirler. Değişen pazar şartları ve ekonomik etkenler dolayısıyla bu platformlar üzerinde kurulmuş süreçlerde zaman zaman değişiklikler yapılması gündeme gelmektedir. Ancak bir süreç, şirketin bir çok farklı departmanını, hatta alt yükleniciler, iş ortakları ve servis sağlayıcılar gibi bir çok kesimi kapsayabilir. Bu yüzden süreçler yeniden düzenlenirken ya da yeni bir süreç tasarlanırken, simülasyon kendisine çok bu platformlar üzerinde çok önemli bir yer edinmiştir. Mevcutta bir çok iş süreç yazılımı şirketleri, bu platformlarda yaratılan modeller için ya gömülü bir simülasyon motoru kullanarak ya da üçüncü parti bir simülasyon yazılımıyla entegrasyon sağlayarak müşterilerine simulasyon imkanı vermeye çalışmaktadır. Bu çalışmada da, bir simülasyon yazılımı ile bir iş süreçleri yönetimi yazılımı ele alınacak ve bu iki yazılımın beraber kullanılabilmesi için entegrasyon sağlanacaktır.

ACKNOWLEDGEMENTS

This study is supported by the Research Projects Fund (Project #08N302) of Boğaziçi University, Istanbul, Turkey.

I'd like to thank to Assoc. Prof. Aslı Sencer Erdem for her great support and commitment to this study.

I also would like to express my gratitude to my family for their never-ending love and support.

CONTENTS

CHAPTER ONE: INTRODUCTION

The term “Business Process Management” has been out there for a long time, but importance of the subject has recently been recognized as businesses started their transition onto integrated systems platforms barely a decade ago. This transition took many forms: stand-alone applications for specific purposes along with optional middleware to help data transfer across these applications; or some kind of a collaborative integrated software - an enterprise resource planning (ERP) software - that offers full integration capabilities across different functions of a company.

In a typical enterprise, organizations are built around discrete business units; such as marketing, financials, manufacturing and logistics where each unit has its own business systems and hierarchy. With aforementioned attempts for integrated systems, these functional units have gained automation and more control individually but the price of this automation has been the creation of new setbacks that hinder management of the end-to-end processes that extend across organizational and system boundaries (Silver, 2006). To achieve desired level of efficiency, responsiveness and to tackle compliance issues, organizations should have a good way of monitoring their processes and take proper action when needed. BPM tools and techniques have been developed to serve this need; to set better directions in terms of managing processes in a company.

In rapidly evolving business and technology era, BPM has emerged to handle ever-changing interactions between different parties and applications. The critical importance of leading organizations from old to new ways of doing business has been well understood by managers, but the challenge remains for them to justify and

accelerate this transformation. Static modeling tools like MS Visio has been widely used in modeling and studying business processes, but exclusion of stochastic behavior of entities within a process has always hindered the ability of process owners or analysts to predict how performance indicators and overall performance would be affected. Once transformed into dynamic models, static flowcharts and workflow diagrams would serve as a much better decision support tool. As M.W. Barnett (2003) from Gensym Corporation sets forth, "simulation in this case provides all the answers: it shows you how the answers are derived; it enables you to trace from cause to effect; and it allows you to generate explanations for decisions" (p.1).

Simulation obviously plays an important role in modeling and (re)designing any business process or in any other field of study. Simulation mainly allows users to determine how their process designs will work and provides a platform to assess risks before design engagements. It helps to identify redundant steps, bottlenecks and opportunities that are involved in complex business processes. Through some citations from relevant literature, Hlupic et al. (2007) list major reasons for the introduction of simulation into process modeling: i.e., the influence of random variables on process development can be investigated and re-engineering effects can be anticipated in a quantitative way by using the metrics like costs, cycle time, serviceability and resource utilization. Furthermore, process visualization and animation are provided, allowing multidisciplinary team members to understand the model and communicate about it. As modeling user interfaces have become so much friendlier, simulation may be used by those who have little or no simulation background. These reasons have all been recognized by companies through struggles in real-world cases and BPM software

vendors have started to develop simulation modeling features to meet these requirements. In a research conducted by Hall and Harmon (2005) as cited by Hlupic, et al. (2007), most modeling tools offer discrete event simulation (DES) capabilities either as part of the tool supported with a simulation engine or as a separate add-on module developed by themselves or through 3rd party software development companies.

The simulation capabilities of the BPM tools where the simulation engine is a part of the tool are generally very limited. This is because testing a model dynamically by using different scenarios with well-defined and approximated input distribution or even with historical data is not the main focus of the vendors of these tools. One the other side, users have not given as much thought and effort on getting help from simulation when designing or reengineering a business process as they have done in manufacturing and production systems. However, it has now been realized that organizations need to identify cost minimizing and value added activities in designing their processes.

The second option in incorporating simulation capabilities to a BPM tool is to integrate separate simulation software. The most obvious shortcoming of this option is the fact that purchasing a simulation tool increases costs. Secondly this requires the knowledge on both platforms. Yet one should note the tradeoff between the increased costs and the broader functionality achieved by integrating a third party simulation software.

In this study a platform is generated that comprises the integration of BPM tool and simulation software. The organization of the study is as follows: In the next chapter, some background information on BPM is presented, followed by a literature survey on

the use of simulation in the context of business process management. After BPM tool -Netflow-, and simulation tool -Arena- are shortly introduced, the comparison between modeling and simulation capabilities are compared and results are presented. Finally, after some remarks on software integration techniques, how integration is developed between Netflow modeling environment and Arena simulation environment is explained. There is also an additional chapter where a sample model is implemented both in Netflow platform and the proposed integration platform to provide better insight to readers. Discussions on the limitations of the proposed integration environment and suggestions regarding this kind of study are provided in the end.

CHAPTER TWO: BACKGROUND

What is BPM?

In academic literature there are various definitions of BPM, all of which converge to the same point of "improving the way of doing business and organization". DeToro and McGabe (1997) suggest that in a BPM perspective, organizations realize that chain of events is horizontal across functional units. Policy and direction are still set at the top, but the authority to examine, challenge, and change work methods are delegated to cross-functional work teams. Consistently, Elzinga, Horak, Chung-Lee and Bruner (1995) set forth that "many companies are engaged in assessing ways in which their productivity, product quality, and operations can be improved. A relatively new area of such improvements is BPM".

There are also organizations emerged to create methodologies and provide skills to help companies actively manage their end-to-end processes. Workflow Management Coalition (WfMC) defines BPM as the practice of developing, running, performance measuring, and simulating business processes to effect the continued improvement of those processes. Simply put, BPM is concerned with the lifecycle of the Process Definition. According to Association of BPM Professionals, ABPMP, BPM is a systematic approach to identify, design, execute, document, monitor, control and measure both automated and non-automated business processes to achieve consistent, targeted results consistent with an organization's strategic goals. BPM involves the deliberate, collaborative and increasingly technology-aided definition, improvement,

innovation, and management of end-to-end business processes that drive business results, create value, and enable an organization to meet its business objectives with more agility.

Components of BPM

The definition given by ABPMP, along with its emphasis on innovative management of cross-functional processes as in the case of other definitions rendered by academia, also draws on important aspects of potential BPM suites. These aspects can be reflected in nine automated components that are required to exist in a fully capable, functional-wise complete BPM tool as stated in the book "Business Process Management: Practical Guidelines to Successful Implementations" by Jeston and Nelis (2006). These components are shown in Figure 1 along with an illustration of full process lifecycle.

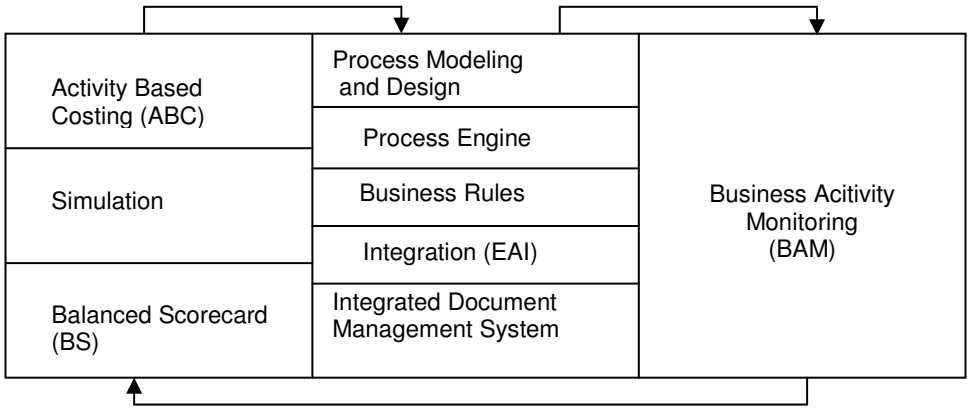


Figure 1: Components and interaction recycle in a BPM tool

Though the relationships between these components might be interpreted in different ways with respect to the way of approach, the identification of components is usually similar to that appears in (Jeston and Nelis, 2006). Process Modeling and

Design component enables many users collaboratively design process models at different times or simultaneously from different terminals. Process Engine component simply automates standardized tasks within a process and directs flow instances to people with respect to organizational structure. Business Rules component provides a pool of parameters and operands to construct constraints in a model without making significant programming effort. Enterprise Application Integration (EAI) is especially useful when there are other software in use and when a process necessitates sharing information back and forth with other software. Document Management component enables users to use documents within a workflow and store them on server with better performance. Business Activity Monitoring (BAM) component is becoming very popular due to the fact that it enables managers to track key performance indicators defined in a model on customized dashboards in real time. Simulation component provides an analysis platform in which users have the chance to see how efficient a process workflow is and whether there are any setbacks that would create costly problems after deployment. Activity Based Costing (ABC) component enables managers to track activity costs more accurately and helps in taking strategic decisions regarding cost accounting. Finally, Balanced Score Card (BS) component is important because it provides the means of associating performance indicators with company's strategies and monitoring whether workflow outputs are in alignment with company's strategic goals.

Considering the input-output relations between these components, the five components in the middle column are the fundamental ones that provide data for BAM. The monitoring activities indicate opportunities to redesign the processes across the

divisions. ABC, Simulation and BS components include recently added features that enhance the process redesign capabilities of the BPM Tools. These components serve management needs to keep daily business in alignment with strategic goals, to monitor associated costs more accurately and to reengineer or implement new processes to meet changing market requirements. In this picture, simulation, when used correctly, provides great decision support and comfort in judgment of new business engagements.

CHAPTER THREE: LITERATURE SURVEY

In literature, there are many studies on the use of simulation in the context of business process modeling. Especially in the area of manufacturing and production systems, simulation has been widely used to be able to walkthrough intricate processes of machinery and workstations. As cited by Paul et al. (1999), Halpin (1999) indicates the uses of simulation in various areas such as military operations, economic studies, healthcare, computer systems design, construction applications, transportation facilities and sociological analyses. In this study the emphasis is on dynamic modeling of business processes where simulation has most recently surfaced. While rendering quite a few of real life applications in this area, Paul et al. (1999) state that holistic approach must be adopted in designing processes that have multiple organizational perspectives (e.g. various departments in a company) and mapping of business processes to workflows is crucial in this aspect. This statement summarizes the need for BPM systems which would integrate process definition, modeling, administration and evaluation under single environment. However, even with the rise of BPM systems that would bring along executable models, Paul et al. (1999) argue that simulation will not become a mainstream modeling tool and they identify important factors that give rise to this fact. First one is the lack of interface capabilities between simulation software and other modeling tools, which is still a gap that this study tries to narrow. Second one is the requirement of a certain amount of expertise for building models with most simulation languages, which is obsolete at the time because there are quite a few

simulation packages that offer modeling capabilities without use of programming skills (Paul et al., 1999). Therefore, interface issue remains the one to be investigated.

The lack of interface issue has been recognized by both the academia and the industry, leading to studies on integration between modeling tools and simulation. Harrell and Field (1996) present an integration study between Design/IDEF, a process mapping tool and ProModel, a simulation tool. Through modifying the underlying code of process mapping tool, they manage to add entity attributes, input buffers, ways to model different entities and additional data fields to capture dynamic information. Similarly, Srinivasan and Jayaraman (1997) come up with a model generator developed in SIMAN simulation language, called EMF-SIMAN. EMF stands for Enterprise Modeling Framework and consists of a methodology that distinguishes between the important aspects of an organization: function, information and dynamics.

There is also a more recent work published about simulation of business process models generated in BPM tools. In this paper, Waller et al. (2006) presents a Java based simulation engine, L-SIM, which would be able to simulate processes constructed with business process management notation (BPMN)[*]. This commercial product provides specific objects to represent each BPMN shape as well as additional objects to support simulation data analysis, enabling easy translation from a static diagram to simulation environment. This engine may be used with any BPMN compliant modeling environment via a BPMN compliant application programming interface (API) and currently implemented in several BPM tools such as IDS Scheer, Telelogic and

[*] Developed by Business Process Management Initiative (BPMI) and is now being maintained by the Object Management Group (OMG) after the merger of these organizations in 2005, Business Process Management Notation (BPMN) offers a standardized graphical notation for drawing business processes in a modeling environment. (Retrieved on September 30, 2008, from http://en.wikipedia.org/wiki/BPMN)

PNMSoft (http://www.lanner.com/en/l-sim.cfm). The essence of this study lies within the conversion of inexplicit dynamic behavior which may exist in complex models, such as queuing procedures. In BPMN, there isn't any object that represents queues and it wouldn't be realistic to expect all business process modelers to understand the nature of simulation and put in the queues with appropriate procedures in simulation model generated by L-SIM. For this reason, model generated via L-SIM would be a result of much detailed design activity and analysis, which introduces a fundamental challenge to making of such an interface (Simulating for Success, n.d.).

The work presented in this study aims to assist users in simulation of business processes using a general purpose simulation tool and a BPM tool. Although the ultimate purpose is the same as other works presented in the literature, which is simulating process models, the methodology introduced is completely different and obviously specific to the BPM tool at hand. In addition, different integration techniques using this widely-implemented general purpose simulation tool are presented as well as a thorough discussion and analysis of alternative methods using different BPM tools.

CHAPTER FOUR: MODELING AND SIMULATION ENVIRONMENTS OF ARENA AND NETFLOW

Rockwell Automation's Arena

Arena, developed by Rockwell Automation, is a simulation and automation software based on SIMAN processor and simulation language. In this simulation tool, the user places different modules and data blocks (process, decision, conditions etc.) together to build the representative model of a real business process. Connection in between these modules specifies the flow of entities. Different modules have specific actions relative to entities, attributes, timing and flow; however with flexible programming such as built-in Visual Basic for Applications (VBA) events and expression customizing capabilities, the precise representation of each module and entity relative to real-life objects may be modified by the user. Statistical input/output data may be recorded, analyzed and reports may be generated not only with standard parameters, but also by parameters of the user's own definition.

All these features place a sound ground to select Arena for a simulation environment. But there are two other important aspects of Arena that enables it to be selected as a simulation platform to be integrated to a BPM tool. Arena integrates very well to Microsoft technologies as it contains VBA for specific, user-defined algorithms. It also includes ActiveX control that provides users the ability to reference an external application and use object variables from these external application's object model such as MS-Excel. On top of this, Arena supports open database connectivity (ODBC) which

enables users to connect right into an SQL Server to retrieve and manipulate data. Being Microsoft-oriented to this degree is important because the BPM software presented in this study is designed on Microsoft technologies, specifically on .NET platform (http://www.arenasimulation.com).

Bizitek's Netflow

Netflow is a BPM solution developed by Bizitek Bilgisayar Yazılım ve İnternet Teknolojileri A.Ş. in Istanbul, Turkey. It allows modeling of related workflows according to companies needs to automate workflows, follow the actions, check status reports, and see tangible results. This software consists of three main modules: Netflow Web Interface where the users log onto the system and carry out their role-specific tasks; Netflow Studio where the process owner models the process by using workflow modules and rules without doing any programming; Netflow Admin where dedicated personnel from IT department manages the whole system; role definitions, hard parameters, exception handling, database processing etc. The underlying architecture of Netflow that supports these three modules is illustrated in Figure 2:

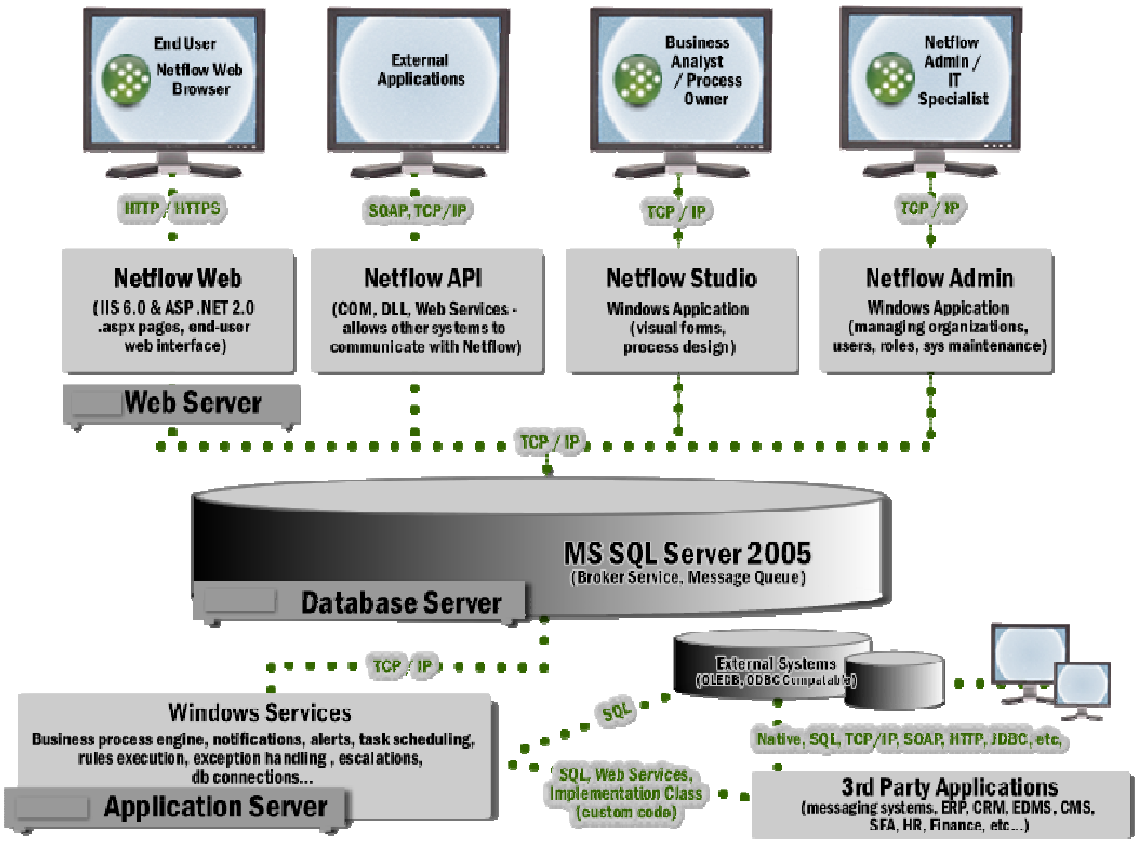


Figure 2: The Architecture of Bizitek's Netflow

Netflow runs on Microsoft's SQL 2005 Server and uses Windows services for most of the functionality. It has been developed with a service oriented framework that integrates the products with the same type of architecture and includes them into the workflows. It is capable of integrating with a wide range of third party products and can communicate with any OLEDB (object linking and embedding database) or ODBC compatible databases as well as other systems through web services and/or application programming interfaces.

In Netflow Studio model, the process owner models the process and embeds the rules so that each instance follows through a pre-established logic. In doing so, however, the process owner doesn't know if the real process is mapped onto the system

accurately and whether all different cases are covered. Netflow offers a “test system” where each process can be deployed first and tested against certain scenarios customer possesses before live deployment takes place, but this way of testing the models may require many resources and time. Besides, testing is limited only to a number of runs of certain scenarios which may not project accurate results for identifying bottlenecks or utilization rates of resources.

Netflow, as a matter of fact, has built-in simulation functionality where inter-arrival times and service times (times required by organizational resources) are randomly generated from a normal distribution with a given mean and standard deviation. There is also the possibility of generating these times using different parameters (mean, standard deviation) by assigning each case discrete empirical probabilities. However, merely randomization of input parameters would not suffice for getting useful simulation results. Lack of elaborate data collection on different attributes of entities and resources, lack of interfaces of data input and most importantly lack of flexibility as compared to other simulation and general purpose programming languages may create problems when designing complex models within different areas of business (http://www.bizitek.com.tr)

Before presenting how integration is constructed between Netflow and Arena, it is important to present a short comparative study of Arena’s simulation features and Netflow’s simulation capabilities by using the framework developed by Hlupic et al. (2007).

Hlupic et al. (2007) basically put together a list of guidelines that would assist managers in selecting a simulation package to be deployed in the context of business process change projects. The authors in their discussions present the most important simulation features and define the criteria for the evaluation of these features. One can definitely pursue the very same guidelines and criteria within BPM implementations because process change and reengineering is an essential part of managing business processes.

These criteria were derived from previous experiences and literature survey and were further tuned to cover only those features required for business process simulation issues (in other words, not very exhaustive to cover manufacturing systems). Moreover, Hlupic et al. (2007) asserted that levels of importance are vastly dependent on user choices, level of modeling details and area of use. When modeling complex processes or using the software for educational purposes, some criteria such as modeling flexibility, ease of debugging and access to source code would attract more attention than ease of learning and user friendliness or vice versa. Consistently, if top management requires output analysis of simulation results, then visual aspects and output capabilities would be more significant.

Hlupic et al. (2007) defined four main groups of categories and features within each category were further classified into subcategories with respect to their character. The main categories and the related subcategories are shown in Table 1.

Table 1 - Overview of the Framework Presented by Hlupic et al. (2007)

Hardware & Software Considerations	Coding Aspects	Software Compatibility	User Support	Financial and Technical Features	Pedigree
Modeling Capabilities	General Features		Modelling Assistance		
Simulation Capabilities	Visual Aspects	Efficiency	Testability	Experimental Facilities	Statistical Facilities
Input / Output Issues	Input and Output Capabilities			Analysis Capabilities	

Hardware and Software Considerations

Coding aspects, included in the hardware and software considerations category, would seek whether additional possibility for programming or at least access to source code is provided. This subcategory also would check whether global variables could be used across the model and whether built-in functions are provided. In Arena, Visual Basic editor comes with ThisDocument object which has a collection of standard events that are triggered at different times by Arena; which provides great additional programming features for users. Users are not allowed to access to source code in neither of the simulation environments but Arena lets users utilize global variables and many built-in functions such as TNOW to describe the current time whereas Netflow simulation engine provides users with rule construction using available variables and

attributes of an instance through which it supports basic programming concepts and offers built-in functions.

Arena outruns Netflow in the software compatibility category in the context of simulation capabilities. Technology exploited by Arena for application integration is the ActiveX Automation (formerly knows as OLE automation). Fundamentally, this technology allows applications to control each other and themselves via a programming interface, VBA in Arena case. The user may use MS Excel or MS Access to import/export model and operand data. Arena also has full compatibility to different input/output statistical analyzer software where inputs into the model and outputs are verified and validated. Netflow utilized Microsoft Office SharePoint Server features to integrate with commonly used desktop applications; however this integration only works during execution of workflow engine, not the simulation engine. However if this capability was extended to cover simulation engine as well, then integration capabilities may be comparable.

In user support category, Arena again has remarkable superiority. Rockwell Automation offers consultancy services and training courses across the world and they have been carrying out these activities for many years now. Obviously much expertise has been accumulated over the years and this expertise is clearly reflected in the so called “SMART” files; sample models each of which illustrates different modeling techniques and simulation capabilities of Arena. Simulation in Netflow, however, has only recently been developed and is not even included in the current version’s (version 4.9) help documentation. This statement also invokes that Arena gets much higher points in the category of pedigree.

Since the simulation platform of Netflow may not be deployed separately without having to purchase BPM software itself, it wouldn't make any sense to make comparison in the financial and technical features category.

Modeling Capabilities

Regarding the modeling capabilities category, actually the comparison would not expose useful information because the scope of this study is to keep modeling environment within Netflow and yet transform models into Arena environment to simulate them. Nevertheless, this comparison will shed some light on why authors have chosen to create an interface between Netflow and Arena instead of having users do modeling also in Arena manually and run simulation afterwards.

Experience and education required for the use of Arena needs to be much greater than for the use of Netflow modeling and simulation features. Netflow has an unstructured modeling environment where each node is represented with the same block and these blocks are differentiated using codes or names. As long as the rules on the connectors are defined accurately and nodes are connected elaborately, modeling is like free diagramming. Due to structured modeling environment of Arena and sophistication of simulation engine written in Siman programming language, ease of learning and user-friendliness are obviously compromised when compared to Netflow modeling environment. This is actually an important point in the favor of justifying the idea to keep modeling environment within Netflow along with single user interface and total cost of ownership.

Owing to more advanced modeling capabilities of Arena, it is expected to score higher from modeling assistance subcategory. With comprehensive online help and user forums, many different modules to meet the requirements of end-to-end processes in various industries and ability to allow simulation runs with different input parameters on the same model (model and data separation), Arena steps up higher than Netflow in this subcategory.

Simulation Capabilities

Within the visual aspects subcategory of simulation capabilities group, animation helps people with no simulation background visualize how the model performs and it is definitely a better representation of the process. However, visual aspect subcategory happens to be relatively less important because in the scope of this study, the animated simulation run is not included although Arena supports full animation of the model and has an extensive animation library to represent different entities, resources or activities. On the other hand, Netflow does not offer any animation features at all.

Efficiency of simulation software is as important as any other type of software. Although functionality (features offered) has more priority when assessing simulation tools in the scope of this study, these tools should demonstrate the ability to recover when exceptional inputs or situations occur. Simulation tools are also expected to have the capability to model a variety of complex systems and show characteristics which can save time needed for modeling and improve the quality of modeling such as

reusability, reliability and time scale for model building. As a general purpose simulation tool, Arena clearly outscores Netflow's simulation capabilities within this category.

Testability subcategory comprises criteria that basically seek for facilities for model verification. Logic checks and detailed error messages are important features Arena simulation software provides from the very first releases. Trace elements and step command may be used in Arena to track certain variables or attributes during run time as well as debugging by more technical users. Netflow also provides logic checks and error messages for its limited simulation features.

Experimentation subcategory basically evaluates a simulation package's facilities that are required for improving the quality of simulation results and for speeding up the process of designing experiments and of experimentation itself. Arena provides users the ability to enter Warm-up period while setting up run time parameters so that steady state analysis could be done. Users may also open "Breakpoint Properties" dialogue to pause the system whenever a specific simulation time, calendar time, condition, entity or module is reached. After a validated model has been constructed, users have the chance to run the simulation in batch mode to collect statistics out of the model. Speed adjustment for the flow of entities is also available during runtime. None of these features are provided by Netflow simulation engine.

Due to the stochastic nature of the simulation models, good statistical facilities are essential. Criteria in this subcategory observe the range and the quality of the statistical analysis facilities offered by simulation packages. Netflow merely offers random number generation out of a normal distribution with given mean and standard

deviation and may be further randomized with discrete empirical coefficients from zero to one. On the other hand, Arena comprises all of theoretical distributions along with the possibility to construct user-defined distributions to generate random variables. Input analyzer tool that comes with Arena installation provides great support for distribution fitting for a given stream of numbers. This powerful and versatile tool can be used to determine the quality of fit of probability distribution functions to input data. It may also be used to fit specific distribution functions to a data file to allow you to compare distribution functions or to display the effects of changes in parameters for the same distribution. In addition, the Input Analyzer can generate sets of random data that can then be analyzed using the software's distribution-fitting features. Also Arena generates standard reports at the end of each simulation run defining critical points and bottlenecks along with confidence intervals if enough input data is presented into the system.

Input / Output Issues

Regarding the input/output issues, criteria included in the subcategory -input and output capabilities- examine how the user could provide data into the package and type and the quality of output reports provided by the package. Criteria for the statistical facilities subcategory above also provide sufficient evidence that Arena offers superior statistical analysis tools.

Criteria under analysis capabilities subcategory investigate a simulation package's facilities to provide scenario analysis and optimization for better decision

making. The scenario analysis in Netflow could only be done by changing the model for every different set of data because there is no model-data separation. There is also no optimization facility within Netflow. However, Arena offers an optimization add-on facility, named OptQuest, which allows users to define various system inputs (controls and constraints) and desired system outputs (objective functions), then guides the process of selection of system inputs, and then executes the model by running several scenarios for each set of inputs in order to achieve the desired system outputs.

This comparison framework is used in the scope of this study to justify the choice of a 3rd party simulation package and interface development in stead of using or enhancing Netflow simulation capabilities within Netflow development environment. In order to show by how much Arena outperforms Netflow in the context of modeling and simulation, comparison tables for each category in the framework is presented in the following tables.

Table 2 - Hardware & Software Considerations Category Results

Comparison of simulation environments -->			Arena	Netflow
Hardware & Software Considerations	Coding Aspects			
		Programming flexibility	Provided	Not Provided
		Access to source code	Not provided	Provided
		Global variables	Provided	Not Provided
		Built-in functions	Provided	Provided
		Support of programming concepts	Provided	Provided
	Software Compatibility			
		Integration with spreadsheet packages	Possible	Not possible
		Integration with statistical packages	Possible	Not possible
		Integration with DBMS	Possible	Not possible
		Integration with legacy applications, ERP..	Not possible	Not possible
		Integration with WFM systems, BAM systems	Not possible	Possible
	User Support			
		Documentation and tutorial	Provided	Not provided
		Consultancy	Provided	Provided
		Training courses	Provided	Provided
		Package maintenance	Provided	Provided
		Demo models, libraries	Provided	Not Provided
	Pedigree			
		Age	High	Low
		Spread	High	Low
		Reputation of supplier	High	Low
		Availability of references	High	Low

Table 3 - Modeling Capabilities Category Results

Comparison of simulation environments -->			Arena	Netflow
Modeling Capabilities	General Features			
		Experience and education required for software use	Substantial	Some
		Ease of learning	Not easy	Easy
		User friendliness	Low	Medium
		Representativeness of models	High	Low
		Formal logic	High	Low
		Simulation modelling approach (process-based, activity-based etc.)	Not provided	Not provided
	Modelling Assistance			
		Documentation notes	Provided	Not rovided
		On-line help	Provided	Not provided
		Modularity	Possible	Not possible
		Model and data separation	Possible	Not possible

Table 4 - Simulation Capabilities Category Results

Comparison of simulation environments -->			Arena	Netflow
Simulation Capabilities	Visual Aspects			
		Animation	Possible	Not possible
		Type of animation	Full Animation	No animation
		Animation with visual clock	Provided	Not provided
		Expressiveness and quality of graphics	High	No animation
		Graphic library	Provided	Not provided
	Efficiency			
		Robustness	High	Medium
		Level of detail	High	Low
		Model reusability	Possible	Not possible
		Model reliability	High	Low
		Time scale for model building	Large	Low
		Model chaining: linking outputs from different models	Possible	Possible
		Queueing policies	Provided	Not provided
	Testability			
		Logic checks	Provided	Provided
		Error messages	Provided	Provided
		Ease of debugging	Easy	Not easy
		Trace files	Provided	Not provided
		Step function (event to event jumping)	Provided	Not provided
		Dynamic display of elements (capacity, events, state etc..)	Possible	Not possible
		Display of the workflow path	Provided	Provided
	Experimental Facilities			
		Warm-up period	Provided	Not provided
		Breakpoints	Provided	Not provided
		Speed adjustment	Provided	Not provided
		Automatic determination of run length	Not provided	Not provided
		Automatic batch run	Possible	Not possible
	Statistical Facilities			
		Theoretical statistical distributions	Provided	Provided (limited)
		User-defined distributions	Possible	Not possible
		Random number streams	Provided	Provided
		Output data analysis	Provided	Provided
		Quality of data analysis facility	High	Low
		Distribution fitting	Provided	Not provided
		Confidence intervals	Provided	Not provided

Table 5 - Input / Output Issues Category Results

Comparison of simulation environments -->			Arena	Netflow
Input / Output Issues	Input and Output Capabilities	Input data reading from files	Provided	Not provided
		Quality and understandability of output reports	High	Low
		User defined output	Possible	Not possible
		Periodic output of simulation results	Provided	Not provided
	Analysis Capabilities			
		What-if-analysis	Provided	Not provided
		Conclusion-making support	Provided	Not provided
		Optimization	Provided	Not provided

CHAPTER FIVE: INTEGRATION of the BPM TOOL into the SIMULATION ENVIRONMENT

Techniques for Integrating Netflow to Arena

There are several techniques that may be used to build the interface to transfer model data from Netflow modeling environment into Arena runtime environment. Before delving into the details of these techniques, some it is vital to know automation and integration possibilities of Arena and Netflow.

The integration environment in this study does not expect users to use Arena modeling interface at all. Therefore automation in Arena should be utilized to make sure that Arena model conversion and simulation run would be carried out from outside Arena interface. For this purpose Arena comprises ActiveX Automation (the technology formerly known as OLE Automation), which allows applications to control each other and themselves via a programming interface. Generally, for an application to be automated using ActiveX automation technology, it must have an object model, which is a list of application objects that can be controlled. This object model is registered when the application is installed; so if one uses an automation programming language and want to utilize the application's functionality, one can establish a reference to its object model and program its objects directly. In the case of Arena, for example, Arena object model library may be referenced from an external application developed in Visual Basic, C++, Excel VBA etc., and Arena may be called from within this application without having to do anything on Arena interface.

A second technology that also takes advantage of ActiveX automation technology for integration is Visual Basic for Applications (VBA) in Arena. VBA is a component of Arena licensed from Microsoft Corporation. It is the same language technology that powers Microsoft Office applications, including Microsoft Word, Excel and PowerPoint, and third-party applications like Visio (by Visio Corporation) and Arena. It is also the same engine behind Microsoft's own Visual Basic version 6.0.

Incorporating VBA into Arena allows for tighter integration by extending the VBA interface to include Arena's own objects, methods, properties and events. The interface to VBA is a separate window, integrated with Arena, in which one can edit, design, and debug Visual Basic code and forms. With the inclusion of VBA in Arena, it would be possible to integrate Arena with other programs that support the Microsoft ActiveX Automation programming interface. Ultimately, users may develop their own program within VBA to generate simulation models that is fully functional within Arena environment or use VBA to read in operand parameter values and variable values from external data sources for models constructed using Arena modeling interface.

There is one important advantage of using VBA over an external environment like Visual Basic in that the VBA program has two-way communication with the model. The code written in VBA would control the model using Arena Object Model as well as respond to events initiated by the model such as simulation events fired during, before or after the active simulation run.

On the Netflow side, an application programming interface (API) exits that allows data transfer from external systems to execute transactions such start a workflow, to advance an entity one step further into the flow or to change the status of

users. It contains dynamic library link file (NetflowAPI.dll) with several object and corresponding methods to execute different function within Netflow, COM interface and WebServices interface. These interfaces are all intended to be used with workflow features of Netflow, not with simulation features. Besides, Netflow simulation environment will be replaced by Arena simulation environment in the scope of this study; therefore this API will not be utilized in this study.

Having mentioned technologies Arena and Netflow offer for integration and automation, the possible techniques using these technologies is listed here:

- Developing an in-process or out-of-process DLL (or EXE).
- Direct modeling in VBA using ActiveX automation
- Indirect modeling in VBA using ActiveX automation (Importing Model from Database)

The first method of developing a DLL or EXE file first requires some knowledge on add-in concept. An add-in is a separate program written external to Arena to automate Arena. This external program may have “exe” as extension if it is going to be run outside Arena, or “dll” as extension if it is going to be called from within Arena user interface. Performance (i.e., speed) is often compromised when automating applications externally (i.e., out-of-process), as is the case when creating an .exe that automates Arena, rather than from inside the application (i.e., in-process) as is the case with VBA. Depending on the program being developed, this may be more (or less) of an issue. Arena provides a development interface, called the Arena AddOn Interface

that allows external applications to be developed which run in-process to Arena, thus eliminating the performance issue when developing add-ins. After a DLL file is built using this development interface in an external application, it should then be moved to the Add-Ins folder located in the Arena installation directory. Once moved into this location, the DLL should then be registered using the regsvr32.exe supplied with your development tools. This will make it show up on Arena's Tools menu. Unlike an .exe, the ActiveX DLL must first be registered in this way in order for it to show up on the Tools menu. (Retrieved from Arena help files for Arena v.10 CPR 7.0 Academic Release)

The second method is direct modeling in VBA using ActiveX automation. In this technique, code is developed within VBA workspace and also run from here. It basically would retrieve data from SQL database where Netflow model data is located, and then parse this data to extract useful information about the model conversion. For data that is required to be manually added such as runtime parameters or any other model values for that matter, an excel sheet may be used in which also a start button may be placed to run the VBA code so that Arena interface is never used by users during this process (Arena interface may be called up during run time if required, by the way). In the same way, the reports generated after simulation run has been completed may be downloaded into excel sheets for user viewing.

“Import Model from Database” option, as the third technique, has been implemented via an ActiveX dynamic library link (DLL) that is installed in Arena’s add-on folder: smImportFromDatabase.dll. The object model of this DLL is registered

when files are installed. Through ActiveX, user can establish a reference to this object model and utilize the functionality of the DLL in another program.

For example, user can write an application that converts data stored in a particular database format to the Arena model database format. Then, as a second step, the program can access the functionality in smImportFromDatabase.dll to automatically import the Arena model database into Arena. This allows user to limit the coding efforts of the new application to database conversion. By taking advantage of the functionality in smImportFromDatabase.dll, users avoid duplicating the effort of actually creating a model in Arena.

In this study, the second technique was chosen, direct modeling in VBA using ActiveX automation. Because if the first technique was to be used, for DLL option users would have to open Arena interface after modeling is done in Netflow and then run the add-on from the menu. For EXE option, there would be performance problems which would be problematic for large models. The third technique sounds much easier considering limitation of coding efforts only with database conversion. However, for this study, this technique goes beyond mere database conversion because in Netflow, model data is kept in an XML file in a totally different format than Arena would accept and retrieving model data and slicing it into structured model tables for Arena requires also logic development. This logic development, as a matter of fact, would be done in VBA in the second method and flow connections would be set easily without worrying strict table structures if model was to be exported to database as in the second technique.

How Integration Works

As previously stated, Arena comes with an embedded VBA and Netflow's architecture fully supports Microsoft technology. Therefore, the mapping code is constructed in VBA. The workflow data is kept on SQL database and most of the necessary data retrieved is in Extensible Markup Language (XML) format. For the data to be retrieved from SQL database directly, OLE Automation and ActiveX libraries are enabled. When the XML data is to be parsed, then Microsoft XML library is enabled along with VBA library in Arena visual basic workspace. For the acquisition and manipulation of data in XML format, Document Object Model Object (DOM)*, which serves as an Application Programming Interface (API) for XML documents is used. The DOM basically represents XML data in a hierarchical format and provides a set of methods to retrieve and/or update this data.

In Netflow, a business process flow is generated in the form of a network, where the nodes are the processes and the arcs are the flows. The arcs originating from a single node point out processes that have to be accomplished simultaneously. These parallel processes would have to be synchronized at some point in the model, which should be handled accordingly. Nodes that are connected in series represent the consecutive processes that have to be accomplished one after another.

The model developed in the Netflow modeling environment is mainly split into two parts when constructing the algorithm that will transform model in the Netflow to

* DOM provides a structural representation of the nodes in an XML document, which makes it easy for a coder to visualize the data and retrieve it.

an executable model in Arena environment: Mapping source nodes and mapping destination nodes.

Mapping Source Nodes

In the mapping algorithm, the code should be developed in a way that each connection should be defined with corresponding source and destination modules and with rules embedded on these connectors. If more than one connection originates from a single source node, it would mean parallel processing of a single entity in the model (Figure 3). If this is the case, then assignments of rule parameters should be made after this single source node so that each parallel entity is checked against rules on their path to destination. Depending on the results of the check, the entity may be let through to destination node or may be disposed.

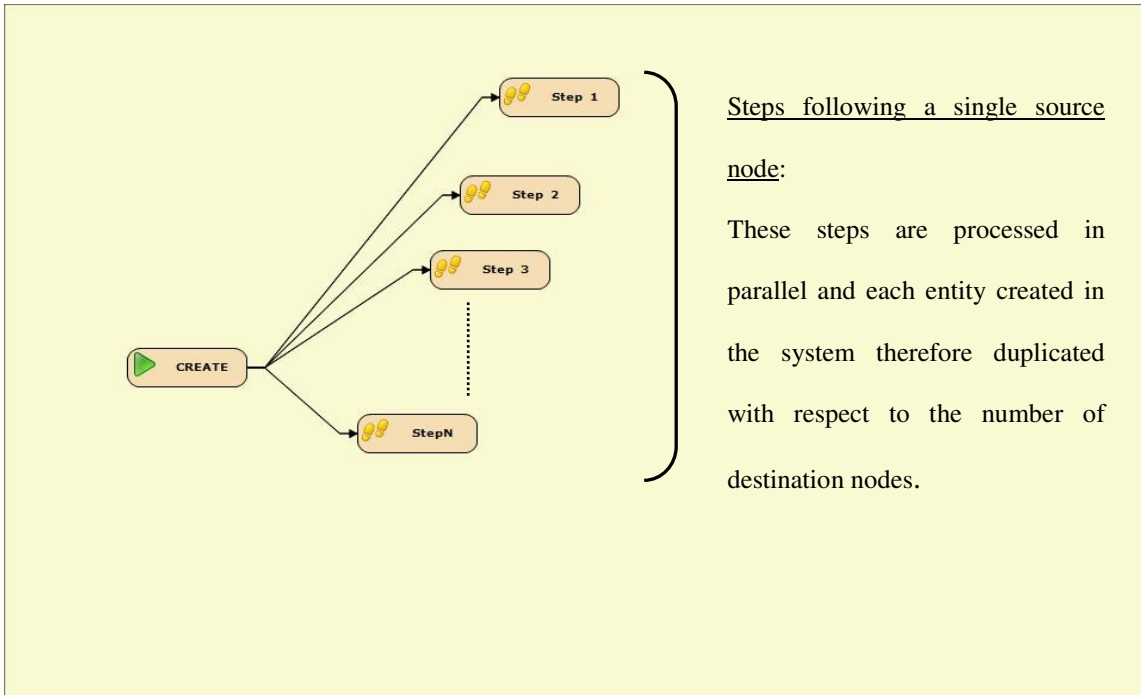


Figure 3: Parallel processing of steps originating from a single source node

An example with three steps is presented in Figure 4. To make it more clear how model data is stored on the database, the XML file of this sample model is also presented in the Figure 5 and highlights are elaborated.

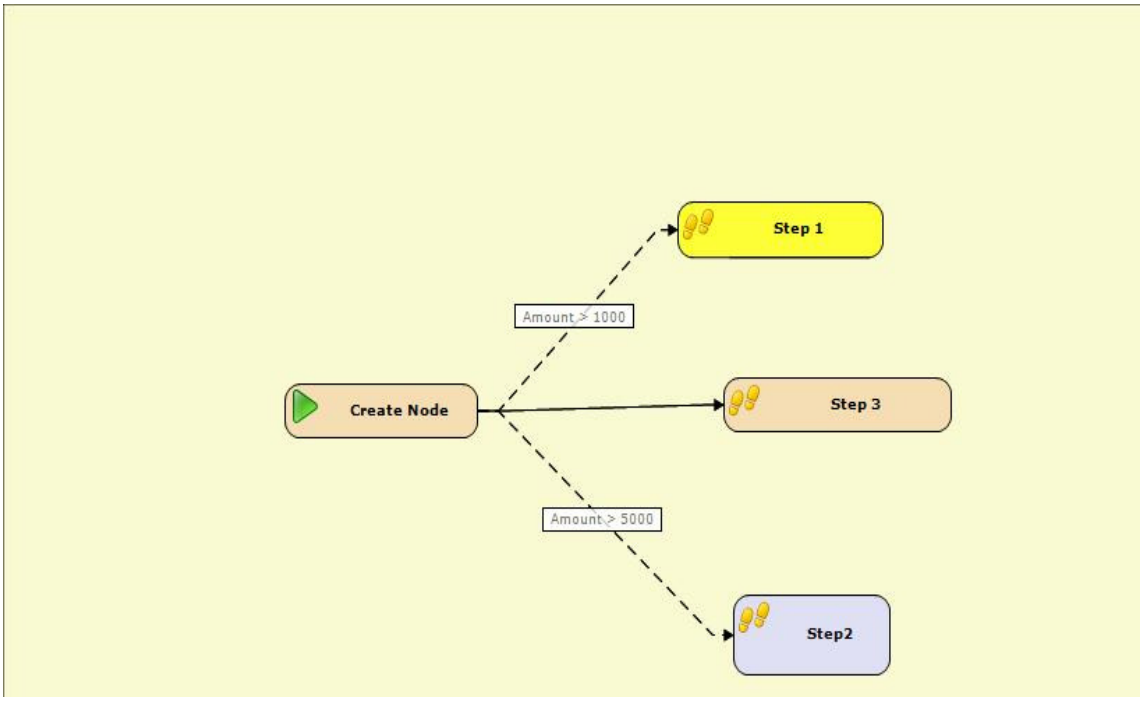


Figure 4: Example for the parallel processing of steps with conditions

```
- <Steps>
-   <UserStep Code="Create" Start="NR" NotAssignedBehaviour="ERR" RequestComment="true"
      Help="true" Delegate="true" Iteration="1" CanSendToPool="true" StepAssignmentType="S">
-    <Directions>
        - <Direction Code="Yönlendirme2" To="Step1" Direction="F">
            <Condition Type="BS" Script="GT(WFDATA( "MAIN", "TUTAR" ),"1000")" />
            <From>Create</From>
          </Direction>
        - <Direction Code="Yönlendirme3" To="Step2" Direction="F">
            <Condition Type="BS" Script="GT(WFDATA( "MAIN", "TUTAR" ),"5000")" />
            <From>Create</From>
        </Direction>
      -   <Direction Code="Direction1" To="Step3" Direction="F">
            <Condition Type="A" />
            <From>Create</From>
         </Direction>
    </Directions>
```

Figure 5: XML data kept in the SQL database corresponding to the example

In an XML file, all the elements within < > brackets are named nodes; using a hierarchal view these nodes may become a parent node, a child note or both. For instance, <Steps> node is a parent node; <Condition> is a child node whereas <Direction> becomes child node with respect to <Steps> node and a parent node with respect to <Condition> node. Every first element (word, character etc.) within brackets < > defines the corresponding tag and the rest are called attributes of that element. For example, “Type” is an attribute of the <Condition> node as well as “Script”. Using DOM interface in VBA, these nodes and their attributes are easily read and used to transform the model.

In Figure 4, the connectors from the “Create” step to “Step 1” and “Step 3” are associated with rules. These rules are basically conditions that each entity will be checked against and determined to advance further into the flow or disposed. If there is no rule, then this is indicated as “All cases” in the XML file; meaning that all attributes will be passed through that connector without any check. If a rule is assigned on a connector, this is kept as “basic script” condition in the XML file. In figure 6, <Condition> element has two attributes: “Type”, which determines whether the condition is a “Basic Script” (BS type) or “All cases” (A type) and “Script”, which holds the logical statement with the help of system variables or attributes.

Before elaboration on the transformation method, the Arena model that corresponds to this example is presented in Figure 6 because in the following description of methodology, references to this figure are given.

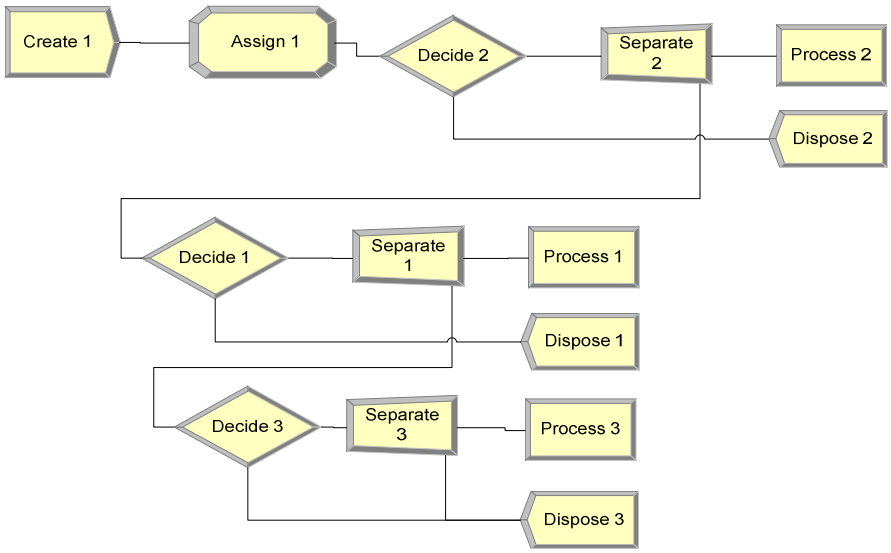


Figure 6: Model for parallel processing of steps from a single source node

The method for converting XML data into the Arena model shown in Figure 6 is described in the following points:

- In VBA code, to transform this flow into an executable Arena model, each source node is put through a loop to extract all the destination nodes and connectors as well as condition types and condition scripts. Afterwards, source node is checked whether it is the initiator step of the flow or not so that “Create” module is determined.

- The “Create” module is always followed by an “Assign” module, which will be used to assign values to entities. The attributes which will carry these values assigned in this module are extracted from “Basic Script” data. This script contains the rule operation (equation, greater than, less than etc.) and parameters and is kept

as a string in the XML file within “Script” attribute. This string is parsed using string functions in VBA with certain logic to extract the attribute to use in this “Assign” module and equation variables to use in “Decide” modules later on.

- Each connection in the Figure 4 corresponds to a group of “Decide, Separate, Process and Dispose” module group in Figure 6. This sequence of modules will serve the following functions:

    - “Decide” is configured differently for “A” type and “BS” type conditions. Basically, “A” type conditions could be seen as probabilistic conditions where the probability of true is always hundred percent. However, “BS” type rules always impose that an entity must satisfy certain condition(s). In this case, decision is 2-way-conditional and the condition is supplied by parsing the string within the “Script” attributes of <Condition> element in the XML file. If the condition is satisfied, the entity will proceed to the “Separate” module where it will be duplicated and passed on to the “Process” module and also to the next “Decision” module (or to the “Dispose” module if end of the flow is reached).

    - “Separate” module merely acts to duplicate the incoming entity and lets it pass through to the “Process” module. The duplicate would be directed to the next decision if anymore exits or to the dispose module if there is no more parallel step to be processed.

    - “Process” module may be seen as the replicate of the process step in Netflow. If the connection rule is satisfied by the attribute of an entity, then

that entity will proceed to “Process” module where it will be seized by a resource, processed and then released back into the flow.

    - “Dispose” module is mainly used to dispose entities that do not satisfy conditions or that are loosely coupled at the end of connections.

- After the modules have been created, the connections in between them should be completed where the “Auto-Connect” option is disabled. When all the modules are created, they are all assigned a unique tag which contains the type of module, a counter that represents the originating step and the name of the destination step. Using these tags and loop constructs in VBA, the connections are completed and this part of the model would be completed.

Mapping Destination Nodes

When more than one connection goes into a single destination module, as in the case of an additional step after parallel processing described above, then the duplicated entities should be batched up with a proper entity attribute (so that exact pairs are batched up) within a batch module. Since this means more than one source nodes, there will be assignment modules after each source step to cover any existing rules on each connector going into the destination node. This sort of flow is represented in the Netflow as in Figure 7. An example with three source steps connecting into one step is presented in Figure 8. The XML data of the model is constructed in the same structure as the single source step model; nevertheless this XML data is illustrated in Figure 9 as well to be able to give a more comprehensive picture of data source.

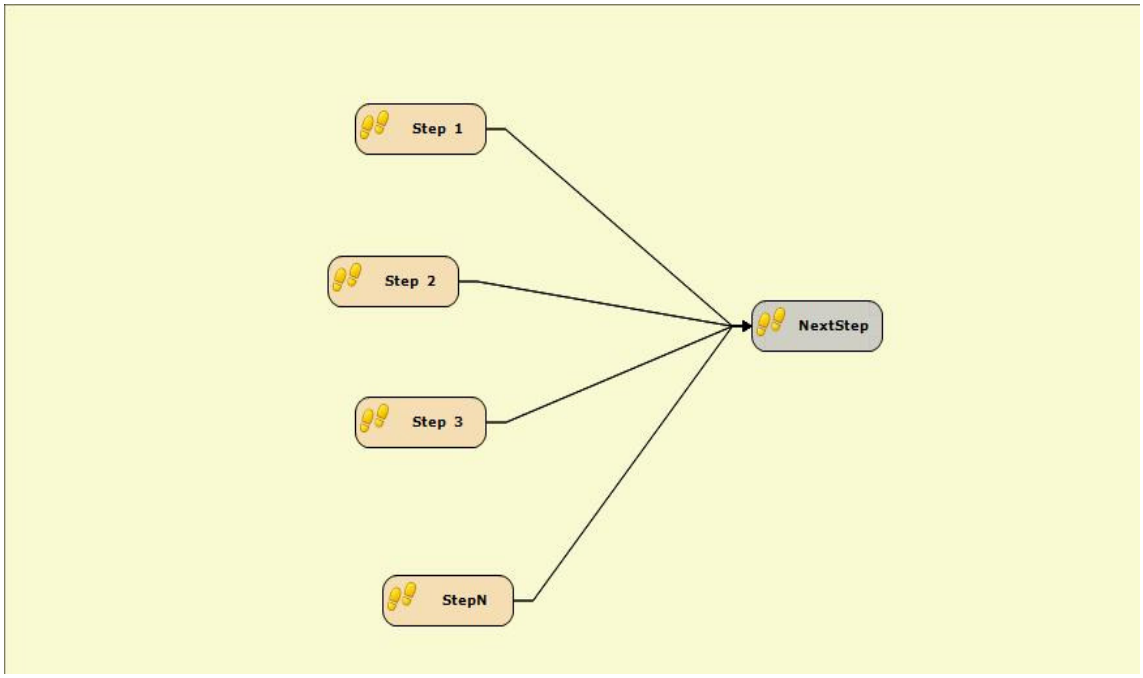


Figure 7: Representation of multiple connections into single destination node

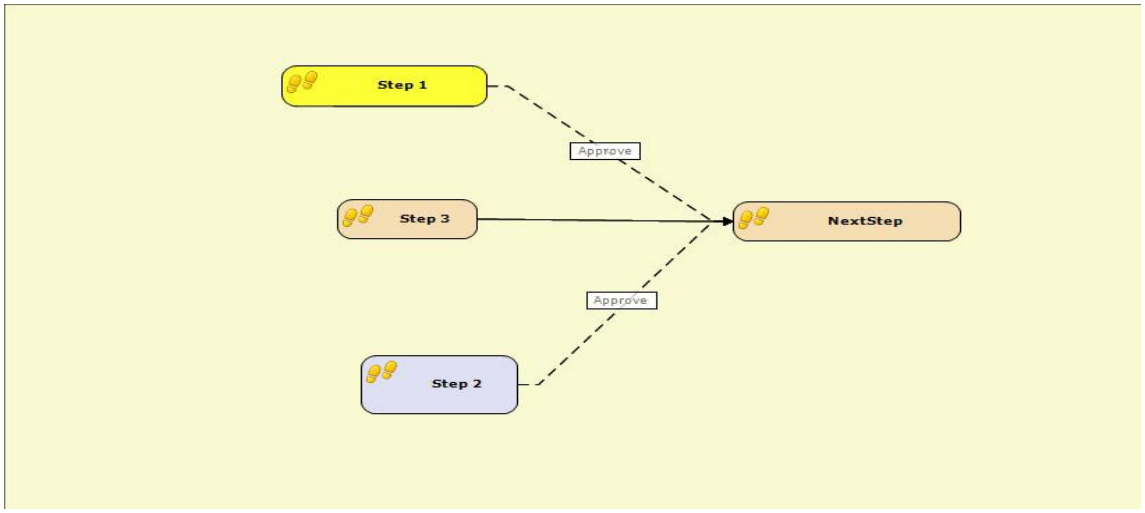


Figure 8: Representation of three steps batching into one destination step

```
- <UserStep Code="Step1" Start="N" NotAssignedBehaviour="ERR" RequestComment="true"
      Help="true" Delegate="true" Iteration="1" CanSendToPool="true" StepAssignmentType="S">
- <Directions>
- <Direction Code="Yönlendirme5" To="NextStep" Direction="F">
  <Condition Type="BS" Script="EQ(WFDATA( "MAIN", "ONAYRET" ),"EVET")" />
  <From>Step1</From>
    </Direction>
    </Directions>
    </UserStep>

- <UserStep Code="Step2" Start="N" NotAssignedBehaviour="ERR" RequestComment="true"
      Help="true" Delegate="true" Iteration="1" CanSendToPool="true" StepAssignmentType="S">
- <Directions>
- <Direction Code="Yönlendirme6" To="NextStep" Direction="F">
  <Condition Type="BS" Script="EQ(WFDATA("MAIN","ONAYRET"),"EVET")" />
  <From>Step2</From>
    </Direction>
    </Directions>
    </UserStep>

- <UserStep Code="Step3" Start="N" NotAssignedBehaviour="ERR" RequestComment="true"
      Help="true" Delegate="true" Iteration="1" CanSendToPool="true" StepAssignmentType="S" >
- <Directions>
- <Direction Code="Direction1" To="NextStep" Direction="F">
  <Condition Type="A" />
  <From>Step3</From>
    </Direction>
    </Directions>
    </UserStep>
    </Steps>
```

Figure 9: XML data of the model with three steps batched into one destination step

The method for converting XML data into the Arena model shown above is described in the following points:

- XML data is read and parsed in exactly the same way as described in the case of connections origination from a single source node; each node is looped around all the other steps until each source and destination steps are determined using “From” and “To” tags in the XML file.

- After each source-destination pairs are determined, an “Assign” module would be assigned to each source step, which would handle any “BS” type conditions on these connectors separately. For “A” type conditions, an “Assign” module would also be created but no assignment procedure would be defined within this module because all attributes without any check should be able to proceed through any connector holding “A” type condition. At these “Assign” modules, relevant attributes would be created and assigned to the attributes. In this example, there is one common attribute, “ONAYRET”, between Step 1 and NextStep, and Step 2 and NextStep. This very same attribute would be assigned to the entities coming through from Step 1 and Step 2 to set a value for “ONAYRET” attribute.

- After the attributes are assigned with values, these values should be checked with a “Decide” module using 2-way-by-condition whether incoming entities would be advanced to next step or disposed (which would be handled by a “Dispose” module assigned to the “false-end” of the “Decision” node. For “A” type conditions, “Decision” module would be created as “2-way-by-chance” where the odds of being

false would be set to zero. In other words, each entity would be passed through this "Decide" module because it is valid for "All Cases" ("A" type condition").

- The "true-end" of the each "Decision" node should be connected to "Batch" module to make sure that the same entity that had been duplicated for parallel processing is now composed back into one original entity and to be advanced through the rest of the flow.

The Arena model that is constructed following this procedure is presented in Figure 10. When two of the mappings are composed, a complete flow would be constructed in Arena which is presented in the following section of this study.

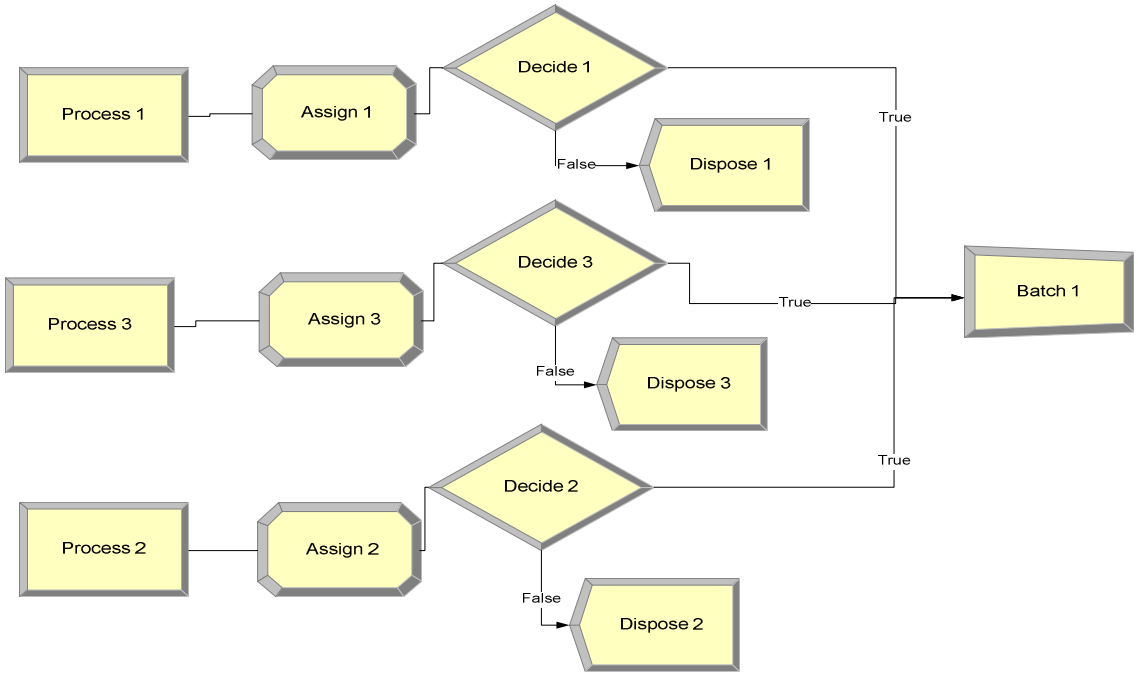


Figure 10: Arena model with multiple steps batched into one destination step

CHAPTER SIX: SIMULATION OF A SAMPLE MODEL IN NETFLOW AND ARENA

In chapter four, existence of an embedded simulation engine in Netflow's modeling module was mentioned. Using the simulation software comparison framework of Hlupic et al. (2007), it was also concluded that Arena's simulation capabilities were superior to Netflow's. Therefore, using a sample business process model already created in Netflow modeling environment, it will be shown in this chapter that how the interface proposed in this study generates better results than those rendered by Netflow.

The sample workflow constructed in Bizitek's Netflow is illustrated in Figure 11. This is a typical purchasing process where goods are externally procured and the requisition of this purchase needs to go through a few approval steps depending on the total amount of the all required items before requisition is finally converted into a purchase order. Each instance is created within "Procurement Demand" step and directly sent to the "Management Approval" step regardless of the total monetary value of the purchases. The management approval step represents the manager in the organizational structure, of the person that creates this demand instance. The solid direction arrow here shows that the following process is unconditional and will always take place. If the total amount of this instance is more than 1000 system currency units, it also goes through "Finance Manager Approval" step. If it is even more than 5000 currency units, then the process flow further asks for "General Manager Approval" step. The dashed direction arrows indicate that the following processes are conditional, i.e., they are valid only if certain conditions are satisfied. When all of the approval steps are completed, then the initial demand passes through "Procurement Manager Approval"

where it is confirmed as a purchasing order upon approval. Finally, an e-mail is sent back to the person who initiated the demand and the purchase order is recorded in the database.

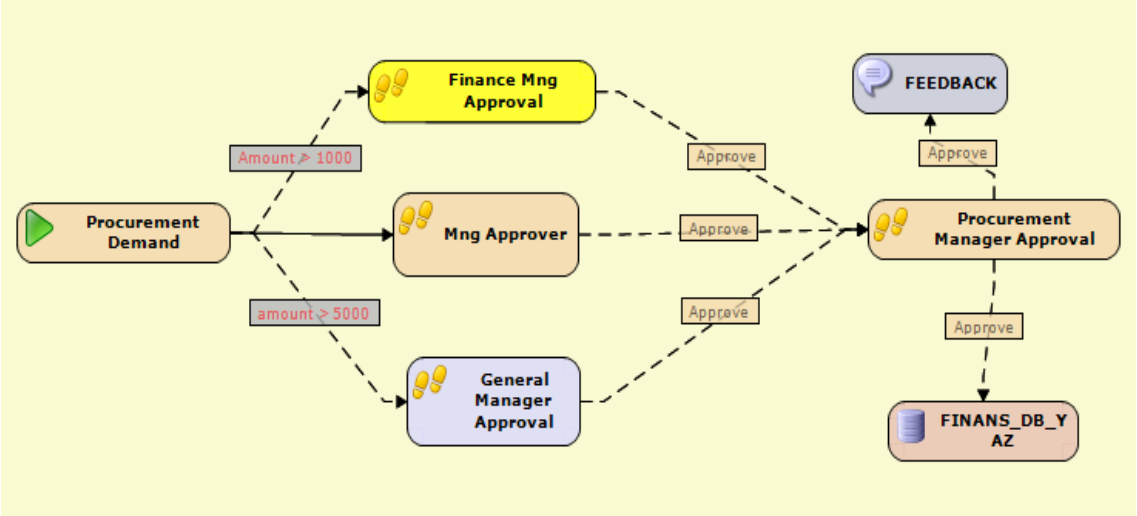


Figure 11: Sample workflow in Bizitek's Netflow

Every simulation model requires certain input data for a successful run. With the purpose of comparing proposed integration environment and Netflow's simulation environment, same set of input data is used. The data used in this chapter is presented in Table 6. It is important to note that Netflow only allows random number generations out of normal distribution therefore all times are assumed to be normally distributed. However, Arena comes with all the theoretical distributions to be able to cover many different data structures.

"NORM" represents normal distribution, "# of res" represents the number of resources available in the corresponding step and condition strings simply shows what the rules are from the step on the left side of the table to the step on the top of the table.

Table 6 - Simulation input data

Steps	Proc Demand	Mng App	Finance Mng App	Gn. Mng. App.	Proc Mng App
Proc Demand	- NORM(3, 0.1)		- Amount > 1000 with % 80	- Amount > 5000 with % 20	
Mng App		- NORM(2, 0.1) '- # of res = 1			- Approved with % 80
Finance Mng App			- NORM(5, 0.1) '- # of res = 1		- Approved with % 80
Gn. Mng. App.				- NORM(6, 0.3) '- # of res = 1	- Approved with % 80
Proc Mng App					- NORM(5, 0.1) '- # of res = 1

The simulation will run until the number of entities that have passed through the system will reach to one thousand. Afterwards, output results will be presented. “Bildirim” and “Finans_DB_Yaz” steps will be conveniently neglected because they only represent secondary transactions called as soon as the flow is finished and the amount of time spent in these activities would be measured in seconds. There is also no way to enter simulation data for these steps in Netflow.

The Application in Netflow’s Current Simulation Environment

When the model is run using the above parameters for about one thousand entities, the result from Netflow merely shows the average time an entity takes to get through the entire model. This output is presented in Figure 12.

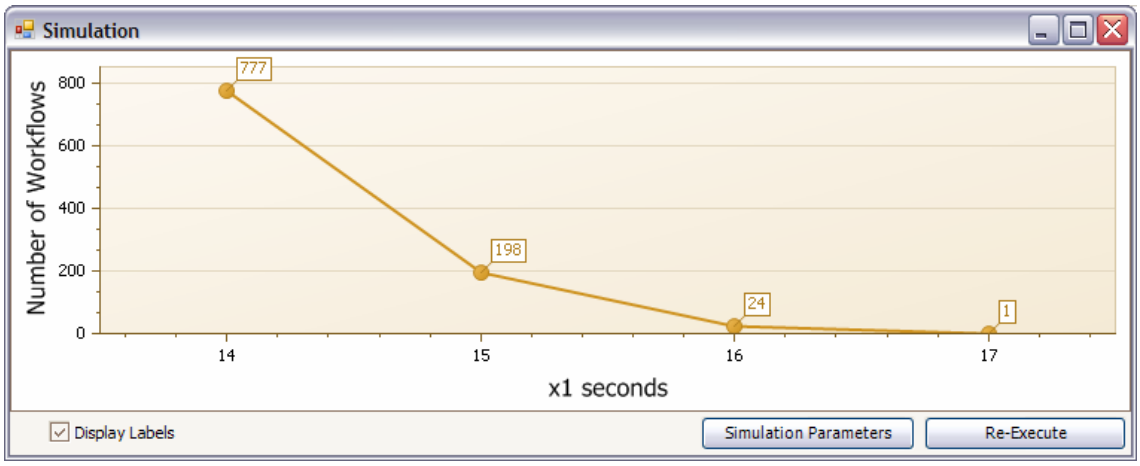


Figure 12: Simulation output by Netflow after one replication

When the same model is executed once more as a second replication, than the output data would look something like in Figure 13:

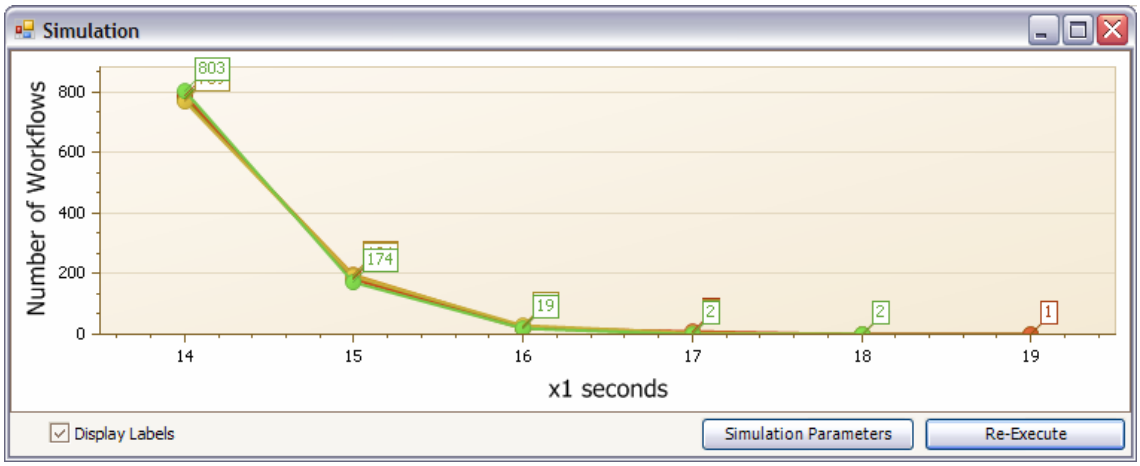


Figure 13: Simulation output by Netflow after multiple replications

This output merely represents the histogram of the distribution of the number of entities that has passed through the model. The output is interpreted easily but it is not very useful to determine bottlenecks or opportunities. In the next section, the same model is simulated using the interface developed in this study and outputs are presented.

The application in Arena using proposed interface

When the same process is modeled using the interface developed in this study, the generated model would be similar to Figure 14 (omitting “Bildirim” and “Finans_DB_Yaz” steps). Since the notation -the way the steps are represented- is different, the workflows are not alike; however the processes are essentially the same in terms of the underlying parameters and rules.

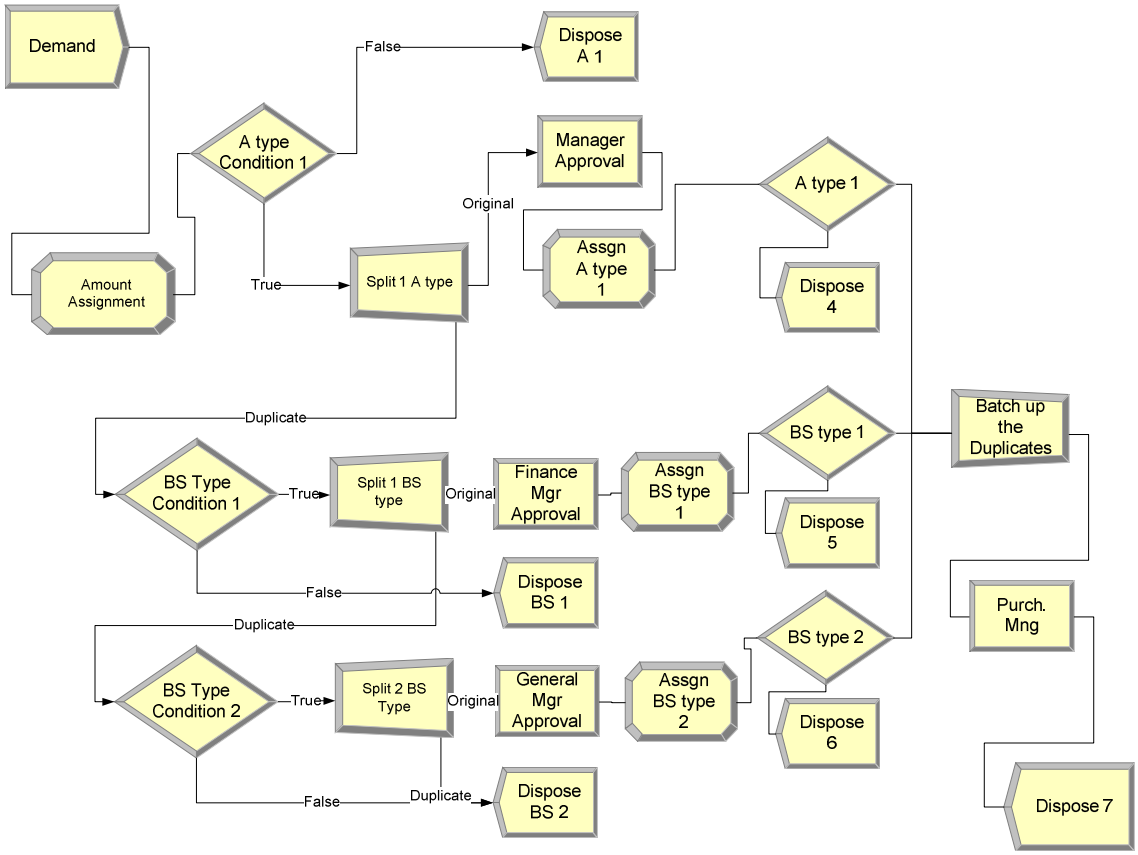


Figure 14: Sample model in Arena

The representation of the process, even though there are some simplifications, looks more complex in Arena. It is because, the separation of instances for parallel processing and decision nodes are handled at different modules, which raises the big

challenge of mapping the same process from Netflow into Arena. To overcome this challenge, the mapping methodology previously presented is developed.

When this model is run in Arena, more sophisticated outputs are generated that would assist users in making more informed decisions. Some of the outputs generated after three replications with the same parameters are presented in the following figures.

Entity

Time

VA Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
TALEP	2.3390	0.64	2.1288	2.6255	0.00	18.8545

NVA Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
TALEP	0.00	0.00	0.00	0.00	0.00	0.00

Wait Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
TALEP	5.1550	3.23	3.9871	6.5585	0.00	45.8818

Transfer Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
TALEP	0.00	0.00	0.00	0.00	0.00	0.00

Other Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
TALEP	0.00	0.00	0.00	0.00	0.00	0.00

Total Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
TALEP	3.9109	1.61	3.3290	4.6107	0.00	29.5233

Figure 15: Entity time statistics generated by Arena

Queue

Time

Waiting Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Batch 5.Queue	6.5331	0.06	6.5125	6.5621	0.00	24.4935
STEP FINANSONAY.Queue	11.5005	0.65	11.2438	11.7696	0.00	23.5084
STEP GMONAY.Queue	0.03973995	0.17	0.00	0.1192	0.00	0.2384
STEP MUDURONAY.Queue	0.00	0.00	0.00	0.00	0.00	0.00
STEP SATINALMAONAY.Queue	0.05131311	0.11	0.00	0.08385835	0.00	0.3310

Other

Number Waiting	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Batch 5.Queue	4.6556	0.47	4.4549	4.8259	0.00	10.0000
STEP FINANSONAY.Queue	4.1604	0.27	4.0487	4.2673	0.00	9.0000
STEP GMONAY.Queue	0.00267005	0.01	0.00	0.00403614	0.00	1.0000
STEP MUDURONAY.Queue	0.00	0.00	0.00	0.00	0.00	0.00
STEP SATINALMAONAY.Queue	0.00652895	0.02	0.00	0.01257875	0.00	1.0000

Figure 16: Queue statistics generated by Arena (especially useful to determine bottlenecks)

CHAPTER SEVEN: DISCUSSION and ANALYSIS

It is apparent that Arena has superior simulation capabilities than Netflow and there are various means of integrating these platforms to take advantage of the more advanced simulation features. The proposed interface in this study could provide a good reference point for other similar applications. Since there are many other BPM tools with different modeling and data storage characteristics, a few suggestions for developing such interfaces to Arena will be given and also the limitations of the proposed methodology and interface will be discussed.

Suggestions on Developing Similar Interfaces

As presented earlier in this study, there are a few integration techniques that could be utilized when developing an interface between an external application, such as Netflow, and Arena simulation software. The advantages and disadvantages of these techniques were discussed earlier and the second technique, direct modeling in VBA using ActiveX automation, is used for this particular study. The choice was based entirely on the ability of two-sided communication between the development environment and the simulation environment and also on the potential amount of effort to be spent developing the interface. The first technique wouldn't allow for two-way communication and the third technique would require much more effort due to an additional database conversion programming on top of mapping algorithm. The first technique would allow developing much nicer interface for end users to input data or to

start simulation runs however two-way communication is a more important aspect. Using Visual Basic Editor embedded in Arena enables to extract data even after the simulation run has begun so that further checks or logics could be placed in the events that would be called during simulation run provides much more flexibility than just the model conversion routines. Putting more weight on this aspect, the choice should be the second or the third technique. At this point, it should be noted that this choice should be entirely contingent on the circumstances: how model data is kept on the database and modeling notation in the BPM software. Even though the second technique, developing the mapping algorithm within the Visual Basic Editor in Arena and run the interface using Arena model file, is used in the scope of this study, the third technique would be more appropriate when model data in BPM software is kept in a standard tabular form in the database.

In the "Import Model from Database" technique, Arena takes advantage of the object model introduced with "smImportFromDatabase.dll" ActiveX dynamic link library installed during Arena setup. This technique simply relies on a certain way of storing model information in any particular model database. Regardless of the type of database the model information is stored; the data should be organized in a standard set of tables in Access or Excel database to be able to use this object model (Microsoft products enable the use of ActiveX DLL). Each model should be divided into separate storage containers called tables. The type of tables required to represent a model as well as corresponding table structures are listed in appendix A.

In order to have similar structures to store model information within any BPM tool, there should be a standardized notation of objects used in modeling so that these

objects may be divided into table structures similar to what Arena offers to fully take advantage of the "Import Model from Database" technique. BPMN may provide a solid basis for this method.

The primary focus of BPMN is to provide a common process modeling language for all parties involved; from business analysts who create initial drafts of processes, to technical developers who would be implementing the technology that will perform those processes and finally to the business responsible for managing and monitoring these processes (White, S. A., 2004). BPMN provides graphical elements to represent many different flows of entities subject to many rules and conditions on the way. BPMN defines Business Process Diagram (BPD) which is based on a flowcharting technique to create graphical models of processes (White, S. A., 2004). BPD is made up of a set of graphical elements. The four basic categories of elements are as follows:

- Flow objects: Events, Activities, Gateways
- Connecting objects: Sequence flow, Message flow, Association
- Swimlanes: Pool, Lane
- Artifacts: Data objects, Group, Annotation.

It is also possible to define custom type of a flow object or an artifact to make diagrams more understandable

To see the graphical objects and a process example modeled in this notation, please refer to appendix B. (http://en.wikipedia.org/wiki/BPMN).

OMG has great influence in BPM community and most of the BPM suite vendors have already switched to BPMN within the modeling environment of their

BPM suite offerings[*]. Modeling in such structured way would make integration much easier because every graphical object in BPD has a counterpart in flowchart modeling stencils which also exist in Arena modeling environment.

As previously explained, BPMN is a visual process notation standard and this standard would create a better way to store process definition into table structures in any suitable database. However, there are also standards defined for how to store or interchange process definition and this is where the XML Process Definition Language (XPDL) comes in.

XPDL is the language proposed by the WfMC to interchange process definitions between different workflows (van der Aalst, 2003). The mission of WfMC is to promote and develop the use of workflow through the establishment of standards for workflow terminology, interoperability and connectivity between workflow products (van der Aalst, 2003). For this purpose, WfMC introduces a workflow reference model where five interfaces are identified. This reference model is presented in Figure 17.

[*] There are fifty two current implementations and four planned implementations of BPMN by the companies who offer BPM tools in their product portfolio. A few examples of current implementations are: IDS Scheer, ILOG, Lombardi, Savvion, Popkin and SAP. Four companies who are planning to go on with BPMN are: Hyland Software, IBM, Staffware and webMethods. Retrieved on September 15, 2008, from http://www.bpmn.org/BPMN_Supporters.htm

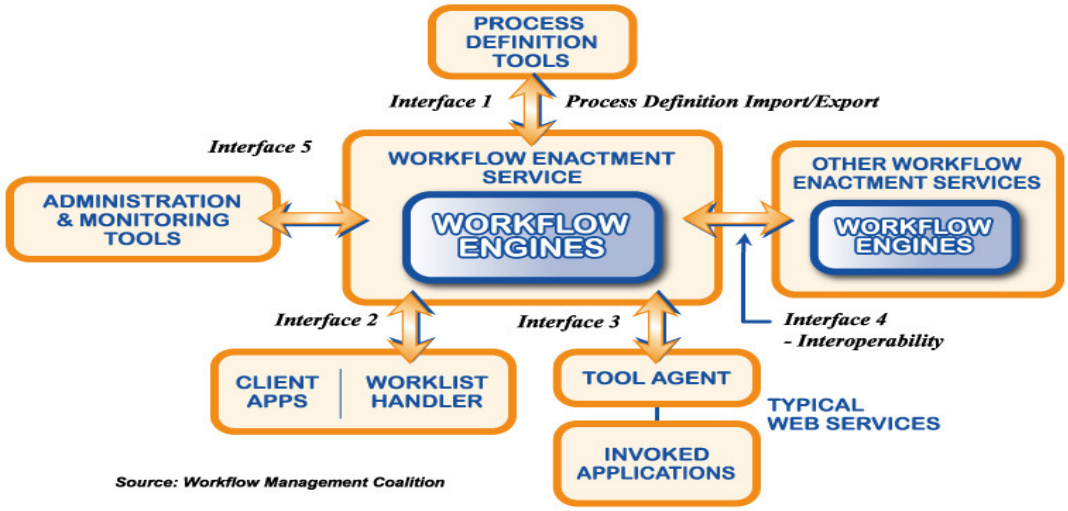


Figure 17: Workflow reference model by WfMC. Retrieved from http://www.wfmc.org/reference-model.html

The primary goal of interface 1 in the above figure is the import and export of process definitions, which consists of network of activities and their relationships, information about individual steps such as participants and etc. Using this reference model, Van der Aalst (2003) explains the need for process definition interchange and describes possible sources of this need. First of all, the connection between a design tool and a run-time environment must be established. Another reason may be the need to desire to use another design tool such as ARIS or Protos and transfer the model back into the process engine of another BPM tool. It could also be the case where a link would be created with analysis software such simulation and verification tools and a design tool, which is the primary goal of this study. There may also be a need to transfer a definition from one engine to another in especially automated processes. To support the interchange of process definitions due to any of these needs, XPDL has been put forward by WfMC as a standardized process definition language.

XPDL provides a file format that supports every aspect of the BPMN process definition notation including graphical descriptions of the diagram, as well as executable properties used at run time. It is designed to exchange both graphics and semantics of a workflow business process. With XPDL, a product can write out a process definition with full fidelity, and another product can read it in and reproduce the same diagram that was sent (http://www.wfmc.org/xpdl.html).

Bizitek's Netflow doesn't offer BPMN and process definition is not kept in XPDL; however, if an integration like the one developed in this study was to be realized with a BPM tool that does offer BPMN and XPDL, then "Import Model from Database" option would be more suitable. This way, each module and connector would be easily recognized and coding efforts would only be limited to database conversion.

Limitations of the proposed methodology

The interface developed in this study has been developed to work not only for a specific model, but also for any model that could be constructed in Netflow environment. The aim was to make this integration algorithm as generic as possible that could map any model drawn in Netflow modeling environment into Arena environment. Nevertheless, there are a few limitations to the model that should be explained in detail. These points could also be seen as improvement points to be developed additionally as a future study.

First of all, the conditions are kept as string in the XML representation of the model. That is why this string is parsed using string functions available in VBA and the

output is presented to the user in a dialog box for confirmation during mapping algorithm runs. However, there may be many different forms of conditions, hence different strings, which may not be divided into logical parameters by a simple string parsing code snippet. It would also be impossible to generate a generic dialogue box for users to enter logical parameters as there may be complex logical operations within a condition. The current dialogue box and string parsing algorithm is suitable for single logical operation with one parameter and one value (e.g. parameter < value, parameter > value etc.), which wouldn't be sufficient to cover many actual business process rules. Therefore, it would be possible to cover this part by creating a user-defined class which would contain the VBA version of very same methods or properties used in the underlying code of Netflow.

There is also another shortcoming of the proposed interface in reverse flows. If a transaction requires sending an entity back to a process or decision module, it would be omitted in the current mapping algorithm. Inclusion of this activity would be crucial in an interface developed for mapping business process models into Arena environment for accurate simulation results, especially on resource and entity statistics.

Maybe the most important improvement opportunities for the interface developed in this study are the ability of historical data feed into the model and generation of custom reports as well as improving the comprehensibility of standard reports generated by Arena. Historical data is essential in improving current processes using simulation because using historical data instead of theoretical distributions for inter-arrival times or resource service times would also form a solid base to validate the accuracy of simulation model generated through the interface. Arena supports data input

from external documents such as spreadsheets or text files so this ability may be used to provide historical data into the Arena model. As well as having ability to read data from external data sources, Arena may also export reports to external data files such as Excel spreadsheet or Access database. Reports are generated on selected statistics such as entities, queues and resources when a simulation run has been completed and this statistical data is saved in the model Access database file. This model file is then used to generate standard reports in Arena environment through a third party statistical package, Crystal Reports. One may find all useful statistics in these reports; however, interpretation of output data may not be always that easy. Therefore interface may be further enhanced to collect many predefined statistics on given selection parameters (entities, queues, resources etc.) and generate a spreadsheet view of requested information with conditional color formatting of significant results such as bottlenecks.

CHAPTER EIGHT: SUMMARY AND CONCLUSION

It has become clearer that organizations need to have some sort of a BPM tool to get better control over their processes and resources. Furthermore, these processes span across most of the entities within an organization, which requires efficient use of time, money and resources. Noting that the real world processes are too complex to design with mere experience and expertise to be deployed in a BPM tool, organizations need simulation capability to successfully predict the effects of new or enhanced process designs.

Although most commercial BPM tools nowadays have some kind of simulation engine, the simulation capabilities of these tools happen to be limited. To overcome this drawback, separate simulation software can be integrated to the BPM tool. However purchasing a simulation tool increases costs and moreover, this requires the knowledge on both platforms. Thus one should make a tradeoff between the increased costs and the broader functionality achieved by integrating a third party simulation software.

In this study an interface is built that can be used to map the processes designed in a BPM tool into simulation software. The problems encountered in integrating process mapping and simulation were mainly due to unstructured modeling environment and use of non-standard notation within Netflow. However, XML data of models provided the basis and due to high integration and automation capabilities of VBA environment, models could be transformed into executable Arena models.

Integration with general purpose simulation software may require high customization and get quite costly depending on the choice of software. But if this

interaction was used efficiently, the benefits obtained from using a dynamic modeling environment that could imitate real process as accurate as possible would be substantial considering ability to model even complex models and being able to generate very sophisticated outputs. Therefore, this study supports that advantages of using a simulation environment in the context of business process management would make the efforts and money spent worthwhile.

REFERENCES

Barnett, M. (2003). *Modeling and simulation in business process management*. White paper retrieved from http://www.bptrends.com.

DeToro, I. & McCabe, T. (1997). How to Stay Flexible and Elude Fads. *Quality Progress, 30*(3), 55-60.

Elzinga, D.J., Horak, T., Chung-Yee, L. & Bruner, C. (1995). Business Process Management - Survey and Methodology. *IEEE Transactions on Engineering Management, 24*(2), 119-28.

Harrell, C.R. & Field, K.C. (1996). Integrating Process Mapping and Simulation. *Proceedings of the 1996 Winter Simulation Conference*.

Hlupic, V., Bosilj Vuksic, V. and Ceric, V. (2007). Criteria for the Evaluation of Business Process Simulation Tools. *Information, Knowledge, and Management, 2*, 73-88.

Jeston, J. & Nelis, J. (2006). *Business Process Management: Practical Guidelines to Successful Implementations*. Burlington: Butterworth-Heinemann Publishing.

Simulation for Success. (n.d.). *Using Simulation to Enhance the Value of BPM*. White paper retrieved from http://www.lanner.com/en/l-sim.cfm

Paul, R.J., Giaglis, G.M. & Hlupic, V. (1999). Simulation of Business Processes. *The American Behavioral Scientist, 42* (10), 1551-1579.

Silver B., (2006). *The 2006 BPMS Report: Understanding and Evaluating BPM Suites*. Retrieved from http://www.bpminstitute.org

Srinivasan, K. & Jayaraman, S. (1997). Integration of Simulation with Enterprise Models. *Proceedings of the 1997 Winter Simulation Conference*

Van der Aalst, W.M.P. (2003). *Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language*. (Report No: FIT-TR-2003-06). Retrieved from http://www.bpm.fit.qut.edu.au/about/publications/technical.jsp

Waller A., Clark M., & Enstone L. (2006). L-SIM: Simulating BPMN Diagrams with a Purpose Built-in Engine. *Proceedings of the 2006 Winter Simulation Conference*

White, S.A. (2004). *Introduction to BPMN*. White paper retrieved from http://www.bpmn.org/.

UNCITED REFERENCES

DeFee, J.M. & Harmon, P. (2004). *Business Activity Monitoring and Simulation*. White paper retrieved from http://www.bptrends.com

Everton, J.G., & Stafford, R.D. (2005). Using Workflow Business Process Tools in Simulation Modeling. *Proceedings of the 2005 Winter Simulation Conference*

Henschen, D. (2005). Business Process Management is Under Construction. *Intelligent Enterprise, 8* (8), 22-27.

Hlupic, V., & Robinson, S. (1998). Business Process Modeling and Analysis Using Discrete-Event Simulation. *Proceedings of the 1998 Winter Simulation Conference*

Kelton, W.D., Sadowski, R.P., & Sturrock, D.T. (2003). *Simulation with Arena, third edition.* Singapore: McGraw-Hill.

Law, A.M., & Kelton, W.D. (2000). *Simulation Modeling and Analysis, third edition*. Singapore: McGraw-Hill.

Lee, R.G., & Dale, B.G. (1998). Business Process Management: A Review and Evaluation. *Business Process Management Journal, 4* (3), 214-225

Lomax, P. (1998). *VB & VBA in a Nutshell: The Language*. California: O'Reilly & Associates.

Luo, W., & Tung, Y.A. (1999). A Framework for Selecting Business Process Modeling Methods. *Industrial Management & Data Systems, 99* (7), 312-319

Microsoft Developer Network Library. (2008). Retrieved from http://msdn.microsoft.com/en-us/library/default.aspx

Schneider, D.I. (1999). *An Introduction to Programming Using Visual Basic 6.0, fourth edition.* New Jersey: Pearson Custom Publishing

Weyland, J.H., & Engiles, M. (2003). Towards Simulation-Based Business Process Management. *Proceedings of the 2003 Winter Simulation Conference*

APPENDICES

A. Import Model from Database - Table Structures

The following information is retrieved from help files of Arena simulation software version 10.0 CPR 7.

- "ModuleTables" Table: This table keeps data about how many different modules are to be created for the model. For example, if there is one "Create" module, two "Process" modules and one "Dispose" module is needed to construct the model, then the number of records within this table would be three; representing the number of unique modules in the model.

Field Name	Data Type	Description
PanelName	Text	Name of Arena panel (e.g., "BasicProcess", "AdvancedTransfer", "AdvancedProcess")
ModuleName	Text	Type of Arena module (e.g., "Create", "Process", "Dispose")
TableName	Text	Name of database table that records all instances of the module

- Module Table: This table holds the all instances of any module in the model. For the same example used above, this table would have four records including one "Create" module instance, two "Process" module instances and one "Dispose" module instance. The "SerialNumber" field differentiates the different instances of the same module. The non-repeatable operands within a module would also be stored in this table such as "Module Name", "Distribution type" and "time units" if relevant. The repeatable entries such as resources in a "Process" module or

attribute/variable assignments in an “Assign” module are recorded in a different table which is also shown below.

Field Name	Data Type	Description
SerialNumber	Long	Unique object identifier
ModelLevelID	Long	Level in model hierarchy that the module exists (foreign key to ModelLevels table)
X	Long	The X position (in world coordinates) of the module in the model level
Y	Long	The Y position (in world coordinates) of the module in the model level
UserDescription	Text	User-specified description (specified in module’s Properties dialog)
Operand Name 1	Text	Value of the first non-repeatable operand in the module
Operand Name 2	Text	Value of the second non-repeatable operand in the module
Operand Name N	Text	Value of the Nth non-repeatable operand in the module

- “RepeatGroupTables” Table: If a module has a repeatable set of operands such as resources in a “Process” module, then a separate table exists to keep this data in the database. This table merely holds data regarding the location of operands. The values entered into these operands are kept in the next table presented.

Field Name	Data Type	Description
PanelName	Text	Name of module panel (e.g., "BasicProcess", "AdvancedTransfer", "AdvancedProcess")
ModuleName	Text	Type of module (e.g., "Create", "Process", "Dispose")
RepeatGroupName	Text	Name of module repeat group including parent repeat groups separated by a "|" character (e.g., "Resources", "Failures", "Steps|Assignments")
TableName	Text	Name of database table that records all data for the repeat group in the model

- A Repeat Group Table: This table stores the data associated with a repeat group for any of the modules whose structure includes repeat group in a model. A repeat group table also contains one or more *|*Index* fields. “These fields identify the index in the repeat group that a record pertains to. For example, suppose we are adding a record to the Assignments repeat group table for Advanced Transfer’s Sequence module. This repeat group table has two index fields: *Steps|Index* and *Assignments|Index*. To enter the second assignment for the third step in the sequence, we would enter a value of "3" in the *Steps|Index* field and a value of "2" in the *Assignments|Index* field.”

Field Name	Data Type	Description
ModuleSerialNumber	Long	Serial number of module that the repeat group record pertains to (foreign key to appropriate module table)
Parent Repeat Group 1|Index	Long	Index into first level parent repeat group (this field may not exist if there are no parent repeat groups)
Parent Repeat Group 2|Index	Long	Index into second level parent repeat group (this field may not exist if there is no second level parent repeat group)
Repeat Group Name|Index	Long	Index into repeat group
Operand Name 1	Text	Value of the first operand in the repeat group
Operand Name 2	Text	Value of the second operand in the repeat group
Operand Name N	Text	Value of the Nth operand in the repeat group

- “Model Levels” Table: Model hierarchy may be used in Arena to represent submodels, each with its own workspace for defining entity flow and animation. These separate but interdependent workspaces are referred to as “Model Levels”. To be able to identify this hierarchal structure, two tables exist: first is the “Model Levels” and the second is the “Submodels” table, whose structure are given below

Field Name	Data Type	Description
ModelLevelID	Long	Unique identifier
ModelLevelName	Text	Name of model level

- "Submodels" Table: This table basically stores all the instances of submodels explained above. Each submodel has a unique serial number that distinguishes submodels from each other if more than one exits in a model.

Field Name	Data Type	Description
SerialNumber	Long	Unique object identifier
ModelLevelID	Long	Level in model hierarchy that the submodel object exists (foreign key to ModelLevels table)
X	Long	The X position (in world coordinates) of the submodel in the model level
Y	Long	The Y position (in world coordinates) of the submodel in the model level
Name	Text	The name of the submodel
NumEntryPoints	Long	Number of submodel entry points
NumExitPoints	Long	Number of submodel exit points
Description	Text	Submodel description

- "Connections" Table: This table keeps all instances of connection objects within a model. Similar to each shape in a model, connectors in the model also have their own unique serial numbers as identifiers. The field "ModelLevelID" indicates whereabouts of a particular connection.

Field Name	Data Type	Description
SerialNumber	Long	Unique object identifier
ModelLevelID	Long	Level in model hierarchy that the connection object exists (foreign key to ModelLevels table)
SourceSerialNumber	Long	Serial number of source object that is being connected (foreign key to a module table or Submodels table)
SourceLabel	Text	Identifies the exit point of the source object. If the source object is a module, enter the name of the module operand that defines the exit point. If the source object is a submodel, enter a 1-based index

		into the submodel's exit points
SourceRepeatIndex	Long	If the exit point of the source object is repeatable (e.g., the repeatable exit of an N-way Decide module), this field specifies the index into the exit point
DestinationSerial Number	Long	Serial number of destination object that is being connected (foreign key to a module table or Submodels table)
DestinationLabel	Text	Identifies the entry point of the destination object. If the destination object is a module, enter the name of the module operand that defines the entry point. If the destination object is a submodel, enter a 1-based index into the submodel's entry points
DestinationRepeat Index	Long	If the entry point of the source object is repeatable, this field specifies the index into the entry point
UserDescription	Text	User-specified description (specified in module's Properties dialog)

- "Named Views" Table: This table records all instances of named views in a model. Named view may be used segment a model into portions that may be easily recalled without using scroll bars. This is helpful if a model is great in size so that it can not be viewed at once in the screen.

Field Name	Data Type	Description
ViewID	Long	Unique identifier
ModelLevelID	Long	Level in model hierarchy that the named view exists (foreign key to ModelLevels table)
Name	Text	View name
HotKey	Text	Optional hot key for view
Left	Long	Left-edge boundary of view (in world coordinates)
Right	Long	Right-edge boundary of view (in world coordinates)
Top	Long	Top-edge boundary of view (in world coordinates)
Bottom	Long	Bottom-edge boundary of view (in world coordinates)

- "Project Parameters" Table: This table records project parameters set up before simulation run. Only one entry may be recorded in this table.

Field Name	Data Type	Description
ProjectTitle	Text	Specifies the project title
AnalystName	Text	Specifies the analyst name
ProjectDescription	Text	Specifies a project description
CostingStatistics	-1 for "Yes" or 0 for "No"	Specifies whether costing statistics are recorded during a model run
QueueStatistics	-1 for "Yes" or 0 for "No"	Specifies whether queue statistics are recorded during a model run
TransporterStatistics	-1 for "Yes" or 0 for "No"	Specifies whether transporter statistics are recorded during a model run
EntityStatistics	-1 for "Yes" or 0 for "No"	Specifies whether entity statistics are recorded during a model run
ConveyorStatistics	-1 for "Yes" or 0 for "No"	Specifies whether conveyor statistics are recorded during a model run
ProcessStatistics	-1 for "Yes" or 0 for "No"	Specifies whether process statistics are recorded during a model run
ResourceStatistics	-1 for "Yes" or 0 for "No"	Specifies whether resource statistics are recorded during a model run
StationStatistics	-1 for "Yes" or 0 for "No"	Specifies whether station statistics are recorded during a model run
ActivityAreaStatistics	-1 for "Yes" or 0 for "No"	Specifies whether activity area statistics are recorded during a model run
TankStatistics	-1 for "Yes" or 0 for "No"	Specifies whether tank statistics are recorded during a model run

- “Replication Parameters” Table: This table records replication parameters set up before simulation run. Only one entry may be recorded in this table.

Field Name	Data Type	Description
NumberOfReplications	Long	Specifies the number of replications to run the model
InitializeStatisticsBetween Replications	-1 for "Yes" or 0 for "No"	Determines whether or not the statistics are cleared between simulation replications
InitializeSystemBetween Replications	-1 for "Yes" or 0 for "No"	Determines whether or not the system is reinitialized between replications. If so, all entities are removed from the model and the system status and user variables are restored to their default values. This state is referred to as "empty and idle"

StartDateTime	DateTime	Specifies the calendar date and time corresponding to simulation time 0.0
WarmupPeriod	Text	Time period after the beginning of the run at which statistics are to be cleared
WarmupPeriodTimeUnits	0 for "Days", 1 for "Hours", 2 for "Minutes", 3 for "Seconds"	Specifies the time units of the warm-up period entry
ReplicationLength	Text	Specifies the run length of each replication
ReplicationLengthTimeUnits	0 for "Days", 1 for "Hours", 2 for "Minutes", 3 for "Seconds"	Specifies the time units of the replication length entry
HoursPerDay	Text	Defines the number of hours to be modeled within a simulated day
BaseTimeUnits	0 for "Days", 1 for "Hours", 2 for "Minutes", 3 for "Seconds"	Specifies the base time units of the simulation clock
TerminatingCondition	Text	Specification of an expression or condition that is evaluated throughout the simulation run to determine whether or not to stop the simulation. If the condition or expression is true (or evaluates to 1), the simulation run will be terminated. This is one method, besides specifying a replication length, in which the simulation run may be ended.

- “Reports” Table: This table holds information about reporting options after simulation run. This table may also contain only one record.

Field Name	Data Type	Description
DisplayDefaultReport	0 for "Always", 1 for "Never", 2 for "Prompt Me"	Specifies whether report is displayed at end of simulation run
DefaultReport	Text	Default report to display

DisableGenerationOfReportDatabase	-1 for "Yes" or 0 for "No"	Specifies whether to suppress the generation of the report database

B. Business Process Diagram as Specified by BPMN

The descriptions and figures in appendix B are retrieved from "http://en.wikipedia.org/wiki/BPMN" in October 2008.

The four basic categories of graphical elements are presented in the following figures.

Flow objects

Flow Objects consist of only three core elements. The three Flow Objects are:

- Event: An Event is represented with a circle and is something that happens. It could be Start, Intermediate or End. This element is a trigger or a result.

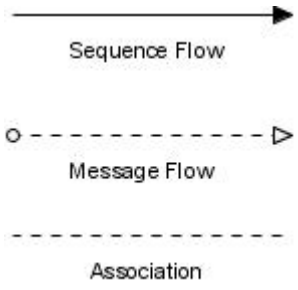
- Activity: An Activity is represented with a rounded-corner rectangle and shows us the kind of work which must be done. It could be a task or a sub-process. A sub-process also has a plus sign in the bottom line of the rectangle.

- Gateway: A Gateway is represented with a diamond shape and will determine different decisions. It will also determine forking, merging and joining of paths.

Connecting objects

The Flow Objects are connected to each other with Connecting Objects. There are three different Connecting Objects:

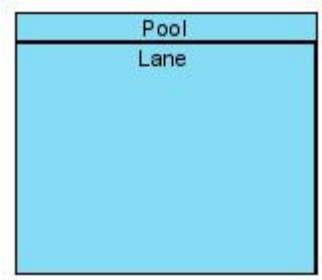
- Sequence Flow: A Sequence Flow is represented with a solid line and arrowhead and shows in which order the activities will be performed. A diagonal slash across the line close to the origin indicates a default choice of a decision.
- Message Flow: A Message Flow is represented with a dashed line and an open arrowhead. It tells us what messages flow between two process participants.
- Association: An Association is represented with a dotted line and a line arrowhead. It is used to associate an Artifact, data or text to a Flow Object.



Swimlanes

A swim lane is a visual mechanism of organizing different activities into categories of the same functionality. There are two different swimlanes, and they are:

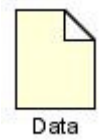
- Pool: A Pool is represented with a big rectangle which contains many Flow Objects, Connecting Objects and Artifacts.
- Lane: A Lane is represented as a sub-part of the pool. The lanes organize the Flow Objects, Connecting Objects and Artifacts more precisely.



Artifacts

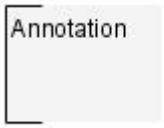
Artifacts allow developers to bring some more information into the model/diagram. In this way the model/diagram becomes more readable. There are three pre-defined Artifacts and they are:

- Data Objects: Data Objects show the reader which data is required or produced in an activity.



- Group: A Group is represented with a rounded-corner rectangle and dashed lines. The Group is used to group different activities but does not affect the flow in the diagram.

- Annotation: An Annotation is used to give the reader of the model/diagram an understandable impression.



An example process model constructed using BPMN is presented here.