UMinho | 2013

André Filipe Couto de Sousa

The Role of Simulation in Business Process Improvement: Difficulties and Solutions

Universidade do Minho
Escola de Engenharia

André Filipe Couto de Sousa

The Role of Simulation in Business Process Improvement: Difficulties and Solutions

outubro de 2013

Universidade do Minho
Escola de Engenharia

André Filipe Couto de Sousa

The Role of Simulation in Business Process Improvement: Difficulties and Solutions

Tese de Mestrado
Mestrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação do
Professor Doutor José Luís Mota Pereira

outubro de 2013

DECLARAÇÃO

Nome: André Filipe Couto de Sousa

Endereço eletrónico: andrefcsousa@gmail.com

Título da dissertação: The Role of Simulation in Business Process Improvement: Difficulties and Solutions

Orientador: Professor Doutor José Luís Mota Pereira

Ano de conclusão: 2013

Designação do Mestrado: Mestrado em Engenharia e Gestão de Sistemas de Informação

Universidade do Minho, ____ /____ /______

Assinatura: ____

Agradecimentos

Este trabalho de dissertação só foi possível graças ao apoio de um conjunto de pessoas, em relação às quais desejo expressar o meu mais profundo agradecimento.

Gostaria de agradecer em primeiro lugar ao meu orientador, o Professor Doutor José Luís Mota Pereira por me ter dado a conhecer o mundo da simulação de processos de negócio, pela sua disponibilidade, criticas e sugestões que enriquecerem este trabalho de dissertação.

Gostaria também de agradecer a toda a comunidade da ferramenta Simio que contribui também com muitas ideias e criticas, e em particular ao David Sturrock por ter reconhecido o valor deste trabalho.

Não podia deixar de agradecer também a um conjunto de amigos que conheci no Departamento de Sistemas de Informação e que me acompanharam na realização deste trabalho de dissertação, como o David Barros, Pedro Fidalgo, Jorge Mendes, Sérgio Santos e o Rui Tiago.

Por último um profundo agradecimento aos meus pais que sempre me apoiaram e contribuíram para o que hoje sou.

Abstract

Due to the constant pressures that nowadays organizations have to face, they need to constantly redefine and improve their businesses. Consequently, business process improvement initiatives are of uttermost importance to overcome these challenges. Business Process Management can be the answer, as it helps organizations to respond adequately and more quickly to the pressures around them. However, business processes are complex systems involving people, activities and technology under a great dependence, variability and complexity, which makes very difficult to forecast the systems performance and behavior.

In this context, computerized simulation can play an important role in business process improvement initiatives. First, business process simulation can be considered as a potential way to evaluate, in a controlled environment, the impact of changes on new or existing business processes, without taking the risks and costs of their implementation; second, it can also be considered as a change management tool since it allows to understand the reasons for changes, thus generating explanations for the decision-making process. However, despite its promises, current business process simulation approaches present some limitations that compromise their relevance as powerful tools to management decision making.

This work aims, in one hand, to demonstrate the importance of computerized simulation in business process improvement initiatives and, on the other hand, to solve particular problems of business processes that are difficult to represent in discrete-event simulation tools. In this context, a catalog of 34 solutions was developed with a specific simulation tool – Simio - addressing different perspectives of recurrent problems related to human behavior in business process execution, targeted for business process modelers and simulationists. This catalog demonstrates that difficult aspects of business processes can be represented in discrete-event simulation tools, contributing to the relevance of business process simulation.

Keywords: Business Process, Business Process Management, Discrete Event Simulation, Simbits

Resumo

Devido às pressões constantes que hoje em dia as organizações têm de enfrentar, estas precisam constantemente redefinir e melhorar os seus negócios. Consequentemente, as iniciativas de melhoria de processos de negócios são de suma importância para superar esses desafios. A Gestão de Processos de Negócio pode ser a resposta, uma vez que ajuda as organizações a responder de forma adequada e mais rapidamente às pressões em seu torno. No entanto, os processos de negócio são sistemas complexos que envolvem pessoas, atividades e tecnologia sob uma grande dependência, variabilidade e complexidade, o que torna muito difícil prever o desempenho e comportamento destes sistemas.

Neste contexto, a simulação computadorizada pode desempenhar um papel importante nas iniciativas de melhoria de processos de negócios. Em primeiro lugar, a simulação de processos de negócio pode ser considerada como uma forma potencial de avaliar, num ambiente controlado, o impacto das mudanças nos processos de negócio novos ou já existentes, sem assumir os riscos e os custos da sua implementação; em segundo lugar também pode ser considerada como uma ferramenta de gestão da mudança, uma vez que permite compreender as razões para tais mudanças, gerando explicações para o processo de tomada de decisão. No entanto, apesar das promessas, as atuais abordagens de simulação de processos de negócio apresentam algumas limitações que comprometem a sua relevância como poderosas ferramentas para a tomada de decisão de gestão.

Este trabalho visa, por um lado, demonstrar a importância da simulação computadorizada em iniciativas de melhoria de processos de negócios e, por outro lado, resolver problemas específicos dos processos de negócio que são difíceis de representar em ferramentas de simulação de eventos discretos. Neste contexto, um catálogo de 34 soluções foi desenvolvido com uma ferramenta de simulação específica - Simio - abordando diferentes perspetivas dos problemas recorrentes relacionados com o comportamento humano na execução de processos de negócios, direcionados para modeladores e profissionais da simulação de processos de negócio. Este catálogo demonstra que determinados aspetos dos processos de negócio podem ser representados em ferramentas de simulação de eventos discretos, contribuindo assim para a relevância da simulação de processos de negócio.

Palavras-chave: Processo de Negócio, Gestão de Processos de Negócio, Simulação de Eventos Discretos, *Simbits*

List of Contents

List of Figures

List of Tables

List of Acronyms

BPM	Business Process Management
DES	Discrete Event Simulation

1. Introduction

This chapter intends to justify the interest in the realization of this research work. First the problem contextualization is explained, which contextualizes and describes the motivations that led to the formulation of the research question. Next, the objectives defined to address the research question, the expected outcomes and the research methodology conducted. Finally, the structure of the document describing the contents of each chapter is also presented.

1.1. Problem Contextualization

Due to an increasingly global market, nowadays organizations are facing an increasing number of challenges and threats, some of them very difficult to forecast. Changes of markets, competitors, customer needs and demands are examples of challenges which organizations embrace today. Therefore, it´s seems obvious that organizations strive for better responses in order to survive and maintain high levels of competitiveness.

Business Process Management (BPM) as a new approach to the management and operation of organizations may be the answer to react adequately to these constant changes. BPM is a management discipline taking place alongside other common management disciplines, which advocates a continuous improvement and innovation of business processes in order to increase business efficiency and effectiveness. BPM implies a constant reformulation of how work is executed. For this purpose, a BPM lifecycle must be performed in order to plan, model, analyze, improve, implement and monitor a business process. In this context, computerized simulation arises as potential tool in the BPM lifecycle. First it can be used to test new scenarios without taking the risks and costs of their implementation. On the other hand, it can be seen as a change management tool in such a manner that helps to visualize the reasons for the changes.

However, despite the potential of business process simulation its relevance is reduced. First, simulation approaches are performed in an unstructured and naïve manner in organizations. Second, the other problem lays on the limitations of simulation tools regarding the modeling and simulation of many subtle aspects of business processes. Some of these aspects are related to the human behavior in business processes which are difficult to model and simulate in current simulation tools, such as the situation of human resources spending their time on multiple business processes or the tendency for batch work. Neglecting aspects like these leads to overoptimistic and unreal simulation results which compromise the relevance and potential of business process simulation. Therefore, emerged the following research question as the main driver of this research work:

How to overcome the limitations of simulation tools in business process improvement initiatives?

1.2. Objectives and Expected Outcomes

In order to answer the research question presented above the following three objectives were defined:

- To perform an exploratory study on the potential of simulation tools in the BPM lifecycle;
- To collect a set of situations concerning business processes which are difficult to model and simulate in current simulation tools;
- For each situation identified as difficult to model and simulate a suitable solution will be created.

The expected outcomes of this work are:

- A demonstration of the potential of computerized simulation tools in business process improvement contexts;
- An identification of a set of limitations of current simulation tools, regarding the modeling and simulation of business processes;
- A set of suitable solutions for each of the identified limitations described in the previous point, using a specific simulation tool.

1.3. Research Methodology

According to Berndtsson et al (2008, p. 12) the purpose of a thesis project "*is to get training in the use of a scientific method, which can then be applied when structuring and solving more complex problems*". The authors also consider a method in the context of a research project as "*an organized approach to problem-solving that includes (1) collecting data, (2) formulating a* *hypothesis or proposition, (3) testing the hypothesis, (4) interpreting results, and (5) stating conclusions that can later be evaluated independently by others*" (Berndtsson et al (2008, p. 12).

In Computer Science and Information Systems there are a range of research methods one can choose from like *qualitative methods* and *quantitative methods*. Typical Business Process Simulation research uses a case study as a research method in which data is collected from a real environment (e.g. an enterprise) to build simulation models. As described earlier, with this dissertation work it is intended to build suitable simulation models in order to overcome the limitations of simulation tools regarding modeling and simulation of human behavior in business process.

Design Science Research is related to the "design sciences", like engineering and medicine, whose core mission "*is to develop knowledge that can be used by professionals in the field in question to design solutions to their field problems*" (Aken, 2005, p. 22). Essentially, Design Science Research refers to the aggregation of new knowledge with the design of solutions related to real world problems. Therefore, the most suitable research approach to use in this research work is Design Science Research.

1.4. Structure of the Document

This document is organized in five chapters, including the present one (Introduction).
Chapter 2 presents BPM as a new approach to process management in organizations. The chapter starts by tracing an evolutionary perspective of several process management approaches that contributed to the emergence of BPM; then it explores the concepts of process-centric organizations and end-to-end business process; the BPM lifecycle and the drivers for business process change. Finally, the critical success factors that can restrain any BPM project are presented.
Chapter 3 presents Business Process Simulation and its applicability in business process improvement efforts. First, the concept of simulation is described, followed by an evolutionary perspective of the use of computerized simulation. The major simulation paradigms are referred, arguing that Discrete Event Simulation tools are the most suitable for business process simulation. Finally, some limitations of current business process simulation approaches and tools are presented.

Chapter 4 intends to describe and justify the catalog of solutions built in a specific simulation tool - Simio - in order to answer the research question. First it is presented the justification for selecting Simio as a suitable DES tool and described Simio's modeling framework. Next are presented each solution.

Chapter 5 presents the Conclusions of this work.

2. Business Process Management

Given to an increasingly global market, in which nowadays organizations have to operate, it is necessary that these respond adequately and more quickly to the pressures around them. The functional-centric organizational structure, which is the norm in traditional organizations, is not prepared to allow rapid response to current market demands, since the highly functional and hierarchical segmentation that characterize these organizations act as a barrier to constant adaptations. In this context the concept of process-centric organization and end-to-end business process emerge enabling organizations to create a global and integrated vision of work, thus allowing faster adaptation and better response (Pereira, 2004).

From an evolutionary path of several process management approaches, a new approach to managing business processes emerged under the label of Business Process Management (BPM). BPM can be seen as an integrated body of knowledge, derived from other approaches, to the management of business processes in order to achieve business objectives. BPM is not a new technology, a software tool or an episodic effort to improve a business process. BPM is an ongoing effort taking place alongside other management initiatives, to improve business processes which create business value.

This chapter starts presenting an evolutionary perspective of the transformations occurred in business markets which led organizations to rethink the way they were operating and which in some way contributed to the emerging of BPM as a new process management approach. Then, the concepts of process-centric organization and end-to-end business process are explored along with the BPM lifecycle. The chapter also presents the drivers for business process change and the critical success factors related to any BPM project implementation.

2.1. The emergence of BPM

BPM is a new label referring to a comprehensive collection of methodologies, methods, technologies and principles, derived from others process management traditions in order to managing business processes. Process management and his primordial aim to improve how work is performed to achieve business objectives is not a new concept strictly related to BPM. Harmon (2010, p. 37) argues that BPM "*is part of a tradition that is now several decades old that aims at improving the way business people think about and manage their businesses*". Process management has had an evolutionary path to the present day, with particular manifestations built on the foundations of others. Therefore it is essential to trace an evolutionary perspective of several process management approaches (see figure 1) and their contributions to what is now understood as BPM.

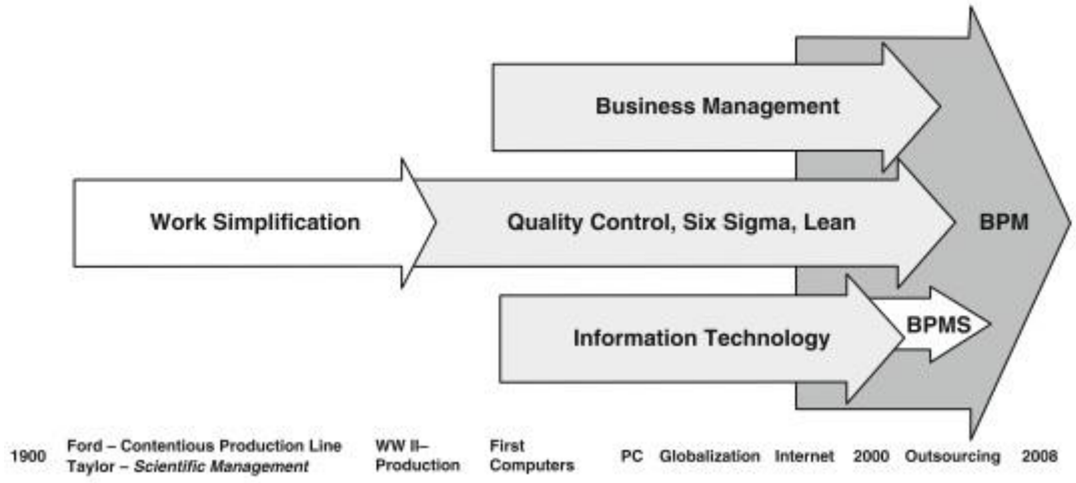


Figure 1. An overview of approaches to process management (Harmon, 2010, p. 38)

The idea that work can be viewed as a process, and then improved, dates at least to the works of Henry Ford and Frederick Taylor at the beginning of the 20th century (Davenport, 2006, p. 13). In 1903, Henry Ford introduced a moving production line bringing a new approach to manufacturing automobiles. Essentially, workers began assembling a new automobile at the beginning of the production line, and then at the end, the automobile was completed. Ford applied the Adam Smith´s principle of specialized work in order that workers on the production line had one specific task to do (Harmon, 2007, p. 11). For instance, a workers group was responsible for putting the chassis into place and another was responsible for lowering the automobiles engine. Therefore, Henry Ford conceptualized the manufacturing of automobiles as a single process with designed and sequenced activities that could be improved. (Harmon, 2010, p.39). Frederick Taylor improving the work of Henry Ford, set the stage for his renowned book, Principles of Scientific Management (Taylor, 1911). Taylor argued for time analysis, work simplification, control systems to measure and reward outputs and systematic experiments in order to identify the best way to perform some task (Harmon, 2010, p. 40). Taylor´s ideas were widely used by managers until a new approach on process management emerged at the end of the World War II by combining his ideas with statistical techniques, thus initiating the quality control tradition.

Quality control tradition emerged essentially with the works of Shewhart and Deming on statistical process control which led to other particular quality control traditions like Total Quality

Management (TQM) in the 70´s and Six Sigma and Lean ideas in the 80´s (Harmon, 2007; Harmon, 2010; Jeston and Nelis, 2006). These traditions sought to reduce the variation of processes performance to achieve consistency by using a collection of statistical techniques to measure the root causes that could be creating performance problems (Hammer, 2010, pp. 3-4). According to Hammer (2010, p. 3) "*much more important than the details of upper and lower control limits or the myriad of other analytical tools that are part of quality´s armamentarium are the conceptual principles that underlie this work*". The author highlights the usage of performance metrics in order to evaluate whether processes were being done satisfactorily or not and the never-ending continuous improvement. Nevertheless, the quality control traditions suffered from some limitations. A process was any sequence of work activities, like putting a parts box on a shelf or checking the customer credit card status, and the quality techniques could be applied to any of these processes towards consistent performance. However as Hammer (2010, p. 4) argues "*consistent is not a synonym for good. A process can operate consistently, without execution flaws, and still not achieve the level of performance required by costumers and the enterprise*". Therefore it lacked to quality control traditions an integrated vision of work which helps to fulfill the customers' expectations.

With the advent of the first computers, managers began to use information technologies to increase functional department's efficiencies. However, in most cases departmental functions hadn´t been redesign but simply automated, and consequently "*departmental efficiencies were maximized at the expense of the overall process*" (Harmon, 2007, p. 10). For example, a financial department could have used a computer to assure more accurate accounting reports forcing production to turn in reports on the status of production process. However, this could provoke delays on production caused by the elaboration of those reports and then sales department could not sell on time what they promised to customers. Concerning these problems, a new process management tradition was born under the label of Business Process Reengineering (BPR) in the early 90´s (Hammer, 1990; Hammer and Champy, 1993).

This tradition brought a new comprehensive vision on business processes enabled by information technologies. Here, a process was not any sequence of work activities but end-to-end work across an enterprise that creates customer value, and then "*BPR theorists urged companies to define all of their major processes and then focus on the processes that offered the most return on improvement efforts*" (Harmon, 2007, p. 9). BPR was also "*the first process management movement to focus primarily on non-production, white-collar processes such as order* *management and customer service*" (Davenport, 2006, p. 14). By focusing on large-scale business processes, BPR addressed the evils of functional-centric organizations: delays, errors, non-value adding, overhead and communication and coordination problems with functional departments with different goals and metrics (Hammer, 2010, p. 4). Another idea was a focus on process design rather than process execution using information technologies as an enabler of news ways of working (Harmon, 2007, p. 12).

Many of the approaches to business process improvement that emerged in the mid-90´s to 00´s were driven by software technologies. For instance, document management systems and workflow systems were used by enterprises. They initially controlled the flow of documents from one employee to another at the same functional department and then evolved into a group of integrated software modules that represented comprehensive business processes called Enterprise Resource Planning (ERP) systems (Harmon, 2007, p. 13). At the turning to the new millennium "*many people on IT realized that they could integrate a number of diverse technologies that had been developed in the late 1990s to create a powerful new approach to facilitate the day-to-day management of business processes*" (Harmon, 2007, p. 19).

The label BPM emerges, with the work of Smith and Fingar (2002), as an amalgam of methodologies, methods, technologies, principles and ideas of prior process management traditions to "*the achievement of an organization´s objectives through the improvement, management and control of essential business processes*" (Jeston and Nelis, 2006, p.11). Smith and Fingar (2002) proposed also that companies should combine workflow systems, software applications integration systems and Internet Technologies. Thus a new type of system was created by several vendors called Business Process Management System (BPMS). In essence, a BPMS allows software to be constructed using process descriptions as a foundation. This caused many vendors to sell BPM as another software tool to improve and automate business processes, thus misinterpreting the essence of BPM (Capote, 2012, p. 119).

Today BPM is understood in a larger context being defined as a "*management discipline that is taking its rightful place alongside Finance, Accounting, HR, and other mature management practices"* (Franz and Kirchmer, 2012, p. 11). This view is supported by Harmon, cited by Jeston and Nelis (2006, p. 11), defining BPM "*as a management discipline focused on improving corporate performance by managing a company´s business processes*". Another contribution to define BPM is the one from the Association of Business Process Management Professionals (ABPMP) which, in the BPM Common Book of Knowledge, defines BPM as a "*a disciplined* *approach to identify, design, execute, document, measure, monitor, and control both automated and non-automated business processes to achieve consistent, targeted results aligned with an organization's strategic goals*" (ABPMP, 2009, pg. 24).

In essence, 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. BPM enables an enterprise to align its business processes to its business strategy, leading to effective overall company performance through improvements of specific work activities either within a specific department, across the enterprise, or between organizations (ABPMP, 2009, p. 24).

2.2. End-to-End Business Process

As a result of several factors like market globalization, financial crisis and much more informed and demanding customers, organizations need to continuously improve, innovate and adapt faster in order to survive and earn competitive advantages (Pereira, 2004). As described in the previous section, until recently many organizations had a functional-centric organizational structure inspired by Adam Smith´s principle on specialized work and later by Henry Ford and Frederick Taylor. A functional-centric organization is characterized by not having a full comprehensive view about business processes. Typically, measures and rewards the outcomes of each functional department leading to communication problems between functional units and consequently the loss of the overall process. This could affect business goals and customers expectations. This view is supported by Harmon (2007, p. 7) about the problems of functional-centric organizations:

"*Each department developed its own standards and procedures to manage the activities delegated to it. Along the way, in many cases, departments became focused on doing their own activities in their own way, without much regard for the overall process. This is often referred as silo thinking, an image that suggests that each department on the organization chart is its own isolated silo*".

Some management gurus like Michael Porter and Geary Rummler viewing the problems of functional-centric organizations set the groundwork towards an integrated view of business processes focused on customer´s expectations. According to Porter, cited in Harmon (2010, p. 46), "*companies succeed because they understand what their customers will buy and proceed to generate the product or service their customers want by means of a set of activities that create, produce, sell, and deliver the product or service*". This idea was the background for Porter´s value chain concept, which emphasized that companies should focus on the essential processes that define the purpose of organizations. For instance, if a company produced jeeps, all the activities on the jeeps production should be integrated into a single value chain, thus creating a comprehensive business process. On the other hand, Rummler emphasized that organizations must align processes both vertically and horizontally. By vertically means that activity goals must be related to process goals, which must, in turn, be derived from strategic goals, and horizontally means that a process must be seen as integrated whole with goals and measures, well implemented, and a management system to assure the process execution (Harmon, 2007, p. 7).

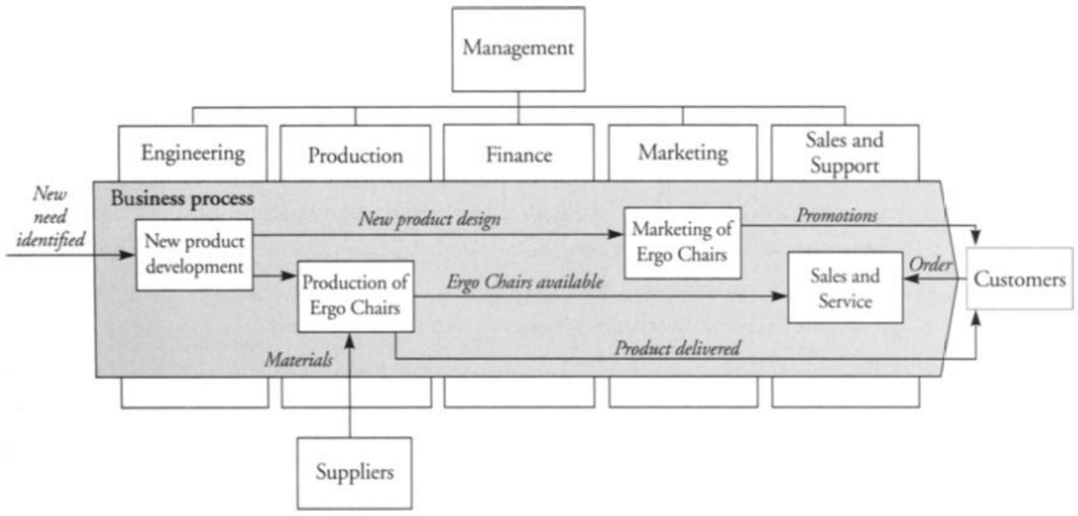


Figure 2. An end-to-end business process (Harmon, 2007, p.5)

Therefore emerged end-to-end or customer-centric business processes based on the ideas of Porter and Rummler. There are many definitions of end-to-end business processes. For instance, Hammer and Champy (1993, p.35) consider that an end-to-end business process is "*a set of activities that receive one or more types of inputs to provide value to the customer*". Davenport (1993, p.5) states that a "*process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action*".

Burlton (2001, p.72) holds the view that "*a true process comprises all the things we do to provide someone who cares with what they expect to receive*". In essence, an end-to-end business process is a sequence of work activities that creates some costumer value (see figure 2 above). Therefore, the overall process is customer-centric, producing the outcome expected by the costumer. It is important mentioning that a costumer can be an external customer or a costumer belonging to the organization.

2.3. BPM lifecycle

At this point, BPM may be understood as a new process management approach to achieve business objectives by the management and improvement of end-to-end business processes. In order to manage organizational business processes, some activities need to be performed. These activities constitute the process management lifecycle. Although there are many distinctive process management lifecycles for management and improvement of business processes the essential BPM lifecycle is pictured in figure 3.

The first step intends to design the process configuration, in other words, to define the process activities and tasks, the human and technologic resources to perform those activities, conditions, circumstances, etc. The design of the business process should be documented (e.g. using a notation like BPMN). This is not a minor, purely formal step. According to Hammer (2010, pg. 6) "*many organizations find that certain aspects of their operations are characterized by wild variation, because they lack any well-defined end-to-end process whatsoever*".

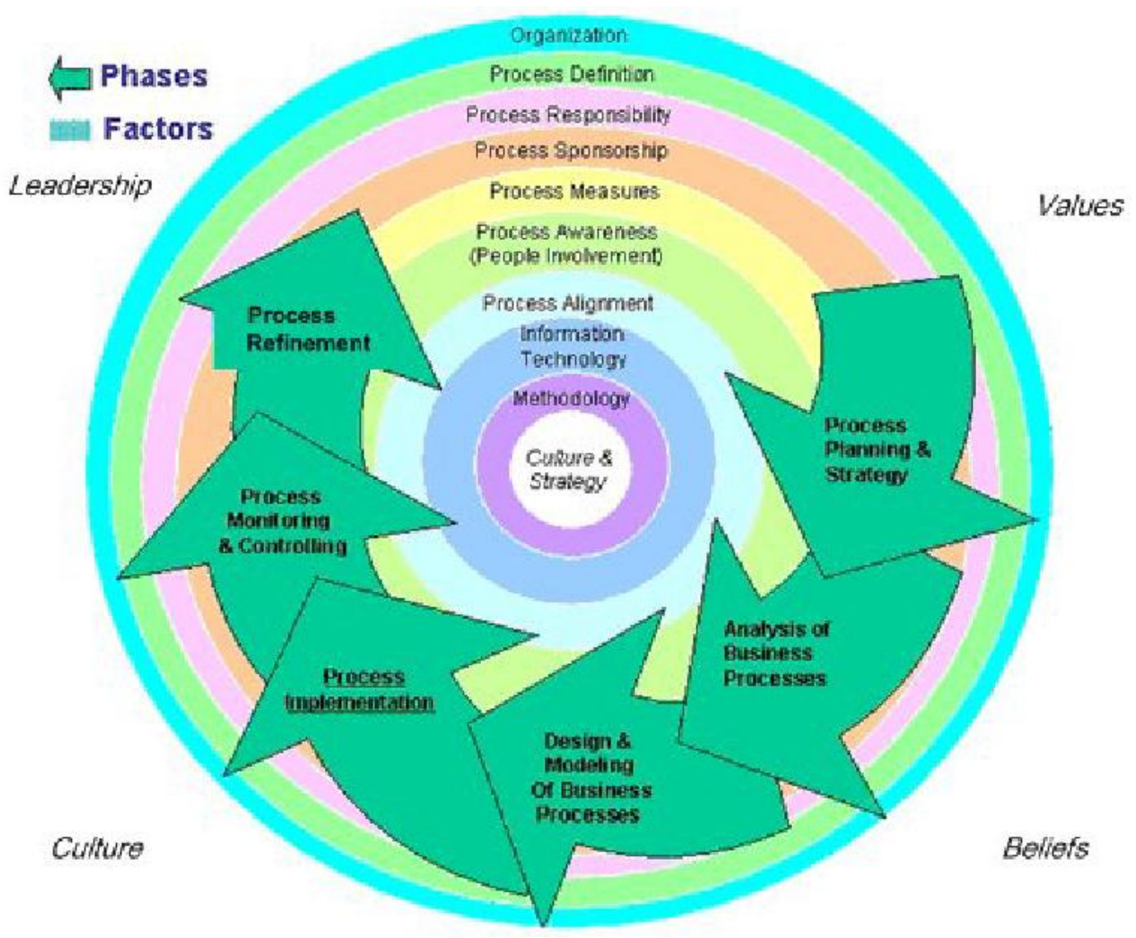


Figure 3.BPM lifecycle (ABPMP, 2009, p.2)

Once a process is in place, it needs to be managed continuously. Its performance, in terms of critical metrics that relate to customer needs and company requirements, needs to be compared to the targets for these metrics. As described earlier, such targets should be based on strategic targets. If performance does not meet targets, the reason for this flaw must be determined. According to Hammer (2010, p.7) "*processes fail to meet performance requirements either because of faulty design or faulty execution*". Once the appropriate intervention has been chosen and implemented, the results are assessed, and the entire cycle begins again.

2.4. Drivers of Business Process Change

After presenting BPM and its evolutionary path, process-centric organizations, end-to-end business processes and the BPM lifecycle, it is important to step back and discuss what aspects are driving the interest in business processes change. The answer for the interest in business process change seems very straightforward, as Harmon (2007, p. 20) states:

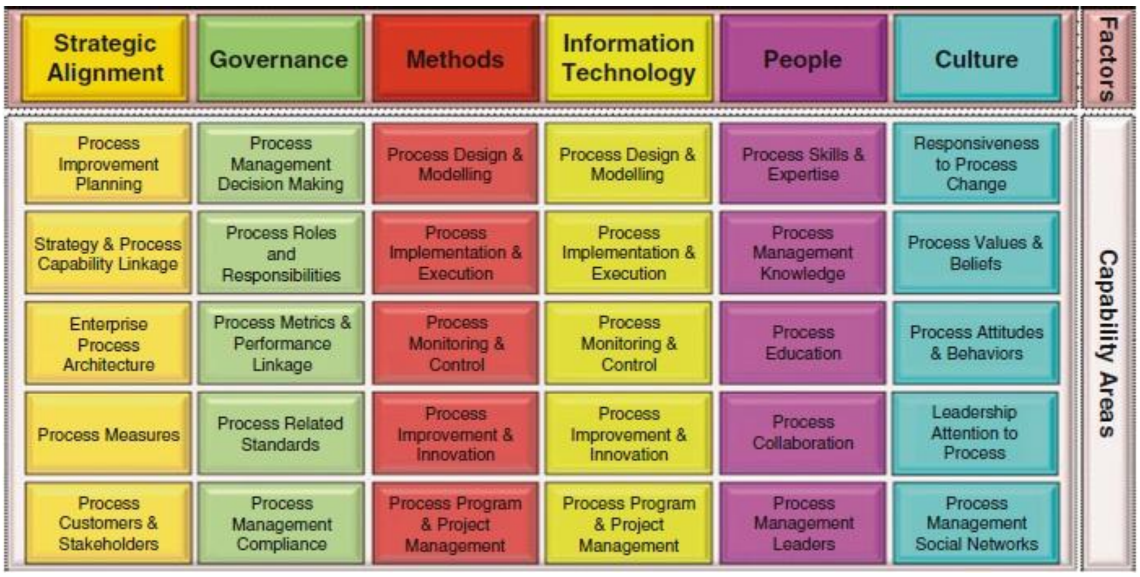
"*In economically bad times, when money is tight, companies seek to make their processes more efficient. In economically good times, when money is available, companies seek to expand, to ramp up production and to enter new markets". They improve processes to offer better products and services in hopes of attracting new customers or taking costumers away from competitors*".

However, there are also other factors beyond economic reasons. New technologies like computers, the Internet and the Web are driving business process change too. The availability of computers made it possible to do things in entirely different and much more productive ways. The Internet and Web made possible for companies to coordinate their efforts with other companies and also facilitated worldwide communication between organizations. The change from local markets to global markets boosted relentless competition which calls for constant innovation and for constant increases in productivity, and both call for an intense focus on how work gets done. To focus on how work is done is to focus on business processes. Every manager know that if his or her company is to succeed it will have to figure out how to do things better, faster, and cheaper than they are being done today, and that´s what the focus on process is all about (Harmon, 2007, p. 22).

2.5. BPM Critical Success Factors

At this point, BPM was described as a management discipline advocating an ongoing effort to continuous improvement of business processes aligned with the business strategy. Now, it is important to consider the aspects of uttermost importance in a BPM project implementation.

A BPM project has major consequences in any organization. Implementing BPM in an organization means realigning systems, authority, modes of operation, and more. Rosemann and vom Brocke (2010, pp. 112-120) identified the following six critical factors of BPM which are heavily grounded in literature: Strategic Alignment, Governance, Methods, Information Technology, People and Culture (see figure 4).



Strategic Alignment	Governance	Methods	Information Technology	People	Culture	Factors
Process Improvement Planning	Process Management Decision Making	Process Design & Modelling	Process Design & Modelling	Process Skills & Expertise	Responsiveness to Process Change	Capability Areas
Strategy & Process Capability Linkage	Process Roles and Responsibilities	Process Implementation & Execution	Process Implementation & Execution	Process Management Knowledge	Process Values & Beliefs	
Enterprise Process Architecture	Process Metrics & Performance Linkage	Process Monitoring & Control	Process Monitoring & Control	Process Education	Process Attitudes & Behaviors	
Process Measures	Process Related Standards	Process Improvement & Innovation	Process Improvement & Innovation	Process Collaboration	Leadership Attention to Process	
Process Customers & Stakeholders	Process Management Compliance	Process Program & Project Management	Process Program & Project Management	Process Management Leaders	Process Management Social Networks	

Figure 4. The Six Critical Success Factors of BPM (Rosemann and vom Brocke, 2010, p. 112)

Each of these elements, sooner or later, needs to be considered by organizations striving for success with a BPM project.

- **Strategic Alignment:** A BPM project adds value to the execution of the organizational strategy and objectives. In this context, it is important that the organizational strategy is understood by all members involved in the project. Organizational strategy "*is the common ground which ensures that all people involved are working towards the same objectives*" (Jeston and Nelis, p. 35). If this is not the case, the project should not exist, unless it is consider as a tactical project. Although Jeston and Nelis (2006, p.35) hold the view that this type of project can solve some current problem at hand in a short-term but not the root of the problem at long-term, which is a purpose of any BPM project implementation. Once the organizational strategy is clarified and understood, process architecture needs to be designed to conduct process change. Process architecture ensures that "*processes to be redesigned, or newly developed, are meeting the organization´s objectives and fit within the organization strategy*" (Jeston and Nelis, p. 81).

- **Governance:** BPM governance establishes appropriate roles and responsibilities for different levels of BPM (portfolio, program, project, and operations). BPM needs a set of governance mechanisms in order to ensure that processes integration do not turn into a new generation of horizontal silos. In this context emerges the concept of process owner. According to Hammer (2010, p. 9), "*in a conventional organization, no one is responsible for an end-to-end process, and so no one will be in a position to manage it on an end-to-end basis (i.e., carry out the process management cycle)".* It is essential for an organization striving for BPM to have process owners with authority and responsibility to conduct the BPM lifecycle.

- **Methods:** The way BPM is implemented by the organization can influence the project success. According to Jeston and Nelis (2006, p.35) "*without an agreed, structured and systematic approach to the implementation of BPM projects that takes into account the organization strategy, how it is to be executed and the significant behavioral aspects of the implementation, a project will be chaotic and have very high risks associated with it"*. Therefore a systematic approach or methodology is considered a fundamental aspect to a BPM project to achieve its objectives.

- **Information Technology:** This factor refers to the software, hardware, and information systems that enable and support the BPM lifecycle activities. The IT factor focus on the specific needs of each process lifecycle stage and its mission is to evaluate organizational needs from many viewpoints, such as customizability, appropriateness of automation, and integration with related IT solutions (e.g., data warehousing, enterprise systems, reporting). Particular IT solutions regarding BPM are Business Process Management Systems (BPMS), a specific kind of complex software system, which support all activities of the BPM lifecycle.

- **People:** Implementing a BPM project brings with it organizational change, which may carry resistance by people. As Davenport (2006) states: "*If they don't want to work in new ways, it is often very difficult to force them to do so*". Thus, it is essential to conduct process change and people management during a BPM project in the organization. Its crucial to emphasize the importance of leadership in this aspect. Without the support of a leadership committed to the project, all the effort to persuade people will vanish. Leadership not always equate to a topmost executive like a CEO. Leadership means having the attention, support, funding, commitment and time of the leader of the BPM project (Jeston and Nelis, 2006, p.34).

- **Culture:** Any organization has its own behaviors, values and habits expressed by its culture. A BPM project assumes a continuous improvement to process management with a focus on costumers, outcomes and an integrated vision of work that consequently entails more or less deep organizational changes. This approach of BPM to process management "*demands that people at all levels of the organization put the costumer first, be comfortable working in teams, accept personal responsibility for outcomes, and be willing to accept change*" (Hammer, 2010, p.10). Neglecting organizational culture and people resistance to change will certainly compromise and affect the implementation of the project. It is essential to place people at the center of the project since its beginning. They need to be informed about project objectives, the reasons for the changes and their effort should be rewarded (Oliveira, 2009, p. 9). Therefore, an organizational culture aligned with the values described above will contribute to a successful implementation of a BPM project.

2.6. Summary

This chapter presented BPM according to the social and economic developments that enabled its emergence and evolution. It was clarified that process management is not a new concept related to BPM, but a particular manifestation of an evolutionary tradition, several decades old, with the main goal to improve how managers see and manage organizations.

The concept of End-to-End Business Process was presented as a new way of seeing business processes at a larger scope and as the turning point from functional-centric organizations to process-centric organizations. Process-centric organizations have a global vision of work integrating resources scattered across the functional units of the organization, allowing greater efficiency and rapid response to changes.

The management of business processes was also explored according to the BPM lifecycle. In essence the BPM lifecycle consists of a continuous cycle to designing and implementing the business processes and measuring its performance targets by comparing them with strategic targets. If performance does not meet targets, it is necessary intervention to identify the process flaws. Once the problem is resolved and the results are assessed, the entire cycle begin again.

The drivers for business process change were also discussed. It was noted that are many factors like change on markets, competitions, Web and IT Technologies are increasing the interest on business process change.

Finally, the critical success factors concerning a BPM project (Strategic Alignment, Governance, Methods, Information Technology, People and Culture), were also presented. It was noted that each of these factors, sooner or later, during a BPM project, will have to be considered in order to reach success.

3. Business Process Simulation

Simulation tools have been used as a mean to analyze, evaluate and understand complex systems. Successfully used in industrial processes, with the emergence of BPM and its focus on the management and continuous improvement of business processes, simulation tools have the potential to be used also in the BPM lifecycle.

According to Shannon, cited in Ingalls (2011, p. 1379), simulation "is *the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system*". With this definition, business processes can be seen as complex systems concerning people, activities and technology under a great dependence, variability and complexity that make difficult forecast the systems performance and behavior. Hence, it is necessary to create simulation models of business processes in order to conduct useful experiments to understand their behavior and the impact of changes.

Indeed, the constant pressures that organizations face today, force them to rethink the way they operate, i.e. thinking in their business processes, and make decisions very quickly to assure organizational survival and competitiveness. In this context, business process simulation can be considered as a potential way to evaluate, in a controlled environment, the impact of changes on new or existing business processes without taking the risks and costs of their concretization (Oliveira, 2009, p. 44). Business process simulation according to Barnett, cited in Oliveira (2009, p.44), can also be considered as a change management support tool since it allows making visible the reasons for existing changes and allows generating explanations for the decision-making process.

This chapter intends to present the potential and relevance of business process simulation in the BPM lifecycle. It will present a brief evolution of simulation, the different simulation paradigms, with special emphasis on Discrete Event Simulation (DES), and the pitfalls of current business process simulation approaches.

3.1. Simulation Evolution

Simulation is one of the oldest applications of computing and, although its origins can be traced to pre-computer era (Goldsman and Wilson, 2010, p. 568), this section emphasizes the major events in the simulation field since the appearance of the first computers. According to Goldsman and Wilson (2010, pp. 568-570) the mid-1940´s brought two major developments in the simulation field. The first one refers to the construction of the first general-purpose electronic computers (e.g ENIAC) and the works of Stanislaw Ulman, John von Neumann and Nicholas Metropolis that used Monte Carlo method on electronic computers in order to solve certain problems in neutron diffusion that arose in the design of the hydrogen bomb. The Monte Carlo method is a statistical technique commonly used in the simulation field that uses random-number generators to simulate physical and mathematic systems (Oliveira, 2009, p. 46).

The increasing availability of general-purpose electronic computers in the 1950s allowed for the rapid proliferation of simulation techniques and applications in other disciplines. Karl Douglas Tocher developed the first general-purpose simulator, the General Simulation Program (GSP), and in 1963 wrote the first textbook about simulation - The Art of Simulation (Tocher, 1963). Also in the 60´s the first simulation languages were developed like SIMSCRIPT and SIMULA that influenced the development of general-purpose languages (Nygaard and Dahl, 1981).

While the initial focus was on programming languages extended with simulation capabilities, in the following decades gradually more and more simulation packages became available that offered some graphical environment that provided building blocks that could be composed graphically like, for instance, the Arena simulation tool (Kelton et al, 2004). Today myriads of simulation tools exist with graphical frameworks, supporting several simulation paradigms (e.g. Discrete Event Simulation).

3.2. Types of Simulation Models

Simulation is a very broad and diverse subject that encompasses many different kinds of approaches to carrying it. Although there are other ways to categorize simulation in terms of model structure, the taxonomy proposed by Sulistio et al (2004), illustrated in figure 5, is suitable to present the different types of simulation. The author categorizes simulation models in terms of presence of time, basis of value and behavior, in which a model can have a combination of properties of each category. Therefore simulation models can be static or dynamic, discrete or continuous and deterministic or probabilistic (stochastic).

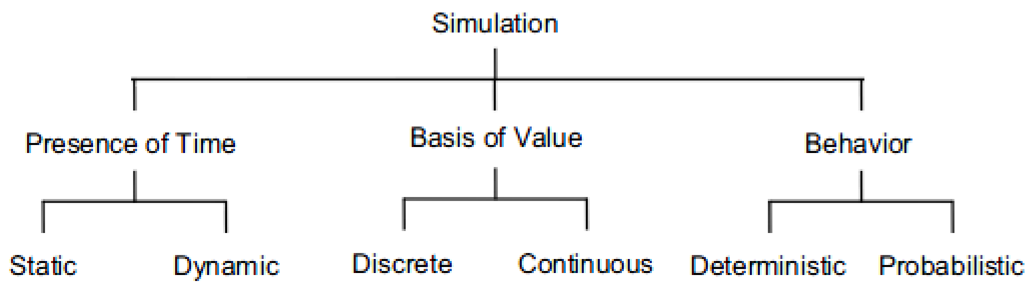


Figure 5. Simulation taxonomy (Sulistio et al., 2004, pg. 657)

Static vs. Dynamic models

Regarding to the presence of time in simulation models, these can be classified as static or dynamic. A static simulation model is one in which the presence of time is irrelevant or don't play a meaningful role in the model's operation and execution. For instance, in a simulation model of a gambling game or lottery the presence of time it's irrelevant. While there may be a notion of time in the model, unless its presence affects the model's structure and operation, it's considered static. On the other hand, in a dynamic model the presence of (simulated) time is essential to the model's structure and operation, otherwise wouldn't be possible to build a correct representation of the system. Examples include many queuing-type systems since simulated time must be present to model for instance arrival or service completion times.

Continuous-change vs. Discrete-change Dynamic Models:

Simulation models where the presence of time plays a meaningful role (dynamic models) can be classified as having continuous-change or discrete-change nature. In dynamic simulation models there are usually state variables that, collectively, describe the state of the model at any point in (simulated) time. Examples are length of queues, arrival times, processing times of services or resource states (busy, idle, etc.). If the values of state variables can change continuously over simulated time, the models are considered to have continuous aspects, like a model of a tank in which its water level (state variable) change continuously. If the values of state variables can only change at instantaneous, separated and discrete points on the simulation time, the model is then considered to have discrete-change aspects. Examples are most of queuing-type systems where state variables like queue lengths or resource states change only at the times of occurrence of discrete events like costumer arrivals and customer service completions respectively. Therefore in discrete-change models, simulated time is realized only at the times of occurrence of relevant events.

Deterministic vs. Probabilistic (Stochastic) Models:

These properties are related to the predictability nature of a simulation model's behavior, and consequently these can be classified as deterministic or probabilistic (also known as stochastic models). In deterministic models all input values are fixed, and unless there are some changes, the results will be the same from multiple runs of the model since these types of models don't have random variables. In Probabilistic (stochastic) models, on the other hand, at least some of the input variables are not fixed constants, but rather random draws (or samples) from probability distributions, for example representing variation of customer arrivals or customer service processing times.

3.3. Business process simulation with Discrete-Event Simulation

In the previous section, the different types of simulation based on model's structure were presented. This work taking into account the concept of business process, involves the modeling and simulation of systems where the presence of time is relevant, its state variables change at discrete points in time and whose behavior has a stochastic nature. This type of modeling and simulation is denominated in literature as Discrete-Event Simulation (Banks et al, 2005, p. 12). This section intends to discuss the suitability of Discrete-Event Simulation tools for business process simulation and aims to describe the common basic constructs of DES simulation tools.

According to Schriber & Brunner (2011), Discrete-Event Simulation involves a transaction-flow world view. In this world view, a system is seen as being composed of discrete units of traffic that flow from point to point in the system while competing with each other for the use of scarce (capacity-constrained) resources. Also, a discrete-event simulation is one in which the state of a model changes only at discrete, but possibly random, set of simulated time points, called event times. Numerous systems fit in the preceding descriptions being business processes one of them. Business processes are complex systems involving people, activities, resources and technology that can be seen in a transaction-flow worldview, since things, like costumers or process cases, flow through the activities and compete for limited resource availability. Therefore, discrete-event simulation tools become essential and suitable to model business processes and consequently to overcome the inherent complexities of studying and analyzing these in order to contribute to higher level of understanding and improvement. In fact, according to Hlupic et al (2003), DES tools are quite well suited for business process modeling and simulation since:

- A business process model can be easily modified to follow changes in the real system and as such can be used as a decision support tool for continuous process improvement.
- A business process in discrete-event simulation terminology relates to a time-ordered sequence of interrelated events which describes the entire experience as it flows through the system.
- The flow of information within and between business processes can be modeled as the flow of temporary entities between processing stations.
- They account for the stochastic variability of the key business process parameters, such as resources availability.
- They can capture the dynamic interdependencies among business process activities.
- They typically have built-in animation options that enable to visually monitor the business processes as they evolve over time.

Common to DES tools are a set of basic constructs to model and simulate business processes that are independent of the tool's complexity and of the discrete-event modeling paradigm used. Typically, according to Schriber & Brunner (2011) in DES tools these constructs involve: entities, resources, control elements, operations, experiments, replications and simulation clock. **Entities** are system elements that flow through the model and consume resource's capacity. They have attributes essential to their performance and behavior within the model. Examples are customers, documents flows, etc. A **resource** is a system element that provides service (such as a Worker or a machine). Typically they are capacity-limited, so entities compete for their use and sometimes must wait to use them, experiencing a delay as an outcome. **Control elements** are constructs that support delay types or logic alternatives based on the system's states. Examples are switches, counters, user data values or system data. **Operations** are steps carried on entities while they move through the system. For instance, customer arriving to a balcony and request help are two examples of operations. **Experiments** are associated with the model's execution for analysis and study. Each experiment consists of one or more replications (trials). A **replication** is a simulation that uses the experiment's model logic and data but its own unique set of random numbers, and so produces unique statistical results that can be analyzed as part of a set of such replications (all of which are independent). Basically, a replication consists of initializing the model, running it until a run-ending condition is met, and reporting results. Another important construct for the running of the model is the **simulation clock**. Its primary function is to tackle the passage of simulated time. The clock advances (automatically) in discrete steps during the run. After all possible actions have been taken at a given simulated time, the clock is advanced to the time of the next earliest event to carry out all the possible actions in the new simulated time. Therefore running a model consists of executing all possible actions at a given point in time and advance the clock to the next simulated time.

3.4. Discrete-Event Simulation paradigms

Depending on the system to be simulated, this may be modeled according to different simulation paradigms. In DES we can approach the modeling of a system with some different paradigms. This section will describe three modeling paradigms relevant to DES: event-oriented, process-oriented and object-oriented paradigms:

Event-oriented paradigm

In this modeling paradigm the system is viewed as a series of instantaneous events that change the state of the system. The modeler just needs to define the events and the logic that take place when these events occur. Although being a flexible and efficient approach to modeling, the conceptual modeling of the systems is relative. This modeling paradigm was dominant in the 60's where was implemented by Simcript tool (Carson, 1993).

Process-oriented paradigm

In process-oriented paradigm the system is described by the movement of entities through the system as a process flow. The process flow is described by a series of process steps that model the changes that take place in the system (Carson, 1993). This paradigm dates back to the 60's but became the dominant modeling approach in 80's and 90's with the emergence of graphical simulation tools such as Arena, BP Simulator or SIMUL8 that permit to build simulation models with domain-specific process libraries and take advantage of its graphical capabilities to view and validate simulation results. The vast majority of discrete event models continue to be built using the same process orientation that has been widely used for the past 25 years (Pegden, 2012).

Object oriented paradigm

This approach to modeling carries the principles of object-oriented programming to discrete-event simulation. Here, a system is described in terms of the objects that make up the system. For instance, a model of a factory is made by describing the workers, machines, conveyors, robot and the other (physical) objects that make up the system. The system behavior emerges from the interaction of these objects. Although might being simpler and less abstract than the dominant process-approach, to date many practitioners are still stick to the latter approach (Pegden, 2012). Such tools as Simio Simulation Software or AnyLogic promote the use of object-oriented paradigm.

3.5. Business Process Simulation methodology

In order to analyze and understand the complex behavior of a business process through discrete-event simulation, a sequence of steps should be performed. These steps, processes or phases are known in the literature as simulation methodologies, simulation models life-cycle or also as simulation best practices (Law and McComas, 1991, pp. 21-27; Sturrock, 2009).

The process of creating a discrete-event business process simulation starts with a detailed analysis on the business process to study. It is intended with this phase to define the simulation requirements and benchmarks. Once this phase is concluded a repetitive iterative cycle starts from the creation and improvement of a suitable simulation model to the execution of the simulation model. At the end of each cycle the simulation results are analyzed. If errors occurred or it is recognized that the simulation events do not reflect the conventional input data of the business process, the simulation model can be altered. The simulation process continues until satisfactory process behavior is achieved. Hence, the number of iterations is not fixed, and it strongly depends on the complexity of the business process in study.

According to Schiefer (2007, pp. 1731-1733) a discrete-event simulation has the following phases (see figure 6):

1) Business knowledge acquisition and analysis phase
2) Modeling phase
3) Execution phase
4) System analysis phase
5) Refactoring and development phase

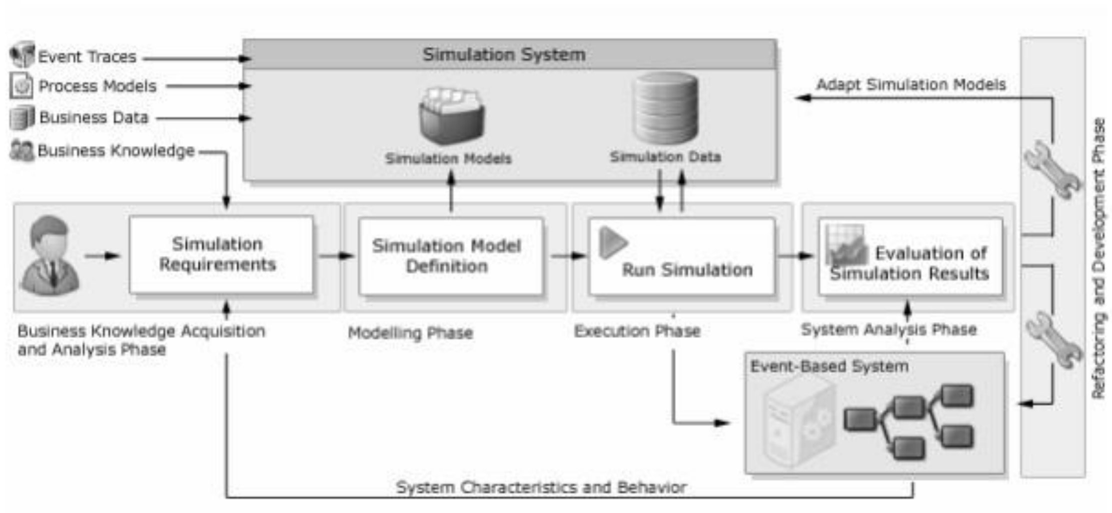


Figure 6. Discrete Event Simulation phases (Schiefer, 2007, p. 1732)

The **business knowledge acquisition and analysis phase** consists in capturing all the relevant information related to the business process in study in order to define the scope, requirements, benchmarks and scenarios of the simulation project. Typically a business process model specification (e.g a BPMN schema) is used as a starting point to business process simulation. A business process model specification gives information about the ordering of tasks, human and technologic resources, conditions, time triggers, etc. However, this information is not sufficient to simulation purposes. First of all the environment of the business process needs to be specified in simulation tools, such as process instances, process time (also called service time) of the tasks, priorities, choices, information about resources, etc. (van der Aalst et al., 2008, p. 2).

In the **modeling phase**, the simulation scenarios defined during the prior phase are transposed to a simulation model that can be executed in a simulation tool. For the simulation events to be coherent and meaningful, it is crucial to identify and model correlations and value sequences. At the end of the modeling phase, simulation scenarios are represented as executable simulation runs that assemble all the necessary operations to generate representative sequences of events and describe how to create these events to be processed by the system.

The **execution phase** starts with generating data according to the simulation model created during the modeling phase. The simulation tool executes a simulation run by generating all events of the defined sequences. The simulation ends when all simulation events are processed.

During the **system analysis phase** the performance and behavior of the simulation model is evaluated by tracing processing steps, analyzing automated decisions, calculating performance metrics and checking the processed input data for completeness.

Finally, the insights gained during the system analysis phase serve as the basis for the following **refactoring and development phase**, which consists of adapting and improving the simulation model.

3.6. Limitations of Business Process Simulation

Despite being considered relevant and highly applicable, business process simulation use is limited, as few organizations apply simulation in an effective and structured manner. Also traditional simulation approaches don't consider correctly all the aspects concerning the complexity of business processes. For instance, Van der Aalst et al (2008, pp. 5-9) argue that traditional simulation approaches focus mainly on the design analysis of new business processes to resolve future abstract problems rather than existing business process to solve concrete problems at hand (operational diagnosis and decision support purposes). Despite the abundance of simulation tools, simulation is rarely used for operational decision making. One of the reasons is the inability of traditional tools to capture the real process, along with limited support of historical data and process models (Van der Aalst et. al, 2008, p. 6).
Another problem of current business process simulation approaches is that human behavior in business processes is modeled in a very naïve manner in current simulation tools. For instance, current simulation tools assume that human resources are always eager to perform work. As a result is not uncommon that simulation models predict flow times of minutes or hours while in

reality flow times are weeks or months. Several distinct behaviors of human resources while performing work were identified in Russell et al (2006) that are relevant to business process simulation.

Van der Aalst et al (2008, pp. 8-9) and Oliveira and Pereira (2009, pp. 17-18) identified the following main problems concerning human resource modeling in current simulation tools:

People are involved in multiple processes: Typically in organizations people perform activities concurrently in several business processes daily. For instance a doctor in his workday may spend 20 percent of his daily time with one process and the remaining 80 percent with another process. Currently simulation tools assume that a resource is 100 percent dedicated to one process, which is far from the reality. In most simulation tools it is impossible to model that a resource is only available 20 percent of the time dedicated to one process. As a result simulation results are too optimistic and far from the reality.

People do not work at a constant speed: Another problem is that people work at different speeds based on their workload and priorities. Therefore, considering that is not just the distribution of attention over various business processes, but also the work speed of each human resource that determines the capacity of a particular process, it is also import to model the performance of people based in their workload in business processes.

People tend to work part-time and in batches: As described earlier, typically people spend their daily time working in multiple business processes. Moreover, they may work part-time (e.g only in the morning or afternoon). Also people have a tendency to work in batches. For instance, in many business processes the same task needs to be executed for different cases (process instances). Often people prefer to let work items related to the same task accumulate, and then process all of these in one batch. In most simulation tools this is not considered.

Priorities are difficult to model: Other important aspect that is typically neglected by simulation models is that people need to continuously choose between work-items and set priorities. A process instance may be more import than other. For instance, fulfill an order of a special client of the organization.

Process may change depending on context: Another problem is that most simulation tools assume a stable process, not considering the sometimes changing nature of business processes. Depending on the context a process configuration may change to address some special situation. Using the previous example, fulfilling a special order of some client can lead to last minute changes to the process configuration.

The work of several authors demonstrates that it is possible to overcome these problems but the legibility and quality of simulation models may become compromised by the inclusion of strange elements in the models. For instance, Oliveira and Pereira (2009) using the Arena simulation tool overcame the batches work tendency by adding, for each activity characterized by a batch work regime, a second activity whose role is only to retain the entities leaving the first activity without consuming simulation time until the batch size reaches a certain value, then releasing them to the next activity.

3.7. Summary

This chapter presented business process simulation, with special emphasis on Discrete Event simulation (DES). It was discussed the applicability and benefits of business process simulation, a very brief perspective of computer simulation evolution, the different types of simulation paradigms, the suitability of DES simulation tools, a business process simulation methodology and some limitations of current business process simulation tools.

This last point, as described above, opens the possibility of conducting experiments of business process simulation for operational purposes rather than a focus on the design of new business processes. Also, the situations identified concerning the modeling of human behavior in business processes in current simulation tools opens the possibility of conducting experiments to verify the possibility to model in a new and different simulation tool these situations, without compromising the quality and legibility of the simulation models.

4. Catalog of Simulation Patterns

In response to the research question of this work, it was intended to identify a set of situations which are difficult to model and simulate in current simulation tools and to each of them to build a suitable solution with a DES tool. The literature review presented in the previous chapters showed that although computerized simulation can be an important tool in business process improvement contexts, the current approaches to business process simulation present some limitations that compromise its relevance. One of these identified limitations comes from oversimplified models that lead to overoptimistic and unrealistic simulation results, as a consequence of the difficulty of most simulation tools to model subtle aspects of human behavior during the execution of business processes. Therefore it was decided to reunite a set of situations related to human behavior aspects and to build a suitable solution to each of them.

Concerning the first objective, the first steps consisted in finding empirical evidence of a set of situations regarding human behavior while executing business processes, and define which situation types or perspectives to address. In the previous chapter there were identified some situations related to human resource's availability and performance. Resource task allocation strategies and execution in workflow technology were also referred. It was then decided to address these four types of situations. The second step involves on how to document each identified situation and its correspondent solution. Based on the Workflow Patterns Initiative, an effort that resulted in a vast number of identified situations documented in patterns from multiple perspectives led to the idea to document each identified situation and solution in form of patterns since they are a common mean to categorize recurrent problems and solutions in a particular domain (van der Aalst & ter Hofstede, 2012). In the context of this work, the catalog's domain is business process simulation and it's addressed to business process modelers and simulationists. The catalog intends to provide a source from which these professionals can select sets of patterns that are suitable to model and simulate a given business process.

The next step involves the identification and selection of a suitable DES tool to build the solutions. Some requirements were defined for this purpose. The tool would have to provide a rapid and easy modeling approach, no necessity to extend the tool's functionality with hard-coding to model more complex and subtle situations, and to provide sufficient visualization capabilities in order to validate the solutions. Arena simulation tool was one of the first candidates in mind due to its relevance and popularity on the market of DES simulation tools (Dias et al., 2011). However the choice fell on Simio simulation tool. Comparing the tools, Simio offers a more rapid and easy to learn environment by the adoption of an object-oriented modeling approach in which systems are modeled by combining objects that represent the physical components of the these systems. The behavior of the objects can be customized using add-on processes to model particular situations in a process-oriented graphical modeling approach without having to resort to programming. Simio also provides 3D modeling and visualization capabilities. Another important aspect of Simio is the Simbit concept. Simbits are small and well-documented models that show how to solve a single common modeling problem within Simio. This concept of models demonstrating how to solve a particular problem reinforced the interest in building the catalog of patterns.

The remaining of this chapter intends to present in more detail Simio simulation software and its modeling framework, the catalog's perspectives and its correspondent pattern descriptions.

4.1. Simio Simulation Software

Simio is a multi-paradigm simulation tool with a rapid modeling framework that lets modelers quickly evaluate alternatives to reduce risks and maximize the impact of investments or decisions in a variety of systems. Simio makes modeling easier by providing a new object-oriented paradigm that radically changes the way models are built and used. The objects are just selected from libraries and graphically placed in a model. Simio includes a Standard Object Library that lets immediately start modeling. This library permits to quickly model a wide range of systems such as business processes. Another powerful aspect is the Simio's unique architecture that permits users to add new objects (e.g. domain-specific objects) or extend ones behavior with graphical processes without programming, and visualize and validate their models with a powerful 3D interface view (see figure 7). This section intends to present the basic fundamental concepts of Simio, important to later understand the solutions described in the next section. These descriptions of the fundamental concepts are based on the works of Joines & Roberts (2012) and Kelton et al (2011).

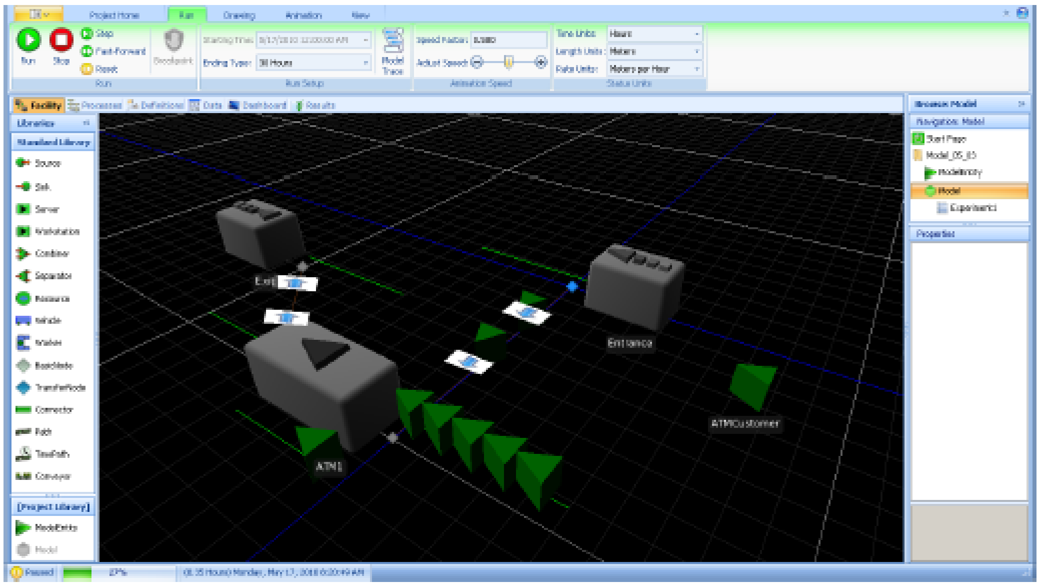


Figure 7.A Simio model in 3-D view (Kelton et al., 2011)

Objects

Object-oriented programming is accepted by many professionals of the computer-programming world as the *de facto standard* for modern software development. Discrete-event simulation is moving down the path of object-oriented modeling. The use of objects allows reducing complex problems into smaller and manageable ones. Objects also help to improve model's reliability, robustness, reusability, extensibility and maintainability. Therefore as a result, overall modeling flexibility and power are improved, and in some cases the modeling expertise required is lower. In Simio, an object is a self-contained model construct that defines that construct's characteristics, data, behavior, user interface and animation. Simio provides the Standard Library of objects (see table 1) – a general-purpose set of objects that can be used to model a range of systems. For instance, the Server object can be used to model any type of capacity-constrained service and a Resource object may be used to represent a human resource like an operator. Simio users can also build their own objects and libraries of objects.

In Simio objects are classified in terms of a three-tier object hierarchy, thus we have object definitions, instances and run spaces. An object definition defines how an object looks, behaves and interacts with other objects. It consists of its properties, states, events, external view and logic. An object instance is created when an object is dragged (instantiated) into a model. It includes the property values and may define one or more symbols but refers back to the corresponding object definition for all of its aspects. It's possible to create many instances corresponding to a single object definition. An object run space is created when a model is executed. Each object run space has its own unique ID and may reference a changeable symbol. The three-tiers of an entity object are shown in figure 8.

Name	Class	Description
Source	Fixed	Creates entities that arrive to the system.
Sink	Fixed	Destroys entities and records statistics.
Server	Fixed	Models a multi-channel service process with input/output queues.
Resource	Fixed	Models a resource that can be used by other objects.
Combiner	Fixed	Combines entities in batches.
Separator	Fixed	Separates entities from batches.
Workstation	Fixed	Models a 3-phase workstation with setup, processing, and teardown.
Vehicle	Transporter	Carries entities between fixed objects.
BasicNode	Node	A simple intersection of links.
TransferNode	Node	An intersection where entities set destination and wait on transporters.
Connector	Link	A zero-time connection between two nodes.
Path	Link	A pathway between two nodes where entities travel based on speed.
TimePath	Link	A pathway with a specified travel time.
Conveyor	Link	An accumulating/non-accumulating conveyor device.

Table 1. Simio's Standard Object Library (Pegden, 2009)

Entities and Tokens

Many simulation tools have the concept of an Entity – generally the physical things like parts and people that move around in the system and require resource's limited capacity. In Simio, entities are part of an object model and can have their own intelligent behavior. They can make decisions, reject requests, decide to take a rest, etc. They have object definitions just like other objects in the model, can be dynamically created and destroyed, move across a network of links and nodes, move through 3D space, etc. For instance, entities can represent the flow of business process cases, customers, patients, work pieces, etc.

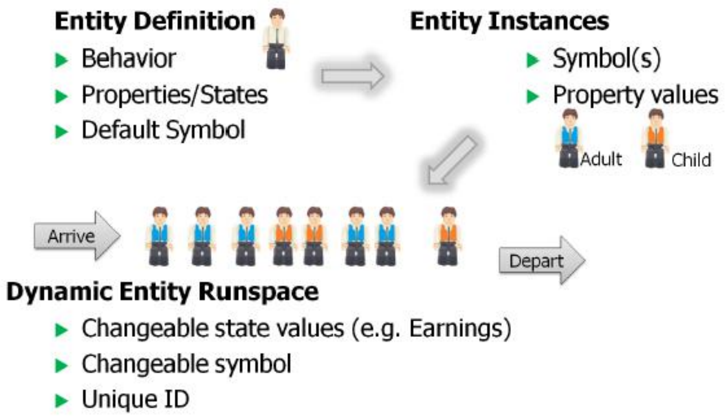


Figure 8. The three-tiers of an entity object (Kelton et al., 2011)

The movement of an entity into and out of other objects may trigger an event, which might execute a process. For instance, an entity arriving into the Processing Station of the Server object triggers a specific process that can change the state of the model. When a process is executed, a Token is created that flows through the steps of the process. A token is a delegate of an object that executes the steps in a process. A token is created at the beginning of a process and is destroyed at the end of that same process. As it moves through the process it executes the actions as specified by each step. A token carries a reference to both its parent object and its associated object (see figure 9). The parent object is an instance of the object in which a process is defined and the associated object is the object that triggered the execution of the process.

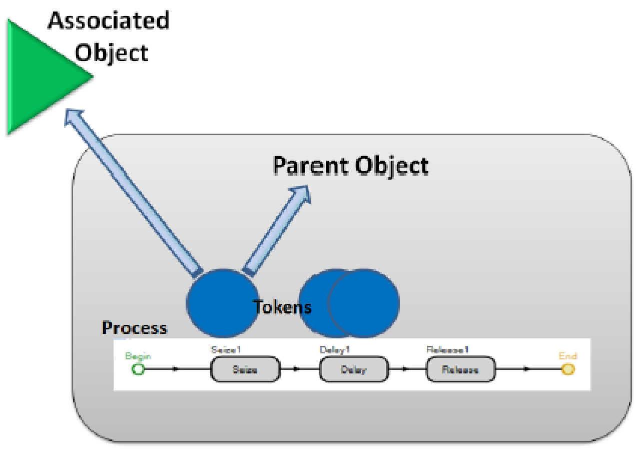


Figure 9. Relationship between Entities, Tokens and Processes (Kelton et al., 2011)

Processes

Every model built in Simio uses processes because the logic for all objects is specified using processes. A process is a set of actions that take place over time, which may change the state of the model. In Simio a process is defined as a flowchart using steps that are executed by tokens. In Simio there are three types of processes:

- *Simio-defined processes* are automatically executed by the Simio engine. For example, the OnInitialized process is executed by Simio for each object on their initialization.
- *Event-triggered processes* are user-defined processes that are triggered by an event that fires within the model
- *Add-on processes* are incorporated into an object definition to allow the user of that object to insert customized logic to model more complex and subtle behaviors.

Simbits

Other important aspect of Simio is its Simbits library. Simbits are small, well-documented models that each illustrates a modeling concept or explain how to solve a common problem. Each Simbit model is accompanied with a detailed guideline that describes how to build the model. In the context of this work, all guidelines of the solutions built can be found in the Appendixes section.

4.2. Catalog's Perspectives

It was mentioned in the previous sections that the catalog is targeted to professionals who are modeling and simulating business processes and provides a set of recurring problems, related to human resource behavior in business process execution, and correspondent solutions built with Simio simulation tool. The catalog addresses four types of perspectives: **resource's availability**, **performance**, **task allocation** and **execution**. Thirty four problems and solutions (Simbits) were identified and built, respectively. The empirical evidence of the real existence of the problems came essentially form the works of van der Aalst (2008, 2010), Oliveira & Pereira (2009) and Russell et al (2005). This catalog aims to offer a common taxonomy for Simio users. For instance, a modeler during a business process simulation project can encounter some problem for which a pattern can provide a solution. For instance, a professional that wants to model a business process in which instances of a given task must be allocated to a single resource can use the Direct Allocation (presented below) pattern which provides a solution to this problem.

Each pattern is described in five sections: **problem**, **example**, **description**, **solution** and **discussion**. The *problem* is a sentence describing a common and relevant problem or situation of human resources behavior in business process execution. The *Example* section gives one or two examples of the situation identified. The *Description* section details the problem identified. The *Solution* section indicates the name of the Simbit built to address the problem and the *Discussion* section presents how the problem can be represented in Simio and the solution built for that purpose. It's important to note that each Simbit model must have sufficient information for validation purposes. All solutions were built with the Simio Academic version. In the Appendix section one can found the guidelines to build each Simbit.

4.2.1. Resource Availability perspective

The Resource Availability perspective contains patterns that capture some phenomena about time distribution of resources over business processes and tasks. Three patterns were identified:

1. Time Distribution per Multiple Business Processes (TDMBP)

Problem: Resources distribute their working time over multiple business processes.

Example: John spends 3 hours per day performing tasks of *Handling Claims* process and 4 hours of *Creating New Contracts* process.

Description: Rarely resources only perform activities of a single process. The typical case is that resources distribute their time over multiple business processes. Another problem is that typically simulation involves the study of a single process of interest and thus in most simulation tools it is difficult or not possible to represent that a resource spends a period of time on that process. Hence, one needs to assume that a resource is there all the time and has a very low utilization. As a result simulation results are too optimistic.

Solution: *TimeDistributionPerMultiplesBusinessProcesses* Simbit.

Discussion: In Simio, resource type objects (e.g. a Worker object) can have two types of capacity: fixed or work schedule types. A fixed capacity means that the resource is available all the time, thus never goes on-shift or off-shift. A Work Schedule capacity means that the resource's capacity varies over time. A Work Schedule consists of a fixed repeating cycle of day patterns (set of work periods where the resource is available), combined with a set of exceptions. A typical repeating cycle is 8 to 5 on Monday through Friday. Exceptions can be superimposed on the pattern to define overtime, planned maintenance, vacation periods, etc. The solution built presents a Worker object that performs activities in two business processes. The Worker follows a day pattern in which works the first 2 hours on activities of process1, then works 3 hours only on process 2 and then returns to process1 activities to work another 4 hours. This is done by using an integer state variable whose values are associated with work periods. The Worker accepts and rejects entity requests as a function of the actual work period.

2. Time Distribution per Task (TDPT)

Problem: Resources distribute their time dedicated to a given business process by its tasks.

Example: Matt dedicates 5 hours daily to the *Elaborate Contract* process. He spends 3 hours on the first task and the other 2 hours on the second one.

Description: Likewise the previous pattern, resources can also distribute their time dedicated to a given business process over its tasks. It is a common difficulty of most simulation tools to define the quantity of time each task consumes from the time period that the resource dedicates to the process.

Solution: *TimeDistributionPerTask* Simbit.

Discussion: This problem is solved in Simio in a similar way of the previous pattern. In the built solution, a Worker object distributes it's time over the tasks of a simple business process by following a work schedule with a Day Pattern containing 2 work periods. An integer state variable is used to set the actual time period. The Worker changes between tasks by evaluating seize requests using the actual time period and evaluating the current entity location. When the worker goes off-shift there is a work period change and thus the variable is incremented before reaching the last work period (time period 2) for the day or initialized when reached (to 1).

3. Resource Working Part-time (RWPT)

Problem: Some resources work only on part-time periods.

Example: Charles works only in the mornings over his work week.

Description: A part-time job is a form of employment in which an individual works fewer hours per week than in a full-time job. Typically, part-time workers work fewer than 35-40 hours per week.

Solution: *WorkingPartTime* Simbit.

Discussion: In Simio this problem can be represented using a Work Schedule with a Day Pattern containing work periods that represent a typical part-time job. The solution built presents two part-time Workers. One Worker follows a morning shift from 8am-13pm and the other follows an afternoon shift from 14pm-19pm.

4.2.2. Resource Performance Patterns

The Resource Performance perspective contains patterns that capture aspect related to resources productivity while executing tasks of business processes. Two patterns were identified:

4. Resources with Different Processing Times (RWDPT)

Problem: Resources have different processing (execution) times in the different tasks of a given business process.

Example: John spends 1h to process instances of the *Review Documentation* task while Louis spents 45m.

Description: Specifying different processing times of resources in the tasks of a given business process is another limitation of simulation tools. Typically an object or element representing the resources is placed into a model and all share the same processing times. This is erroneous abstraction since in reality persons have different processing times from factors such as expertise or experience of resources in a given task.

Solution: *DifferentProcessingTimesPerTask* Simbit.

Discussion: In Simio this problem can be surpassed by creating within Simio a data model in which resources data such as processing times are related to tasks data. Using these relations one can specify for a given user which is its processing time for a given task. In the solution built there two Workers with different processing times in a task. A Data table is used to store data about each entity type and another to store information about the workers processing times for each entity type. A relationship between the tables is created using primary/foreign keys. When an entity seizes one Worker from a list at the Server, it will search the seized Worker in the list to get its position. This position will be used as an Index to obtain the corresponding processing time of the Worker to that entity type from the related rows. The processing time is assigned into the Priority property of the entity to set it as the new processing time on Server.

5. Resources with Dynamic Processing Times (DPT)

Problem: Resources productivity varies over the time based on their workload.

Example: Bill productivity decreases when he has few work items to do.

Description: Another problem of simulation tools regarding person performance is that the latter work at different speeds based on their workload. Their workload determines their productivity in a given business process. Some persons can be more productive when they many work items to dispatch while other can decrease.

Solution: *DynamicProcessingTimes* Simbit.

Discussion: In Simio this problem can be represented using a Lookup Table. A Lookup table is used to model situation where a value is dependent of other value. In the solution built, there is a Worker whose processing time for a given task varies according to its workload. This information is represented in a Lookup Table and used to get the right processing time.

4.2.3 Resource Task Allocation Patterns

The Resource Task Allocation perspective contains patterns that capture several forms of work allocation strategy that occur in business process and workflows. Twenty one patterns were identified:

6. Direct Allocation (DA)

Problem: The ability to specify at design time that instances of a given task must be allocated to a specific resource.

Example: Instances of the *Create New Contract* task must be allocated to *John*.

Description: Direct Allocation offers the ability to specify at design time the identity of the resource to which instances of a given task must be allocated at run-time. This is particularly useful when it's known that a task of a workflow can only be effectively performed by a specific resource, thus avoiding unexpected or non-suitable run-time allocations.

Solution: *DirectAllocation* Simbit.

Discussion: This problem is solved by defining specific resource objects for entities to seize capacity units. The Seize step provides functionality to set a specific resource object for seizing in a given process, e.g. in the add-on before processing trigger of a Server object. The Simbit built presents a scenario in which only one of two Worker objects can process entities at a Server.

7. Role-based Allocation (RBA)

Problem: The ability to specify that instances of a given task must be allocated to resources which correspond to a given role.

Example: Instances of the *Approve Documentation* task must be allocated to a *Manager*.

Description: Role-based Allocation offers the ability to route instances of tasks to resources based on their corresponding role within the organization. This mechanism is useful where some tasks of a given business process must be undertaken to resources with a given role.

Solution: *RolebasedAllocation* Simbit.

Discussion: This problem is surpassed in Simio using lists of resource objects to define organizational roles and using the functionality provided by the Seize step for an entity to seize an available resource object from a list. The Simbit provides a scenario with two Servers in series and where two organizational roles (A and B) are represented by two lists of Workers with two Workers each. Role A Workers are able to process entities at the first Server while Role B Workers are able to process entities at the second Server.

8. Deferred Allocation (DefA)

Problem: The ability to defer specifying the identity of the resource that will execute a task instance until run-time.

Example: During execution of a case, instances of the *Create Form* task will be executed by the resource named in the *next_value* field.

Description: Deferred Allocation regards indirect allocation in which the identity of a resource to undertake instances of a given task is postponed until run-time. Typically this is solved using a data field whose value is updated dynamically by the workflow engine with the identity of the resource to which instances of a task will be routed, thus varying the resource allocation of future instances of the task.

Solution: *Deferred Allocation* simbit.

Discussion: Indirect Allocation can be represented in Simio by seizing a specific resource using an Object Reference State Variable whose value changes by the occurrence of one or more events. There are plenty of options to change the variable's value like using a Monitor element to monitor a discrete or crossing state change, creating an event-triggered process that changes the variable's value, etc. This is particularly useful when in certain conditions the entities need to be processed by other resources than those defined at design-time. The Simbit built uses an Object Reference State Variable whose value changes between a reference of the two Workers placed in the model when the current Worker processes ten entities in the Server.

9. Authorization (Auth)

Problem: The ability to specify the range of resources that have authorization to undertake instances of a given task.

Example: Only *John*, *Matt* and *Susan* are authorized to undertake instances of the *Create Loan* task.

Description: By specifying authorizations, the range of resources that can view or execute a work-item is restricted. This is useful to prevent some events that may arise during a case execution (e.g. delegation by a resource or reallocation to other resources outside the normal process execution) that lead to unauthorized resources undertake work-items.

Solution: *Authorization* simbit.

Discussion: Likewise as Role-based Allocation problem, the concept of lists of objects is sufficient to represent the range of resources that are authorized to process entities in a given server. This solution presents two types of entities (process cases types) that require different authorized workers for processing in a server. Using a data table, each entity type is associated with a datable row containing the list of workers for processing in the server.

10. Separation of Duties (SOD)

Problem: The ability to specify that two tasks must be allocated to different human resources in a given process case.

Example: Instances of the *Countersign check* task must be allocated to a different resource to that which executed the *Prepare check* task in a given workflow case.

Description: Separation of Duties is a concept or principle of internal controls of having more than one person to complete a business process in order to prevent fraud or error. This ensures that a task cannot be performed by the same resource that executed another task within the same process case.

Solution: *Separation of Duties* simbit.

Discussion: In Simio this problem is surpassed by defining an Object Reference State variable for the entities in the model, store the reference of the worker seized in a given task and use this reference to avoid seizing it again in the task to which the separation of duties must be applied. In the built solution, each entity arriving at the first server will seize randomly an available worker from a list of workers and store its reference into an object reference state variable. At the third server the entity uses this reference to seize other worker than the one seized at the first server.

11. Case Handling (CH)

Problem: The ability to allocate all work items within a given process case to the same resource.

Example: All tasks in a given case of the *Handle Claims* process are allocated to the same *Advisor*.

Related Patterns: Retain Familiar (RA-RF), Chained-Execution (RA-CE).

Description: Case Handling is an approach to work distribution based on the premise that all work-items of a given process case are so closely related that they should all be undertaken by the same resource. Typically, the identification of the responsible resource is made when the first work-item of the process case requires allocation.

Solution: *CaseHandling s*imbit.

Discussion: Case Handling can be represented in Simio using entities (representing process cases) that always seize the same resource for processing in each Server (case's work-items) placed into the model. The identification of the resource who will conduct the case is made at the first Server. This assures that all work-items will be performed by the same resource and not by others. A case can be performed entirely meaning that an entity will seize the resource at a first Server and only releasing it after the last one (see Chained-Execution pattern) or seizing and realizing it at each Server and thus letting him processing former or later work-items of other cases allocated to it (this solution presents the latter form). If the process is performed by multiple resources an entity must save a reference of the resource seized in the first Server in order to assure that the same resource will be seized at the other Servers placed into the model.

12. Retain Familiar (RF)

Problem: The ability to allocate a work item within a given process case to the same resource that undertook a preceding work item where several resources are available.

Example: If there are several resources available to undertake the *Create Report* work item, it should be allocated to the same resource that undertook the *Prepare Report* work item in a given process case.

Description: Retain Familiar is mechanism of work distribution that when there are multiple suitable resources, favors the allocation of a work-item to the resource who already undertook a precedent work-item in a given case. This is particularly useful to expedite a process case since the resource is already aware of the details of the case and thus reduces familiarization time

Solution: *RetainFamiliar* simbit.

Discussion: This pattern and Case Handling differs from the fact the latter works on a case basis, i.e. all work-items of a given case must be allocated to some resource, while Retain Familiar works on work-item basis, i.e. the work-items can be offered and allocated to multiple resources but allocation favors the resource who undertook a precedent work-item in a given case. In the built solution, add-on Before Processing and After Processing triggers are used in Servers for seizing/releasing Workers. At Server1, entities randomly seize an available Worker. At the other Servers, entities will first check if the last seized resource is available. If not, they will attempt to seize other available Worker. The color of each entity is changed at a Server to match the one of the seized Worker in order to see which Worker the entity should attempt to seize first in the next Server.

13. Capability-based Allocation (CBA)

Problem: The ability to offer or allocate instances of a given task to resources based on their specific capabilities.

Example: Instances of the *Approve Document* task should be allocated to a Manager with a Master Degree in Economics and with a minimum of 5 years of experience.

Description: Capability-based Allocation is a mechanism of work-distribution that permits to offer or allocated work-items through the matching of specific requirements of work-items with capabilities of the potential range of resources available to undertake them. The evaluation of the resource capabilities is made at run-time to offer or allocated a work-item.

Solution: *CapabilitybasedAllocation* Simbit.

Discussion: Allocation through resources capabilities can be represented in Simio using a Resource or a Worker object representing a resource with specific capabilities or using a list of objects to represent a class of resources with specific capabilities. This is similar to Role-based Allocation or Authorization simbits since lists were also used to represent roles and authorized resources. The solution built to solve this problem uses 3 Workers representing resources with specific capabilities. Two types of entities require different Workers and a data table is used to store data about which objects each entity should attempt to seize in a given a Server.

14. History-based Allocation (HBA)

Problem: The ability to offer or allocate work-items to resources based on their previous execution history.

Example:

- Allocate the *Finalise heart bypass* task to the Surgeon who has successfully completed the most of these tasks.
- Allocate the *Core extraction task* to the drill that has the lowest utilization over the past 3 months.

Description: History-based allocation involves the use of information on the previous execution history of resources to offer or allocate them work-items. It's a method for selection of who is the best resource to conduct a task.

Solution: *History-basedAllocation* Simbit.

Discussion: In Simio this problem is solved using Matrixes to store execution's history of the different resources in the model. Data tables would permit a more flexible solution but Simio's Academy edition don't allow the use of State variables in data tables (variables whose value changes dynamically), thus is not possible to update the values of table cells. In the solution built, the resources are seized by the number of successful completions per task. This information is stored in a matrix. No failure occurs while processing an entity. An entity before processing at a server seizes an available resource based on who has the most number of successful completions for that task. After processing the entity, the number of successful completions is increased by one.

15. Organizational Allocation (OA)

Problem: The ability to offer or allocate instances of tasks to resources based on their position within the organization and their relationship with other resources.

Example: The *Authorize Expenditure* work item must be allocated to the *Manager* of the resource that undertook the *Claim Expenditure* work item in a given case.

Description: People in organizations are allocated to groups, teams or functional units and they have relationship with each other. In some processes, these relationships decide how the work is allocated to resources, and thus the capacity of representing this type of work allocations helps to get a more faithful abstraction into a simulation model.

Solution: *OrganizationalAllocation* Simbit.

Discussion: In Simio this problem can be easily surpassed using either state variables or data tables to store organizational relationships between the resources. In this Simbit a vector is used to store object references of the Managers of Worker1 and Worker2. At Server1, an entity seizes one Worker from the list, changes it's picture to match the seized Workers color and assigns an Index variable with the Worker's priority value. At Server2 the Manager of the Worker is seized using the Index.

16. Automatic Execution (AE)

Problem: The ability for an instance of a task to execute without using the services of a resource.

Example: The *Send Document* work item executes without needing to be allocated to a resource.

Description: In some processes, not all tasks require the services of a resource to be executed. Some are able to be executed automatically once the specified enabling criteria are met.

Solution: *AutomaticExecution s*imbit.

Discussion: This problem is solved in Simio using a standard Server object. The internal model of this object permits to define a capacity of resource units able for processing entities without requiring secondary resources. The solution built consists of one Source-Server-Sink model that processes entities without any secondary resource.

17. Distribution By Offer – Single Resource

Problem: The ability to offer a work item to a given resource without committing it for execution.

Example: The *Review Claim* work item is offered to David but he has the choice to accept or reject it.

Description: Distribution by offer to a single resource means that once a work item has been created and it has been determined that the work item should be offered to a single distinct resource for potential execution, the work item is offered to a resource without allocate it to him directly, i.e. the resource has the choice to accept or reject the work item.

Solution: *DistributionByOffertoSingleResource* Simbit.

Discussion: In Simio this problem can be solved using the Add-on Evaluating Seize Request process trigger in a Worker object. When an entity attempt to seize a given resource, the later can accept or reject the seize request, thus using customized logic one can model this situation. In the solution built, there are two entity types in the system. Each entity type will attempt to seize a Worker's capacity but until the Worker process 5 entities of type A it will reject any seize requests from entities of type B.

18. Distribution By Offer – Multiple Resources

Problem: The ability to offer a work item to multiple resources without committing them for execution

Example: The *Create Contract* work item is offered to David and John but either can accept or reject the work item.

Description: Offering a work-item to multiple resources is the workflow analogy to the act of “calling for a volunteer” in the real life. It provides a means to notify a group of resources that a work item exists but leaves the choice to them as to actually commit undertaking the task.

Solution: *DistributionByOffertoMultipleResources* Simbit.

Discussion: This pattern is similar to Distribution by Offer to Single Resource, the difference is in the fact that the entity attempt to seize from a list of workers. The Simbit represents a situation in which entities at a Server will seize randomly a Worker from a list of two. One of them rejects to process entities until 1h of simulation time has passed.

19. Random-basis Allocation (RBA)

Problem: The ability to allocate work-items to suitable resources on a random basis.

Example: The *Create Document* work item is allocated to an *Advisor* on a random basis.

Description: Random-based Allocation provides a non-determinist mechanism for allocating work-items to suitable resources. At run-time when the range of suitable resources is identified to undertake a work-item, one of these is selected randomly.

Solution: *RandombasedAllocation* Simbit.

Discussion: This problem is solved in Simio with a functionally provided by the Seize Step, in which we can specify the method to select Workers from a list. This is done by defining the Selection Goal property as Random. In the built solution, each entity will seize randomly a Worker from a list of three.

20. Round-Robin Allocation (RRA)

Problem: The ability to allocate a work item to available resources on a cyclic basis.

Example: Work items corresponding to the *Repair Machine* task are allocated to each available Operator on a cyclic basis.

Description: This type of cyclic allocation is also known as round robin allocation which provides a means of ensuring that all resources are allocated the same number of work items.

Solution: *RoundRobinAllocation* Simbit.

Discussion: This problem is solved in Simio by using the Routing Logic functionality of a Transfer Node object, in which entities can be routed to destinations, where it will seize specific resources, on a cyclic basis. In the Simbit, the routing logic of a Transfer Node is changed to entity to select one worker-associated transfer node from a list of nodes on a cyclic basis. When an entity enters its worker-associated transfer node assigns the Worker reference into his object reference state variable in order to seize it at Server.

21. Shortest Queue (SQ)

Problem: The ability to allocate a work item to the resource that has the least number of work item allocated to it.

Example: The *Approve Insurance Contract* is allocated to the *Actuary* who has the least number of work items allocated to them.

Description: This allocation mechanism seeks to expedite the throughput of a workflow process by ensuring that work items are allocated to the resource that is able to undertake them in the shortest possible timeframe.

Solution: *ShortestQueue* Simbit.

Discussion: This problem is solved in Simio using the Number Waiting Function of a Worker Object that return the actual number of entities waiting in the Worker's allocation queue. In the built solution, entities will seize from a list at a Server the Worker that has the least number of seize requests waiting in its allocation queue using Number Waiting function.

22. Early Distribution (ED)

Problem: The ability to advertise and potentially allocate work items to resources ahead of the moment at which the work item is actually enabled for execution.

Example: The *Create Documentation* work item is offered to a Matt at least one hour ahead.

Description: Early Distribution provides a means of notifying resources of upcoming work items ahead of the time at which they need be executed. This is useful where is necessary that resource are able to provide some form of forward commitment indicating that they will complete a work item at some future time.

Solution: *Early Distribution* Simbit.

Discussion: This problem is solved in Simio by randomly attribute a Worker to an entity for execution and transferring them into a Station Element for waiting until their time for execution. In this simbit, there two types of entities. EntTypeB entities have assigned a Worker based randomly. They are enabled for execution at intervals of 1 hour and until it is time for execution they will wait at a Station and then transferred into a Server for seizing and processing.

23. Distribution on Enablement (DOE)

Problem: The ability to advertise and allocate work items to resources at the moment they are enabled for execution.

Example: The *Approve Contract* work item is allocated to a Manager at the time it is required to commence.

Description: Distribution of work items at the time that they are enabled for execution is effectively the standard mechanism for work distribution in a workflow system. The enablement of a work item serves as the trigger for the workflow engine to make it available to resources for execution.

Solution: *DistributionOnEnablement* Simbit.

Discussion: This problem is all similar to Early Distribution since it differs from the fact that entity will attempt to seize a resource only at the time at which the entity is ready for execution. In this Simbit, EntTypeB entities are used to demonstrate this situation. They are enabled for execution at intervals of 1 hour and until it is time for execution they will wait at a Station and then transferred into a Server for seizing and processing.

24. Late Distribution (LD)

Problem: The ability to advertise and allocate work items to resources after the work item has been enabled for execution.

Example: The Service Car work item is allocated to a Mechanic after the car has been delivered for repair.

Description: Late Distribution of work item effectively provides a means of demand driving workflow process by only advertising or allocating work items to resources when the work item has already been enabled for execution, possibly at some previous time.

Solution: *Late Distribution* Simbit.

Discussion: Late Distribution can be represented in Simio simulation some condition or event that forces entities to seize workers after their time for execution. In this Simbit, there are two entity types. EntTypeB entities are used to demonstrate this situation. They are enabled for execution at intervals of 1 hour, but they will wait at a Station until the Number of entities waiting in Worker1 allocation queue is lesser or equal 3. Then only after this condition is met they are transferred into a Server for seizing and processing.

25. Escalation (EA)

Problem: The ability to offer or allocate a work item to resources other than those it has previously been offered or allocated to in an attempt to expedite the completion of the work item.

Example: The *review earning* work item was reallocated to John since it had previously been allocated to Matt but the deadline for completion had been exceeded.

Description: Escalation provides the ability to intervene in the conduct of a work item and assign it to alternative resources. Generally this occurs as a result of a specified deadline being exceeded.

Solution: *Escalation* Simbit.

Discussion: This situation can be represented in Simio by executing a process that removes an entity waiting in a Worker's allocation queue when some condition such as a deadline is met. In this Simbit, if a entity waiting in the Worker's allocation queue waits 3 or more minutes, it is removed from the queue in order to other worker process it to expedite the processing.

26. Chained Execution (CE)

Problem: The ability to allocate the next work item in a case once the previous one has completed.

Example: Immediately start the next work item in the *Handle Claims* process when the preceding one has been completed.

Description: Chained Execution is a form of work allocation that intends to expedite case throughput when a resource is allocated sequential work items within the case and when a work item is completed, it's successor is immediately initiated.

Solution: *ChainedExecution* Simbit.

Discussion: In Simio this situation can be represented using the functionality provided by the Move step to force a given seized worker to move between Servers. This problem is solved in Simio by seizing at the first server a Worker object and releasing it only at the last server. In this solution, each entity seizes the Worker at Server1 and releases it after processing in Server3. Move Step is used to force the worker to move between the Servers.

27. Additional Resource (AR)

Problem: The ability to allocate the same work item to additional resources for completion.

Example: The *Handle Order* work item needs two Operators to process it.

Description: In some situations, a given work item may require the services of multiple resources in order for it to be completed.

Solution: *AdditionalResource* Simbit.

Discussion: This problem is solved in Simio by specifying the number of units of resource object to seize in the Seize Step. In the Simbit, there are two types of entities. EntTypeB entities require 2 workers for processing while EntType only requires 1. A productivity function is used to calculate the processing times of each entity type.

4.2.4 Resource Execution Patterns

The Resource Execution perspective addresses patterns related to the way resources can manipulate work item business processes. Seven patterns were identified:

28. Resource-Determined Work Queue Content (RDWQC)

Problem: The ability for resources to specify the ordering of work items in the work queue for execution.

Example: Catherine has its work list ordered by time of receival.

Description: Enabling resources to specify the ordering of their work queue provides them with a greater flexibility in how they go about tackling the work items to which they have committed.

Solution: *ResourceDeterminedWorkQueueContent* Simbit.

Discussion: In Simio we can define in a resource-type object the order in which entities waiting in the Worker's allocation queue through the ranking rule property. Simio provides four methods for ordering waiting entities: First in First Out, Last In First Out, Largest Value First and Smallest Value First. The two latter forms need to use an expression. In the built solution, one Worker processes entities with the highest priority first. This is done using the Largest Value First ranking rule of the Worker's allocation queue and using the entity priority as the ranking expression.

29. Delegation (DL)

Problem: The ability for a resource to allocate a work item previously allocated to it to another resource.

Example: Matt passed to John is *Create Report* work item before going on leave.

Description: Delegation provides a resource with a means to re-routing work items that it is unable to execute.

Solution: *Delegation* Simbit.

Discussion: This problem is solved in Simio specifying a probability of a Worker to delegate a work item to a given resource. In this simbit, Worker1 has a probability to delegate an entity at a Server. This information is stored in a Data Table and is associated to entities by Table Assignments. If an entity is delegated it will carry a reference of another worker and will be transferred again to the Server. The entity color changes to match the color of the resource that will process the delegate entity. Green color means that the entity wasn´t delegated.

30. Deallocation (DEA)

Problem: The ability of a resource to relinquish a work item which is allocated to it and make it available for reallocation to another resource.

Example: As progressing on the Review Contract work item is not sufficient, Charles has made it available for reallocation to another resource.

Description: Deallocation provides resources with a means of relinquishing work items allocated to them and making them available for re-allocation to other resources. This may occur for a variety of reasons including insufficient progress, availability of a better resource or a general need to unload work from a resource.

Solution: *Deallocation* Simbit.

Discussion: This problem can be surpassed by defining a probability of a Worker to relinquish a given entity. Next, we have to guarantee that the relinquished entity will not seize the Worker who in turn has relinquished it. In the Simbit, a Worker has a probability to relinquish an entity at a Server. This information is stored in a Data Table and is associated to entities by Table Assignments. Entities at the Server will attempt to seize a Worker from a list. If an entity is relinquished it will be transferred to Output@Source1 node and carry a reference to the Worker that has relinquished it to prevent its seizing again from the list. Relinquished entities color changes to blue.

31.Stateful Reallocation (SFRA)

Problem: The ability of a resource to allocate a work item to another resource without loss of state data.

Example: Susan has suspended work on the *Create Report* work item and passed it to Matt for further work.

Description: Stateful reallocation provides a means to suspend a work item and reallocate it to other resources maintain the current state and results undertaken on it to date.

Solution: *StatefulReallocation* Simbit.

Discussion: The functionality provided by the Interrupt Step permits to store the remaining processing time of a suspended entity. Therefore when a Worker suspends the processing of an entity and passes it to another for execution, this latter will only process the entity by the remaining time. The solution built, there are two entity types. When an EntTypeB arrives to Server2 it will attempt to seize Worker1. If Worker1 is already seized, the entity being processed is interrupted. The interrupted entity will release Worker1 and seize Worker2 in order to process it by the remaining time.

32. Stateless Reallocation (SLRA)

Problem: The ability for a resource to reallocate a work item currently being executed to another resource without retention of state.

Example: As progress on the Review Health Plan work item is not sufficient, it has been reallocated to another resource that will restart it.

Description: Stateless reallocation provides a lightweight means of reallocation a work item to another resource without needing to consider the complexities of state preservation. A resource suspends a work item and another will restart the same.

Solution: *Staless Reallocation* Simbit.

Discussion: This problem in Simio, using the default functionality of the Interrupt Step that permits to interrupt an entity without storing the remaining processing time. In the built solution there are two entity types. When an EntTypeB arrives to Server2 it will attempt to seize Worker1. If Worker1 is already seized, the entity being processed is interrupted. The interrupted entity will release Worker1 and seize Worker2 in order to restart the suspended entity.

33. Skip

Problem: The ability for a resource to skip a work item allocated to it and mark the work item as complete.

Example: Bill has elected to skip the *Elaborate Contract* work item previously allocated to it.

Description: Skipping work items reflects the ability to expediting work processes by simply ignoring non-critical activities and assuming them to be complete such that subsequent work items can be commenced.

Solution: *Skip* Simbit.

Discussion: In Simio we can model resources skipping work items by specifying a probability of an entity to be skipped and attribute it a process time with value 0. In the built solution, a Worker has a probability to skip an entity at a Server. This information is stored in a Data Table and is associated to entities by Table Assignments. At the Server before processing if the entity is skipped its processing time is 0. The entity color changes to match the color of the resource that skipped it. Green color means that the entity wasn´t skipped.

34. Redo

Problem: The ability for a resource to redo a work item that has previously been completed in a case.

Example: Emily has decide to redo the *Review Insurance Plan* work item.

Description: Redoing work items allow a resource to repeat a work item that has previously been completed. This may happen based on a decision that the work item was not undertaken properly or because more information has become available that alters the potential outcome of the work item.

Solution: *Redo* Simbit.

Discussion: This problem can be surpassed by using a weight of an entity to be redone. This weight will in the routing logic of the link objects to route the entity back to Server from which has been redone. In the built solution a Worker has a redo weight. This information is stored in a Data Table and is associated to entities by Table Assignments. At Output Node of the Server, the path selection is made by Link Weight using the Redo Weight. An entity to redo changes his color to match the workers color.

5. Conclusions

This chapter intends to present a brief summary of this research work, to mention its contributions and limitations and finally to present some considerations about future work.

5.1 Thesis Summary

Today, as never happened before, organizations around the world face constant pressures that put them in a need to constantly reformulate and improve their businesses. Such pressures result from financial and economical conjectures, rise of emergent economies and markets, paradigm shifts, needs of consumers increasingly volatile and demanding, etc. Indeed, an effort for continuous improvement must be on the top priorities of organizations for their survival and sustainability on the actual markets.

Business Process Management can be the answer organizations need today to face these constant pressures. BPM is not just an episodic reengineering program or technology but is an ongoing effort taking place alongside other management initiatives to improve and innovate business processes while striving for efficiency and effiectiveness outcomes. BPM helps organizations to respond adequately and more quickly to the pressures around them. It advocates the implementation of a process-centric organizational structure and an integrated vision of work since the highly functional and hierarchical segmentation that still characterize many organizations act as a barrier to constant adaptation. However, business processes are complex systems involving people, activities and technology under a great dependence, variability and complexity that make difficult to forecast the systems performance and behavior, thus carrying some degree of uncertainty about the outcomes of improvement initiatives.

In this context, computerized simulation can play an important role on business process improvement initiatives. First, business process simulation can be considered as a potential way to evaluate, in a controlled environment, the impact of changes on new or existing business processes without taking the risks and costs of their concretization, and second can also be considered as a change management tool since it allows making visible the reasons for existing changes and allows generating explanations for the decision-making process. However, despite its promises, current business process simulation approaches present some limitations that compromise its relevance as a powerful tool to management decision making. These limitations come mainly from naïve and oversimplified models that don't reflect faithfully the reality. An important reason for this aspect is due to the incapacity of most simulation tools to model more complex and subtle aspects of human resources behavior in business process execution such as human resource's availability and performance concerns.

For this reason, a catalog of small simulation solutions (Simbits), built with Simio simulation tool, addressing different perspectives of recurrent problems related to human behavior in business process execution for business process modelers and simulationists was created in response to the research question behind this work.

5.2 Contributions and Limitations

During the development of this work it was intended to demonstrate the important role that DES tools can have in business process improvement contexts. This work has as a main outcome: a catalog of thirty four patterns addressing different perspectives about human resource behavior in business process execution that are difficult to model and simulate in current simulation tools. The catalog contributes to increase the relevance of business process simulation, since the solutions built with Simio simulation tool demonstrate that more complex and subtle aspects of business processes can be overcome in DES tools and, simultaneously, the catalog provides a source of situations from which a professional of business process simulation can lookup upon to represent more faithful business processes.

For Simio simulation users it also provides a relevant set of solutions to problems that may appear during the modeling of systems. It is important to underline that the director of the development team of Simio, Mr. David Sturrock, accompanied the evolution of this work and has manifested the intention to add some of the built Simbits into the next Simio's release versions. In fact some of the presented solutions are already available in Simio recent releases. Nevertheless, this work has some limitations essentially related to the Simio simulation software:

1. The solutions contained in the catalog are restricted for Simio users, therefore professionals working with other simulation tools can't benefit of these to build new models.

2. Another limitation of this work comes from the Simio software edition used to build the solutions. The majority of Simio users use the Express Edition of the software where users can only use the Standard Library of Objects to model a range of systems and can't use the flexibility of add-on processes. As seen in the patterns descriptions the majority of the patterns use add-on processes to model more complex and subtle behaviors.

5.3 Future Work

Once finalized this project, it is evident that the business process simulation domain is vast and there are always opportunities to address new problems. In this context two proposals of future work are presented, which aim to somehow continue the development of this work:

1. It would be interesting to model the thirty four patterns identified with other DES tools in order to cover a majority of business process simulation professionals.

2. Another proposal is to apply the patterns contained in the catalog in the context of a real business process simulation project to get some insights on the contribution of these in real projects.

Hopefully, this work will evolve in the future, contributing to a higher level of utilization of simulation tools in the BPM domain.

References

ABPMP, 2009. Guide to the Business Process Management Common Body of Knowledge, Version 2.0, ABPMP Release.

Aken, J.E., 2005. Management Research as a Design Science: Articulating the Research Products of Mode 2 Knowledge Production in Management. British Journal of Management, Vol.16, nº 1, pp. 19-36.

Banks, J., Carson II, J.S., Nelson, Barry, Nicol., D., 2005. Discrete-Event System Simulation, 4th Edition, Prentice Hall.

Berndtsson, M., Hansson, J., Olsson, B., Lundell, B., 2008. *Thesis Projects: A Guide for Students in Computer Science and Information Systems*, Springer-Verlag.

Burlton, R.., 2001. *Business Process Management*, Sams Publishing.

Capote, Gart., 2012. *BPM Para Todos: Uma Visão Geral, Abragente, Objetiva e Esclarecedora sobre Gerenciamento de Processos de Negócio*, BOOkeSS.

Carson, J.S., 1993. *Modeling and Simulation worldviews*. Paper presented at 25th Conference on Winter Simulation, L.A, California, USA.

Davenport, T.H., 1993. *Process Innovation: Reengineering Work Through Information Technology*, Harvard Business School Press.

Davenport, T.H., 2006. Foreword. In: Jeston, J. & Nelis, J., 2006. *Business process management: practical guidelines to successful implementations*, Elsevier, Ltd.

Dias, L. M. S., Pereira, G. A. B., Vik, P., & Oliveira, J. A., 2011. *Discrete Simulation Tools Ranking – a Commercial Software Packages comparison based on popularity.* Paper presented at 9th Annual Industrial SimulationConference, Venice, Italy.

Frank, Peter., Kirchmer, Mathias., 2012. Value-Driven Business Process Management: The Value-Switch for Lasting Competitive Advantage, McGraw Hill.

Goldsman, D., Nance, R.E. & Wilson, J.R., 2010. A Brief History of Simulation Revisited. In B. Johanson et al., eds. *Proceedings of the 2010 Winter Simulation Conference*.

Hammer, M., 1990. Reengineering Work: Dont Automate, Obliterate. *Harvard Business Review*.

Hammer, M. & Champy, J., 1993. *Reengineering the corporation: A manifesto for business revolution*, Harper Collins Publishers.

Hammer, M., 2010. What is Business Process Management?. In: Vom Brocke, J. & Rosemann, M., 2010. *International Handbooks on Information Systems - Handbook on Business Process Management Part 1*, Springer-Verlag.

Harmon, P., 2007. *Business Process Change* Second Edi., Morgan Kaufmann Publishers.

Harmon, P., 2010. The Scope and Evolution of Business Process Management. In: Vom Brocke, J. & Rosemann, M., 2010. *International Handbooks on Information Systems - Handbook on Business Process Management Part 1*, Springer-Verlag.

Hlupic, V., de Vreede, G.-J., Orsoni, A., 2003. Modeling and Simulation Techniques for Business Process Analysis and Re-Engineering. *I.J of Simulation*, 7(4), pp.1–8.

Hlupic, V., 1997. Business Process Modelling Using Discrete Event Simulation: Potential Benefits and Obstacles for Wider Use. *I.J of Simulation*, 4(1), pp.62–67.

Hook, G., 2011. Business Process Modelling and Simulation. In S. Jain et al., eds. *Proceedings of the 2011 Winter Simulation Conference*.

Ingalls, R., 2011. Introduction to simulation. In S. Jain et al., eds. *Proceedings of the 2011 Winter Simulation Conference*.

Jeston, J. & Nelis, J., 2006. *Business process management: practical guidelines to successful implementations*, Elsevier, Ltd.

Joines, J., Roberts, S., 2012. Simulation Modeling with Simio: A Workbook, Second Edition, North Carolina State University.

Kelton, W.D., Smith, J.S., Sturrock, D., 2011. Simio and Simulation: Modeling, Analysis, Applications, Second Edition, Simio LLC.

Kelton, D.W., Sadowski, R., Sturrock, D., 2004. Simulation With Arena, McGraw-Hill Professionals.

Law, A. & McComas, M., 1991. Secrets of Successful Simulation Studies. In *Proceedings of 1991 Winter Simulation Conference*.

Nygaard, K. & O.-J. Dahl., 1981. The development of the SIMULA languages. In *History of programming languages*, ed. R. L. Wexelblatt, 439Q493. New York: Academic Press.

Oliveira, P., 2009. *Simulação de Processos em Projectos de Reengenharia Organizacional*. M.Sc thesis. University of Minho, Portugal.

Oliveira, P. & Pereira, J.L., 2009. A Simulação Computorizada no Suporte à Optimização e Melhoria Contínua de Processos Organizacionais. In *Proceedings da Conferência da Associação Portuguesa de Sistemas de Informação 2009*.

Pegden, Dennis C., 2012. Intelligent Objects: The Future of Simulation, Simio LLC. Available at: http://www.simio.com/resources/white-papers/Intelligen-objects/Intelligent-Objects-The-Future-of-Simulation-Page-1.htm

Pegden, Dennis C., 2009. An Introduction to Simio ® for Arena Users. *White Paper at Simio LLC.*,pp.1–6. Available at: http://www.simio.com/resources/white-papers/For-Arena-Users/Introduction-to-Simio-for-Arena-Users.pdf

Pereira, J. L., 2004. Sistemas de Informação para o Novo Paradigma Organizacional: O Contributo dos Sistemas de Informação Cooperativos, Tese de Doutoramento, Universidade do Minho.

Robinson, Stewart., 2012. Tutorial: Choosing What to Model – Conceptual Modeling for Simulation. In *Proceedings of the 2012 Winter Simulation Conference*.

Rosemann, Michael., vom Brocke, Jan., 2010. The Six Core Elements of Business Process Management. In: Vom Brocke, J. & Rosemann, M., 2010. *International Handbooks on Information Systems - Handbook on Business Process Management Part 1*, Springer-Verlag.

Russell, N., van der Aalst, W.M.P., Hofstede, Arthur., Edmond, David., 2005. Workflow Resource Patterns : Identification , Representation and Tool Support. In O. Paster & J. Falcão e Cunha, eds. *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05), volume 3520 of Lecture Notes in Computer Science*. Berlin: Springer-Verlag, pp. 216–232.

Schiefer, J., Roth, Heinz., Suntinger, Martin., Schatten, Alexander., 2007. Simulation Business Process Scenarios for Event-based systems. In *Preceedings of 15th European Conference on Information Systems (ECIS'07)*. University of St. Gallen, pp. 1729–1740.

Schriber, T.J. & Brunner, D.T., 2011. Inside Discrete-Event Simulation Software: How it works and why it matters. In *Proceedings of the 2011 Winter Simulation Conference*.

Smith, H. & Fingar, P., 2002. *Business Process Management – The Third Wave*, Meghan-Kiffer Press.

Sturrock, D.T., 2009. Tips for successful practice of simulation. In *Proceedings of the 2009 Winter Simulation Conference*.

Sulistio, A., Yeo, C.S. & Buyya, R., 2004. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Software: Practice and Experience*, 34(7), pp.653–673.

Tocher, K. D., 1963. *The Art of Simulation*. London: The English Universities Press Ltd.

Aalst, W.M.P. & ter Hofstede, a. H.M., 2012. Workflow patterns put into context. *Software & Systems Modeling*, 11(3), pp.319–323.

van der Aalst, W.M.P., 2010. Business Process Simulation Revisited, Eindhoven University of Technology, Eindhoven, The Netherlands.

van der Aalst, W.M.P., Nakatumba, J., Rozinat A., Russell, N., 2008. Business Process Simulation : How to get it right ? , Eindhoven University of Technology, Eindhoven, The Netherlands.

Appendixes

A – Resource Availability Simbit Guidelines

Time Distribution per Multiple Business Processes (TDMBP)

Problem:

I want to model a resource that distributes its time over multiple business processes.

Categories:

Entity - - Workers

Key Concepts:

Worker, Server, Source, Sink, Path, Add-On Before Processing Trigger, Add-on After Processed Trigger, Seize Step, Release Step, Assign Step, List, Integer State Variable, Work Schedule, Day Pattern, Add-on On Evaluate Seize Request trigger, Add-on Off-Shift trigger

Assumptions:

There are two workers in the system. Worker1 distributes it's time over the two business processes while Worker2 only works in Business Process2. Worker1 has the following work periods: From 8:00 am to 10:00 am works in business process 1, 10:01am to 13:00 pm in business process 2 and 14:00 pm to 18:00 pm in business process1.

Technical Approach:

Worker1 distributes it's time over multiple business processes by following a work schedule with a Day Pattern containing 3 work periods. An integer state variable is used to set the actual time period. Worker1 changes between processes by evaluating seize requests using the actual time period. When the worker goes off-shift there is a work period change and thus the variable is incremented before reaching the last work period (time period 3) for the day or initialized when reached (to 1).

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server, and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Set Server1 *Initial Capacity* to ‘Infinity’.
- Drag 1 Source, 2 Servers, and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Set Server2 and Server3 *Initial Capacity* to ‘Infinity’.
- Add 2 Worker objects into the facility view. Set ‘Output@Source1’ Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to ‘Go To Home’. Lastly, set the Initial Network as ‘No Network (Free Space)’.
- Change the color of Worker1 to Yellow by clicking in Color in the ribbon.

<u>Defining the Entities:</u>

- Add 2 ModelEntity objects into the Facility View and add 3 additional symbols. Change their names and color respectively to ‘Process1Entities’ and ‘Process2Entities’ and Yellow and Red.

<u>Creating a list of Workers:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it ‘ResourceList’ and add Worker 1 and Worker 2 into the List. This List will be used to a Process2Entity seize one Worker.

<u>Creating an Integer Variable associated with Worker1 Work Periods:</u>

- In the Definitions window of the model, go to the States panel
    - Create a new Integer State Variable by clicking on Integer in the ribbon. Name this new state ‘Worker1TimePeriod’.

<u>Create a Work Schedule and Day Pattern for Worker1:</u>

- In the Data Window, select the Schedules panel and add a Work Schedule with the *Name* ‘Worker1Schedule’.
- Create a Day Pattern by clicking on the Day Pattern tab and enter the *Name* ‘Worker1DayPattern’.
- Click on the ‘+’ sign to expand the Work Periods.
- Enter the Start Time and End Time from 8:00 am to 10:00am, 10:01 am to 13:00 pm and 14:00 pm to 18:00 pm. Leave everything else empty, which indicates Off Shift times.
- Under Work Schedules, select ‘Worker1DayPattern’ for Monday through Friday.

- Click in Worker1 and change *Capacity Type* to 'WorkShedule' and *Work Schedule* to 'Worker1Schedule'.

<u>Adding Process Logic for Evaluate Seize Requests in Worker1:</u>

- Click on Worker1 and create a new process in the Evaluating Seize Request add-on process trigger. This process will be executed each time an entity is attempting allocation of Worker1.
    - Add a Decide Step into the process to check if the entity is of Type A. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Is.Process1Entities'.
    - In the True path add a Assign step. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'True'.
    - In the False path of the step add other Decide to check if the actual time period is 2. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'Worker1TimePeriod == 2. In the true path of this step add an Assign Step to accept EntTypeB seize requests. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'True'. In the False step add an Assign Step to reject EntTypeB seize requests. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'False'.

<u>Adding Process Logic to change time periods in Worker1 when it goes off-shift:</u>

- Click on Worker1 and create a new process in the Off Shift add-on process trigger.
    - Add a Decide Step into the process to check if the entity is of Type A. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'Worker1TimePeriod == 3'.
    - In the True path add a Assign step. Set the *State Variable Name* to 'Worker1TimePeriod' and *Expression* to '1'.
    - In the False path of the step add an Assign Step to accept. Set the *State Variable Name* to 'Worker1TimePeriod' and *Expression* to 'Worker1TimePeriod+1'.

<u>Adding Process Logic for Seizing and Releasing the Workers at Servers:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change

Object Type to 'Specific', *Object Name* to 'Worker1', Request Move from 'None' to 'To Node' and *Destination* to 'Input@Server1'.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific, *Object Name* to 'Worker1'.
- Click on Server2 and create a new process in the Before Processing add-on process triggers and add a Seize Step. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourcesList', Request Move from 'None' to 'To Node' and *Destination* to 'Input@Server2'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to release the worker. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourcesList'.
- Click on Server3 and create a new process in the Before Processing add-on process triggers and add a Seize Step. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourcesList', Request Move from 'None' to 'To Node' and *Destination* to 'Input@Server3'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.
- Click on Server3 and create a new process in the After Processing add-on process triggers. Add a Release Step to release the worker. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourcesList'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Time Distribution per Task (TDT)

Problem:

I want to model a resource that distributes its time over the tasks of a given business process.

Categories:

Entity - - Workers

Key Concepts:

Worker, Server, Source, Sink, Path, Add-On Before Processing Trigger, Add-on After Processed Trigger, Seize Step, Release Step, Assign Step, List, Integer State Variable, Work Schedule, Day Pattern, Add-on On Evaluate Seize Request trigger, Add-on Off-Shift trigger

Assumptions:

There is one Worker in the system. Worker1 distributes it's time over the tasks of a business process. Worker1 follows two work periods: From 8:00 am to 12:00 am works in Server1 while from 14:00 pm to 17:00 pm works in Server2.

Technical Approach:

Worker1 distributes it's time over tasks of a business process by following a work schedule with a Day Pattern containing 2 work periods. An integer state variable is used to set the actual time period. Worker1 changes between processes by evaluating seize requests using the actual time period and evaluating the current entity location. When the worker goes off-shift there is a work period change and thus the variable is incremented before reaching the last work period (time period 2) for the day or initialized when reached (to 1).

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server, and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Set Server1 *Initial Capacity* to 'Infinity'.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Lastly, set the Initial Network as 'No Network (Free Space)'.

<u>Creating an Integer Variable associated with Worker1 Work Periods:</u>

- In the Definitions window of the model, go to the States panel

    - Create a new Integer State Variable by clicking on Integer in the ribbon. Name this new state 'TimePeriod'.

Create a Work Schedule and Day Pattern for Worker1:

- In the Data Window, select the Schedules panel and add a Work Schedule with the *Name* 'Worker1Schedule'.
- Create a Day Pattern by clicking on the Day Pattern tab and enter the *Name* 'Worker1DayPattern'.
- Click on the '+' sign to expand the Work Periods.
- Enter the Start Time and End Time from 8:00 am to 12:00am and 14:00 pm to 17:00 pm. Leave everything else empty, which indicates Off Shift times.
- Under Work Schedules, select 'Worker1DayPattern' for Monday through Friday.
- Click in Worker1 and change *Capacity Type* to 'WorkSchedule' and *Work Schedule* to 'Worker1Schedule'.

Adding Process Logic for Evaluate Seize Requests in Worker1:

- Click on Worker1 and create a new process in the Evaluating Seize Request add-on process trigger. This process will be executed each time an entity is attempting allocation of Worker1.
    - Add a Decide Step into the process to check if the entity is of Type A. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'TimePeriod == 1 || TimePeriod == 0.
        - In the True path add another Decide Step to check if the actual location of the requesting entity is Server1. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Location.Parent == Server1'.
            - In the True path add a Assign step. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'True'.
            - In the False path add a Assign step. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'False'.
        - In the False path add another Decide Step to check if the actual location of the requesting entity is Server2. Set *Decide Type* property to

'ConditionBased' and the *Expression* to 'ModelEntity.Location.Parent == Server2'.

        - In the True path add a Assign step. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'True'.
        - In the False path add a Assign step. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'False'.

Adding Process Logic to change time periods in Worker1 when it goes off-shift:

- Click on Worker1 and create a new process in the Off Shift add-on process trigger.
    - Add a Decide Step into the process to check if the entity is of Type A. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'TimePeriod == 2'.
    - In the True path add a Assign step. Set the *State Variable Name* to 'TimePeriod' and *Expression* to '1'.
    - In the False path of the step add an Assign Step to accept. Set the *State Variable Name* to 'TimePeriod' and *Expression* to 'TimePeriod+1.

Adding Process Logic for Seizing and Releasing the Workers at Servers:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', Request Move from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific, *Object Name* to 'Worker1'.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', Request Move from 'None' to 'To Node' and *Destination* to 'Input@Server2'.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the

Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific, *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Working Part-time (RWPT)

Problem:

I want to model two resources that work part-time.

Categories:

Entity - - Workers

Key Concepts:

Worker, Server, Source, Sink, Path, Add-On Before Processing Trigger, Add-on After Processed Trigger, Seize Step, Release Step, Assign Step, List, Integer State Variable, Work Schedule, Day Pattern

Assumptions:

There are two Workers in the system. Worker1 follows a morning shift from 8am-13pm and Worker2 follows an afternoon shift from 14pm-19pm.

Technical Approach:

Representing resources working part-time is done by setting capacity types of Worker1 and Worker2 with Work Schedule, in which Day Patterns define a single work period representing a part-time job.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server, and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Set Server1 *Initial Capacity* to 'Infinity'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Workers. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Lastly, set the Initial Network as 'No Network (Free Space)'. Color Worker2 with Red by clicking in Color in the ribbon.

<u>Creating a list of Workers:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'ResourceList' and add Worker 1 and Worker 2 into the List.

<u>Create a Work Schedule and Day Pattern for the Worker1:</u>

- In the Data Window, select the Schedules panel and add a Work Schedule with the *Name* 'Worker1PartTimeSchedule'.

- Create a Day Pattern by clicking on the Day Pattern tab and enter the *Name* ‘MorningShift’.
- Click on the ‘+’ sign to expand the Work Periods.
- Enter the Start Time and End Time from 8:00 am to 13:00pm.
- Under Work Schedules of Worker1PartTimeSchedule, select ‘MorningShift’ for Monday through Friday.
- Click in Worker1 and change *Capacity Type* to ‘WorkSchedule’ and *Work Schedule* to ‘Worker1PartTimeSchedule’.

<u>Create a Work Schedule and Day Pattern for the Worker2:</u>

- In the Data Window, select the Schedules panel and add a Work Schedule with the *Name* ‘Worker2PartTimeSchedule’.
- Create a Day Pattern by clicking on the Day Pattern tab and enter the *Name* ‘Afternoon Shift’.
- Click on the ‘+’ sign to expand the Work Periods.
- Enter the Start Time and End Time from 14:00 am to 19:00pm.
- Under Work Schedules of Worker1PartTimeSchedule, select ‘Afternoon’ for Monday through Friday.
- Click in Worker1 and change *Capacity Type* to ‘WorkSchedule’ and *Work Schedule* to ‘Worker2PartTimeSchedule’.

<u>Adding Process Logic for Seizing and Releasing the Workers at Server 1:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘From List’, *Object List* to ‘ResourcesList’, Request Move from ‘None’ to ‘To Node’ and *Destination* to ‘Input@Server1’. Expand Advanced Options and change the value of *Selection Goal* to ‘Random’.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘FromList’, *Object List* to ‘ResourcesList’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

B – Resource Performance Simbit Guidelines

Different Processing Times per Task (DPTPT)

Problem:

I want to model two resources that have different processing times in a given task.

Categories:

Entity - - Workers

Key Concepts:

Worker, Server, Source, Sink, Path, Add-On Before Processing Trigger, Add-on After Processed Trigger, Seize Step, Release Step, Assign Step, List, Data Table, Search Step, Add-on On Created Entity trigger, Real State Variable, Set Row Step

Assumptions:

There are two Workers in the system and two entity types from an entity instance.

Technical Approach:

A Data table is used to store data about each entity type and another to store information about the workers processing times for each entity type. A relationship between the tables is created using primary/foreign keys. When an entity seizes one Worker from a list at Server1, it will search the seized Worker in the list to get its position. This position will be used as an Index to obtain the corresponding processing time of the Worker to that entity type from the related rows. The processing time is assign into the Priority property of the entity to set it as the new processing time on Server1.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server, and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Set Server1 *Initial Capacity* to 'Infinity' and *Processing Time* to 'ModelEntity.Priority'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Lastly, set the Initial Network as 'No Network (Free Space)'.
- Change the color of Worker2 to Yellow by clicking in Color in the ribbon.

<u>Defining the Entities:</u>

- Add 1 ModelEntity object into the Facility View and add an additional symbol. Change additional symbol number 1 to Black.
- In the Definitions window of the model, go to the States panel
    - Create a new Real State Variable by clicking on Integer in the ribbon. Name this new state 'StaWhichRes'.

<u>Creating a list of Workers:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'WList' and add Worker 1 and Worker 2 into the List.

<u>Creating 2 related Data Tables:</u>

- Go to the Data Window, and within the Tables panel, click the Add Data Table icon the ribbon to add a new table. Name it 'EntityInfo'.
- Add 2 columns that are a Integer standard properties by selecting each data type from the Standard Property drop down in the ribbon. Change the names of each column to 'Type' and 'Percentage' respectively.
- Add a third column to the table – an Object List reference by selecting Object List from the Object Reference drop down in the ribbon. The column can be renamed to 'Resources'.
- Click on Type column and click on *Set Column As Key* button to set this column as the primary key for the table.
- Fill the table with data – such as:
    - 0, 60, WList
    - 1, 40, WList
- Add another Data Table. Name it 'Worker Processing Times'. Add 2 columns that are an Integer and Expression standard properties by selecting each data type from the Standard Property drop down in the ribbon. Change the names of each column to 'Type' and 'ProcessingTime' respectively.
- Click on Type column and change Table Key property to 'EntityInfo.Type' to create a relationship between the tables.
- Fill the table with data – such as:
    - 0, 20
    - 0, 7

    - 1, 10
    - 1, 15

<u>Adding Process Logic to associate a table row with a created entity:</u>

- Click on Server1 and create a new process in the Created Entity add-on process trigger. Add a Set Row step in this process. Change *Table Name* property to 'EntityInfo' and *Row Number property* to 'EntityInfo.Percentage.RandomRow'.
- Add an Assign Step to the process. Set *State Variable Name* to 'ModelEntity.Picture' and *New Value* to 'EntityInfo.Type'.

<u>Adding Process Logic for Seizing and Releasing the Workers at Server1 and set the corresponding processing time.</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers and add a Seize Step. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'EntityInfo.Resources', Request Move from 'None' to 'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.
- Add a Search Step to the process. Change the following properties:
    - *Collection Type* to 'Object List'
    - *Object List* Name to 'EntityInfo.Resources',
    - *Match Condition* to 'ModelEntity.SeizedResources.LastItem==Candidate.Worker'
    - *Save Index Found* to 'ModelEntity.StaWhichRes'.
- Add an Assign Step to the process. Set *State Variable Name* to 'ModelEntity.Priority' and *New Value* to '60 * WorkerProcessingTimes[ModelEntity.StaWhichRes].ProcessingTime'.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to release the worker. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'EntityInfo.Resources'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Dynamic Processing Times (DPT)

Problem:

I want to model a resource whose processing time varies in function of its workload.

Categories:

Entity - - Workers

Key Concepts:

Worker, Server, Source, Sink, Path, Add-On Before Processing Trigger, Add-on After Processed Trigger, Seize Step, Release Step, Lookup Table

Assumptions:

There is one Worker in the system. The standard processing time of Worker is 10 minutes. Its efficiency is reduced when 2 or more entities are waiting in Server1 Input Buffer.

Technical Approach:

In this Simbit, a Lookup table is used to vary Worker1 processing time in function of the number of entities waiting in Server1 Input Buffer.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server, and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Set Server1 *Initial Capacity* to 'Infinity'.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Lastly, set the Initial Network as 'No Network (Free Space)'.

Creating a Lookup Table and using it in Server1 Processing Time expression:

- Go to the Data Window, and within the Lookup Tables panel, click on Lookup Table button to create a lookup table. Name it 'Efficiency'.
- Fill the table with data – such as:
    - 0, 1
    - 1, 1
    - 2, 0,8
    - 5, 0,5
- Click on Server1 and change Processing Time property to '10/Efficiency[Server1.InputBuffer.Contents.NumberWaiting]'.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', Request Move from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific, *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

C – Resource Task Allocation Simbit Guidelines

Direct Allocation (DA)

Problem:

I want to model the ability of some process-aware information systems to allocate instance of a task to a specific resource.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step

Assumptions:

There are two servers in series and two workers in the system. Only Worker2 can process entities at Server2.

Technical Approach:

Add-on process triggers are used for seizing/releasing the Workers at each Server objects placed into the model. Entities can seize both Workers from a list at Server1 while on Server2 only Worker2 is able to process entities.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 2 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each Worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

<u>Creating a list of Workers:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'ResourcesList' and add Worker 1 and Worker 2 into the List. This List will be used to an entity seize one Worker at the Server1.

Adding Process Logic for Seizing and Releasing the Workers at the Servers:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourceList', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourcesList'.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize Worker2. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker2', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process to release Worker2. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific, *Object Name* to 'Worker2'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Role-based Allocation (RBA)

Problem:

I want to model the ability of some process-aware information systems to specify that instances of a given task must be allocated to resources which correspond to a given role.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step

Assumptions:

There are two servers in series and four workers in the system. Role A Workers are able to process entities at Server1 while Role B Workers are able to process entities at Server2.

Technical Approach:

Roles are represented by Lists of Workers with two workers each. Add-on Before and After processing triggers are used at both Servers for seizing/releasing workers from each role list.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 2 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 4 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

<u>Creating lists of Workers representing Roles:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add 2 new Object lists. Name them 'Role A' and 'Role B' respectively. Add Workers 1 and 2 into the Role A list and Workers 3 and 4 into Role B list. These lists represent roles.

<u>Adding Process Logic for Seizing and Releasing the Workers at the Servers:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the Role A list. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'RoleA', *Request Move* from 'None' to

'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'RoleA'.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the Role B list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'RoleB', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'RoleB'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Deferred Allocation (DefA)

Problem:

I want to model the ability of some process-aware information systems to defer specifying the identity of the resource that will execute a task until run-time. This is typically done using a data field whose value is updated during run-time when certain events happen, thus providing different resource allocations.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Object Reference State Variable, Integer State Variable, Monitor

Assumptions:

Worker1 is the initial value of the state variable.

Technical Approach:

An Object Reference State Variable is used to seize a specific Worker at Server1 and a Counter (Integer State Variable) is used to track the number of entities processed by the current Worker. When a Worker processes 10 entities, a Monitor element starts the execution of a process that assigns the reference of the other Worker into the Object Reference State Variable and initializes the counter to 0.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.
- Clicking in Worker 2 and change his color to Yellow by selecting Color in the ribbon.

Creating a Monitor element and State variables:

- In the Definitions window of the model, go to the States panel
    - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'next_resource'.

  - Create 2 new Integer State Variables. Name them ‘Counter’ and ‘Which Resource’. Set the *Initial Value* of Which Resource to ‘1’. The latter state has two possible values: 1 and 2 to indicate that the current Worker is Worker1 and 2 respectively.
- In the Definitions window of the model, go to the Elements panel. Create a new Monitor Element. Set *Monitor Type* to ‘CrossingStateChange’, *State Variable Name* to ‘Counter’, *Initial Threshold* to ‘9’ and *On Change Detected* to ‘Process1’.

Initializing the Object Reference State Variable value:

- Click on the Process window of the model and select OnRunInitialized process by clicking on Select Process button. This is a standard process executed by the Simio engine during the initialization of the model. In this context is used to set Worker1 as the first value of the next_resource state variable.
- Add an Assign Step to the process. Set the *State Variable Name* to ‘next_resource’ and *Expression* to ‘Worker1’.

Creating a Process that changes the Object Reference State Variable value:

- Click on the Process window of the model and create a new process by clicking on the Create Process button.
- Add a Decide Step to the Process. Set *Decision Type* as ‘ConditionBased’ and *Expression* as ‘WhichResource ==1’. This step is used to evaluate which worker should be assigned as the next resource.
  - In the True path of the step add an Assign Step. This step is used to set Worker2 as the next resource. Set the *State Variable Name* to ‘next_resource’ and *Expression* to ‘Worker2’. Also add two additional state assignments. In the first set the *State Variable Name* to ‘Counter’ and *Expression* to ‘0’ and for the second set *State Variable Name* to ‘WhichResource’ and Expression to ‘2’.
  - In the False path of the step add an Assign Step. This step is used to assign Worker1 as the next resource. Set the *State Variable Name* to ‘next_resource’ and *Expression* to ‘Worker1’. Also add two additional state assignments. In the first set the *State Variable Name* to ‘Counter’ and *Expression* to ‘0’ and for the second set *State Variable Name* to ‘WhichResource’ and Expression to ‘1’.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'next_resource', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'next_resource'.
- Add an Assign Step to the process. This step is used increment the Counter variable. Set the *State Variable Name* to 'Counter' and *Expression* to 'Counter+1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Authorization (Auth)

Problem:

I want to model the ability of some process-aware information systems to specify the range of authorized workers able for processing entities in a given server.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Before Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Data Table, Table Reference Assignments, State Assignments, Entity Instance

Assumptions:

There are two entity types created from the same entity instance in the system. Green and Red workers are authorized to process green and red entities while the yellow worker can process both.

Technical Approach:

Lists of Workers are used to represent authorized workers per each entity type. A Data Table is used to store for each entity type its symbol reference, percentage of arrival and a reference to the list of authorized resources for processing in Server1. A table row reference is associated randomly to each created entity at the Source object by percentage of arrival. Add-on process triggers are used to seize and release the authorized workers in Server1.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 3 Worker objects into the facility view. Set 'Output@Source1' Node as the Initial Home for each Worker. Set the Idle Action and Off Shift Action properties of each worker to 'Go To Home'. Set the Initial Network as 'No Network (Free Space)'.
- Clicking in Worker1, 2 and 3 and change their color to Green, Yellow and Red respectively by selecting Color in the ribbon.
- Place a ModelEntity into the facility view and change it's the name to 'ProcessCaseType'. Click it on and add an additional symbol. Change the color of active symbol 1 to Red in order to distinguish the entity types.

Creating lists of Workers representing Authorized Resources:

- In the Definitions tab of the Facility View go to the Lists panel. Add 2 new Object lists. Name them 'AuthorizedGroup_One' and 'AuthorizedGroup_Two' respectively. Add Workers 1 and 2 into the AuthorizedGroup_One list and Workers 3 and 4 into AuthorizedGroup_Two list. These lists represent authorized resources.

Creating a Data Table:

- Go to the Data Window, and within the Tables panel, click the Add Data Table icon the ribbon to add a new table. Name it 'EntityData'.
- Add 3 columns that are a String, Integer and Real standard properties by selecting each data type from the Standard Property drop down in the ribbon. Change the names of each column to 'Name', 'Symbol' and 'Percentage' respectively.
- Add fourth column to the table – an Object List reference by selecting Object List from the Object Reference drop down in the ribbon. The column can be renamed to 'AuthorizedResources'.
- Fill the table with data – such as:
    - TypeA, 0, 40, AuthorizedGroup_One
    - TypeB, 1, 60, AuthorizedGroup_Two

Associating a Data Table Row Reference to the ModelEntity and changing its color:

- Click on the Source object and expand Table Reference Assignments property category. Expand On Created Entity and set Table Name property to 'EntityData' and RowNumber property to 'EntityData.Percentage.RandomRow'. This will associate a table row reference to a created entity based on the percentage of arrival specified in the table.
- Click on the Source Object and expand State Assignments property category. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change State Variable Name to 'ModelEntity.Picture' and New Value to 'EntityData.Symbol'. This will change the color of a new created entity by its corresponding active symbol.

Adding Process Logic for Seizing and Releasing the Workers at the Servers:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change

Object Type to ‘From List’, Object List to ‘EntityData.AuthorizedResources’, Request Move from ‘None’ to ‘To Node’ and Destination to ‘Input@Server1’. Expand Advanced Options and change the value of Selection Goal to ‘Random’.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘From List’, Object List to ‘EntityData.AuthorizedResources’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Separation of Duties (SOD)

Problem:

I want to model the ability of some process-aware information systems to specify that two tasks must be allocated to different resources in a given process case.

Categories:

Workers

Key Concepts:

Worker, Server, Source, Sink, Path, Add-On Before Processing Trigger, Add-on After Processed Trigger, Seize Step, Release Step, Assign Step, Object Reference State Variable, List of Objects

Assumptions:

There are two Workers and three Servers in the system. The second server doesn't need any secondary resources. An entity represents a process case (instance) and must seize different workers at Server 1 and Server 3.

Technical Approach:

Each entity arriving at the first server will seize randomly an available worker from a list of workers and store its reference into an object reference state variable. At the third server the entity uses this reference to seize other worker than the one seized at the first server.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 3 Servers, 3 Basic Nodes and 1 Sink into the Facility View. Put the 3 Servers in line and connect them with paths. Connect the Source to Server 1 and Server 3 to the Sink using also paths. Put one Basic Node below each Server and connect them with bidirectional paths.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the Initial Home for each Worker. Set the Idle Action and Off Shift Action properties of each worker to 'Go To Home'. Lastly, set the Initial Network as 'No Network (Free Space)'.

<u>Defining the Entities:</u>

- Add a ModelEntity object into the Facility View and add 3 additional symbols. Paint the Active Symbol number one with 'Red color' and the second with 'Blue color'. This will be used to change the entities color.
- In the Definitions window of the model, go to the States panel

  - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'SeizedResource'. This will store the reference of the last seized object at Server 1 and then used in Server 3 to seize other worker than this.

Creating a list of Workers:

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'ListResources' and add Worker 1 and Worker 2 into the List. This List will be used to an entity seize one Worker.

Adding Process Logic for Seizing and Releasing the Workers at the First Server:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change Object Type to 'From List', Object List to 'ListResources', Request Move from 'None' to 'To Node' and Destination to 'BasicNode1'. Expand Advanced Options and change the value of Selection Goal to 'Random'.
- Add an Assign Step to the process. This step is used to store the last seized resource reference into the object reference state variable. Set the State Variable Name to ´ModelEntity.SeizedResources' and the expression 'ModelEntity.SeizedResources.LastItem' to New Value. Also add an additional state assignment, set the State Variable Name to 'ModelEntity.Picture' and the New Value to '(ModelEntity.SeizedResources.LastItem.Worker.Priority) + 1'. This will change the entities color to match the color of the seized worker.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change Object Type to 'From List', Object List to 'ListResources'.

Adding Process Logic for Seizing and Releasing the Workers at Server 3:

- Click on Server3 and create a new process in the Before Processing add-on process triggers and add a Seize Step. Now this step is used to seize the other worker. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change Object Type to 'From List', Object List to 'ListResources', Request Move from 'None' to

‘To Node’ and Destination to ‘BasicNode3’. Expand Advanced Options and change the value of Selection Goal to ‘Random’ and Selection Condition to ‘ModelEntity.SeizedResource!= Candidate.Worker’.

- Click on Server3 and create a new process in the After Processing add-on process triggers. Add a Release Step to release the worker. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘From List’, Object List to ‘ListResources’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Case Handling (CH)

Problem:

I want to model the ability of some process-aware information systems to allocate all work items of a given process case to the same resource.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, Basic Node, Add-on Before Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step.

Assumptions:

There are two workers in the system.

Technical Approach:

Add-on Before and After Processing triggers are used for seizing/releasing Workers at the Servers. An entity will randomly seize an available Worker at Server1 and store its reference into an Object Reference State Variable to seize the same Worker in Server2 and 3. The color of each entity is changed at Server1 to match the color of the seized Worker.

<u>Simple System setup:</u>

- Drag 1 Source, 3 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the Initial Home for the Workers. Set the Idle Action and Off Shift Action properties of each worker to 'Go To Home'. Set the Initial Network as 'No Network (Free Space)'.
- Clicking on Worker1 and Worker2 and change their color to Blue and Red respectively by selecting Color in the ribbon.
- Change Priority property of Worker1 and Worker2 to '1', '2' respectively.

<u>Defining the Entities:</u>

- Place a ModelEntity into the facility view and change its name to 'ProcessCase'. Click on it and add 2 additional symbols. Change the color of active symbol 1 and 2 to Blue and Red respectively. These symbols will be used to change the color of each entity to match the color of a seized Worker at Server1.
- In the Definitions window of the model, go to the States panel

    - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state ‘CaseWorker’. This will store the reference of the last seized resource in Server1.

<u>Adding Process Logic for Seizing/Releasing the Workers at Servers:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the “…” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘From List’, Object List to ‘ListResources’, Request Move from ‘None’ to ‘To Node’ and Destination to ‘Input@Server1’’. Expand Advanced Options and change the value of Selection Goal to ‘Random’.
- Add an Assign Step to the process. This step is used to store the reference of the last seized resource. Set the State Variable Name to ‘ModelEntity.CaseWorker’ and New Value to ‘ModelEntity.SeizedResources.LastItem’. Also add an additional state assignment, set the State Variable Name to ‘ModelEntity.Picture’ and the New Value to ‘ModelEntity.SeizedResources.LastItem.Worker.Priority’. Worker’s priority value is used to match with the value of the Active Symbol of each entity in order to change their color according to the seized Workers.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “…” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘From List’, Object List to ‘ListResources’.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize the “CaseWorker”. Click on the “…” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘Specific’, Object Name to ‘ModelEntity.CaseWorker’, Request Move from ‘None’ to ‘To Node’ and Destination to ‘Input@Server2’.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “…” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘Specific’, Object Name to ‘ModelEntity.CaseWorker’.
- Repeat the former 2 steps for Seizing/Releasing the “Case Worker” at Server3.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Retain Familiar (RF)

Problem:

I want to model the ability of some process-aware information systems to favor the allocation of a work item to the resource who executed a previous in a given process case.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, Basic Node, Add-on Before Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Assign Step, Resource State

Assumptions:

Entities at Server2 and Server3 will always try to seize first the Worker seized in the former Server (1 or 2) otherwise they will try to seize other available Worker.

Technical Approach:

Add-on Before Processing and After Processing triggers are used in Servers for seizing/releasing Workers. At Server1, entities randomly seize an available Worker. At the other Servers, entities will first check if the last seized resource is available. If not, they will attempt to seize other available Worker. The color of each entity is changed at a Server to match the one of the seized Worker in order to see which Worker the entity should attempt to seize first in the next Server.

<u>Simple System setup:</u>

- Drag 1 Source, 3 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 3 Worker objects into the facility view. Set 'Output@Source1' Node as the Initial Home for each Worker. Set the Idle Action and Off Shift Action properties of each Worker to 'Go To Home'. Set the Initial Network as 'No Network (Free Space)'.
- Clicking on Worker1, 2 and 3 and change their color to Blue, Red and Yellow respectively by selecting Color in the ribbon.
- Change Priority property of Worker1, 2 and 3 to '1', '2','3' respectively.

<u>Defining the Entities:</u>

- Place a ModelEntity into the facility view and change its name to 'ProcessCase'. Click on it and add 3 additional symbols. Change the color of active symbol 1, 2 and 3 to Blue, Red and Yellow respectively. These symbols will be used to change the color of each entity to match the color of a seized Worker.

- In the Definitions window of the model, go to the States panel
    - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'LastSeizedWorker'. This will store the reference of the last seized resource in a Server.

Creating a list of Workers:

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'ListResources' and add Worker 1, 2 and 3 into the List. This List will be used to an entity seize one Worker from the list.

Adding Process Logic for Seizing/Releasing the Workers at Servers:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change Object Type to 'From List', Object List to 'ListResources', Request Move from 'None' to 'To Node' and Destination to 'Input@Server1''. Expand Advanced Options and change the value of Selection Goal to 'Random'.
- Add an Assign Step to the process. This step is used to store the reference of the last seized resource. Set the State Variable Name to 'ModelEntity.LastSeizedWorker' and New Value to 'ModelEntity.SeizedResources.LastItem'. Also add an additional state assignment, set the State Variable Name to 'ModelEntity.Picture' and the New Value to 'ModelEntity.SeizedResources.LastItem.Worker.Priority'. Worker's priority value is used to match with the value of the Active Symbol of each entity in order to change their color according to the seized Workers.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change Object Type to 'From List', Object List to 'ListResources'.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change Object Type to 'From List', Object List to 'ListResources', Request Move from 'None' to 'To Node' and Destination to 'Input@Server2'. Expand Advanced Options and change the

value of Selection Goal to ‘Random’. Set Selection Condition as ‘Math.If(ModelEntity.LastSeizedWorker.Worker.ResourceState == 0, Candidate.Worker == ModelEntity.LastSeizedWorker, Candidate.Worker.ResourceState == 0)’. This function is used to entity seize the Worker seized in the former Server if it is available, otherwise it will seize other available Worker.

- Add an Assign Step to the process. Repeat the same instructions given for Server 1.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘From List’, Object List to ‘ListResources’.
- Click on Server3 and create a new process in the Before Processing add-on process triggers. Add a Seize Step and follow the same instructions given for Server2, except that the Destination Node is to ‘Input@Server3’.
- Click on Server3 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘From List’, Object List to ‘ListResources’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Capability-based Allocation (CBA)

Problem:

I want to model the ability of some process-aware information systems to allocate instances of a task to resources based on their specific capabilities.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Before Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Data Table, Table Reference Assignments, State Assignments, Entity Instance

Assumptions:

There 3 workers in the system and a red and yellow colored entity types. Red workers are capable to process red entities while the Yellow worker is able to process yellow entities.

Technical Approach:

Lists of Workers are used to represent Workers with specific capabilities. A Data Table is used to store for each entity type its type number, percentage of arrival and a reference to the list of capable Workers for processing in Server1. Table Reference and State Assignments are used in the Source object to associate randomly to each created entity a table row reference by percentage of arrival and to change the color of an entity to match its type. Add-on process triggers are used to seize and release the capable workers in Server1.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 3 Worker objects into the facility view. Set 'Output@Source1' Node as the Initial Home for each Worker. Set the Idle Action and Off Shift Action properties of each worker to 'Go To Home'. Set the Initial Network as 'No Network (Free Space)'.
- Clicking in Worker1, 2 and 3 and change their color to Red, Red and Yellow respectively by selecting Color in the ribbon.
- Place a ModelEntity into the facility view and change it's the name to 'ProcessCaseType'. Click it on and add an additional symbol. Change the color of active symbol 1 and 2 to Red and Yellow respectively in order to distinguish the entity types.

Creating lists of Workers representing Workers with specific capabilities:

- In the Definitions tab of the Facility View go to the Lists panel. Add 2 new Object lists. Name them 'EntA_CapableResources' and 'EntB_CapableResources' respectively. Add Workers 1 and 2 into the EntA_CapableResources list and Workers 3 into EntB_CapableResourceslist.

Creating a Data Table:

- Go to the Data Window, and within the Tables panel, click the Add Data Table icon the ribbon to add a new table. Name it 'EntityData'.
- Add 3 columns that are a String, Integer and Real standard properties by selecting each data type from the Standard Property drop down in the ribbon. Change the names of each column to 'Name', 'Type' and 'Percentage' respectively.
- Add fourth column to the table – an Object List reference by selecting Object List from the Object Reference drop down in the ribbon. The column can be renamed to 'Capability Resources'.
- Fill the table with data – such as:
    - TypeA, 1, 50, EntA_CapableResources
    - TypeB, 2, 50, EntB_CapableResources'

Associating a Data Table Row Reference to the ModelEntity and changing its color:

- Click on the Source object and expand Table Reference Assignments property category. Expand On Created Entity and set Table Name property to 'EntityData' and RowNumber property to 'EntityData.Percentage.RandomRow'. This will associate a table row reference to a created entity based on the percentage of arrival specified in the table.
- Click on the Source Object and expand State Assignments property category. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change State Variable Name to 'ModelEntity.Picture' and New Value to 'EntityData.Type'. This will change the color of a new created entity by its corresponding active symbol.

Adding Process Logic for Seizing and Releasing the Workers at the Servers:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change

Object Type to ‘From List’, Object List to ‘EntityData.CapabilityResources’, Request Move from ‘None’ to ‘To Node’ and Destination to ‘Input@Server1’. Expand Advanced Options and change the value of Selection Goal to ‘Random’.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change Object Type to ‘From List’, Object List to ‘EntityData.CapabilityResources’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

History-based Allocation (HBA)

Problem:

I want to model the ability of some process-aware information systems to allocate work items to resources based on their previous execution history.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Find Step, Integer State Array, Integer State

Assumptions:

Number of successful completions is the selection criteria to seize a Worker at a Server. Neither are failures or off-shifts.

Technical Approach:

A Matrix-dimensioned state array initialized from a data table is used to store the number of successful completions per Server. When an entity arrives to a Server it will search the matrix to find the max number of successful completions in that Server using an Index. That Index is then used to seize a worker whose index value matches a Worker's priority. After processing the number of completions is updated for a Worker at a Server.

Details for building the model:

Simple System setup:

- Drag 1 Source, 2 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.
- Clicking in Worker1 and 2 Blue and Red respectively by selecting Color in the ribbon.
- Change Worker2 *Initial Priority* property to '2'.

Creating a list of Workers:

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'ResourcesList' and add Worker 1 and 2 into the list. This List will be used to an entity seize one Worker from the list.

Creating a Data Table containing the Number of Successful Completions per Server:

- Go to the Data Window, and within the Tables panel, click the Add Data Table icon the ribbon to add a new table. Name it 'Completions'.
- Add 2 columns that are 2 Integer standard properties by selecting Integer data type from the Standard Property drop down in the ribbon. Change the names of each column to 'Task', 'Task B'.
- Fill the table with data – such as:
  - 60, 20
  - 50, 40

Creating a Matrix-Dimensioned State Array and an Index variable:

- In the Definitions window of the model, go to the States panel
  - Create a new Integer State Variable by clicking on Integer in the ribbon. Name this new state 'TaskCompletions'. Change *Dimension Type* property to 'Matrix From Table' and *Table Name* property to 'Completions'. This initializes the matrix with the values store in Completions data table.
  - Create a new Integer State Variable by clicking on Integer in the ribbon. Name this new state 'Index'.

Adding Process Logic for Seizing and Releasing the appropriate Workers at the Servers:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Find Step in this process to find the max number of successful completions. Change *Index Variable Name* property to 'Index', *Starting Index* to '1', *Ending Index* to '2', *Search Type* to 'Maximize' and *Search Expression* to 'TaskCompletions[Index, 1]'.
- Add a Seize Step in this process to seize the Worker with the most number of successful completions from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourceList', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Condition* to 'Candidate.Worker.Priority == Index'.
- Add an Assign Step to the process. Change *State Variable Name* to 'TaskCompletions', *Row* to 'Index', *Column* to '1' and *New Value* to 'TaskCompletions[Index,1] + 1'.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourceList'.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Find Step in this process to find the max number of successful completions. Change *Index Variable Name* property to 'Index', *Starting Index* to '1', *Ending Index* to '2', *Search Type* to 'Maximize' and *Search Expression* to 'TaskCompletions[Index, 2]'.
- Add a Seize Step in this process to seize the Worker with the most number of successful completions from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourceList', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server2'. Expand Advanced Options and change the value of *Selection Condition* to 'Candidate.Worker.Priority == Index'.
- Add an Assign Step to the process. Change *State Variable Name* to 'TaskCompletions', *Row* to 'Index', *Column* to '2' and *New Value* to 'TaskCompletions[Index,2] + 1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourceList'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Organizational Allocation (OA)

Problem:

I want to model the ability of some Process-Aware Information Systems to allocate work items to resources based on their position within the organization and their relationship to other resources. For instance, instances of some task B must be allocated to a Manager of the resource who conducted a previous instance of task A.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Assign Step, State Array,

Assumptions:

Worker1 and Worker 2 priority values are used to get the corresponding Managers from the vector and to change the color of the entities to match the color of their last seized resource.

Technical Approach:

A vector-dimensioned state array is used to store the Managers of Worker1 and Worker2. At Server1, entities will seize either Worker1 or Worker2 randomly and assign into an Index variable the last seized resource's priority value. The Index is then used at Server2 to seize the corresponding Manager of the Worker seized at Server1.

Details for building the model:

Simple System setup:

- Drag 1 Source, 2 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker1 and Worker2 colors to Yellow and Red respectively by clicking in Color in the ribbon.
- Add other 2 Worker objects into the facility view. Set 'Input@Sink1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change their names to 'ManagerOfWorker1' and 'ManagerofWorker2'.

- Paint the heads of ManagerOfWorker1 and ManagerOfWorker2 objects to Yellow and Red respectively by clicking in Color in the ribbon.

<u>Creating lists of Workers for seizing/releasing at Server1:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'ResourcesList' and add Worker 1 and 2 into the list. This List will be used to an entity seize one Worker from the list.

<u>Creating a Vector and an Index variable:</u>

- In the Definitions window of the model, go to the States panel
    - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'Managers'. Change *Dimension Type* property to 'Vector and *Rows* property to '2'.
    - Create a new Integer State Variable by clicking on Integer in the ribbon. Name this new state 'Index'.

<u>Initializing the Vector:</u>

- Click on the Process window of the model and select OnRunInitialized process by clicking on Select Process button. This is a standard process executed by the Simio engine during the initialization of the model.
- Add an Assign Step to the process. Set the *State Variable Name* to 'Managers', *Row* property to '1' and *New Value* to 'ManagerOfWorker1'. Add an Additional State Assignment. Set the *State Variable Name* to 'Managers', *Row* property to '2' and New *Value* to 'ManagerOfWorker2'.

<u>Adding Process Logic for Seizing and Releasing the Workers at the Servers:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to ResourcesList', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.
- Add an Assign Step to the process. This step is used to assign to the Index variable the last seized resource's priority. Set the *State Variable Name* to ΄Index' and the expression

‘ModelEntity.SeizedResources.LastItem.Worker.Priority’ to *New Value*. Also add an additional state assignment, set the State Variable Name to ‘ModelEntity.Picture’ and the New Value to ‘ModelEntity.SeizedResources.LastItem.Worker.Priority’. This will change the entities color to match the color of the seized worker.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘From List’, *Object List* to ‘ResourcesList’.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize the Manager of the last seized worker. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘Managers[Index]’, *Request Move* from ‘None’ to ‘To Node’ and *Destination* to ‘Input@Server2’.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘Managers[Index]’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Automatic Execution (AE)

Problem:

I want to model the ability of some Process-Aware Information Systems to execute instances of a task without needing to utilize the services of a resource.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step

Assumptions:

There are two servers in series and one Worker in the system.

Technical Approach:

Add-on process triggers are used for seizing/releasing Worker1 at Server1. Entities at Server2 only seize the server's capacity for processing without seizing any secondary resource.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 2 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of each Worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

<u>Adding Process Logic for Seizing and Releasing the Workers at the Servers:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step to seize Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process to release Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific, *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Distribution by Offer – To Single Resource (DBOS)

Problem:

I want to model the ability of some Process-Aware Information Systems to offer a work item to a selected individual resource without committing it for execution.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Integer State Variable, Assign Step, On Evaluating Seize Request

Assumptions:

There is only one Worker in the system.

Technical Approach:

The Worker's Add-on Evaluating Seize Request process trigger is used to reject any seize attempts from entities of type B until Worker1 processes 5 entities of type A. Add-on before and after processing triggers are used for seizing/releasing Worker1 at Server1.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of each Worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

Defining the Entities types:

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.

Creating a Counter Variable:

- In the Definitions window of the model, go to the States panel

- Create a new Integer State Variable by clicking on Integer in the ribbon. Name this new state ‘Counter’.

Adding Process Logic for Evaluate Seize Requests:

- Click on Worker1 and create a new process in the Evaluating Seize Request add-on process trigger. This process will be executed each time an entity is attempting allocation of Worker1.
    - Add a Decide Step into the process to check if the entity is of Type A. Set *Decide Type* property to ‘ConditionBased’ and the *Expression* to ‘ModelEntity.Is.EntTypeA’.
    - In the False path of the step add other Decide to check if Worker1 already processed 5 entities of type A. Set *Decide Type* property to ‘ConditionBased’ and the *Expression* to ‘Counter >= 5’. In the true path of this step add an Assign Step to accept EntTypeB seize requests. Set the *State Variable Name* to ‘Token.ReturnValue’ and *Expression* to ‘True’. In the False step add an Assign Step to reject EntTypeB seize requests. Set the *State Variable Name* to ‘Token.ReturnValue’ and *Expression* to ‘False’.

Adding Process Logic for Seizing and Releasing the Workers at the Servers:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘Worker1’, *Request Move* from ‘None’ to ‘To Node’ and *Destination* to ‘Input@Server1’.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Assign Step to increment the Counter. Set the *State Variable Name* to ‘Counter’ and *Expression* to ‘Counter+1’.
- Add a Release Step to the process to release Worker1. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific, *Object Name* to ‘Worker1’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Distribution by Offer – To Multiple Resources (DBOM)

Problem:

I want to model the ability of some process-aware information systems to offer work-items to multiple resources without committing them for execution.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Assign Step, On Evaluating Seize Request

Assumptions:

There are two Workers in the system.

Technical Approach:

Entities at Server1 will randomly seize a Worker from a list. Using Worker2 Add-o On Evaluating Seize Request trigger, until 1h of simulation time has passed it will reject any seize attempts from entities in order to demonstrate that the Worker doesn't become committed to process the entity until some conditions are met.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker1 and Worker2 colors to Yellow and Red respectively by clicking in Color in the ribbon.

<u>Creating a list of Workers for seizing/releasing at Server1:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'ListResources' and add Worker 1 and 2 into the list. This List will be used to an entity seize one Worker from the list.

<u>Adding Process Logic for Evaluate Seize Requests:</u>

- Click on Worker2 and create a new process in the Evaluating Seize Request add-on process trigger. This process will be executed each time an entity is attempting allocation of Worker2.
    - Add a Decide Step into the process to check if the entity if one hour of simulation time has passed. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'Run.TimeNow >=1'.
        - In the true path of this step add an Assign Step to accept seize requests. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'True'.
        - In the False step add an Assign Step to reject seize requests. Set the *State Variable Name* to 'Token.ReturnValue' and *Expression* to 'False'.

<u>Adding Process Logic for Seizing and Releasing the Workers at the Servers:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'FromList', *Object List* to 'ListResources', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process to release Worker1. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'FromList'', *Object List* to 'ListResources'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Random Allocation (RMA)

Problem:

I want to model the ability of some Process-Aware Information Systems to offer or allocate work items to resources on a random basis.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step.

Assumptions:

There are 3 workers in the system.

Technical Approach:

Each entity at Server1 will randomly seize at Server1 one Worker from a list. This is done by defining the Selection Goal of a Seize step as Random.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Add 3 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 and Worker3 colors to Red and Yellow respectively by clicking in Color in the ribbon.

<u>Creating a list of Workers:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'ResourcesList' and add Worker 1,2 and 3 into the list. This List will be used to an entity seize one Worker from the list.

<u>Adding Process Logic for Seizing and Releasing the Workers at the Servers:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to ResourcesList', *Request Move* from 'None' to

'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Goal* to 'Random'.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'ResourcesList'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Round-robin Allocation (RRA)

Problem:

Represents the ability of some Process-Aware Information Systems to allocate a work item to available resources on a cyclic basis.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, TransferNode.

Assumptions:

There are 3 Workers in the system. TransferNode1, 2 and 3 are associated with Worker1, 2 and 3 respectively.

Technical Approach:

The routing logic of Output@Source1 is changed to an entity select one worker-associated transfer node from a list of nodes. When an entity enters its worker-associated transfer node assigns the Worker reference into his object reference state variable in order to seize it at Server1.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Add 3 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 and Worker3 colors to Red and Yellow respectively by clicking in Color in the ribbon.
- Place 3 Transfer Nodes into the facility view between Output@Source1 and Input@Server1 in parallel and connect them with paths. The Transfer Nodes cannot be connected with each other.

<u>Defining the Entities:</u>

- In the Definitions window of the model, go to the States panel
    - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'WorkerReference'.

Creating a List of Worker-associated nodes:

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Node list. Name it 'NodeList' and add TransferNode 1, 2 and 3 into the list.

Defining Routing Logic at Output@Source1 and Assigning a Worker reference to entities at TransferNodes:

- Click in Output@Source1 Transfer Node and expand Routing Logic property category. Change *Entity Destination* property to 'Select From List', *Node List* Name to 'NodeList' and *Selection Goal* to 'Cyclic'.
- Click in TransferNode1 and create a new Entered add-on process trigger. Add an Assign step in this process. Change *State Variable Name* to 'ModelEntity.WorkerReference' and *New Value* to 'Worker1'.
- Click in TransferNode2 and create a new Entered add-on process trigger. Add an Assign step in this process. Change *State Variable Name* to 'ModelEntity.WorkerReference' and *New Value* to 'Worker2'.
- Click in TransferNode3 and create a new Entered add-on process trigger. Add an Assign step in this process. Change *State Variable Name* to 'ModelEntity.WorkerReference' and *New Value* to 'Worker3'.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'ModelEntity.WorkerReference', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'ModelEntity.WorkerReference'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Shortest Queue Allocation (SQA)

Problem:

I want to model the ability of some Process-Aware Information Systems to allocate a work item to the resource that has the least number of work items allocated to it.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step.

Assumptions:

There are two Workers in the system and Worker2 is the only worker that is able to process EntTypeB entities. Worker2 has a bigger workload to demonstrate that at certain points, entities at Server1 will seize Worker1 which has the least number of entities waiting.

Technical Approach:

Entities at Server1 will always seize from a list the Worker that has least number of entities waiting in its allocation queue. This is done by selecting from the candidates the smallest value of the NumberWaiting function of each Worker's allocation queue.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Drag another Source object and another Server object. Put the objects in series and connect them with paths. Connect Server2 to Sink1. Set *Entity Type* property of Source2 to 'EntTypeB'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 color to Red by clicking in Color in the ribbon.

Defining the Entities types:

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.

Creating a list of Workers:

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'Resources' and add Worker 1 and 2 into the list. This List will be used to an entity seize one Worker from the list.

Adding Process Logic for Seizing and Releasing the Workers at the Servers:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize one Worker from the list. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'Resources', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'. Expand Advanced Options and change the value of *Selection Goal* to 'SmallestValue' and *Selection Expression* to 'Candidate.Worker.AllocationQueue.NumberWaiting'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'From List', *Object List* to 'Resources'.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize Worker2. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker2', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server2'.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker2'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Early Distribution (EA)

Problem:

I want to model the ability of some process-aware information systems to allocate work items to resources before their time for execution.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Add-on Entered Process Trigger, Seize step, Release Step, Station Element, Timer Element, Event

Assumptions:

There are two entity types in the system and EntTypeB is used to demonstrate early distribution (allocation). EntTypeB entities will have assigned a Worker using a Random Discrete distribution at Output@Source2. Only Worker1 can process EntTypeA entities.

Technical Approach:

EntTypeB entities are enabled for execution in intervals of 1 hour. Before their time for processing, EntTypeB entities will be transferred from Input@Source1 node to WaitingEntitiesStation. At each hour a Timer element triggers a process that fires an event that will transfer all entities waiting at the Station to the Input Buffer of Server1 in order to seize their Worker for processing.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Drag another Source object and another Server object. Connect the Output@Source2 to Input@Sever1 using a path. Set *Entity Type* property of Source2 to 'EntTypeB'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 color to Red by clicking in Color in the ribbon.

Defining the Entities types:

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.
- In the Definitions window of model entity, go to the States panel
    - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'RandomWorker'.

<u>Creating a Timer, a Station and an Event:</u>

- In the Definitions window of the model, go to the Elements panel. Create a new Timer Element. By default the timer is set to trigger an event at intervals of 1 hour.
- Create a Station element. Change its name to 'WaitingEntitiesStation'.
- Go to the Events panel. Create a new Event and change its name to 'EnabledForExecution'.

<u>Adding Process Logic at Output@Source2 to assign a random Worker to an EntTypeB entity:</u>

- Click on Output@Source2 and create a new process in the Entered add-on process triggers.
    - Add a Decide Step. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'Random.Discrete(1,.67,2,1) == 1'.
    - In the true path of this step add an Assign Step to assign Worker1. Set the *State Variable Name* to 'ModelEntity.RandomWorker' and *Expression* to 'Worker1'. In the False path add an Assign Step assign Worker2. Set the *State Variable Name* to 'ModelEntity.RandomWorker' and *Expression* to 'Worker2'.

<u>Creating 2 processes:</u>

- Click on the Process window of the model and create 2 new processes by clicking on the Create Process button.
    - In Process1 add an EndTransfer step. Add after a Wait step and change *Event Name* property to 'EnabledForExecution'. Add a Transfer step and set *From* property to 'CurrentStation', *To* to 'Station', *Station Name* to 'Server1.InputBuffer'.
    - Click in Process2 and change Triggering *Event Name* property to 'Timer1.Event'. Add a Fire step into the process and set *Event Name* to 'EnabledForExecution'.

Adding Process Logic at Input@Server1 to transfer EntTypeB entities into the Station element:

- Click on Source2 and create a new process in the Entered add-on process triggers.
  - Add a Decide Step to check if the entity type is EntTypeB. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Is.EntTypeB'.
- In the true path of this step add an Transfer step. Set *From* property to 'CurrentNode', *To* to 'Station', *Station Name* to 'WaitingEntitiesStation' and *Token Wait Action* to 'WaitUntilTransferring' in the advanced options. Add an Execute Step and set *Process Name* to 'Process1'.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Decide Step to check if the entity type is EntTypeB. Set Decide Type property to 'ConditionBased' and the Expression to 'ModelEntity.Is.EntTypeB'.
  - Add a Seize Step in the true path process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'ModelEntity.RandomWorker', *Request Move* from 'None' to 'To Node' and *Destination* to 'Output@Server1'.
  - Add a Seize Step in the false path process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Output@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Decide Step to check if the entity type is EntTypeB. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Is.EntTypeB'.
  - Add a Release Step in the true to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'ModelEntity.RandomWorker'.
  - Add a Release Step in the true to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal - Contact: andrefcsousa@gmail.com

Distribution on Enablement (DE)

Problem:

I want to model the ability of some process-aware information systems to allocate work items to resources at the moment they are enabled for execution.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Add-on Entered Process Trigger, Seize step, Release Step, Station Element, Timer Element, Event.

Assumptions:

There are two entity types in the system and EntTypeB is used to demonstrate distribution on enablement. Only Worker1 can process EntTypeA entities.

Technical Approach:

EntTypeB entities are enabled for execution in intervals of 1 hour. Before their time for processing, EntTypeB entities will be transferred from Input@Source1 node to WaitingEntitiesStation. At each hour a Timer element triggers a process that fires an event that will transfer all entities waiting at the Station to the Input Buffer of Server1 in order to seize a Worker from a list for processing.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Drag another Source object and another Server object. Connect the Output@Source2 to Input@Sever1 using a path. Set *Entity Type* property of Source2 to 'EntTypeB'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 color to Red by clicking in Color in the ribbon.

<u>Defining the Entities types:</u>

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.

Creating a list of Workers:

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it 'Resources' and add Worker 1 and 2 into the list. This List will be used to an entity seize one Worker from the list.

Creating a Timer, a Station and an Event:

- In the Definitions window of the model, go to the Elements panel. Create a new Timer Element. By default the timer is set to trigger an event at intervals of 1 hour.
- Create a Station element. Change its name to 'WaitingEntitiesStation'.
- Go to the Events panel. Create a new Event and change its name to 'EnabledForExecution'.

Creating 2 processes:

- Click on the Process window of the model and create 2 new processes by clicking on the Create Process button.
    - In Process1 add an EndTransfer step. Add after a Wait step and change *Event Name* property to 'EnabledForExecution'. Add a Transfer step and set *From* property to 'CurrentStation', *To* to 'Station', *Station Name* to 'Server1.InputBuffer'.
    - Click in Process2 and change Triggering *Event Name* property to 'Timer1.Event'. Add a Fire step into the process and set *Event Name* to 'EnabledForExecution'.

Adding Process Logic at Input@Server1 to transfer EntTypeB entities into the Station element:

- Click on Source2 and create a new process in the Entered add-on process triggers.
    - Add a Decide Step to check if the entity type is EntTypeB. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Is.EntTypeB'.
- In the true path of this step add a Transfer step. Set *From* property to 'CurrentNode', *To* to 'Station', *Station Name* to 'WaitingEntitiesStation' and *Token Wait Action* to

‘WaitUntilTransferring’ in the advanced options. Add an Execute Step and set *Process Name* to ‘Process1’.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Decide Step to check if the entity type is EntTypeB. Set Decide Type property to ‘ConditionBased’ and the Expression to ‘ModelEntity.Is.EntTypeB’.
    - Add a Seize Step in the true path process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘FromList’, *Object List* to ‘Resources’, *Request Move* from ‘None’ to ‘To Node’ and *Destination* to ‘Output@Server1’. Expand Advanced Options and change the value of *Selection Goal* to ‘Random’.
    - Add a Seize Step in the false path process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘Worker1’, *Request Move* from ‘None’ to ‘To Node’ and *Destination* to ‘Output@Server1’.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Decide Step to check if the entity type is EntTypeB. Set *Decide Type* property to ‘ConditionBased’ and the *Expression* to ‘ModelEntity.Is.EntTypeB’.
    - Add a Release Step in the true to the process. Click on the ‘...’ button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘FromList’, *Object List* to ‘Resources’.
    - Add a Release Step in the true to the process. Click on the ‘...’ button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘Worker1’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Late Distribution (LD)

Problem:

I want to model the ability of some process-aware information systems to allocate work items to resources after they have been enabled for execution.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Add-on Entered Process Trigger, Seize step, Release Step, Station Element, Timer Element, Event.

Assumptions:

There are two entity types in the system and EntTypeB is used to demonstrate late distribution. There is only one Worker in the system.

Technical Approach:

EntTypeB entities are enabled for execution in intervals of 1 hour. Before their time for processing, EntTypeB entities will be transferred from Input@Source1 node to WaitingEntitiesStation. At each hour a Timer element triggers a process that fires an event that will transfer all entities waiting at the Station to the Input Buffer of Server1, but they will wait at the Station until the Number of entities waiting in Worker1 allocation queue is lesser or equal than 3. Then they are transferred into InputBuffer of Server1 for seizing and processing.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Drag another Source object and another Server object. Connect the Output@Source2 to Input@Sever1 using a path. Set *Entity Type* property of Source2 to 'EntTypeB'.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

<u>Defining the Entities types:</u>

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.

<u>Creating a Timer, a Station and an Event:</u>

- In the Definitions window of the model, go to the Elements panel. Create a new Timer Element. By default the timer is set to trigger an event at intervals of 1 hour.
- Create a Station element. Change its name to 'WaitingEntitiesStation'.
- Go to the Events panel. Create a new Event and change its name to 'EnabledForExecution'.

<u>Creating 2 processes:</u>

- Click on the Process window of the model and create 2 new processes by clicking on the Create Process button.
    - In Process1 add an EndTransfer step. Add after a Wait step and change *Event Name* property to 'EnabledForExecution'. Add a Decide step into the process to check if the number of entities waiting in Worker1 allocation queue is lesser or equal to 3.
        - In the true path, add a Transfer step and set *From* property to 'CurrentStation', *To* to 'Station', *Station Name* to 'Server1.InputBuffer'.
        - If the condition is false link the false path to the previous Wait step.
    - Click in Process2 and change Triggering *Event Name* property to 'Timer1.Event'. Add a Fire step into the process and set *Event Name* to 'EnabledForExecution'.

<u>Adding Process Logic at Input@Server1 to transfer EntTypeB entities into the Station element:</u>

- Click on Source2 and create a new process in the Entered add-on process triggers.
    - Add a Decide Step to check if the entity type is EntTypeB. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Is.EntTypeB'.
- In the true path of this step add a Transfer step. Set *From* property to 'CurrentNode', *To* to 'Station', *Station Name* to 'WaitingEntitiesStation' and *Token Wait Action* to 'WaitUntilTransferring' in the advanced options. Add an Execute Step and set *Process Name* to 'Process1'.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step to the process in order to seize Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Output@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step in the true to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Escalation (E)

Problem:

I want to model the ability of some Process-Aware Information Systems to offer or allocate work items to other resources than those it has previously been offered or allocated in order to attempt to expedite the completion of the work item.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Add-on On Created Entity trigger, Add-on Entered Process Trigger, Seize step, Release Step, Delay Step, Remove Step, Assign Step, Object Reference State Variable, Process

Assumptions:

There are two workers in the system. Worker2 will only process entities removeD from Worker1 allocation queue.

Technical Approach:

In this Simbit, if an entity waiting in worker1 allocation queue waits 3 or more minutes, it is removed from the queue in order to Worker2 process it to expedite the entity.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Workers. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 color to Red by clicking in Color in the ribbon.

Defining the Entities types:

- Place 1 ModelEntity objects into the facility view. Add an Additional Symbol and color it Red.
- In the Definitions window of model entity, go to the States panel
  - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'WorkerToSeize'.

Setting Worker1 as the first Worker to seize by an entity:

- Click on Server1 and create a new process in the Created Entity add-on process triggers. Add an Assign Step and set *State Variable Name* to 'ModelEntity.WorkertoSeize' and *New Value* to 'Worker1'.

Creating 1 process:

- Click on the Process window of the model and create a new process by clicking on the Create Process button.
- In Process1 add a Delay Step. Set *Delay Time* to '3' and *Units* to 'Minutes'.
- Add a Remove Step. Set *Queue State Name* to 'Worker1[1].AllocationQueue'.
    - In the removed path add an assign step. Set *State Variable Name* to 'ModelEntity.WorkertoSeize' and *New Value* to 'Worker2'. Add an Additional Assignment and set *State Variable Name* to 'ModelEntity.Picture' and *New Value* to '1'.
    - Add a Seize Step to process1 in order to seize Worker2. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker2', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add an Execute Step and set *Process Name* to 'Process1' and *Token Wait Action* to '(None) Continue'.
- Add a Seize Step to the process in order to seize Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'ModelEntity.WorkertoSeize', *Request Move* from 'None' to 'To Node' and *Destination* to 'Output@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step in the true to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'ModelEntity.WorkertoSeize'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Chained Execution (CE)

Problem:

I want to model the ability of some process-aware information systems to immediately start the next work item in a case once the previous one has completed.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Add-on Entered Process Trigger, Seize step, Release Step, Move Step

Assumptions:

There is one Worker in the system.

Technical Approach:

In this Simbit, Worker1 is only seized at Server1 and only released at Server3. Using the move step the model shows the Worker moving between the Servers.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 3 Servers and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

<u>Adding Process Logic for Seizing/Releasing and Moving the Worker at Servers:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step to the process in order to seize Worker1. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add a Move to the process in order to force Worker1 to move to Input@Server2. Click on the "…" button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1' and *Destination* Node to 'Input@Server2'.

- Click on Server3 and create a new process in the Before Processing add-on process triggers. Add a Move to the process in order to force Worker1 to move to Input@Server3. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1' and *Destination* Node to 'Input@Server3'.
- Click on Server3 and create a new process in the After Processing add-on process triggers. Add a Release Step in the true to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Additional Resources (AR)

Problem:

I want to model the ability of some process-aware information systems to allocate a work item to an additional resource.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Function

Assumptions:

There are two entity types in the system and two Workers. A productivity function is used to calculate the processing times of each entity type.

Technical Approach:

Entities will seize randomly a Worker from a list. EntTypeB entities require 2 resources for processing. This is specified in Number of Objects property of the Seize step in order to an EntTypeB entity seizes 2 workers.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Drag another Source object and another Server object. Connect the Output@Source2 to Input@Sever1 using a path. Set *Entity Type* property of Source2 to 'EntTypeB'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 color to Red by clicking in Color in the ribbon.

<u>Defining the Entities types:</u>

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.

<u>Creating a list of Workers:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it ‘Resources’ and add Worker 1 and 2 into the list. This List will be used to an entity seize one Worker from the list.

<u>Creating a function and use it to set processing times:</u>

- In the Definitions tab of the Facility View go to the Functions panel. Add a new Function. Name it ‘Productivity’ and change *Expression* property to ‘Math.If(Entity.SeizedResources.NumberItems > 1, 1.5, 1)’.
- Click in Server1 and change *Processing Time* property to ‘Random.Triangular(3,5,7)/Productivity’.

<u>Adding Process Logic for Seizing and Releasing the Workers at Server1:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in the true path process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘FromList’, *Object List* to ‘Resources’, *Request Move* from ‘None’ to ‘To Node’ and *Destination* to ‘Output@Server1’. Expand Advanced Options, change the value of *Number of Objects* property to ‘Math.If(ModelEntity.Is.EntTypeB,2,1)’ and change the value of *Selection Goal* to ‘Random’.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step in the true to the process. Click on the ‘...’ button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘FromList’, *Object List* to ‘Resources’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

D – Resource Execution Simbit Guidelines

Resource-determined Work Queue Content (RDWQC)

Problem:

I want to model the ability for a resource to determine (in some PAIS) the format and content of its work queue for execution. For instance, order its work items by priority.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Add-on Entered Process Trigger, Seize step, Release Step, Largest Value First ranking rule

Assumptions:

There are two entity types in the system with different priorities and one Worker in the system that processes the entities with the highest priority first.

Technical Approach:

In this Simbit, Worker1 processes first EntTypeB entities. This is done by set Largest Value First as the raking rule of Worker1 allocation queue and using the priority property as the ranking expression.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Servers' *Initial Capacity* property to 'Infinity'.
- Drag another Source object and another Server object. Connect the Output@Source2 to Input@Sever1 using a path. Set *Entity Type* property of Source2 to 'EntTypeB'.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

Defining the Entities types:

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.

<u>Adding Process Logic for Seizing and Releasing the Worker at Server1:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step to the process in order to seize Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step in the true to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Delegation (DL)

Problem:

I want to model the ability of a resource in some process-aware information systems to delegate a work item allocated to it to another resource.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Assign Step, Transfer step, Data Table, Table Reference Assignments

Assumptions:

There are two Workers and two entity types in the system. Worker2 only processes delegated entities to it at Server1.

Technical Approach:

In this simbit, Worker1 has a probability to delegate an entity at Server1. This information is stored in a Data Table and is associated to entities by Table Assignments. If an entity is delegated it will carry a reference to Worker2 and will be transferred again to Output@Source1. The entity color changes to match the color of the resource that will process the delegate entity. Green color means that the entity wasn´t delegated.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Server's Initial Capacity to 'Infinity'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 color to Red by clicking in Color in the ribbon.

<u>Defining the Entities types:</u>

- Place 1 ModelEntity object into the facility view.
- Add an additional symbol and change the second symbol's color to Red.
- In the Definitions window of model entity, go to the States panel

  - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state ‘Delegate’.
  - Create a new Boolean State Variable by clicking on Boolean in the ribbon. Name this new state ‘Delegated’.

<u>Creating a Data Table:</u>

- Go to the Data Window, and within the Tables panel, click the Add Data Table icon the ribbon to add a new table. Name it ‘EntData’.
- Add 2 columns that are a Object reference and a Real standard property. Change the names of each column to ‘Worker’, ‘DelegationProbability’.
- Fill the table with data – such as:
  - Worker1, 0,5

<u>Associating a Data Table Row Reference to the ModelEntity:</u>

- Click on the Source object and expand Table Reference Assignments property category. Expand On Created Entity and set *Table Name* property to ‘EntData’ and RowNumber property to ‘1’.

<u>Adding Process Logic for Seizing and Releasing the Workers at Server1:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Decide Step. Set *Decide Type* property to ‘ConditionBased’ and the *Expression* to ‘ModelEntity.Delegate == true’.
  - In the true path of this step add a Seize Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘Worker2’, *Request Move* from ‘None’ to ‘To Node’ and *Destination* to ‘Input@Server1’.
  - In the false path of this step add a Decide Step. Set *Decide Type* property to ‘ProbabilityBased’ and the *Expression* to ‘EntData.DelegationProbability’.
  - In the true path of the Decide step:
    - Add an Assign step to the process. Set *State Variable Name* property to ‘ModelEntity.Picture’ and *New Value* property to ‘1’.
    - Add an Assign step to the process. Set *State Variable Name* property to ‘ModelEntity.Delegated’ and *New Value* property to ‘True’.

    - Add a Transfer step. Set *From* property to 'CurrentStation', *To* to 'Node', *Station Name* to 'Output@Source1'.
  - In the false path of the Decide step:
    - Add Seize Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Decide Step. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Delegated == true'.
  - In the true path of this step add Release Step to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker2.
  - In the false path add a Release Step to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Deallocation (DEA)

Problem:

I want to model the ability of a resource to relinquish a work item which is allocated to it and make it available for allocation to other resources.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Assign Step, Transfer step, Data Table, Table Reference Assignments

Assumptions:

There are three Workers and two entity types in the system. Worker2 and 3 only processes relinquished entities at Server1. Relinquished entities color changes to blue.

Technical Approach:

In this Simbit, Worker1 has a probability to relinquish an entity at Server1. This information is stored in a Data Table and is associated to entities by Table Assignments. Entities at Server1 will attempt to seize a Worker from a list. If an entity is relinquished it will be transferred to Output@Source1 node and carry a reference to the Worker that has relinquished it to prevent its seizing again from the list again.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Server's Initial Capacity to 'Infinity'.
- Add 3 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 and Worker 3 colors to Red and Yellow respectively by clicking in Color in the ribbon.

<u>Defining the Entities types:</u>

- Place 1 ModelEntity object into the facility view.
- Add an additional symbol and change the second symbol's color to Blue.
- In the Definitions window of model entity, go to the States panel

  - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state ‘SeizedWorker’.
  - Create a new Boolean State Variable by clicking on Boolean in the ribbon. Name this new state ‘Relinquished’.

<u>Creating a list of Workers:</u>

- In the Definitions tab of the Facility View go to the Lists panel. Add a new Object list. Name it ‘Resources’ and add Worker 1,2 and 3 into the list. This List will be used to an entity seize one Worker from the list.

<u>Creating a Data Table:</u>

- Go to the Data Window, and within the Tables panel, click the Add Data Table icon the ribbon to add a new table. Name it ‘EntData’.
- Add 2 columns that are a Object reference and a Real standard property. Change the names of each column to ‘Worker’, ‘RelinquishProbability’.
- Fill the table with data – such as:
  - Worker1, 0,5

<u>Associating a Data Table Row Reference to the ModelEntity:</u>

- Click on the Source object and expand Table Reference Assignments property category. Expand On Created Entity and set *Table Name* property to ‘EntData’ and RowNumber property to ‘1’.

<u>Adding Process Logic for Seizing and Releasing the Workers at Server1:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Decide Step. Set *Decide Type* property to ‘ConditionBased’ and the *Expression* to ‘ModelEntity.Relinquished == True’.
  - In the true path of this step add a Seize Step to the process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘FromList’, *Object List* to ‘Resources’, *Request Move* from ‘None’ to ‘To Node’ and *Destination* to ‘Input@Server1’. Expand Advanced Options, change *Selection Goal* property to ‘Random’ and change the value of *Selection Condition* to ‘Candidate.Worker != Worker1’.

    - In the false path of this step add a Decide Step. Set *Decide Type* property to 'ProbabilityBased' and the *Expression* to 'EntData.RelinquishProbability'.
    - In the true path of the Decide step:
        - Add an Assign step to the process. Set *State Variable Name* property to 'ModelEntity.Picture' and *New Value* property to '1'.
        - Add an Assign step to the process. Set *State Variable Name* property to 'ModelEntity.Relinquished' and *New Value* property to 'True'.
        - Add a Transfer step. Set *From* property to 'CurrentStation', *To* to 'Node', *Station Name* to 'Output@Source1'.
    - In the false path of the Decide step:
        - Add Seize Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Decide Step. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Delegated == true'.
    - In the true path of this step add Release Step to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'FromList', *Object List* to 'Resources'.
    - In the false path add a Release Step to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Stateful Reallocation (SFRA)

Problem:

I want to model the ability of a resource in some process-aware information systems to suspend a work item and reallocate it to another resource that will resume it without loss of state data (progress of the work item).

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Assign Step, Interrupt Step, Interrupted Process Action

Assumptions:

There are two Workers and two entity types in the system. Worker2 only processes suspended entities at Server1.

Technical Approach:

When an EntTypeB arrives to Server2 it will attempt to seize Worker1. If Worker1 is already seized, the entity being processed is interrupted. The interrupted entity will release Worker1 and seize Worker2 in order to process it by the remaining time.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Server's *Processing Time* property to '1' and *Units* to 'Hours'. Change Source's *Interarrival Time* property to '60' and *Units* 'Minutes'.
- Drag another Source object and another Server object. Connect Output@Source2 to Input@Sink1. Set *Entity Type* property of Source2 to 'EntTypeB'. Change Server's *Processing Time* property to '25' and *Units* to 'Minutes'. Change Source's *Time Offset* property to '30' and *Interarrival Time* property to '90' and *Units* 'Minutes'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 color to Yellow by clicking in Color in the ribbon.

<u>Defining the Entities types:</u>

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.
- Add an additional symbol to EntTypeA and change the second symbol's color to Yellow.
- In the Definitions window of model entity, go to the States panel
    - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'SeizedWorker'.

<u>Adding Process Logic for Seizing and Releasing the Workers at Server1:</u>

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Add an Assign step to the process. Set *State Variable Name* property to 'ModelEntity.SeizedWorker' and *New Value* property to 'ModelEntity.SeizedResources.LastItem'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'ModelEntity.SeizedWorker'.

<u>Adding Process Logic for Interruption at Server2:</u>

- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add an Interrupt step to interrupt the activity taking place at Server1. Select the *Process Name* to 'Server1.OnEnteredProcessing' from the dropdown list. And Change the *Interrupted Process Action* to 'ResumeProcess'.
- On the "Original" branch from the Interrupt step, place a Seize step that seizes Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server2'.

- On the “Interrupted” branch of the Interrupt step, place an Assign step that changes the picture of the interrupted entity. Set the *State Variable Name* to ‘ModelEntity.Picture’, the *New Value* is ‘1’.
    - Add a Release Step to the process. Click on the ‘...’ button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘ModelEntity.SeizedWorker’.
    - Add a Seize Step in the true path process. Click on the “...” button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘Worker2’, *Request Move* from ‘None’ to ‘To Node’ and *Destination* to ‘Input@Server1’.
    - Add an Assign step to the process. Set *State Variable Name* property to ‘ModelEntity.SeizedWorker’ and *New Value* property to ‘ModelEntity.SeizedResources.LastItem’.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the ‘...’ button to open the Repeating Property Editor. Click the Add button and change *Object Type* to ‘Specific’, *Object Name* to ‘Worker1’.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Stateless Reallocation (SLRA)

Problem:

I want to model the ability of a resource in some process-aware information systems to suspend a work item and reallocate it to another resource that will undertake it without retention of state data (progress of the work item).

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, Basic Node, Add-on Processing Trigger, Add-on After-Processing Trigger, Seize step, Release Step, Assign Step, Interrupt Step, Interrupted Process Action, Largest Value First

Assumptions:

There are two Workers and two entity types in the system. Worker2 only processes suspended entities at Server1.

Technical Approach:

When an EntTypeB arrives to Server2 it will attempt to seize Worker1. If Worker1 is already seized, the entity being processed is interrupted. The interrupted entity will release Worker1 and seize Worker2 in order to restart the suspended entity.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths. Change Server's *Processing Time* property to '5' and *Units* to 'Hours'. Change Source's *Interarrival Time* property to '4' and *Units* 'Minutes'.
- Drag another Source object and another Server object. Connect Output@Source2 to Input@Sink1. Set *Entity Type* property of Source2 to 'EntTypeB'. Change Server's *Processing Time* property to '25' and *Units* to 'Minutes'. Change Source's *Time Offset* property to '10' and *Interarrival Time* property to '90' and *Units* 'Minutes'.
- Add 2 Worker objects into the facility view. Set 'Output@Source1' Node as the *Initial Home* for each Worker. Set the *Idle Action* and *Off Shift Action* properties of each worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'. Change Worker2 color to Yellow by clicking in Color in the ribbon.

- Set Worker1 Ranking Rule to 'Largest Value First', with a Ranking Expression of 'ModelEntity.Priority'.

Defining the Entities types:

- Place 2 ModelEntity objects into the facility view and change their names to 'EntTypeA' and 'EntTypeB'. Click on EntTypeB and change its color to Red by clicking in Color in the Ribbon.
- Add an additional symbol to EntTypeA and change the second symbol's color to Yellow and set it's priority to 2.
- In the Definitions window of model entity, go to the States panel
    - Create a new Object Reference State Variable by clicking on Object Reference in the ribbon. Name this new state 'SeizedWorker'.
    - Create a new Boolean State Variable by clicking on Boolean in the ribbon. Name this new state 'Suspended'.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Decide Step. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Suspended == true'.
    - In the true path of this step add a Seize Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker2', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
    - Add an Assign step to the process. Set *State Variable Name* property to 'ModelEntity.SeizedWorker' and *New Value* property to 'ModelEntity.SeizedResources.LastItem'.
    - Add an Assign step to the process. Set *State Variable Name* property to 'ModelEntity.Suspended' and *New Value* property to 'False'.
    - In the false path of this step add a Seize Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.

    - Add an Assign step to the process. Set *State Variable Name* property to 'ModelEntity.SeizedWorker' and *New Value* property to 'ModelEntity.SeizedResources.LastItem'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Decide Step. Set *Decide Type* property to 'ConditionBased' and the *Expression* to 'ModelEntity.Suspended == true'.
    - In the true path of this step add a Transfer step. Set *From* property to 'CurrentStation', *To* to 'Station', *Station Name* to 'Server1.InputBuffer'.
    - In the false path add a Release Step to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'ModelEntity.SeizedWorker'.

<u>Adding Process Logic for Interruption at Server2:</u>

- Click on Server2 and create a new process in the Before Processing add-on process triggers. Add an Interrupt step to interrupt the activity taking place at Server1. Select the *Process Name* to 'Server1.OnEnteredProcessing' from the dropdown list. And Change the *Interrupted Process Action* to 'EndDelay'.
- On the "Original" branch from the Interrupt step, place a Seize step that seizes Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server2'.

- On the "Interrupted" branch of the Interrupt step, place an Assign step that changes the picture of the interrupted entity. Set the *State Variable Name* to 'ModelEntity.Picture', the *New Value* is '1'.
    - Add a Release Step to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1'.
    - Add a Seize Step in the true path process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker2', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.

    - Add an Assign step to the process. Set *State Variable Name* property to 'ModelEntity.Suspended' and *New Value* property to 'True'.
- Click on Server2 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the '...' button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'Worker1'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Skip (SK)

Problem:

I want to model the ability of a resource in some process-aware information systems to skip a work item allocated to it.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Before Processing Trigger, Add-on After-Processing Trigger, Add-on Entered Trigger, Seize step, Release Step, Data Table, Table Reference Assignments, Real State Variable

Assumptions:

Entities have a default processing of 3 minutes.

Technical Approach:

Worker1 has a probability to skip an entity at Server1. This information is stored in a Data Table and is associated to entities by Table Assignments. At Server1 before processing if the entity is skipped its processing time is 0. The entity color changes to match the color of the resource that skipped it. Green color means that the entity wasn´t skipped.

Details for building the model:

<u>Simple System setup:</u>

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of the worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

<u>Defining the Entities and using default processing time in Server1:</u>

- In the Definitions window of model entity, go to the States panel
    - Create a new Real State Variable by clicking on Object Reference in the ribbon. Name this new state 'NewProcessingTime'. Set *Initial Value* to '3'.
- Click in Server1 and change *Processing Time* property to 'NewProcessingTime'.

Creating a Data Table:

- Go to the Data Window, and within the Tables panel, click the Add Data Table icon the ribbon to add a new table. Name it 'EntData'.
- Add 2 columns that are a Object reference and a Real standard property. Change the names of each column to 'Worker', 'SkipProbability'.
- Fill the table with data – such as:
    - Worker1, 0,5

Associating a Data Table Row Reference to the ModelEntity

- Click on the Source object and expand Table Reference Assignments property category. Expand On Created Entity and set *Table Name* property to 'EntData' and RowNumber property to '1'.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'EntData.Worker', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Add a Decide Step. Set *Decide Type* property to 'ProbabilityBased' and the *Expression* to 'EntData.SkipProbability'.
    - In the true path of this step add an Assign step. Set *State Variable Name* to 'ModelEntity.Picture', *New Value* to '1'. Add another Assign step and set *State Variable Name* to 'ModelEntity.NewProcessingTime', *New Value* to '0'.

- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'EntData.Worker'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com

Redo (RD)

Problem:

I want to model the ability of a resource in some process-aware information systems to redo a work item allocated to it after completion.

Categories:

Entity – Workers

Key Concepts:

Source, Server, Sink, Path, Workers, List, Basic Node, Add-on Before Processing Trigger, Add-on After-Processing Trigger, Add-on Entered Trigger, Seize step, Release Step, Data Table, Table Reference Assignments, By Link Weight

Assumptions: None.

Technical Approach:

Worker1 has a redo weight. This information is stored in a Data Table and is associated to entities by Table Assignments. At Ouput@Server1, the path selection is made by Link Weight using the Redo Weight. An entity to redo changes his color to match the workers color.

Details for building the model:

Simple System setup:

- Drag 1 Source, 1 Server and 1 Sink into the Facility View. Put the objects in series and connect them with paths.
- Create a new path to link Output@Server1 to Input@Server1.
- Add 1 Worker object into the facility view. Set 'Output@Source1' Node as the *Initial Home* for the Worker. Set the *Idle Action* and *Off Shift Action* properties of the worker to 'Go To Home'. Set the *Initial Network* as 'No Network (Free Space)'.

Creating a Data Table:

- Go to the Data Window, and within the Tables panel, click the Add Data Table icon the ribbon to add a new table. Name it 'EntData'.
- Add 2 columns that are a Object reference and a Real standard property. Change the names of each column to 'Worker', 'RedoWeight'.
- Fill the table with data – such as:
    - Worker1, 50

Associating a Data Table Row Reference to the ModelEntity

- Click on the Source object and expand Table Reference Assignments property category. Expand On Created Entity and set *Table Name* property to 'EntData' and RowNumber property to '1'.

Defining By Link Weight Rule at Output@Server1 and set Path Logic:

- Click on Output@Server1 and change *Outbound Link Rule* property to 'By Link Weight'.
- Click in Path 3, expand Routing Logic property category and change *Selection Weight* property value to 'EntData.RedoWeight'.
- Create a new process in the Entered add-on process trigger. Add an Assign Step to the process. Set *State Variable Name* to 'ModelEntity.Picture' and *New Value* to '1'.
- Click in Path 3, expand Routing Logic property category and change *Selection Weight* property value to '100 - EntData.RedoWeight'.
- Create a new process in the Entered add-on process trigger. Add an Assign Step to the process. Set *State Variable Name* to 'ModelEntity.Picture' and *New Value* to '0'.

Adding Process Logic for Seizing and Releasing the Workers at Server1:

- Click on Server1 and create a new process in the Before Processing add-on process triggers. Add a Seize Step in this process to seize Worker1. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'EntData.Worker', *Request Move* from 'None' to 'To Node' and *Destination* to 'Input@Server1'.
- Click on Server1 and create a new process in the After Processing add-on process triggers. Add a Release Step to the process. Click on the "..." button to open the Repeating Property Editor. Click the Add button and change *Object Type* to 'Specific', *Object Name* to 'EntData.Worker'.

Created by:

André Sousa – University of Minho, Guimarães, Portugal

Contact: andrefcsousa@gmail.com