A simulator prototype for an ERP system

Oscar Alfonso Caceres Mendoza

LYNGBY 2005
MASTER THESIS PROJECT
NR. 2005/87

Preface

This Master Thesis was carried in order to fulfill the curriculum for the M.Sc.. in Computing and Mathematics program at the IMM department in the summer of 2005.

Working on this project has been a challenging and educational experience, all the more so because its conception and application are both based on my professional interest in simulation modeling for ERP systems, as well as my academic interest in computing and mathematics.

I would like to express my sincere gratitude to Bo Friis Nielsen from IMM-DTU for his invaluable guidance and support during my time on this project.

Abstract

The following document presents an investigation based on the idea of improving the value of Enterprise Resource Planning (ERP) systems by adding a discrete event simulation application to the ERP framework.

First, I conducted a brief investigation to assess whether discrete event simulation is already included in most (or any) ERP applications. I found that this type of simulation is not common to ERP systems, and is particularly lacking in systems designed for small- and mid-sized organizations.

Perceiving an opportunity to enhance the functionality of existing ERP systems, I decided to develop a simulation prototype for an ERP application. This project is the subject of my Master thesis.

To achieve the goal of developing a simulation tool, I conducted research on the structural and functional requirements for my prototype, placing emphasis on ease of use and the interaction between the data-fitting and modeling phases of simulation. I chose to develop this simulation prototype for The Microsoft Navision ERP software solution, with which I have several years of experience as a software developer.

In the course of developing the prototype, I found that Navision lacked certain desired functions for performing statistical operations and designing graphical models. I solved this problem by creating external components that can be used within the Navision development environment. Basic testing was performed in order to provide reasonable assurance that the external components would function correctly.

In the end, my efforts to develop simulation capabilities within the Navision ERP system proved successful. As often occurs during the development process, I was inspired to create new functionality that I had not included in my original requirements. I attribute this creative inspiration, in part, to the wide scope of simulation modeling as a concept. I found that my simulation prototype has the potential to add value to many areas of the modular and integrated framework of an ERP system for a result that is greater than the sum of its parts.

In conclusion, I believe that simulation modeling can add significant value to existing ERP systems, and that future research should be done in order to automate the data analysis process so as to enhance the adoption of simulation techniques for the current users of ERP applications.

Table of Contents

Table of Contents *5*

Introduction *7*

PART I: Concepts and Definitions *9*

1. ERP Systems *9*

1.1. What is an ERP system? **9**

1.2. Simulation on ERP software **10**

1.3. Microsoft Navision **12**

1.3.1. Microsoft Navision Development environment 14

1.3.2. History of Microsoft Navision - former Navision A/S 15

2. Simulation *17*

2.1. Definition **17**

2.1.1. Simulation concepts 17

2.2. The modeling process **19**

2.3. When to use simulation **21**

2.4. Stochastic processes in simulation **22**

PART II System Design *31*

3. Conceptual Design *31*

3.1. Data Analysis **34**

3.2. Simulator **37**

3.2.1. Drawing Tool 37

3.2.2. Simulation Execution 47

3.3. Summary **53**

4. Logical Design *55*

4.1. NaviMath and NaviSim automation servers **56**

4.1.1. NaviMath.dll 57

4.1.2. NaviSim.dll 59

4.2. Navision Objects **65**

4.2.1. Data Fitting 69

4.2.2. Simulation Model Creation 90

4.2.3. Simulation Model Execution 97

PART III Implementation and Testing 108

5. Implementation 108

5.1. Implementation Plan....108

5.2. NaviMath.dll....109

5.2.1. INaviMath Interface....109
5.2.2. NaviMath Class....110
5.2.3. RandomGen Class....112

5.3. Data Fitting....113

5.3.1. Creation of Temp Data Table....113
5.3.2. Kolmogorov-Smirnov Test....117
5.3.3. Chi-Square Test....119
5.3.4. Creation of Frequency table....122

5.4. Simulation Execution....123

5.4.1. Engine....123
5.4.2. Next Event Method....126
5.4.3. Joiner implementation....128
5.4.4. Router Implementation....129

6. Testing 132

6.1. Data Fitting....132

6.1.1. Goodness-of-fit tests....136

6.2. Simulation Execution....137

7. Conclusions 139

7.3. Future Work....142

Appendix A Script for creating Interval Data 144

Appendix B Exponential data used for testing the goodness-of-fit tests 145

Appendix C Simple 1 model log 146

Appendix D Simple 2 model log 148

Appendix E Joiner Test model log 151

Appendix F Router Test model log 155

Appendix G Test Script for the Router Shape 159

References 162

Introduction

Since the introduction of Material Requirement Planning (MRP) systems in the 1960s, Management Information Systems (MIS) have enjoyed a steady growth and have become an essential tool for managing the operations of a company. Originally aimed at manufacturing companies and large corporations, MRP grew into Enterprise Resource Planning (ERP) – which is itself now a dated acronym.

ERP systems are currently used in all the Fortune 500 companies, a market that in 2003 was worth a staggering sum of 61 billion dollars. In both my first job and my current position of employment, I am involved in the development of ERP software for small- and mid-size businesses. Each year, we learn of new requirements and technologies that make this market an interesting and challenging and attractive space in which to compete.

The main goal of ERP software is to act as the central interaction point of different areas of an organization in order to facilitate sound, informed decision-making about operations. In effect, the ERP system becomes the main repository for the company's data. This centralized approach suggests and inspires a variety of applications, such as handheld applications that retrieve and update data from and into the main system; e-commerce applications, human workflow, e-billing and e-procurement.

Simulation modeling is another tool designed to aid the decision-making process. A simulation takes input data and processes that data against a model or template that represents a simplification of a certain reality or situation. There exists an observable relationship between the concepts or ERP and simulation modeling that, in my opinion, is underutilized and worthy of serious academic investigation.

Although I have worked for an ERP software vendor for more than five years, I have never heard the concept of simulation modeling applied in that context. It is my belief that simulation modeling software could become an indispensable tool for ERP and other areas of business operations. The very thought of being able to simulate various elements of a company's operation, observe the probable outcomes of predefined scenarios, and make decisions accordingly is a managers dream.

All these ideas were the foundation to what is now presented to you as a Masters Thesis project. With this project I would like to answer some questions concerning the use of simulation software in ERP systems. Some of these questions are:

- Is simulation used in some ERP systems today? If so, in what areas?
- Is it possible to create a simulation module in the ERP system that I have been working for?
- If possible, What would this system need? What functionality could it have and how could it use the potential that is already part of the ERP software.
- What do I think can be done to enhance the simulation application developed?

The answers to the questions concerning the possibility of including a simulation application in an ERP system would be obtained by developing a simulation prototype for the Navision application. Navision is the ERP system were I have had all my working experience and is an application that is used by small and medium size companies. This market is very large and makes it the perfect environment to test the usability of simulation software for average users that do not have extensive knowledge of mathematics or statistics.

This document is partitioned in four parts. Part I describes introductory concepts related to simulation in ERP systems, the history of the Navision and ERP systems in general, simulation and related concepts that would be use on the project. Part II describes the ideas and the design of the proposed simulation prototype system. Part III describes the implemented parts of the design, how were they tested and what conclusions were found in this project. Potential areas of research and thoughts in regards to the future of simulation in the ERP market are also given.

PART I: Concepts and Definitions

This section briefly describes the basic concepts and definitions that are going to be used through out this document.

1. ERP Systems

1.1. What is an ERP system?

The contents of the chapter are a summary of the concepts found in the Enterprise Resource Planning definition in the Wikipedia encyclopedia [11].

Enterprise resource planning systems are management information systems that integrate and automate many of the business practices associated with the operations or production aspects of a company.

Enterprise resource planning is a term derived from material resource planning. ERP systems typically handle the manufacturing, logistics, distribution, inventory, shipping, invoicing and accounting for a company. Enterprise Resource Planning or ERP software can aid in the control of many business activities, such as, sales, delivery, billing, production, inventory management, and human resources management.

ERPs are cross-functional and span the entire enterprise wide. All functional departments that are involved in operations or production have their functions integrated in one system. In addition to manufacturing, warehousing, and shipping, this integration also includes: accounting, human resources, marketing, and strategic management.

To implement ERP systems, companies often seek the help of an ERP vendor or of third-party consulting companies. Consulting in ERP operates on two levels: business consulting and technical consulting. A business consultant studies an organization's current business processes and matches them to the corresponding processes in the ERP system, thus 'configuring' the ERP system to the

organisation's needs. Technical consulting often involves programming. Most ERP vendors allow changing their software to suit the business needs of their customer.

1.2. Simulation on ERP software

In order to have an overview of the existence of simulation software in the ERP industry a small investigation was conducted. This investigation consisted in searching the offerings concerning simulation of all the big players of the ERP arena. The result of this research is presented in the following table:

Vendor	Has a Simulation software?	Similar products
SAP [5]	Yes, but in a different degree. The modules is called Business Intelligence and has a modules called BPS, business planning and simulation. It is mainly oriented to be used for planning of projects and people involved and run simuilations on expected results on the project.	Powersim, add on product that is not part of the basic offering but that it uses the BPS engine.
Oracle (PeopleSoft and J.D. Edwards) [14]	Workforce simulation. Use to simulate different scenarios regarding compensations and budgeting. Traffic simulation based on radio frequency.	Includes business execution process manager language, a tool for modeling business processes that is aim at integrating services. In other words a service orchestration tool [15]
Microsoft Business Solutions division[12]	Limited offering	Human workflow solution. Based on notifying user on events that occur on the system. The workflow engine can be use for creating a simulation model.
The Sage Group[13]	No offering	Simulation for quotes prices based on different material configurations
SSA Global Technologies (which acquired Baan) [6]	Manufacturing scheduling. A simulator for manufacturing scenarios.	
Intentia International AB)[4]	Yes, but focused on the manufacturing area. A part of the Movex module. Can	

	be used to simulate different scenarios concerning the manufacturing or maintenance strategies.	

Table 1 Simulation in ERP software

As it can be seen from table 1, many large software vendors do include simulation in their current offering. However, there are a few, if any, global or generic simulation application that can be used for any user-defined areas (the focus is mostly in the areas of supply chain and finance). Most of the simulation capabilities in theses products do not relate to discrete event simulation and is more closely linked to impact analysis tools based on different data configurations. For this reason, it is possible to say that the use of a generic discrete event simulation application to be included in an ERP system it is, upon first glance, an unexplored area of opportunity.

With these points in mind, the aim of this project is to design and develop a simulator prototype for an ERP system, more precisely, the Navision software package.

It is important to clarify that there is a plethora of simulation applications available in the software market e.g. Arena, GoldSim, Simul8. However, all of them are stand-alone applications that offer integration capabilities with other systems and are not included in ERP systems.

1.3. Microsoft Navision

Microsoft Navision is a well-established ERP solution aimed at the smaller end of the market and is particularly suited to wholesale distribution and manufacturing. One key area where Navision adds value is in a 'hub-and-spoke' environment, where Navision is used at the 'spokes' and reports to a central HQ solution. Customers are typically small- and medium-sized organisations, with 1-50 users, and the largest customer base is in the core industry sectors of wholesale distribution, manufacturing, and services.

Microsoft Navision is delivered to customers in modular packets known as 'granules'. This feature of Navision enables customers to select the specific components of the Navision system that they require, with the freedom to add extra granules over time as the needs of the business change. All modules are based around a core financials application, with a user-based pricing model that enables customers to pay only for what they use.

The core Navision functions provide a scope of coverage that is typical of ERP solutions–financial accounting, inventory management, supply-chain management, transaction tracking and auditing, as well as management systems for other company resources such as personnel. However, Navision also supports functional areas that are not traditionally part of ERP, such as professional services, business analytics, and e-business infrastructure.

Microsoft Navision Version 4.0 provides a fuller level of integration with the Microsoft family of software products. The user interface has a new look and feel that takes its cue from Microsoft Outlook, and integrations with Microsoft Word and Excel have been improved, especially for reporting functions. The latest version of Navision also has sophisticated analytics capability, which allows small businesses to achieve the level of visibility into underlying data, including pre-defined yet flexible Key Performance Indicators.

In addition, Navision 4.0 introduces the concept of business notification–a workflow technique designed to improve productivity and collaboration by alerting users to important issues that pertain to their role in the business. For example, if a sales margin is particularly low. Navision uses a technique called Sum Index Flow

Technology (SIFT) to enable consolidation of data based on user-defined dimensions, which can be easily changed. SIFT is a proprietary method for rapidly recalculating totals as data flows into the system. This method enables the user to run queries at a range of different levels; for example, monthly or yearly.

Navision can be deployed on an Intel platform running Microsoft Windows 2000 or Windows XP. Navision uses Active Directory for directory services on distributed networking environments and stores information on primary settings for users. The underlying database can be either the proprietary C/SIDE database or Microsoft SQL Server. In cases where OLAP-querying capabilities are required, the SQL Server database is necessary.

Deployment is usually carried out by the Microsoft Business Solutions partner, in conjunction with a team from the customer organisation. After deployment, limited resources are likely to be needed for maintenance and administration, and these are usually provided by the customer themselves. Navision is known for its ability to be rapidly implemented, and Microsoft Business Solutions now provides templates and a methodology to help partners implement the Navision system as quickly and efficiently as possible.

As one would expect from an ERP system, integration with third-party applications is possible and can be carried out using different technologies, including XML, ODBC. Navision is tightly integrated with Microsoft's technology stack, including Microsoft BizTalk Server, Word, and Outlook (for integration between Outlook contacts and Navision).

While the 'sweet spot' for Navision is between 20 and 45 concurrent users, the company states that the solution can be scaled upwards, although it is not appropriate for hundreds of users.

The C/SIDE database is very robust and includes a 'version principle' that allows reports to be generated without locking other users out of the database. Each time a transaction is committed, a new version of the database is created. This function enables employees and applications to access and modify the system concurrently and provides a failsafe against catastrophic power failure.

1.3.1. Microsoft Navision Development environment

C/SIDE (Client Server Integrated Development Environment) is the environment in which it is possible for developing applications for the Navision product. A C/SIDE application is composed from seven types of application objects. Each type of application object is created using a specific tool called a designer. The application objects you create using these designers are all based on some general concepts. A fundamental knowledge of these concepts speeds up the C/SIDE application development process.

There are seven basic objects in C/SIDE: The table, form, report, dataport, codeunit, xmlport and the menusuite objects. The Table object is used to describe how data will be stored in the database and how it will be retrieved. The Form object is used to display data to the user in a familiar and useful way. Most forms allow the user to add records to a table, view and modify records as well.

The Report object enables the user to summarize and print out detail information using the filters and sorting that he or she chooses. The Dataport object allows the user to export or import table data. The Codeunit object allows that work with Navision to organize and group the code that they write. enables automated XML-based communication 'agents' to harvest and transform table data. XMLport is used primarily in conjunction with some communication component as a means of integrating with third-party applications. Finally, the Menusuite object allows the creation of custom menus that can be used to group functionality in common areas.

C/SIDE is not object-oriented, but rather it is object-based. This is an important distinction to note. In an object-oriented language or environment, the developer can create new types of objects based on the ones already in the system. In C/SIDE, you have only the seven fore-mentioned types of application objects to choose from.

While this feature limits Navision in a certain respect, it is intended to optimize the speed and performance factor of C/SIDE. Navision is also optimized in this manner to simplify and streamline development, allowing coders and engineers to work with greater speed and efficiency while reducing the risk of severe bugs and defects.

C/AL is the language that the C/SIDE application compiles and run. This language is based significantly on the PASCAL computer language and is very intuitive and easy to learn.

1.3.2. History of Microsoft Navision - former Navision A/S

The following section is an extract from the Software Advisor web site [10] where the history of Navision is described.

Navision was originally founded in 1983 by Jesper Balser, Peter Bang and Torben Wind in Copenhagen, Denmark. In 1984 the product was launch as "PCPlus" in Denmark & Norway. This was a character-based accounting solution targeted towards the SOHO (small office/home office) market. In 1984 "Beauty of Simplicity" was adopted as the first company slogan. In 1987 the company changed its name to Navision and the product was renamed to Navigator.

In 1990 the company launched Navision 3.0, and expanded the market beyond Scandinavia into Germany, Spain and the United Kingdom. In 1992, the company also reached an agreement to distribute Navision in the United States. In 1993 Navision initiated a major development effort to create a new generation of Navision solutions based on the Microsoft Windows 32-bit client/server platform.

The company continued to enhance the product by adding contact management functionality in 1997, manufacturing capabilities in 1998, and advanced distribution in 1999.

2000 was a stellar year for Navision. In 2000 Navision Financials received Microsoft Windows 2000 Professional Certification and Microsoft Windows 2000 Server Certification. The company launched the Navision Commerce Gateway – world's first solution based on Microsoft's BizTalk Server. The company also launched the Navision User Portal – the world's first solution based on Microsoft's Digital Dashboard. In an industry shaking move, Navision Software merged with their long-time Danish rival Damgaard Software. In 2001 the company made many enhancements including:

1. *Re-branded "Navision Financials" as "Navision Attain" and "Damgaard Axapta" as "Navision Axapta";*
2. *Integrated the e-commerce applications, Commerce Gateway, Commerce Portal into both products;*
3. *Introduced User Portal, browser-based access to both products;*

4. *Introduced supply chain collaboration functionality, manufacturing and distribution capabilities, and new financial management functionality;*
5. *Navision received the Designed for Microsoft XP logo.*

In 2002 Microsoft acquired Navision for $1.4 billion – the largest acquisition ever made by Microsoft. Today Navision has been the fastest growing accounting system solution offered by Microsoft. Since then, the Navision software product line has grown steadily and today has approximately 35,000 customers worldwide, and more than 400,000 individual users. The Navision product's strength and future outlook has never looked stronger.

Navision is mainly targeted to mid market companies located in the mid-low end of the market. The typical Navision user is a company with around 70 to 100 employees per location and sometimes is part of a bigger conglomerate. This market is characterized by a focus on solving day to day operations and requires fast and ease to understand solutions.

2. Simulation

This chapter describes the simulation concepts and definition that would be use throughout the document. The contents of this chapter are a summary of the descriptions found in Banks and Carson [1] and Law and Kelton [2].

2.1. Definition

Simulation is the imitation of the operation of a real-world process or system over time. Simulation involves the generation of an artificial history of the system, and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system that is represented.
Simulation is an indispensable problem-solving methodology for the solution of many real-world problems. Simulation is used to describe and analyze the behaviour of a system, ask "what if" questions about the real system, and aid in the design of real systems. Both existing and conceptual systems can be modelled with simulation.

2.1.1. Simulation concepts

There are several concepts underlying simulation. These include:

Model: A model is a representation of an actual system. Immediately, there is a concern about the limits or boundaries of the model that supposedly represent the system. The model should be complex enough to answer the questions raised, but not too complex as explained in the section above.

Events: Consider an event as an occurrence that changes the state of the system. As an example, events include the arrival of a customer for service at the bank, the beginning of service for a customer, and the completion of a service. There are both internal and external events, also called endogenous and exogenous events, respectively. For example, an endogenous event in the example is the beginning of service of the customer since that is within the system being simulated. An exogenous event is the arrival of a customer for service since that occurrence is outside of the simulation.

System State Variables: The system state variables define what is happening within the system to a sufficient level at a given point in time. The determination of system state variables is a function of the purposes of the investigation, so what may be the system state variables in one case may not be the same in another case even though the physical system is the same.

Entities and Attributes: An entity represents an object that requires explicit definition. An entity can be dynamic in that it "moves" through the system, or it can be static in that it serves other entities. For instance, a customer in a supermarket queue is a dynamic entity, whereas the cashier teller is a static entity. An entity may have attributes that pertain to that entity alone. Thus, attributes should be considered as local values e.g. the time of arrival, colour, shape, etc. This can be later used to differentiate the data and collect statistics accordingly.

Resources: A resource is an entity that provides service to dynamic entities. The resource can serve one or more dynamic entities at the same time. These entities can request one or more units of a resource. If denied, the requesting entity joins a queue, or takes some other action (i.e., diverted to another resource, ejected from the system). If permitted to capture the resource, the entity remains for a time, then releases the resource. There could be many possible states of the resource e.g. idle and busy (minimal case), failed, blocked, or starved.

List Processing: Entities are managed by allocating them to resources that provide service, by attaching them to event notices thereby suspending their activity into the future, or by placing them into an ordered list. Lists are used to represent queues. Lists are often processed according to FIFO (first-in-first-out), but there are many other possibilities.

Activities: An activity whose duration is known before the activity begins. Thus, when the activity starts, its end can be scheduled. The duration of an activity can be a constant, a random value from a statistical distribution, the result of an equation, input from a file, or computed based on the event state.

Delays: A delay is an indefinite duration that is caused by some combination of system conditions. When an entity joins a queue for a resource, the time that it will remain in the queue may be unknown initially since that time may depend on other events that may occur.

Discrete event simulation

After defining relevant concepts of simulation modelling we are now able to introduce the concept of discrete event simulation. This type of simulation is the one that offers most interesting aspects to the author in relation with ERP systems and would be the focus of the project.

A discrete-event model attempts to represent the components of a system and their interactions to such an extent that the objectives of the study are met. These models include a detailed representation of the actual internals of the system and are conducted over time (run) by a mechanism that moves simulated time forward. The system state is updated at each event along with capturing and freeing of resources that may occur at that time. State variables change only at those discrete points in time at which events occur. Events occur as a consequence of activity times and delays. Entities may compete for system resources, possibly joining queues while waiting for an available resource. Activity and delay times may hold entities for durations of time.

2.2. The modeling process

The application of simulation involves specific steps in order for the simulation study to be successful. Regardless of the type of problem and the objective of the study, the process by which the simulation is performed remains constant. The following briefly describes the basic steps in the simulation process:

1. **Problem Definition**: The initial step involves defining the goals of the study and what needs to be solved. The problem is further defined through objective observations of the process to be studied. Care should be taken to determine if simulation is the appropriate tool for the problem under investigation (more clarification on this topic in the next section).
2. **Project Planning**: The tasks for completing the project are broken down into work packages with a responsible party assigned to each package. Milestones are indicated for tracking progress. This schedule is necessary to determine if sufficient time and resources are available for completion.
3. **System Definition**: This step involves identifying the system components to be modelled and the performance measures to be analyzed. Often the

system is very complex, thus defining the system requires an experienced modeller who can find the appropriate level of detail and flexibility.

4. **Model Formulation**: Understanding how the actual system behaves and determining the basic requirements are necessary in developing the right model. Creating a flow chart of how the system operates facilitates the understanding of what variables are involved and how these variables interact.
5. **Input Data Collection and Analysis:** After formulating the model, the type of data to collect is determined. New data is collected and/or existing data is gathered. Data is fitted to theoretical distributions. For example, the arrival times of a specific part to the manufacturing plant may follow an exponential distribution curve.
6. **Model Translation**: The model is translated into programming language. Choices range from general purpose languages such as Fortran or simulation programs such as Arena.
7. **Verification and Validation**: Verification is the process of ensuring that the model behaves as intended, usually by debugging or through animation. Verification is necessary but not sufficient for validation that is a model may be verified but not valid. Validation ensures that no significant difference exists between the model and the real system and that the model reflects reality in an acceptable way. Validation can be achieved through statistical analysis. Additionally, face validity may be obtained by having the model reviewed and supported by an expert.
8. **Experimentation and Analysis**: Experimentation involves developing the alternative model(s), executing the simulation runs, and statistically comparing the alternative(s) system performance with that of the real system.
9. **Documentation and Implementation**: Documentation consists of the written report and/or presentation. The results and implications of the study are discussed. The best course of action is identified, recommended, and justified.

2.3. When to use simulation

Although knowing the basic steps in the simulation study is important, it is equally important to realize that not every problem should be solved using simulation. When simulation is applied inappropriately, the study will not produce meaningful results. The failure to achieve the desired goals of the simulation study is, many times, caused by the inappropriate application of simulation.

To recognize if simulation is the correct approach, four items should be evaluated before deciding to conduct the study:

1. Type of Problem
2. Availability of Resources
3. Costs
4. Availability of Data

Type of Problem: If a problem can be solved by common sense or analytically, the use of simulation is unnecessary. Additionally, using algorithms and mathematical equations may be faster and less expensive than simulating. Also, if the problem can be solved by performing direct experiments on the system to be evaluated, then conducting direct experiments may be more desirable than simulating. The real system itself plays another factor in deciding to simulate. If the system is too complex, cannot be defined, and not understandable then simulation will not produce meaningful results. This situation often occurs when human behaviour is involved.

Availability of Resources: People and time are the determining resources for conducting a simulation study. An experienced analyst is the most important resource since such a person has the ability and experience to determine both the model's appropriate level of detail and how to verify and validate the model. Without a trained simulator, the wrong model may be developed which produces unreliable results. The schedule should allow enough time for the implementation of any necessary changes and for verification and validation to take place if the results are to be meaningful.

Costs: Cost considerations should be given for each step in the simulation process, purchasing simulation software if not already available, and computer resources. Obviously if these costs exceed the potential savings in altering the current system, then simulation should not be pursued.

Availability of Data: The necessary data should be identified and located, and if the data does not exist, then the data should be collectible. If the data does not exist and cannot be collected, then continuing with the simulation study will eventually yield unreliable and useless results. The simulation output cannot be compared to the real system's performance, which is vital for verifying and validating the model.

2.4. Stochastic processes in simulation

In order to model reality many simulation models include the use of random components. These components take the form of random numbers that follow a specific distribution (previously obtain in the data analysis process). In order to generate these distributions there are several methods that are base on random number generators.

As mentioned in section 1.2 the main objective of this project is to build a simulation prototype that will reside inside the Navision application. Therefore, an important decision to make is to find out the distributions that would be more relevant for a typical Navision user. To aid the decision process the following considerations have been specified based on the needs of these users:

a) There should be appropriate distributions to use when there is not enough data or only assumptions of the behavior of the data. In many occasions the user would not have the data needed in the system to estimate a distribution for it, hence the need for distributions that can be used with some assumptions about some plausible ranges.
b) There should be a distribution that is suited when there is an abundance of data but is not desire to do a perfect fit or complicated calculations and formulas to model that behavior. Sometimes, obtaining the best distribution for a set of data is a demanding task in terms of experience and knowledge. In this case it should be possible to find a good approximation so the Navision users could still obtain satisfactory results.
c) There should be a distribution suited for arrival and waiting processes. Arrival and waiting process would be, in my opinion, the most common modeling cases that a user would encounter. With this process types it can be possible to model, arrival of customers, service, sales, purchase documents, waiting time for machinery,

work performed by an employee, machine failures, to name but a few examples.

With the above considerations in mind the following distributions have been chosen:

Consideration Index	Distribution Name	Explanation	Algorithm to generate the distribution
a)	Uniform	With this function the user can give a range of values when there is absence of data.	U(a,b): 1.- Generate U=U(0,1) 2.- Return X = a + (b-a)*U
	Triangular	With this function the user can give a maximum, minimum and most likely value when there is absence of data.	triang(a,b,c) 1.- Generate U=U(0,1) 2.- c´ = (c-a)/(b-a) 2.- if U ≤ c´, then X = $\sqrt{c'U}$. Otherwise, X = $\sqrt{(1-c')(1-U)}$. 3.- Return X´ = a + (b-a)*X
b)	Normal	Besides the fact that many data sets can be normally distributed. Because of the central limit theorem[7][8] this distribution can be applied to wide variety of cases in the presence of an average and variance in the observations.	N(μ,σ²) 1.- Generate U_1=U(0,1) and U_2=U(0,1) 2.- $X_1 = \sqrt{-2\ln U_1} * \cos 2\pi U_2$ 3.- $X_2 = \sqrt{-2\ln U_1} * \sin 2\pi U_2$ 4.- return $X'_1 = \mu + \sigma * X_1$ 5.- return $X'_2 = \mu + \sigma * X_2$
c)	Exponential	A very good and widely used distribution for arrival processes. It can be also used instead of a Poisson process too, since the inter-arrival time is exponentially distributed.	Exp(β) 1.- Generate U=U(0,1) 2.- Return X = - β lnU
	Weibull	It is widely used to simulate failures and waiting times. Also is a versatile distribution that can take the characteristics of other types of distributions.[9]	Weibull(α,β) 1.- Generate U=U(0,1) 2.- Return X = $\beta(-\ln U)^{1/\alpha}$

Table 2 Considerations for chosen distributions and algorithms to generate random variates.

The above mentioned distributions would also be use on the data analysis process since they are the only ones that the user would be able to use on the prototype.

The reader should notice the omission of discrete distributions such as: Poisson, Geometrical and Binomial. These distributions are commonly used to quantify the number of entities arriving in a fix interval of time and estimation of success or failure events respectively. This kind of events will not be in the scope of the prototype.

A fundamental part of the data analysis process is the goodness-of-fit tests that are going to be used. The most common ones to use are the Kolmogorov-Smirnov(K-S) and Chi-Square tests, which would be prefer choice in this prototype. It is worth mentioning that there are other methods available but they are mostly heuristic ones that are not going to be implemented. Some of these methods are: histograms, box plots and probability plots.

The goodness-of-fit tests are based on the null hypothesis (H_o) of no significant difference between the sample and the theoretical distributions. The major shortcoming that they present is that they are not very powerful for small to moderate sample sizes since they are not very sensitive for small disagreements between the data and the fitted distribution. Also, for large amounts of data points, they will almost always reject the H_o (small departures from the hypothesized distribution would be detected). This is unfortunate since sometime is sufficient to have “almost” correct distributions.

The following table briefly describes the advantages and disadvantages of these tests. For further details on how the tests are calculated please refer to the literature.

Test	Advantages	Disadvantages
Chi-Square	- It can be applied to any distribution for which you can calculate the cumulative distribution function. - It does not require that the distributions parameters are known.	- It requires grouping the data in intervals for the calculation of its test statistics. This can be troublesome since it affects the result of the test and there aren't standard methods of how to define them. Heuristic methods are commonly use. The most common one is to make $np_i \geq 5$, where n is the sample size and p the probability of interval i. - It requires a sufficient sample size in order for the test to be unbiased. - It applies to continuous and discrete distributions.
Kolmogorov-Smirnov	- It does not require grouping the data in intervals. - Is valid for any sample size.	- The range of applicability is limited since the original test requires the knowledge of all the parameters in advance. There are some modifications on the original test, so it is possible to use on some limited number of distributions when the parameters have been estimated from the data. These distributions include the Exponential, Weibull and Normal. - It only applies to continuous distributions

Table 3 Advantages and Disadvantages of the K-S and Chi-Square Goodness-of-Fit tests

Based on table 3, both of these tests should be part of the data fitting functionality since they both complement each other in some areas. Also, they can be applied to the distributions explained in table 2 with the benefit that the distribution parameters can be estimated using the maximum likelihood estimators. It has to be said, that these tests are a formal way of proving wether a distribution fits certain sample data. Therefore, it is the basis of automating the data fitting process.

However, there are several exceptions, and many different kinds of tests and method in use. This fact makes the complete automation of a data analysis process a broad subject on its own and an issue that is beyond the scope of this project.

As stated before, both of these tests require the calculation of the maximum likelihood estimator (MLE) and the calculation of probabilities based on the cumulative density function.

The Chi-Square test needs a set of critical points (for the chi-square distribution for different degrees of freedom) to compare it to the X^2*(ki-square)* statistic to accept or reject H_0. The critical points are tabulated data that is available in the literature. It is suggested to use the equiprobable approach, where the expected value of each interval is calculated with the same probability (p). For the test to be safe it is recommended that this equation is satisfied: $np_i \geq 5$, where p_i are the probabilities of each interval.

The number of intervals k is recommended to satisfy the equation: $k \leq n/5$. The application of the Chi-Square test depends on the sample size value. An explanation of how to calculate the intervals for the weibull, exponential and Normal distributions can be found in Banks and Carson, pages 353-356[2].

Recommendations on the number of intervals and the probabilities to use to calculate the expected value are presented in the following table:

Sample Size, n	Number of class intervals, *k*	Interval number to use in the project	Probability to use
<20	Do not use the Chi-Square test		
50	5 to 10	10	1/10
100	10 to 20	15	1/15
>100	$\sqrt{n}$ to n/5	n/5	5/n

Table 4 Recomendations for the Chi-Square test

The method to create the intervals for the distributions are listed in the following table, the value a_i is the end of each interval:

Distribution Name	Equation to use
Uniform	Choose *p,* *Compute* $a_i=(b-a)ip+a$. Start from $(a,a_1), (a_1, a_2)\ldots(a_k, b)$
Normal	Choose the value of *p*, find the value (*1-*α) of *z* from the table for the normal cumulative distribution. Compute $a_i= \mu - (1-\alpha)\ \sigma$. Start from $(-\infty,a_i), (a_i,a_{i+1}) \ldots (a_i ,\infty)$
Exponential	Choose the value of *p,* compute $a_i = -\beta \ln(1-ip)$ Start from $(0,a_1), (a_1, a_2)\ldots(a_k, \infty)$
Weibull	Choose the value of *p,* Compute $a_i = \alpha\left[-\ln(1-ip)\right]^{\frac{1}{\beta}}$, Start from $(0,a_1), (a_1, a_2)\ldots(a_k, \infty)$

Table 5 Methods to create intervals for the Chi-Square test

The altered Kolmogorov-Smirnov test uses different critical points depending on the distribution to be examined. This test uses the *D* statistic to accept or reject H_0. The values of the *D* statistics to use in this project are described in the following table:

Distribution	*D* test statistic
Uniform	$\left(\sqrt{n}+0.12+\frac{0.11}{\sqrt{n}}\right)D_n > c'$
Triangular	Not needed
Normal	$\left(\sqrt{n}+0.01+\frac{0.85}{\sqrt{n}}\right)D_n > c'$
Exponential	$\left(D_n-\frac{0.2}{n}\right)\left(\sqrt{n}+0.26+\frac{0.5}{\sqrt{n}}\right)D_n > c'$
Weibull	$\sqrt{n}D_n > c'$

Table 6 Kolmogorov-Smirnov test statistic calculations

Finally, the following table describes the formulas and methods to calculate the density functions, MLE and X^2 and *D* test statistics, for the distributions mentioned in table 2.

Distribution Name	Distribution function	Maximum Likelihood Estimator(MLE)
Uniform	$F(x)=\begin{cases}0 & x<a\\ \dfrac{x-a}{b-a} & a\le x\le b\\ 1 & b<x\end{cases}$ a<b	a=minimum value of the data, the data should be in ascending order. b=maximum value of the ordered data. the data should be in ascending order.
Triangular	$f(x)=\begin{cases}\dfrac{(x-a)^2}{(b-a)(c-a)} & a\le x\le c\\ 1-\dfrac{(b-x)^2}{(b-a)(b-c)} & c\le x\le b\\ 1 & b<c\end{cases}$ a<c<b	There is no need to have a MLE for this distribution since it is used when there is absence of data.
Normal	$\Phi(z)=\int_{-\infty}^{z}\frac{1}{\sqrt{2\pi}}e^{-z^2/2}dz \qquad -\infty\le x\le\infty$ σ>0 and $z=\dfrac{x-\mu}{\sigma}$	$\hat{\mu}=\overline{X}(n),$ $\hat{\sigma}=\left[\dfrac{n-1}{n}S^2(n)\right]^{1/2}$
Exponential	$F(x)=\begin{cases}1-e^{-x/\beta} & x\ge 0\\ 0 & otherwise\end{cases}$ β>0	$\hat{\beta}=\overline{X}(n)$

Weibull	$F(x)=\begin{cases}1-e^{-(x/\beta)^{\alpha}} & x\geq 0\\ 0 & otherwise\end{cases}$ α, β>0	$\frac{\sum_{i=1}^{n}X_i^{\hat{\alpha}}\ln X_i}{\sum_{i=1}^{n}X_i^{\hat{\alpha}}}-\frac{1}{\hat{\alpha}}=\frac{\sum_{i=1}^{n}\ln X_i}{n},\ \hat{\beta}=\left(\frac{\sum_{i=1}^{n}X_i^{\hat{\alpha}}}{n}\right)^{1/\hat{\alpha}}$ It should be solved by Newton's method. The recursive step is: $\hat{\alpha}_{k+1}=\hat{\alpha}_k+\frac{A+1/\hat{\alpha}_k-C_k/B_k}{1/\hat{\alpha}_k^2+(B_kH_k-C_k^2)/B_k^2}$ Where, $A=\frac{\sum_{i=1}^{n}\ln X_i}{n},\ B_k=\sum_{i=1}^{n}X_i^{\hat{\alpha}_k},C_k=\sum_{i=1}^{n}X_i^{\hat{\alpha}_k}\ln X_i,$ $H_k=\sum_{i=1}^{n}X_i^{\hat{\alpha}_k}(\ln X_i)^2$ As a starting point this estimate can be use: $\hat{\alpha}_k=\left\{\frac{\frac{6}{\pi^2}\left[\sum_{i=1}^{n}(\ln X_i)^2-\left(\sum_{i=1}^{n}\ln X_i\right)^2/n\right]}{n-1}\right\}^{-1/2}$

Table 7 Distribution Density functions and MLE's

PART II System Design

3. Conceptual Design

As stated in section1.3, the Navision application is targeted at the mid-low end of the ERP market. This market is characterized by companies with a staff of 100 to 500 employees and less than $1 billion in annual revenue. The typical Navision user is knowledgeable on many areas of the company since usually has to manage different tasks concerning other departments and is accustom to the design pattern and look and feel of the standard Navision modules. The design principle of the application is based on simplicity and intuition. A user should accomplish tasks in simple steps with the guidance of an intuitive user interface without the need of extensive training or comprehension of assistance material.

The Navision application handles most of the areas of a company; which in Navision terms are called granules. The most complex granules are the Manufacturing and distribution ones. In these granules it is possible to set and calculate different kind of related functionality like: reordering stock policies based on parameters such as lead time and safety stock; capacity requirement planning for productions orders and resource planning tasks.

Even though these are complicated operations, the application does not require any deep knowledge of mathematics, statistics, inventory control or manufacturing processes. With a few concepts and definitions in mind, the user is able to register information and use the functionality. These characteristics give strong design principles to follow and should be viewed as a guiding tool for any new applications that would like to be developed.

With this in mind, several considerations can be stated. The following list describes the design principles to be use as a foundation to develop the simulation functionality:

1. A user with limited knowledge of mathematics and statistics should be able to work with the simulator.
2. The data analysis tasks should be automated as much as possible.
3. The modeling process should be done on a user interface that allows making diagrams.
4. It should be possible to use the existing data and functionality of the ERP software as much as possible.
5. It should be possible to work on data analysis or process modeling independently and concurrently.
6. It should be possible to add new distributions or functionality easily.
7. The system should be ease to maintain and test.
8. It should be possible to reuse as much data as possible from the data analysis process.
9. It should be clear to the user how long a simulation would take and it should be possible to cancel it at any time.
10. The system should help the user as much as possible, indicating when errors or missing information is present so it can be fixed easily.

After having these principles in place, the next task is to figure out what components should the simulator have and how should it look like. Based on the different steps that need to be carried on a simulation process, some ideas came to mind in relation to what kind of personas would interact with the different tasks. In this regard, two personas came to mind: a Modeler and a Data analyst.

The Modeler is the person that is knowledgeable on the business processes and is acquainted on how things work in the company; the Modeler is the most suited candidate to depict processes and create model drawings. The Data , on the other hand, will have a deeper understanding on the application and on technical issues. In this case he/she is (or will be) knowledgeable in basic concepts of statistics and the static structure of the application (tables and fields).

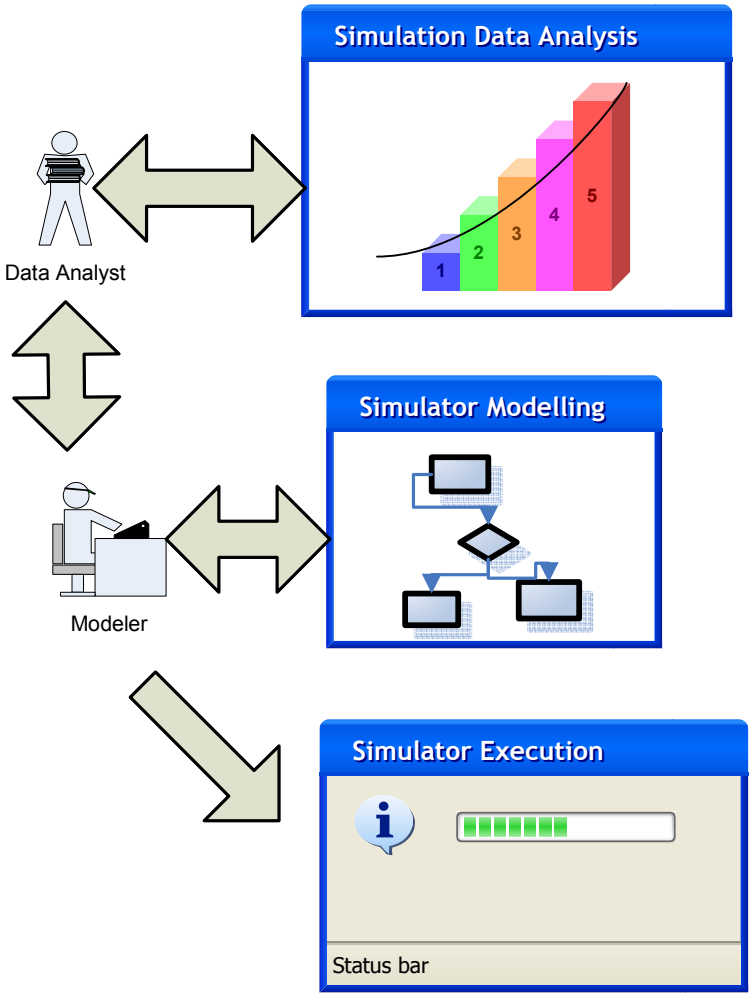


Figure 1 Simulator actors and interactions points

As shown in the above figure, the system would mainly have two interaction points: The simulator itself (modeling and execution) and the data analysis part; both being independent from each other. This fact makes possible for any of the actors to work in parallel and focus on their respective fields of expertise. When both actors are finished with their respective duties, they should coordinate their efforts in order to have a model ready for simulation. Most likely, it would be a coordination coming from the Data Analyst assisting the Modeler in the use of more complex parameters.

When these steps are done, the execution of the simulation may begin. The actor involved in this task would most likely be the Modeler, since is he or she will need to maintain knowledge of how things should work and can asses and inspect that the obtained results are reasonable an according to reality.

It is important to mention that one of the main goals for a future simulator should be that the activities concerning data fitting and modeling are performed by one person. This is because the main idea is that the data fitting tasks should eventually be fully automated, making the simulation phases primarily be concentrated on the modeling tasks.

The following sections would describe the components that, in my opinion, should be part of the modeling and data analysis areas of the prototype.

3.1. Data Analysis

This module should allow the user to analyze existing data in order to find the statistical distribution that fits it in the best way. As stated before, this functionality requires a very basic knowledge in mathematics and statistics–something that could be explained by reading a few sections in the online help or training.

The user should have a repository where he or she can set an identification number and description of the data that is being analyzed. For example, if the user is going to find a distribution for the rate of incoming service orders in a repair shop, he or she should be able to register this information somehow. The most probable outcome is that the data will take the form of an 'estimator' table in which the data can be stored.

Since the data to be analyzed is most likely existing information registered in Navision, the Estimator table should allow the user to select the table-field duple that is most relevant for the case of study. Following the example of the repair shop, if the user realizes that in order to estimate the rate of incoming service orders, he or she must find out in which table are the service orders registered and what field would be suited to obtain the data to analyze. By inspection, the user observes that the service orders are stored in the Service Header table and that a good prospect to obtain the information is the Document Date field.

The aboved-mentioned example leads to some important reflections concerning the usage of data registered in Navision. These considerations are:

1. Navision must have the means to allow the inspection of any table and field just by specifying some table, field identifiers.
2. Table fields with numeric data are the ones that should be mainly targeted in data analysis studies since the majority of the phenomena that would like to be modeled has this characteristics e.g. item quantities, sales amounts, budget amounts, hours worked, etc. In computer terms this translates to: integer, decimal, short, float type fields.
3. Special cases include the Date and Time type fields. As mentioned before in the repair shop example, the Date type field is best suited for arrival studies. However, this information should be further processed to become numeric data, e.g. by inspecting the date of arrival of a service order, we can determine the relative frequency of service order arrivals within a given period of time.

Fortunately, Navision has the means of analyzing any field and table that is part of the system, this takes care of the concern made in point one of the above list.

For the points concerning Date and Time type fields there is a logical aggregation that would make the user capable of analyzing data related to time studies. When the user wants to estimate a parameter that relates to time, he or she would be concerned in one of two things: duration studies or occurrences studies. The duration studies relate, as the name suggests, the study of **how long** it takes for an event to occur. In this regard, it is obvious that there must be a point of reference from which to make calculations. For example, to calculate how long it takes to finish a task, one must include a starting and ending time, as well as a starting and ending date. The occurrence studies, relate to **how often** an event occurs. This also would need a frame of reference, e.g. in the case of incoming service orders, the user can check the date of creation with respect to a month, quarter, year, day, etc. in order to calculate the frequency of creation.

Another tool or functionality that should be present in the data analysis module is a process to make a frequency table in order to make a histogram. With this tool the user can observe the shape of the data so he/she can speculate on the most fitted matching distribution and perform relevant tests. This tool can be use for a user most knowledgeable in these techniques but by no means should be a required step.

After the user has finished selecting the table or fields, then he or she is ready to analyze the corresponding data. This analysis should be performed through a Navision report, where the user can select the distributions that will be used to fit the data. Later, the results will be presented with the different fitted parameters and goodness-of-fit test results.

As mentioned in section 2.4, the goodness-of-fit tests to use in this project are the Kolmogorov-Smirnov and Chi-Square tests. These tests are a formal way to analyze if a distribution fits some data sets and it's a process that can be automated.

To calculate all the distribution parameters and statistical tests, a computer language with mathematical functions is needed. However, C/SIDE does not include mathematical or statistical libraries, therefore it would be necessary to create an external component that uses a language that includes this kind of libraries. Navision can interact with components done in .NET. C#--a language that:

1. Can be compiled using the .NET framework; and
2. contains all the required mathematical functions.

Thus, .NET C# is the language of choice for developing the external components in this project.

Finally, it is very probable that the user will want to analyze data that is not present in the Navision application. An ease way to solve this shortcoming is to allow the import of data in a worksheet style table created inside Navision. Later, after the data has been imported, the user can use the functionality described above and create a new estimator.

To summarize, the following figure depicts the different components that will be part of the data analysis functionality:

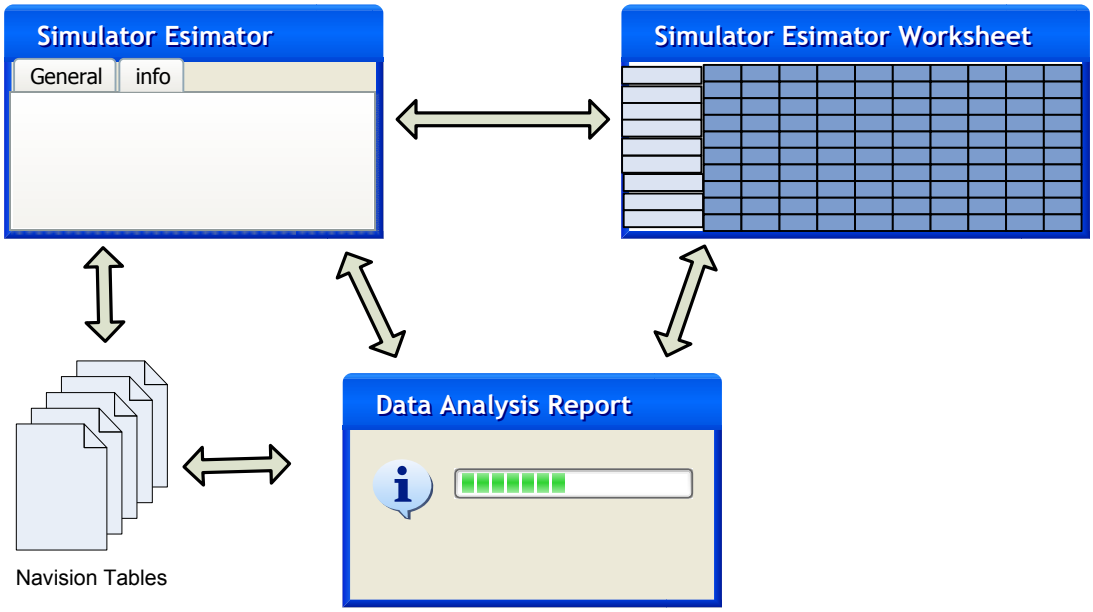


Figure 2 Simulator data fitting functionality overview

3.2. Simulator

The simulator should consist of two parts: model drawing and simulation execution. The user should be able to draw the model, enter some parameters and execute the simulation. The description of the model drawing tool and the simulation execution is described in the following sections.

3.2.1. Drawing Tool

The C/SIDE development environment does not allow the creation of extra controls to make possible the development of a drawing functionality. Nevertheless, it makes possible the interaction with external components that can have embedded functionality. That is why; the best solution to be applied in this case is to create the model-drawing functionality in an external component or dll. The main disadvantage in this approach is that all the business logic and data from Navision would not be present since it is handled in a foreign environment. There are ways to interact with external applications but only on the data level side, Navision is not able to expose business logic to external components at the time of writing.

The best way to tackle this inconvenience is to make the component only a drawing tool and transport the model details into Navision in order to run the simulation. Nonetheless, it would be ideal to have the data and business logic available when making a model since it would make possible the creation of very detail and complex situations that would really leverage the functionality stored in the Navision system.

The drawing tool should make possible the creation of models that include processes, entities and the interactions between them. The user should be able to select different kinds of shapes with different meanings. The shapes present on the tool should be sufficient to model a broad variety of circumstances. To come up with the right shapes, it is necessary to elaborate on the "situations" that most probably the user will face when modeling processes.

The situations that where thought of, where inspired by the different modules of the Navision system and the possible problems that could be solved using simulation in each of them. After some analysis the suggested shapes to use are listed in the following table:

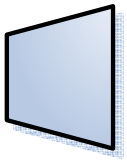
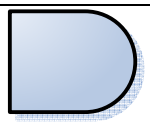
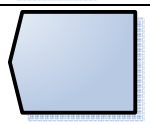
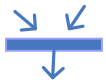
Shape	Shape Name	Description of use
	Process	This shape would allow the user to specify a process. E.g. someone performing a job, or waiting time for transportation.
	Router	This shape would be use to route to other shapes depending on certain criteria. The criteria and the methodology would be described further in this section.
	Creator	This is the shape that will initiate the process of generating the flow in the system
	Terminator	This shape will indicate when the flow ends.
	Arrow	This shape will allow the user to connect the shapes together.
	Joiner	This shape would allow joining entities together to be passed to the connected shape. Typically, it would be use in manufacturing or assembly models where many components are joined together to form one combined output.

Table 8 Simulation modeling shapes

The process and creator shapes should be linked to estimators created in Navision. The Router and Joiner shapes cannot have estimators associated with them and behave in a special manner. The behavior of each shape is further described below.

Creator Shape

The creator shape is the one that models the creation of entities that are going to flow through the system. It is associated with an estimator and has two parameters. The first one is use to specify how many entities are going to be delivered through the system each time. The second one is to tell the system if the creation of entities would be triggered based on a linked process that has a minimum queue value. The default creation quantity value would be one unit at a time. With this feature, it would be possible to model inventory system or systems that present feeding and storage characteristics.

Process Shape

This shape models delays based on activities. It is associated with an estimator and has parameters to trigger the creation of more entities. The created entities will come from its input shape based on its minimum queue value. This shape can have input from more than one shape, unless it is linked to a creator one. It has an *idle* status variable that is triggered when the process is "working" on a entity or not.

Router Shape

The Router shape is used to route the outputs of one process to the input of another. This can be use as a tool to model the behavior of a single queue that feeds many processes. The choice of the next process to choose from would depend on different criteria.

The following picture would be use as an aid to describe the behavior in different situations:

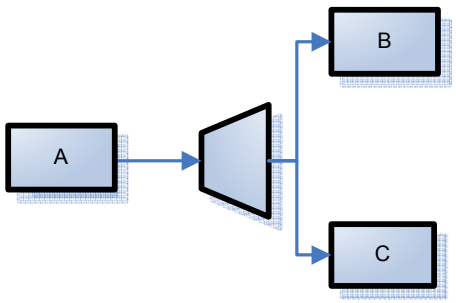


Figure 3 Router shape modeling

When an input comes from A the Router can be set up in two ways to route to shapes B and C. The first method -which would be the default one- consist of sending the output from A to the first shape that is idle. If more than one shape is idle then the decision of which shape to choose would be at random. If all the shapes are not idle then the shape with the smallest queue would be chosen.

The second method consists of setting probabilities for the next shape to be chosen. As an example, if 55.8% of the times shape B should be chosen then the user would setup the router with shape B to have a probability of 0.558 and shape C of 0.442. Then a uniformly distributed random number would be generated between 0

and 1000 and the choice of which shape to choose would be given by the following table:

Shape	Probability	Cumulative Probability	Random number Assignment
B	0.558	0.558	0 – 558
C	0.442	1.0	559 – 1000

Table 9 Probabilities for the Router shape example

For instance, if the numbers 5 and 650 where generated then shapes B and C would have been chosen accordingly.

Joiner Shape

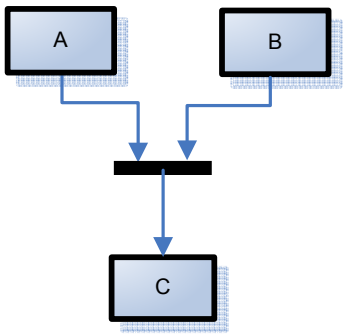


Figure 4 Joiner shape modeling

The joiner shape works in the way that the output of many shapes is combined into one input for another one. As an example, figure 4 shows processes A and B joining together to serve process C. When and output from A arrives at the joiner then as soon as an output from B arrives (or vice versa) it is pass as one output to the C process. If it is not possible to route to the next shape then a queue system would take care of keep tracks of all the inputs arriving at the joiner.

Limitations

Based on the features of each shape the user should be limited on the freedom to associate shapes together, the limitations are described in the following table:

Shape	Condition	Input Shape(s)	Output Shape(s)	Comments
Creator		None	All Except terminator shapes but only one connected.	This shape is not allowed to have any inputs.
Process	Min queue parameter off	Any, one or many connected.	Any but only one connected	
	Min queue parameter on	Only one or many creator shapes.	Any	A router and a joiner shape could be connected in between making the link to the creator shape hard or impossible to obtain.
Router		No router shapes.	No router shapes, one or many shapes connected.	There is no practical use in connecting two router shapes together
Joiner		One or many connections allowed..	only one shape connected.	
Terminator		All except creator shapes one or many connected.	None	This shape is not allowed to have any outputs

Table 10 Shape limitations

Model Validation

Since there are many rules that must be followed in the creation of a model, it would be necessary to have a validating functionality. This feature would allow the system to notify the user on the errors made in the modeling process. The notification should be clear enough so he or she can take corrective measures just by reading the error messages.

The validations performed would be the following:

- All shapes must be connected to another shape and no missing links must be present in the model. This can be done by checking each shapes input and output connections.

- The connections between shapes must be valid and according to table 9.
- All Creator and process shapes must have an estimator assign to them.
- All Router shapes must have correct values assign to them.
- All the flows in the model must begin with a creator and end on terminator shapes.
- There must not be any circular references on the model. A circular reference is when there is a loop on the flow.

Modeling Tool Overview

After describing the shapes that would be part of the drawing tool, it is possible to further visualize how the tool would look like to the user:

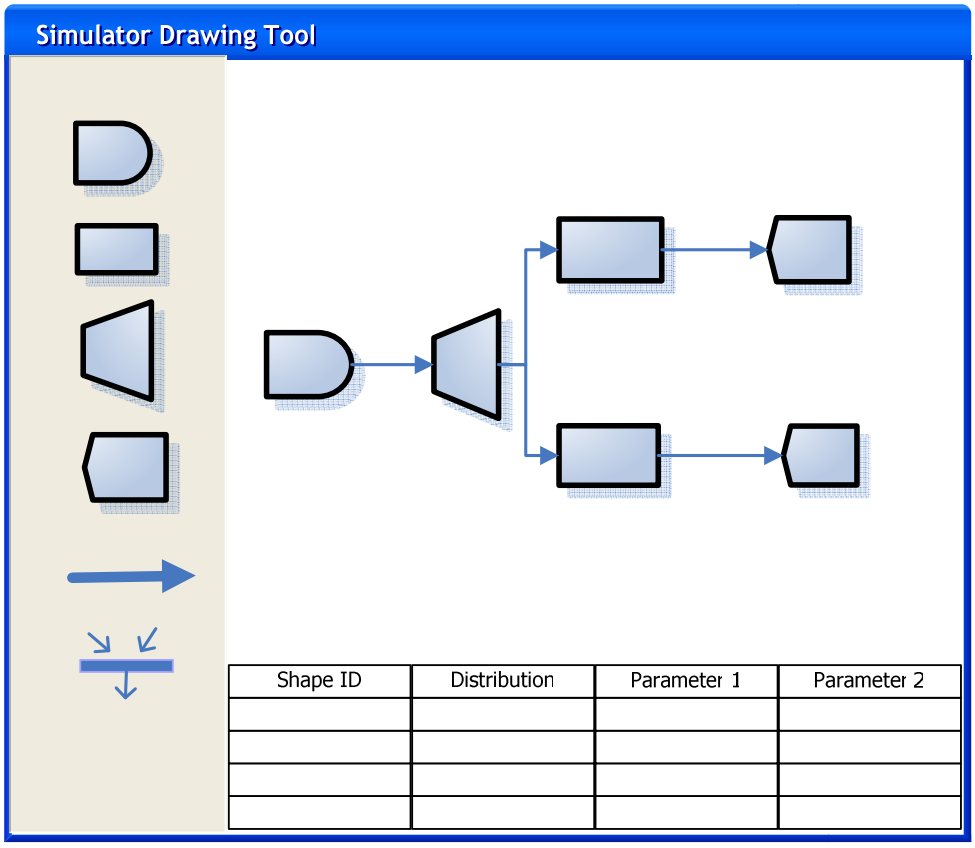


Figure 5 Picture of the model drawing tool

As it can be seen from Figure 5 the drawing tool would consist of 3 panels. The left panel where the shapes described would be available to drag onto the right pane. The right pane would have a drawing canvas and a grid section describing the contents of each of the shapes.

Modeling Examples

As a form of illustrating the flexibility of the shapes to be used, three examples would be given and modeled using the above described shapes.

Example 1: A service department model

The service department of a company has 3 employees. Each of them is allocated to a service request based on the skill needed to fulfill it. Each employee has certain skills that can be unique or overlap with someone else's skill set. A table with an example of the employee's skills is shown below.

Employee	Skills
1	A,B
2	B,C
3	D

Table 11 Service department example - employee skills.

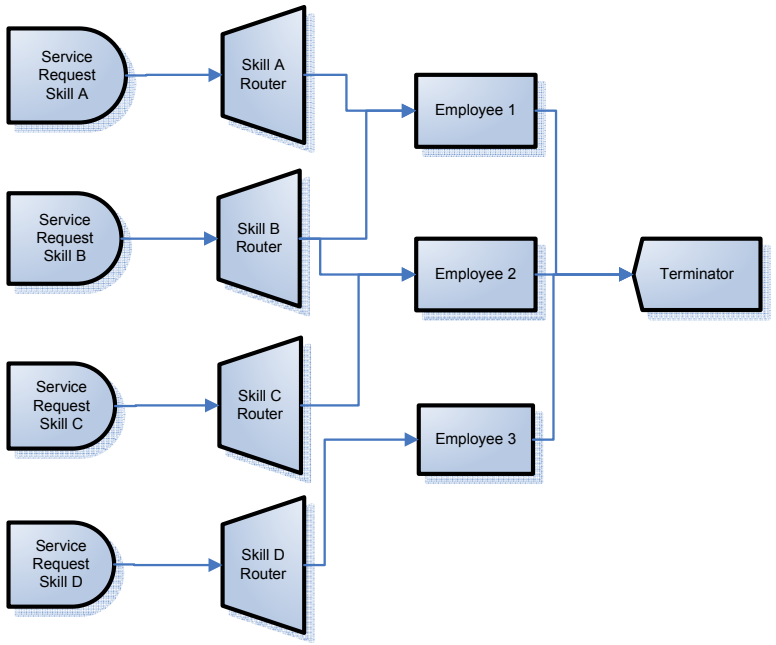


Figure 6 Service department model

Example 2: Inventory model

Items A, B and C arrive at the warehouse to be sold later. A model for this situation is described below.

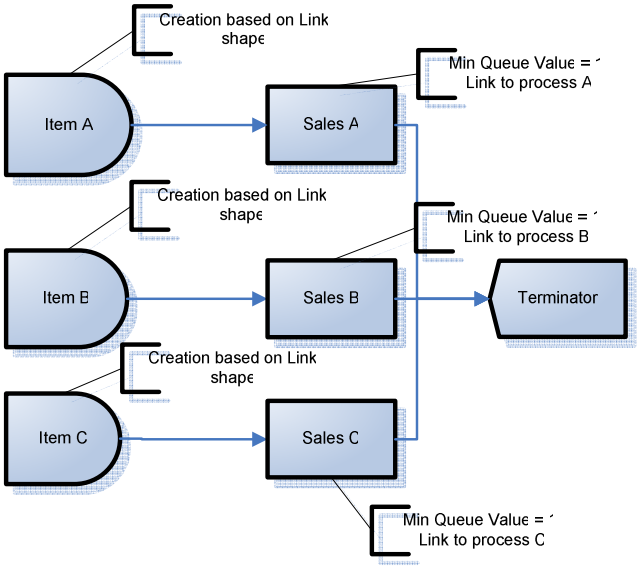


Figure 7 Inventory model

Example 3: Manufacturing model

Two machines A, and B, need items I1, I2, I3, and I4 to manufacture items M1 and M2 respectively. The manufactured items are then assembled and sold. A model for this process is described below.

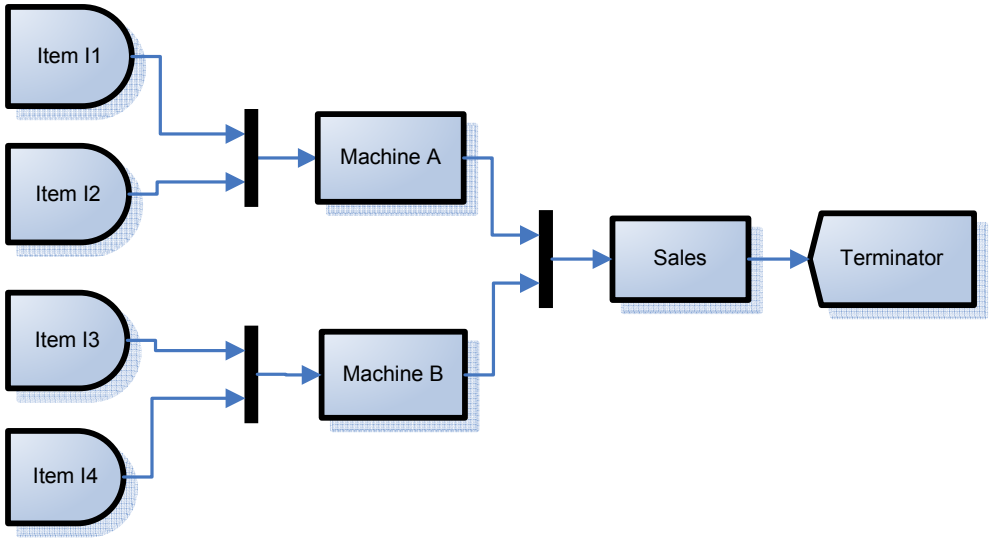


Figure 8 Manufacturing Example

3.2.2. Simulation Execution

After the user has created a complete and valid model the simulation execution phase is ready to commence. The simulation execution is divided in three phases: initiation, execution and termination.

The **initiation** phase deals mainly with the validation and setup steps necessary to perform a successful simulation. The first step in the simulation engine is the validation (re-validation) of the model. Here the system checks for missing data or links in the drawing, if any faults are found the execution is suspended and an error message is given to the user. After the model has been validated the system creates the termination event. A termination event could be a clock time (stop the simulation after 100 units of time) or a number of created "entities" from a creator shape (stop the simulation after 100 entities have been created on shape A). When the termination event has been created is later inserted on the event list; the event list is guiding element of the execution engine since it bases its operations by reading the scheduled events. The last step in the initiation phase is the scheduling of the arrival events from the "creator" type shapes. These events are later inserted into the event list.

The **execution** phase starts after the last arrival events have been inserted. The methodology used in the execution is based on the principles described in the literature (see [1] and [2]). For the most part the execution engine uses the next-event time-advance approach, and the standard queuing mechanism for arrival and departure events. The next-event time-advance approach deals with advancing the internal clock to the time specified in the next unfinished event of the event list. The event types are arrival from the creator shapes and departure from the process shapes.

The mechanism to use when an arrival event occurs is depicted in the following figure:

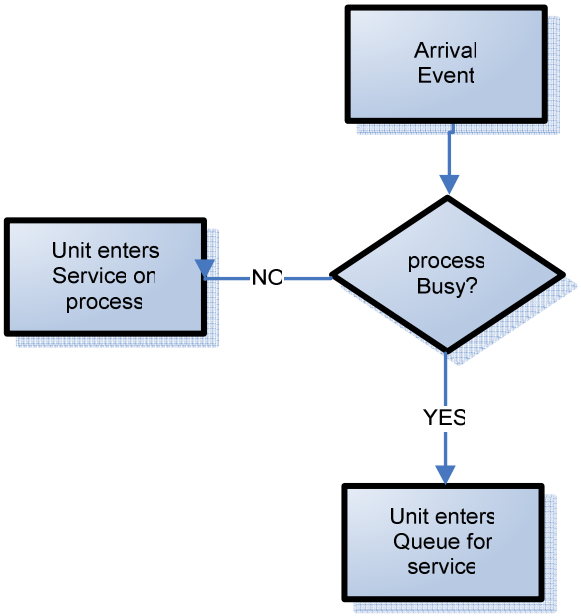


Figure 9 Mechanism when an arrival event occurs

When an arrival event occurs, the system checks weather the process is busy. If so, the unit enters the shape's queue to wait until the process is idle again. If the process is idle, the unit starts to be serviced and a departure event is inserted in the event list. After this is done a new arrival event is created if the shape is not linked to a process minimum queue value.

On departure events, the mechanism used in the execution engine is shown in the following figure:

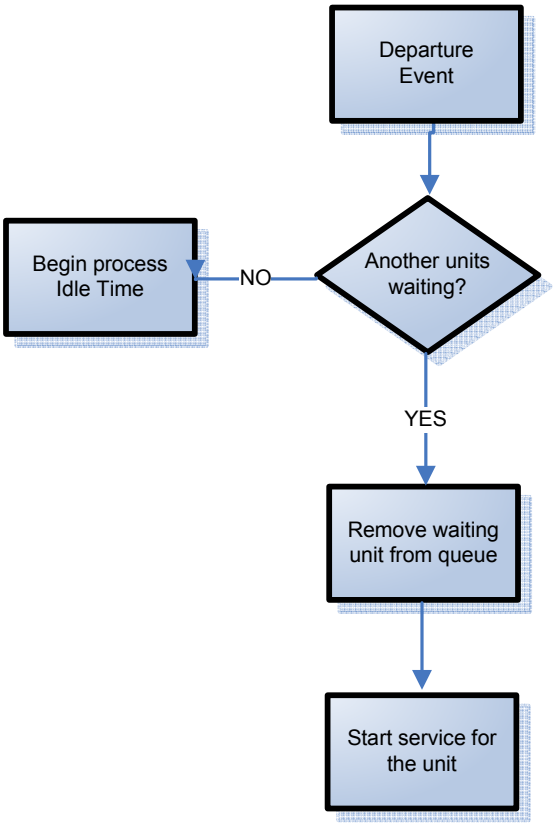


Figure 10 Mechanism when a departure event occurs

When a departure event occurs, the system checks if another unit is waiting to enter the process. If no units are found, the unit is taken from the queue and a departure event is created. If there are no units waiting the process is set to idle.

The engine would continue its execution until a termination event is encountered on the event list. If no more runs are scheduled the termination phase begins. Here the engine calculates the collected statistics and shows the user a report with the simulation results.

Statistics

The collection of statistics is a fundamental part of the simulation process; therefore it is necessary to specify what kind of statistics would be necessary to gather during the simulation execution in order to later present valid results to the user.

The choice of the statistics is based solely on what, in my opinion, would be relevant for the user. Sources of inspiration where found in the literature, but the main ideas are based on the events that are likely to occur in a typical simulation. Such events include entity creation/destruction, entity entrance/exit to and from queues, and the occurrence of idle/busy states within process.

Based on these events, the following questions should be answered after performing a simulation:

- What is the throughput of the system?
- What is the time required for throughput of the system?
- How efficient is the processes?
- How many entities were created? How many of them finish the flow?
- How long does it take to wait to enter a process?
- How long does a process take to service an entity?
- How long does each entity spend in queues?

Based on the above questions the following statistics would be gathered:

- Average time on the system for an entity.
- Average time on queue for an entity. .
- Average processing time for an entity.
- Average idle time for a process.
- Average busy time for a process.
- Average queue time for a process.
- Average number of processed entities on a process.
- Average Number of created entities.
- Average Number of disposed entities.

The way in which this statistics are calculated is described in the section of this paper that deals with logical design of the system.

Finally, the following flow chart depicts an overview of the different phases in the execution of a simulation:

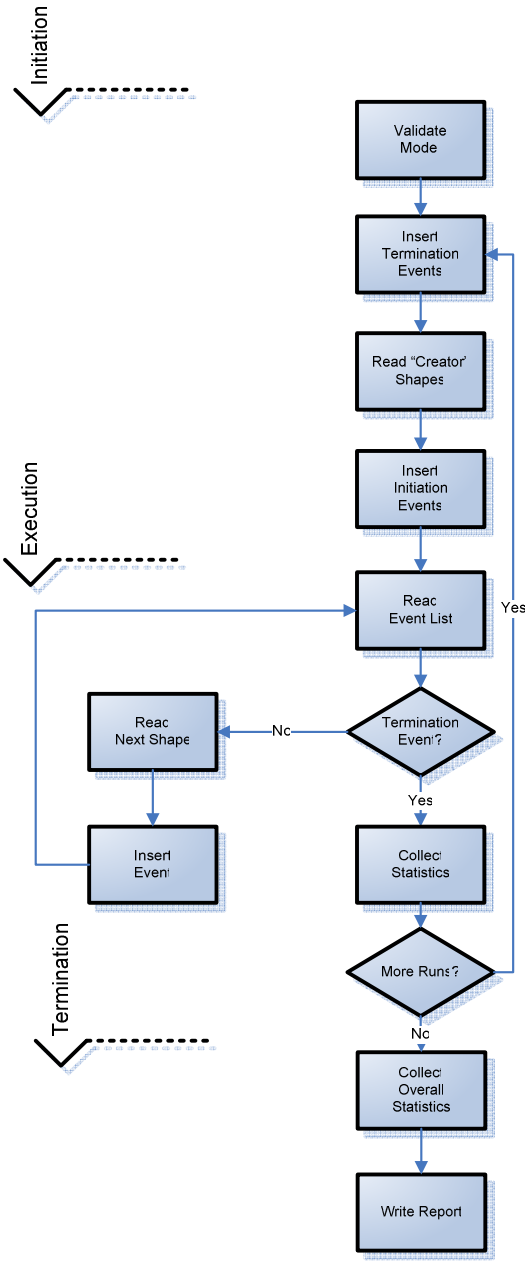


Figure 11 Simulation execution steps

3.3. Summary

After reading the above sections, the reader should have an idea of the overall design concepts for the Navision simulator. In short, the tool should allow two users to work in parallel: one focus on the modeling, and the other on data analysis. The data analyst would specify which of the table and fields the data that are to be analyzed reside in Navision, and then perform goodness-of-fit tests to determine whether there are distributions that fit the given data in a satisfactory manner. If the data does not exist in Navision, it must be imported for analysis.

Once estimators have been created, they can be inserted into an existing model making it ready for simulation execution. When the execution is finished, the user is presented with results based on statistics gathered during the execution. The statistics are based on the different events that happened during the simulation execution.

The overview of the components of the simulator and the interaction between them are shown in Figure 12

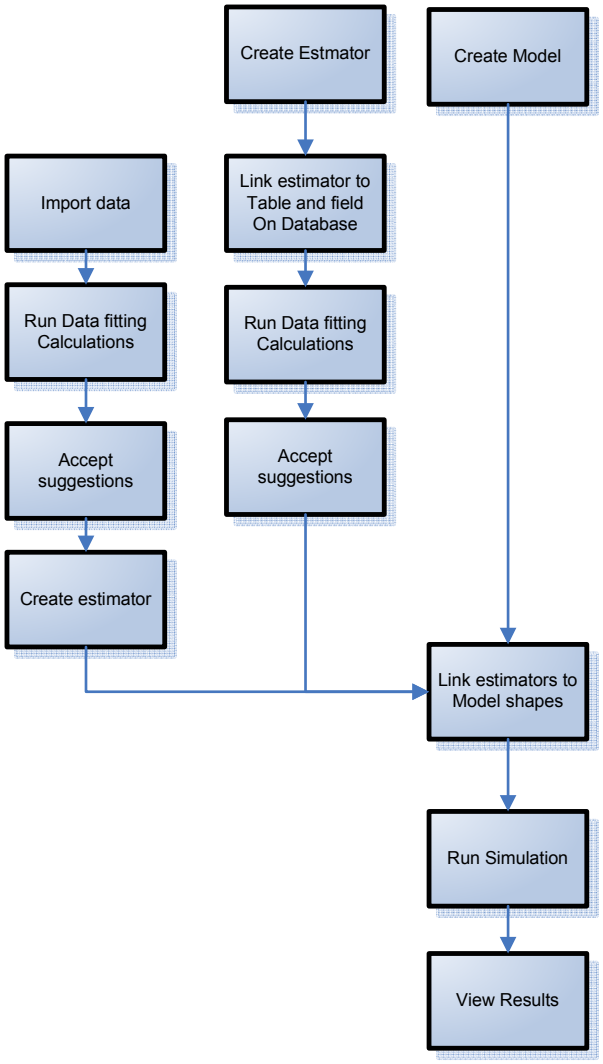


Figure 12 Summary of the simulation steps

4. Logical Design

This chapter presents the description of the different elements that compose the simulation application. An overview of the system's components is depicted in the following figure:

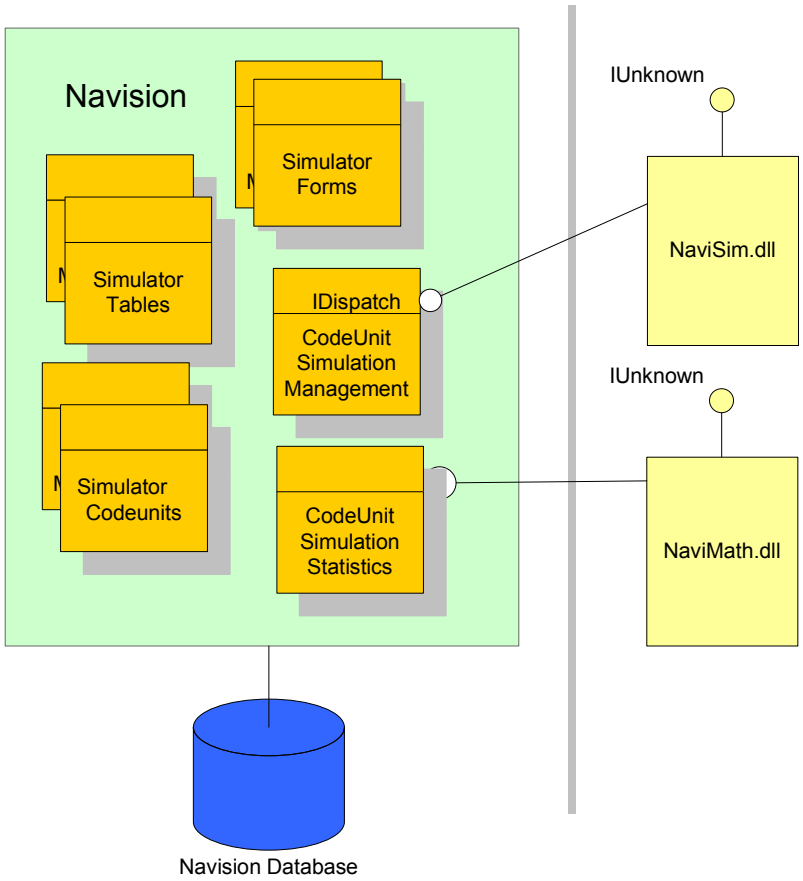


Figure 13 Simulator object overview

The application would be composed of a set of tables, forms, and codeunits in the Navision side. These codeunits use external components (NaviSim and NaviMath DLLs) that deliver missing Navision functionality. As described in section 1.3, Navision does not provide the necessary tools to create drawings or compute complex mathematical functions. The drawing and mathematical functions must be handled by the NaviSim and NaviMath DLLs, respectively.

This chapter is divided in two sections. The first one described the design of the NaviMath and NaviSim components and the second one delves into the description of the Navision application objects.

4.1. NaviMath and NaviSim automation servers

This section describes the interfaces and classes of the NaviMath and NaviSim .NET assemblies.

4.1.1. NaviMath.dll

The following figure depicts the structure of the NaviMath assembly:

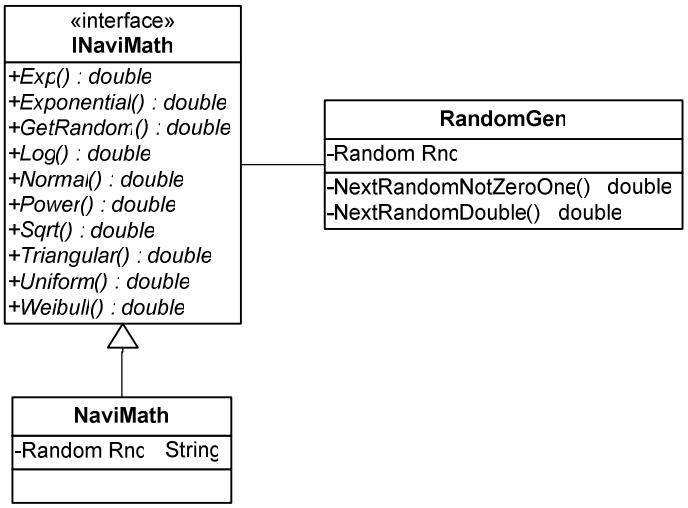


Figure 14 NaviMath component object overview

As it can be observed in figure 14 the NaviMath assembly basically is a wrapper class of mathematical functions and random number variates to be use by the Navision Client. The following table describes in greater detail the classes and methods that are part of the NaviMath assembly:

Object Name	Method	Description
Class NaviMath		The class that expose the methods to Navision
	Exponential(double beta)	Returns a random variate based on the exponential distribution.
	Normal(double mu, double sigma)	Returns a random variate based on the Normal distribution.
	Uniform(double a, double b)	Returns a random variate based on the Uniform distribution.
	Weibull(double alpha,double beta)	Returns a random variate based on the Weibull distribution.
	Exp(double beta)	Return the exponent of a number.

	Log(double number)	Returns the natural logarithm of a number.
	Power(double number,double power)	Returns a number raised to the power of another number
	Sqrt(double number)	Returns the square root of a number
	GetRandom	Returns a random number between 0 and 1. This can be use for Navision if it needs a decimal random number generator.
Interface INaviMath		The interface that is needed for COM interoperability. It is the interface that the NaviMath class inherits therefore it has the same methods.
Class RandomGen		The class that handles the random number generation.
	NextRandomDouble	Returns a random variable with range from [0, l].
	NextRandomNotZeroOne	Returns a random variable with range]0,1[

Table 12 Description of the NaviMath classes and methods

The implementation of the random variates for the different distributions is according with the algorithms described in section 2.4.

4.1.2. NaviSim.dll

The purpose of this component is to make possible the creation of drawings for the modeling process of simulation. This component uses the Visio Active X control[1]. that offers the full functionality of the Visio drawing application and exposes all its member objects and events. By using this control the Navision user would be able to draw the model in Visio from within the Navision client. Later the information of the model can be imported back into the database in order to run the simulation.

The following design has been based on the information found on the Visio Software Development Kit[2]. The structure of the NaviSim assembly is depicted in figure 15.

[1]For further reference see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odc_vis2003_ta/html/odc_vsprogrammingwithvisioactivexcontrol.asp

[2]For further reference visit: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vissdk11/html/Visio2003SDK_ReleaseNotes_HV01083290.asp

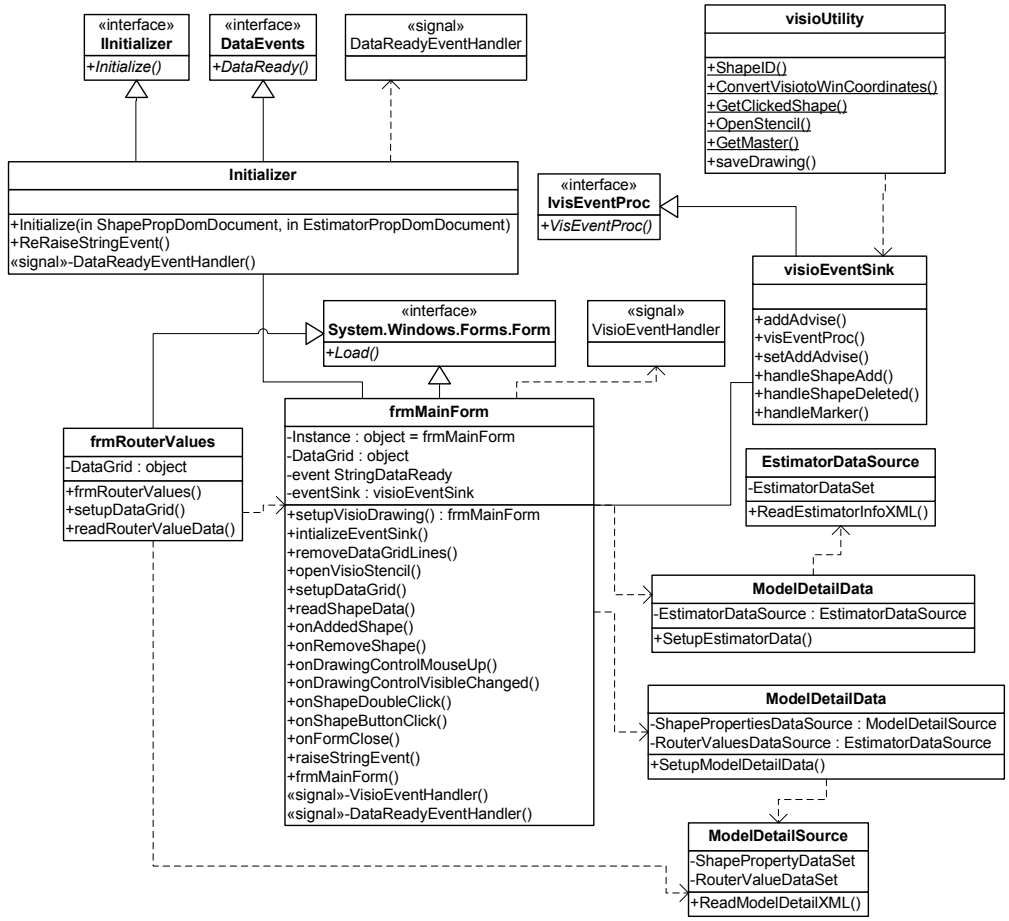


Figure 15 NaviSim.dll assembly

The Initializer class will be the interaction point with the Navision application. Navision will call the initializer method by sending two XML documents (the form of this documents is described in section 4.2.2) this method will import the xml documents and call the MainForm class where the Visio Active X component is embedded. When the user closes the form the onFormClose method is called, this method triggers the SaveDrawing method from the visioUtility class that raises the raiseStringEvent Method on the MainForm, this methods calls the DataReady delegate which notifys the ReRaiseStringEvent method on the Initializer class. Finally, this method raises an event in Navision and sends the ModelDetail XML for storage.

The rest of the classes in the system are aimed at managing the XML data coming from Navision and linking the events raised by the Visio control to the modeling form. A detail description of the classes and its methods are given in the table below.

It is important to mention that due to time limitations the implementation of this functionality was not performed. Nonetheless, this functionality only covers the drawing capabilities of the modeling phase and can be overcome in a different way inside Navision by using the Simulation Model Details table, the description of this table and its functionality is described in the following sections.

Object Name	Method	Description
Interface IInitializer		The interface that is needed for COM interoperability. It is the interface that the Initializer class inherits therefore it has the same methods.
Class Initializer		The Class that is visible to Navision.
	Initialize (XMLDocument ShapeXml, DOMDocument Estimator Xml)	This method is called from Navision. and imports the received XmlDocuments and loads the MainForm
	ReRaiseStringEvent (DOMDocument ModelDetails)	This is the event that is raised to Navision when sending model data
Delegate DataEvents	DataReady(DOMDocument ModelDetails)	The delegate to use for com interoperability with navision
Delegate DataReadyEventHandler		The delegate for the event that will send data to Navision
frmMainForm		The form where the Visio control would reside. Is the main interaction point to the user.
	setupVisioDrawing	This method initializes the Visio drawing that is displayed in the drawing control
	initializeEventSink	This method initializes the event sink on the document.
	removeDatagridLines	This method takes care of removing the data from the datagrid when a shape has been deleted.
	setupDataGrid	This method sets up the data grid by adding the columns and linking the grid to the data. If a previous state of the grid has been saved this state is restored
	readShapeData	This method reads the shape data from the ModelDetail xml into a data set that will be used while the

		application is running. After the data set is populated, the data grid on the form is set up and populated
	onAddedShape	This method handles the shape added event that is raised by the EventSink class. It adds the information associated with the new shape to the information on the form
	onRemoveShape	This method handles the shape delete event that is raised by the EventSink class. It removes the information on the form that is from the specified shape
	onDrawingControlMouseUp	This method handles the event that is raised when the user clicks on the drawing control. If the end user clicked the right mouse button on a shape the shortcut menu is displayed at the click location
	onDrawingControlVisibleChange	This method handles the event that is raised when the Visio ActiveX control has been fully initialized. This handler is an alternative to the form-load event for initializing the ActiveX control
	onShapeDoubleClick	This method handles the marker event that is raised when the user double-clicks on a shape from the sample office furniture stencil
	onShapeButtonClick	This method handles the event that is raised when one of the shape buttons is clicked. The button text field contains the corresponding name of the master shape on the product stencil
	onFormClose	This method is called when the form closes, it will trigger the event to send data to Navision.
	raiseStringEvent	This method is used to raise the event to Navision.
	frmMainForm	The constructor of the form.
frmRouterValues		This is the form that holds the information about the router values
	frmRouterValues	The constructor of the form, Is initialized with the shape id in order to obtain the shape information
	setupDataGrid	This method sets up the data grid by adding the columns and linking the grid to the data. If a previous state of the grid has been saved this state is restored
	readRouterValueData	This method reads the shape data

		from the shape xml into a data set that will be used while the application is running. After the data set is populated, the data grid on the form is set up and populated
ModelDetailData		This class handles the shape properties and router values information.
	setupModelDetailData	This method triggers the reading of the RouterValues and the shape properties data from the ModelDetail XML into data sets that will be used while the application is running
ModelDetailSource		This Class is used to read the ModelDetail XML.
	ReadModelDetailXML	This method reads the shape properties and the router values from the xml file into the shape properties and router values storage. These data is stored in DataSets while the application is running to simplify access and improve performance.
EstimatorData		This class handles the reading of the EstimatorInfo XML
	setupModelDetailData	This method triggers the reading of the estimatorinfo XML into a data set that will be used while the application is running
EstimatorDataSource		This class is used to read the estimatorinfoXML
	ReadEstimatorInfoXML	This method reads Estimatorinfo XML file into the estimator storage. This data is stored in a DataSet while the application is running to simplify access and improve performance.
visioUtility		This Class contains various utility methods that are used in conjunction with the events raised by the Visio active x.
	ShapeID	The ShapeID method reads the shape ID from the given shape
	ConverVisioToWinCoordinates	This method converts Visio coordinates to Windows coordinates
	GetClickedShape	The GetClickedShape method finds a shape at the specified location within a default tolerance
	OpenStencil	The OpenStencil method opens the specified Visio document stencil
	GetMaster	The GetMaster method gets the master by name
	SaveDrawing	The SaveDrawing method prompts the user to save changes to the Visio

		document
visioEventSink		This class handles events from Visio which are specified in the AddAdvise method.
	addAdvise	AddAdvise method takes the Visio application and document and adds events for this solution to their event lists
	visEventProc	The VisEventProc function is called by Visio when an event which has been added to an events list collection has been raised. The events were added to the event lists in the SetAddAdvise method
	setAddAdvice	The SetAddAdvise method adds the events that need to be responded to with solution-specific behavior to the Visio objects. Marker events are handled to respond to shape double-click from shapes in the sample office furniture stencil.
	handleShapeAdd	The handleShapeAdd method is called when a shape is added. The shape exists, so the shape reference can be queued for use later. The actual processing of the add will be done when all Visio events have been processed.
	handleShapeDeleted	The handleShapeDelete method is called when a shape is being deleted.
	handleMarker	The handleMarker method is called when Visio raises a marker event. When the user double-clicks on a shape the formula in the shape's EventDblClick cell will run the QueueMarkerEvent addon which will raise a marker event

Table 13 NaviSim classes

4.2. Navision Objects

Table 14 describes the objects that are part of the Navision simulator.

Object Type	Object Name	Description
Table	Simulation Model	Table that will encapsulate all the information relevant to a simulation model.
Table	Simulation Estimator	Table that holds the information related to a random variable that could be use by a simulation model.
Table	Simulation Model Estimators	Table that holds the estimators that are part of a simulation model.
Table	Simulation Model Detail	Table that holds the information of the simulation model details imported from the drawing made in the NaviSim.dll.
Table	Sim. Model Router Values	Table that hold the information relevant to a Router shape type that is part of a model.
Table	Simulation Event List	Temporary table that is use as the simulation event list of a simulation run.
Table	Shape Queue	Table that holds the information of the elements that are waiting to be processed by a shape.
Table	Entity Statitics	The statistics relevant to a single member that is part of a simulation.
Table	Data Fitting Worksheet Line	Table that would be use as a repository of imported data.
Table	Temp data Calculation	Temporary table that would be use to calculate data that correspond to an estimator. The purpose of this table is to ensure a better performance since it could happen that the relevant indicator does not contain an index or the data in the database is too large.
Table	Temp Time Data Calculation	Temporary table that is used to calculate time intervals when an estimator has been set up to be of that type.
Table	Estimator Frequency Table	Table that holds the information relevant to the calculations of a frequency table, in order for later to do data fitting calculations.
Table	Estimator Fitted Dist. Parameters	Table that holds the information of the Fitted distribution parameters and a lookup to the test results.
Table	Data Fitting Test Result	Table that holds the information regarding the results of the goodness of fit test performed on the data
Table	Simulation Model Result Detail	Table that holds the information of the gathered statistics from a simulation run
Table	Model Shape In Connection	Table that holds the information of the shape's connections
Table	Simulation Model Result	Table that consolidates the results from the simulation model result detail table
Form	Simulation Model	View of the simulation model table
Form	Simulation Model Subform	View of the simulation estimator table
Form	Simulation Estimator Card	View of the simulation estimator table
Form	Simulation Estimator Worksheet	View of the simulation estimator table. This form would offer the data fitting functionality. The user could select the desire ones and the system would perform a batch calculation of them.
Form	Simulation Estimators List	List view for the simulation estimator table.
Form	Simulation Model Details	View for the simulation model details.
Form	Data Fitting Worksheet	View for the data fitting worksheet line. This form would offer the functionality to calculate data fitting calculations

		on external data.
Form	Frequency Table	View for the Estimator Frequency table.
Form	Estimator Dist. Parameters	View for the Esimator Dist. Parameters table.
Codeunit	SimulatorManagement	Unit of code that holds the functionality in charge of calling and manage the data flow between Navision and the NaviSim.dll
Codeunit	SimulatorEngine	Unit of code in charge of running a simulation.
Codeunit	SimStatisticsMgt	Unit of code in charge of the data fitting and calling the NaviMath.dll.
DataPort	Import DataFitting Data	Object in charge of importing external data to the Data fitting worksheet line table.

Table 14 List of the Navision objects that are a part of the simulator

Figure 16 depicts an overview of the relationship between the tables of the simulator

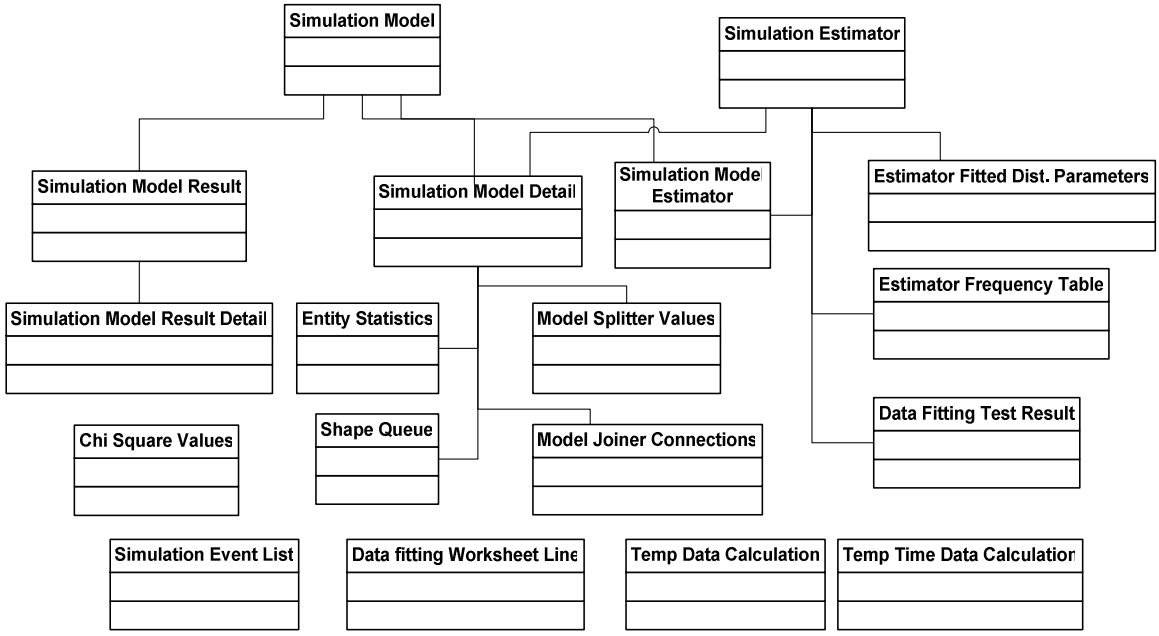


Figure 16 Simulator Tables relationships in Navision

It can be observed from figure 16 that there are some pillar tables which form a basic structure in order to run a simulation model. These tables are: Simulation Model, Simulation Model Detail and the Simulation Event List.

The Simulation Model table is the one that holds the data relevant to one model. When the user creates the model, the details are later transported to the Simulation Model Detail table. This table is also used when the simulation is executed since it keeps track of property values for some shapes.

There are four types of tables that do not have any physical relation to other tables in the system; these are: the Simulation Event List, Data Fitting Worksheet Line, Temp Data Calculation, and Temp Time Data Calculation tables.

The Simulation Event List table is used when running the execution of a simulation , it keep track of the different events created. The Data fitting Worksheet Line table is used to analyze data external to the Navision database. The Temp Data Calculation table is used when calculating the parameters and statistical tests for an estimator. The Temp Time Data Calculation is used to calculate the time calculations for an interval type estimator; finally, the Chi Square Values table holds the critical values for the Chi Square distribution to be use in the goodness-of-fit test. All of these tables are further described in the next sections.

The functionality will be divided in three areas in order to explain the relations and operations more easily, these are:

a) Data Fitting
b) Simulation Model Creation
c) Simulation Model Execution.

4.2.1. Data Fitting

The data fitting area consist of the following objects:

Object Type	Object Name
Table	Simulation Estimator
Table	Estimator Frequency Table
Table	Estimator Dist. Parameters
Table	Data Fitting Worksheet Line
Table	Temp Data Calculation
Table	Data Fitting Test Results
Table	Temp Time Data Calculation
Table	Table Filter
Table	Chi Square values
Form	Simulation Estimator Card
Form	Simulation Estimator Worksheet
Form	Data Fitting Worksheet
Form	Estimator Dist. Parameters
Form	Frequency Table
Form	Data Fitting Test Results
Form	Table Filters
Codeunit	SimStatisticsMgt
DataPort	Import Data Fitting Data
Report	Fit Simulation Distributions

Table 15 Objects of the data fitting functionality

The main purpose of this functionality is to allow the user to analyze existing or imported data to find a suited distribution that can be use in a simulation model. The information collected is gathered around the simulation estimator table. The fields that are part of the estimator table are shown in the following table:

Field No.	Field Name	Data Type	Length	Primary Key	Description
1	Code	Code	10	Yes	The Identifier of the estimator
2	Table No.	Integer			The table Number that the estimator would be based on.
3	Field No.	Integer			The field number that the estimator would be based on if the (type is normal)
6	Filters	Text	250		The filters that will be used on this field
7	Start Time Field No.	Integer			The start time field of the interval
8	End Time Field No.	Integer			The end time field of the interval
9	Start Date Field No.	Integer			The Start date field of the interval
10	End Date Field No.	Integer			The end date field of the interval
17	Description	Text	80		The description of the estimator
18	Type	Option			The type of the estimator. The options are: Deterministic,Table-Field, Interval.

21	Table Name	Text	30		The table name of the selected table no.
22	Field Name	Text	30		The field name of the selected field no.
25	Start Interval Date	Date			The start date of the interval
26	End Interval Date	Date			The end date of the interval
29	Start Date Field Name	Text	30		The name of the start date field
30	End Date Field Name	Text	30		The name of the end date field name
31	Start Time Field Name	Text	30		The name of the start time field
32	End Time Field Name	Text	30		The name of the end time field
38	Selecte	Boolean			Field use in the worksheet to select the estimators
39	Status	Option			Field use in the worksheet
40	Fitted Distribution Exists	Boolean			Shows if there are fitted distribution calculations. This field is a lookup to the Simulation Fitted Distributions
41	Interval Calculation Type	Option			The type of interval, options are: Date Occurrence, Date Duration, Time Duration.
42	Value	Decimal			The value of the estimator in case it is of type deterministic
43	Shape Type	Option			The type of shape this estimator would be use with. Options are creator and process shapes.
44	Entities created at a time	Integer			The entities that would be created at a time in case of a creator type shape.
45	Using Temp Tables	Boolean			This field is set when a user is performing data fitting calculations. In this ways if he would like to calculate data for another estimators then the data would be re-imported.

Table 16 Simulation Estimator table fields

There are two ways to create an estimator: from the Estimator and Data Fitting Worksheet forms. The estimator form is aimed at analyzing data that is present on the Navision tables. The Data fitting worksheet form is use to make possible the analysis of data external to the ERP system. The functionality related to these two form is described in the following sections.

Fitting Data from the Estimator Card

The following figure depicts the different steps that are part of the data fitting process based on the estimator card:

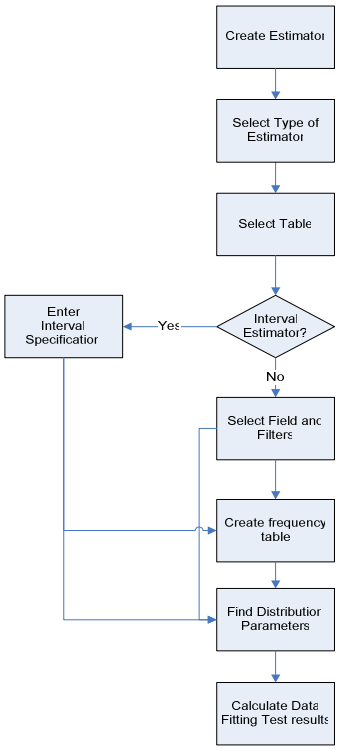


Figure 17 Fitting data from the estimator card

As shown in figure 17 the first step is to create an estimator from the Simulation Estimator Card, the form is shown in the following figures:

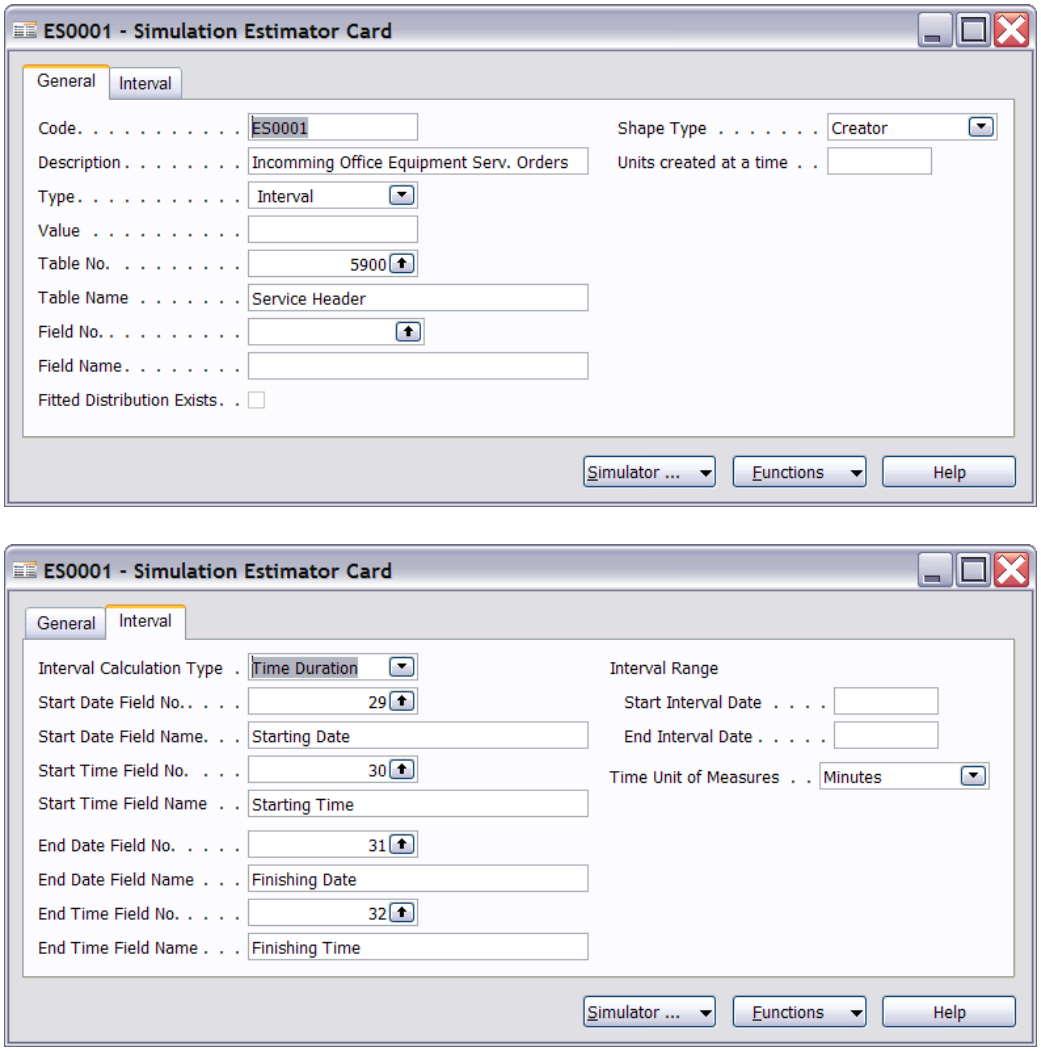


Figure 18 the Simulator Estimator Card

Later the user has to choose the type of estimator to be created (this is done by selecting a value from the *Type* field) and the table the estimator would use to retrieve the values for calculation. Since there is the need to filter a table in order to retrieve the right records, the user can specify filtering possibilities by using the Estimator Table Filters form. The form and the table fields are shown below.

Field No.	Field Name	Data Type	Length	Description
1	Estimator Code	Code	20	The code of the estimator this table relates to
2	Table No.	Integer		The table no the filters are going to be applied to
3	Field No.	Integer		The field that is being filtered
4	Field Name	Text	250	The name of the field
5	Filter	Text	250	The filter value

Table 17 Estimator Table Filter

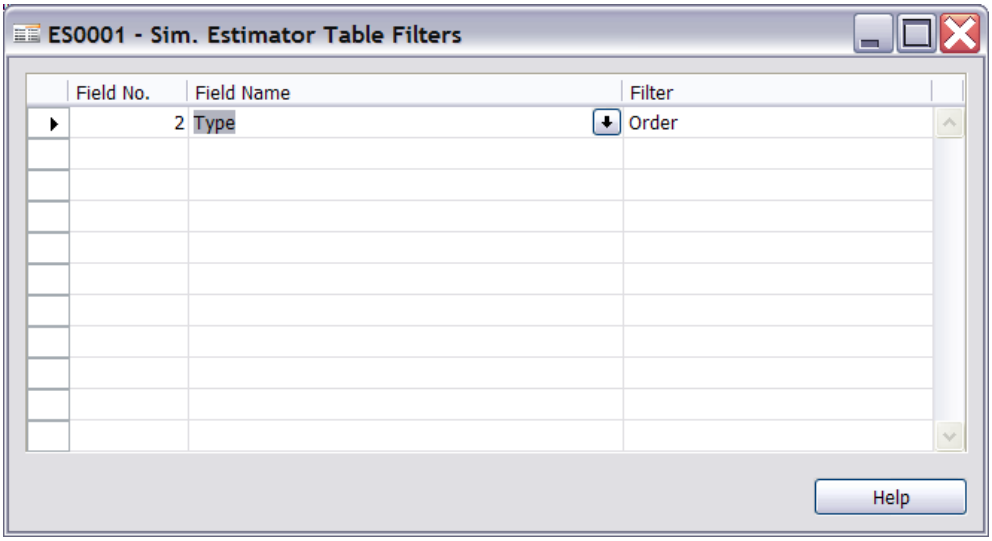


Figure 19 Sim. Estimator Table Filters form

When selecting the type of estimator there are values to choose from: Deterministic, Table-Field and Interval. A **Deterministic** estimator is the one that is not based in any table and has a constant value; this type would be used if a process does not present any random behavior or for testing purposes in a model. A **Table-Field** estimator is the one that is based on a single field on a table (the data to base the calculations on are the different field values). Since table fields can have different types of values e.g. text, date, time, binary, etc. only numeric field types (Integer, BigInteger and Decimal) would be allowed.

An **Interval** estimator is concerned with counting the occurrences or duration of an event in a defined period of time. When this type is chosen the user should defined what kind of calculation is going to be performed. This is done using the *Interval Calculation Type* field. There are three options to choose from: date occurrence, time occurrence, date duration and Time duration.

The **date occurrence** option deals with how often an event occurs (as mention in section 4.1), for instance it can be used to answer questions like: how often does a sales order arrives in a month, or a production order is created. To make these calculations, the user has to select the relevant table and date type field which the system would use to retrieve the data and perform the computations. The selection of the table and date fields is done using the *Table No.* and *Start Date Field No.* fields. An example of the fields to use is shown in table 10. The **time duration** option operations are analogous to the date duration ones, the only addition is of an extra field (the *Start Time Field No.* field*)* to be specified only the mentioned fields are taken into consideration for the calculations.

Field Name	Field Value
Table No.	36
Table Name	Sales Header
Start Date Field No.	23
Start Date Field Name	Document Date
Start Time Field No.	12
Start Time Field Name	Time Created

Table 18 Time calculation example data

The **date duration** option deals with how long it takes to accomplish a task. For this the user has to specify the fields that would be use to calculate the interval length value, this is done using the *Start Date Field No.* and the *End Date Field No.* fields only these fields are taken into consideration for the calculations. For instance, if a user would like to get an estimation on the time it took to handle a sales order, he would have to specify an starting and ending date fields - an example of the fields values is shown in table 16. This option, as the name suggest, deals only with date durations. If a user would like to work with time calculations then he would have to use the time duration option and specify the *Start Time Field No.* and *End Time Field No.* additionally.

So far we have explained the fields that are going to be use and the type of calculation to perform. One missing step is the frame of reference to perform the

calculations. For example, if the user would like to see how often a sales order was created in February 2004, he has to specify, as we mentioned earlier, the table and field that has this information and the frame of reference to perform the calculations- in this case February 2004. This is done by using the *Start Interval Date* and *End Interval Date* fields. To continue with the example, the user would choose the values 01-02-2004 and 28-02-2004, respectively.

Finally, the user has the option to specify what unit of measure the interval estimator would have. The options are: Year, Month, Day, Hour, Minute, Second.

As an example, let us assume that a user would like to calculate and estimator for the number of days that occurs between Service Orders in the period from the 01-01-2001 and 30-01-2001. For this, the user would have to identify the table that holds this information. In the system he finds out that it is the "Service Header" table. Later, the user would need to determine the field to use for the calculation; by further inspection the user realizes that the "Order Date" field is the appropriate one. With this information, the user is ready to create an estimator and should perform the following steps:

1. Enter an estimator Code and an appropriate description
2. Set the type as Interval
3. Set the Interval Calculation type as Date occurrence
4. Set the table no equal to the one from the "Service Header" table.
5. On the interval Tab, set the "Start Interval Date" and "End Interval Date" as 01-01-2001 and 30-01-2001 respectively.
6. Set the "Star Date Field No." equal to the number of the "Order Date" field.
7. To display the results to in units of days, set the *Time Unit of Measure* field to Day.

Auxiliary Tables

When calculating the frequency table, running the goodness-of-fit tests or performing the interval operations described above, it is needed to sort the data in some way –commonly in ascending order. Since the user has the possibility of selecting any field that is part of a table, there is no assurance that the table selected would have an index that would easily allow the data to be sorted in an appropriate manner. Therefore, it is needed to copy the data of the selected field(s) of a table to a temporary table in order to sort the data and perform the needed calculations. For the data fitting and interval-occurrence calculations the Temp Data Calculation and

Temp Time Data Calculation would be use accordingly. The fields and indexes of each of these tables are shown in the following tables:

Field No.	Field Name	Data Type	Description
1	Line No.	Integer	Primary key
2	Value	Decimal	The value stores

Table 19 Temp Data Calculation Fields

Field No.	Field Name	Data Type	Description
1	Line No.	Integer	Primary key field
2	Start Date	Date	Index Field
3	Start time	Time	Index Field

Table 20 Temp Time Data Calculation fields

An important observation that has to be made is that these tables are not linked to any other tables in the system; this means that there are only relevant to the current estimator that is being worked at the moment. This is done since copying existing data to a temp table should be done on a case by case basis, otherwise the system would be filled with duplicate and unnecessary data. The field that is used to keep track of which estimator is currently using these temp tables is the *Lock Temp Table* field from the Simulation Estimator table

On Table-Field type of estimators the data would be directly copied to the Temp Data Calculation table. For interval estimators the data would processed first and later inserted into the Temp Data Calculation table.

For interval-duration estimators, the system would find the records in the table specified in the *Table No.* field and calculate the duration according to the following guidelines:

Case 1: Date Duration

1. Calculate the days based on the values from the *End Date Field No. – Start Date Field No.* Convert the results according to the value of the *Time Unit of Measure* field.
2. Insert this result into the Temp Data Calculation Table.

Case 2: Time Duration

1. Calculate the days based on the values from the *End Date Field No. – Start Date Field No.* Convert the results according to the value of the *Time Unit of Measure* field.
2. Calculate the days based on the values from the *End Time Field No. – Start Time Field No.* Convert the results according to the value of the *Time Unit of Measure* field.
3. Sum the results from steps 1 and 2 and insert this result into the Temp Data Calculation Table.

In the case of an interval-occurrence estimator the data would be first copied to the Temp Time Data Calculation table and later to the Temp Data Calculation table after running the occurrence interval algorithm. This algorithm is further described below.

Occurrence interval algorithm

For the occurrence interval algorithm to function the data has to be ordered in ascending order. Since there are two types of occurrence calculations the algorithm is divided in two cases, the steps corresponding to each case are described below.

Case 1: Date Occurrence calculation type:

1. Find the first record from the Temp Time Calculation Table.
2. Calculate *Start Date(Temp table) - Start Interval Date(estimator table)*. Convert the days to the value specified on the *Time Unit of Measure* field from the estimator table and insert this record in the Temp Data Calculation table.
3. Let AuxDate = *Start Date* on the record just found on the temp table.
4. Move to the next record of the temp table and let NewDate = *Start Date* field of the new record found.
5. Calculate NewDate – AuxDate, convert the result as in step 1 and insert it on the temporary table.
6. If there are more records to be examined go to step 2.

Case 2: Time Occurrence calculation type:

1. Find the first record from the Temp Time Calculation Table.
2. Calculate *Start Date(Temp table) - Start Interval Date(estimator table)*. Convert this value into seconds.
3. Calculate, *Start Time* - 00:00:00 to the time of the first occurrence in seconds.
4. Sum the results from 2 and 3 and convert the result to the value specified on the *Time Unit of Measure* field from the estimator table and insert this record in the Temp Data Calculation table.
5. Let, AuxDate = *Start Date* and Aux Time = *Start Time*.
6. Move to the next record from the temp table. Let NewDate = *Start Date* and NewTime = *Start Time.*
7. There are two cases:
    a) If AuxDate = NewDate then calculate the seconds NewTime – AuxTime.
    b) If AuxDate < NewDate then calculate the second from AuxTime to 00:00:00 and the seconds from 00:00:00 to NewTime, then sum the results.
8. Insert and convert the results from step 7 into the Temp Calculation Table as done for step 4.
9. Set, AuxDate = NewDate and AuxTime=NewTime
10. If there are more records to be found go to step 6.

Frequency Table

After the user has entered all the necessary information in the estimator card, the next step on a data analysis process is the creation of a frequency table. Even though this step is optional it would be commonly done since it aids the user to visualize the data and observe if the data resembles a known distribution. The methods used to create the frequency table are based on the methodology describe in A.M. Law, page 366 [1]. In broad terms, the creation of a frequency table consist in grouping the data into intervals and count how many data points are included in it(the frequency). The number of intervals would be user defined but commonly this value ranges from 5 to 15[2]. The user would access this functionality by running the Create Frequency Table report, the report is shown in the following figure:

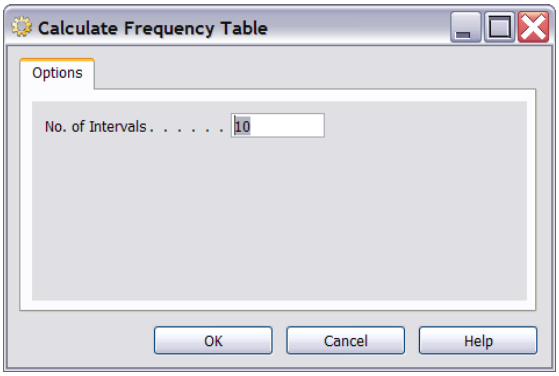


Figure 20 Create frequency table report

The user has to enter the amount of intervals the frequency table will be based on, this parameter is later passed to the CreateFreqTable function in codeunit SimStatisticsMgt which performs the calculations and inserts the results on the Estimator Frequency table. The calculations are based on the Temp Data Calculation table; if no data is present in this table then the system would populate the table based on the information from the estimator card. If the table contains data that does not belong to the estimator currently in use, then the existing data would be deleted and new records would be inserted based on the estimator selected at that moment. The user can view and plot the results in order to select the most adequate distribution when running the data fitting engine, the plotting of the results should be done in external software commonly a spreadsheet program with graph functionality. ´

The user can view the results in Navision by opening the Frequency table form, the form and the Frequency Table fields are shown below:

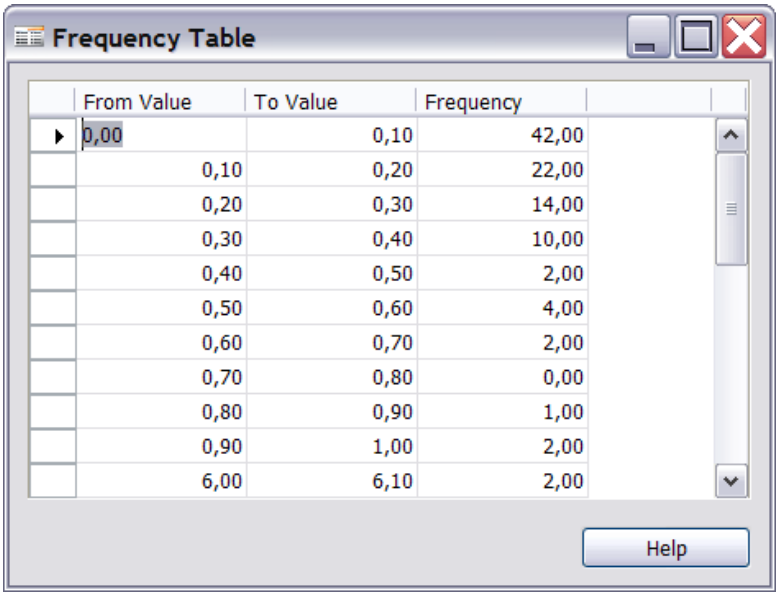


Figure 21 Frequency Table form

Field No.	Field Name	Data Type	Length	Description
1	Estimator Code	Code	20	The code of the estimator where the frequency table has been calculated for.
2	From Value	Decimal		The start of the interval
3	To Value	Decimal		The end of the interval
4	Frequency	Decimal		The frequency of the occurrences

Table 21 Frequency Table field description

Parameter estimation and Goodness-of-fit test calculation

After the user has created the frequency table, he/she would probably have an idea of which distributions to calculate parameters for. The selection of distributions is done on the Fit Simulation Distributions report as seen in the figure 21:

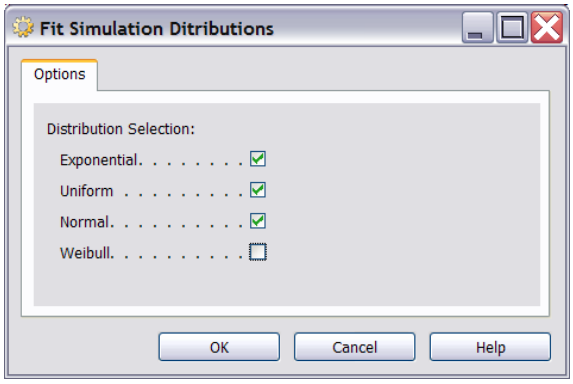


Figure 22 Fit Simulation distributions report

Here the user selects the distributions and starts the calculations. The report will call the function FitDistributions(EstimatorNo,Expo,Uniform,Normal,Weibull) from codeunit SimStatisticsMgt This function fits the data to a selected distribution based on the input parameters and also calculates the Chi-Square and Kolmogorov-Smirnov goodness-of-fit tests.

The first thing the codeunit does is to check what kind of estimator the calculation is going to be done for. For occurrence-interval estimators the system would copy the data from the table and fields specified in the estimator card into the Temp Time Data Calculation table. For the other type of estimators (excluding the deterministic type) the data from the table and fields specified would be copied to the Temp Data Calculation table. Later, the system would calculate the estimators for all the distributions specified in the report (see figure 17) based on the formulas from section 3.4 and insert the results into the Estimator Fitted. Dist. Params. table.

The fields belonging to this table are shown in the following table:

Field No.	Field Name	Data Type	Length	Description
1	Estimator Code	Code	20	The code of the estimator
2	Distribution Name	Option		The name of the distribution the options are: ,Exponential, Uniform, Normal, Weibull and Triangular
3	Parmeter 1 Name	Option		The Parameter of the relevant fitted distribution the options are: ,Alpha,Beta,Mu,a
4	Parameter 1 Value	Decimal		The value of the parameter based on the parameter 1 field.
5	Parameter 2 Name	Option		The options are: ,Beta,b,SigmaSqr
6	Parameter 2 Value	Decimal		Same as field 3
7	Parameter 3 Name	Option		The options are: ,c
8	Parameter 3 Value	Decimal		Same as field 3
9	Chi Square test Approved	Boolean		A lookup field that checks if the Chi Square test was approved
10	K-S test Result Approved	Boolean		A lookup field that checks if the Kolmorov-Smirnov test was approved.
11	Default	Boolean		Indicates that this distribution is the detault one to be used in order to run the simulations.
12	Normal Generation Data type	Option		This field is used to allow the user to specify if the generated values in case of a normal distribution are going to be positive, negative or noth. he options are:Positive, Negative,Both.

Table 22 Fields from the Estimator Fitted Dist. Params. table

Afterwards, the system would calculate the Chi-Square and Kolmogorov-Smirnov tests. The methods to use are the ones described in Law and Kelton, pages 382-391. [1]

For the Chi-Square test is needed to calculate a frequency table based on the values from the Temp Data Calculation table. The lengths of the intervals have to be made according to the methods described in the section 3.4. After the frequency table is created the X^2 test statistic is calculated. When all the calculations have been performed the results are inserted in the Data Fitting Test Result Table.

The fields from the Data Fitting Test Result table are described in the following table:

Field No.	Field Name	Data Type	Description
1	Estimator Code	Code 20	The code that of the estimator that the calculations relate to
2	Distribution Name	Option	The name of the distribution the test was made for. The options are: ,Exponential,Uniform,Normal,Weibull
3	Test Name	Option	The name of the test carried out. The options are: Chi- Square, Kolmogorov-Smirnov
4	Test Statistic Name	Option	The name of the test statistic. The options are: X2,D
6	Test Statistic Value	Decimal	The value of the test statistic
7	Critical Value	Decimal	The critical value that the test statistics was compared to.
8	Result	Option	The results of the test: Approved, Failed

Table 23 Data Fitting Test Result fields

For the Kolmogorov-Smirnov test, no frequency table is needed. The system browses the Temp Data Calculation table and computes the *D* statistics depending on the distribution the test is applied to. The *D* statistic formulas are described in section 3.4. After all the calculations have been done the results are inserted in the Data Fitting Test Result Table.

The user can view the results the parameters estimation and the goodness-of-fit tests from the Estimator Fitted Dist. Params. and Data Fitting Test Results forms. These forms are depicted in the following figures:

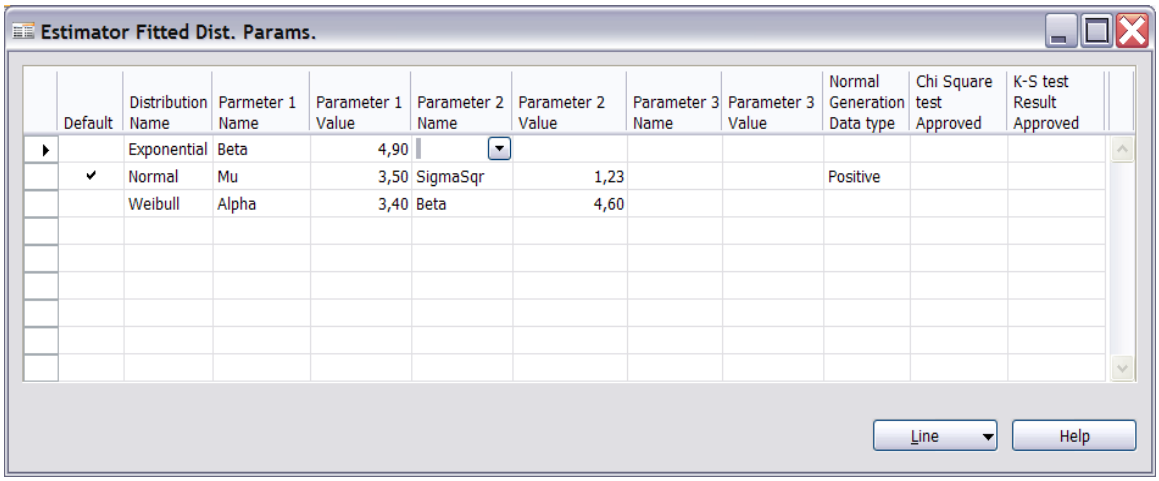


Figure 23 Estimator Fitted. Dist. Params. form

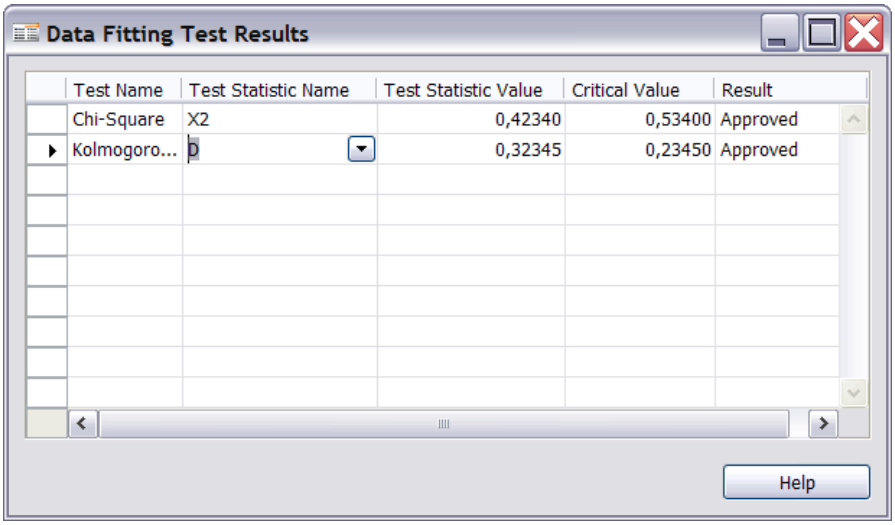


Figure 24 Data Fitting Test Results

A summary of the steps taken by the SimStatisticMgt codeunit are illustrated in the following figure:

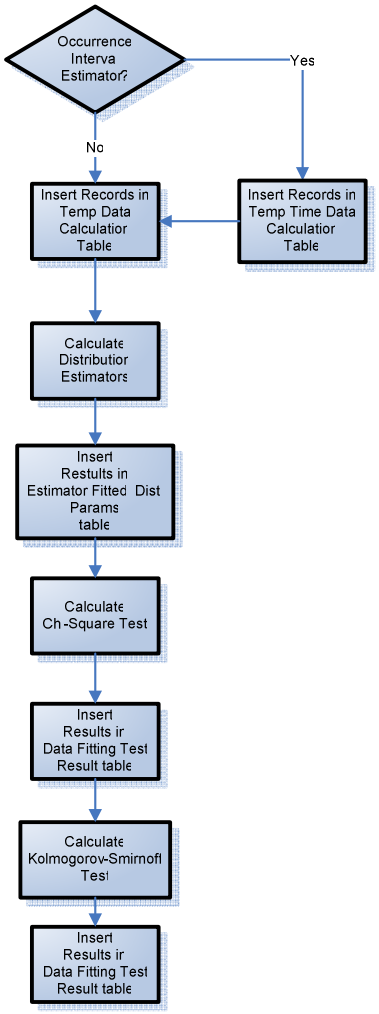


Figure 25 Step taken when calculating the data fitting operations

Finally, a user would probably create several estimators at a time; on some occasions, the user may want to perform the fitting calculations or have an overview of a selection of them. For these reasons, there is a need to have a task pane of some sort in order to allow the user to perform batch calculations and obtain an overview of the created estimators. The solution to this requirement can be found in the Simulation Estimator Worksheet Form. This form is simply a tabular view of the Simulation Estimator table where the user can maintain an overview and can perform the calculations for a selection of estimator.

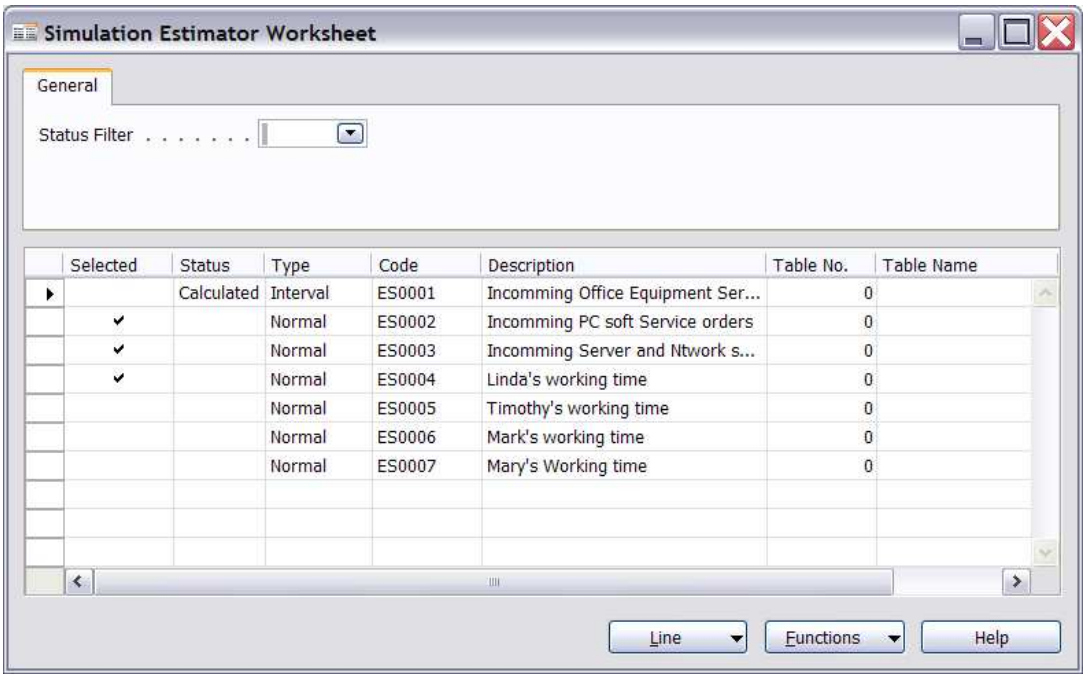


Figure 26 Simulation worksheet form

Fitting Data from the Worksheet Form

Since the ERP software does not contain all the data that a company handles, it is necessary to have a tool that allows user to import data in order to perform a data fitting analysis and create an estimator. The Data Fitting Worksheet would allow the user to perform these tasks. Here the user can import data into three columns and fit the imported data to later create an estimator. The worksheet is based on the Data Fitting Worksheet Line table, the form and the table are depicted and described below respectively.

Field No.	Field Name	Data Type	Length	Description
1	Line No.	Integer		Key of the table
2	Column 1	Decimal		A column for data
3	Column 2	Decimal		A column for data
4	Column 3	Decimal		A column for data

Table 24 Data Fitting Worsheet Line table fields

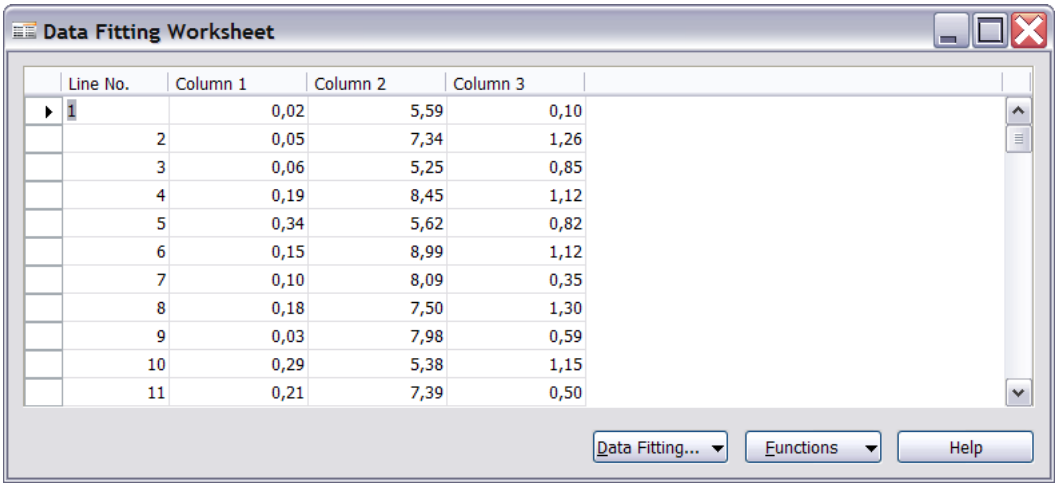


Figure 27 Data Fitting Worksheet

The functions present on this form are:

a) Create Estimator
b) Calculate Frequency Table
c) Run Data fitting Calculations
d) Import Data

Create Estimator

This function runs the Create Estimator report, this report first checks if there is any data on the Estimator Fitted Dist. Parameters table. If that is true it will create an estimator for the column selected by the user.

The Create Estimator report is shown in the following figure:

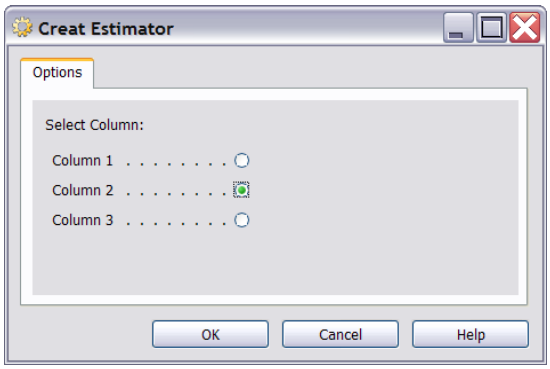


Figure 28 Create Estimator Report

Calculate Frequency Table

This function runs Calc. Freq. Table worksheet report. This report calculates a frequency table for a selected column. The functionality is analogous to the one from the estimator card.

The Calc. Freq. Table worksheet report is shown in the following figure:

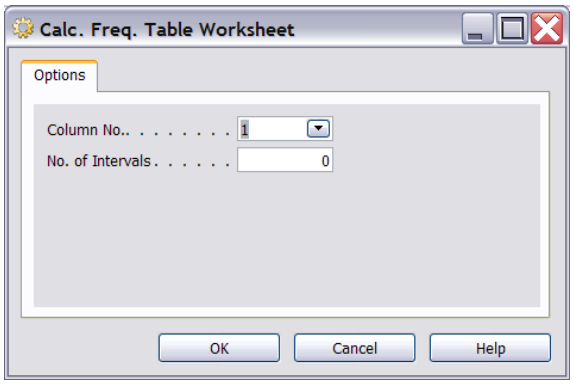


Figure 29 Calc. Freq. Table Worksheet report

Run Data Fitting Calculations

This function runs the "Fit Sim. Distributions Wrksht." report. Here the user selects the column that would be referenced to do the data fitting calculations. The functionality is analogous to the one from the estimator card.

The "Fit Sim. Distributions Wrksht." report is shown in the following figure:

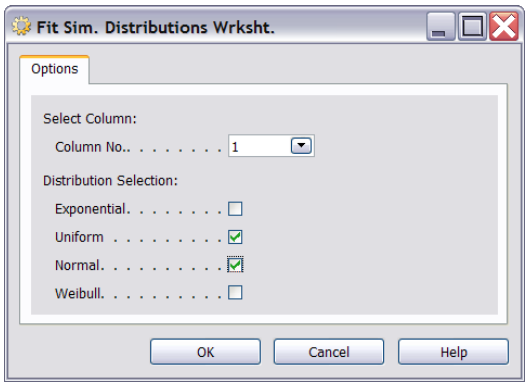


Figure 30 Fit Sim. Distributions Wrksht. report

Import Data

This functions runs the dataport Import Data to D.F. Worksheet. The report reads a tab separated text file with data to be imported to any referenced column. The column reference is done on the text file in order to support multi-column imports.

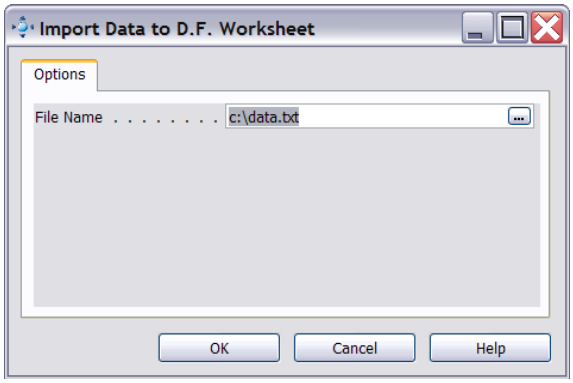


Figure 31 Import Data to D.F. Worksheet report

4.2.2. Simulation Model Creation

The simulation model creation area consists of the following objects:

Object Type	Object Name
Table	Simulation Model
Table	Simulation Model Estimators
Table	Simulation Model Detail
Table	Sim. Model Router Values
Table	Model Shape In Connection
Form	Simulation Model
Form	Simulation Model Subform
Form	Simulation Model Details
Codeunit	SimulationManagement

Table 25 Objects members of the simulation model creation functionality

Another step in the simulation process is the creation of the model to based the simulation on. As mentioned in section 4 there would probably be two actors working on simulation tasks, one of them being the Modeler. The modeler would create the model in the NaviSim.dll form and later execute the simulation. Before creating any drawings the actor has to create a simulation model in Navision. This is done on the Simulation Model form as shown below.

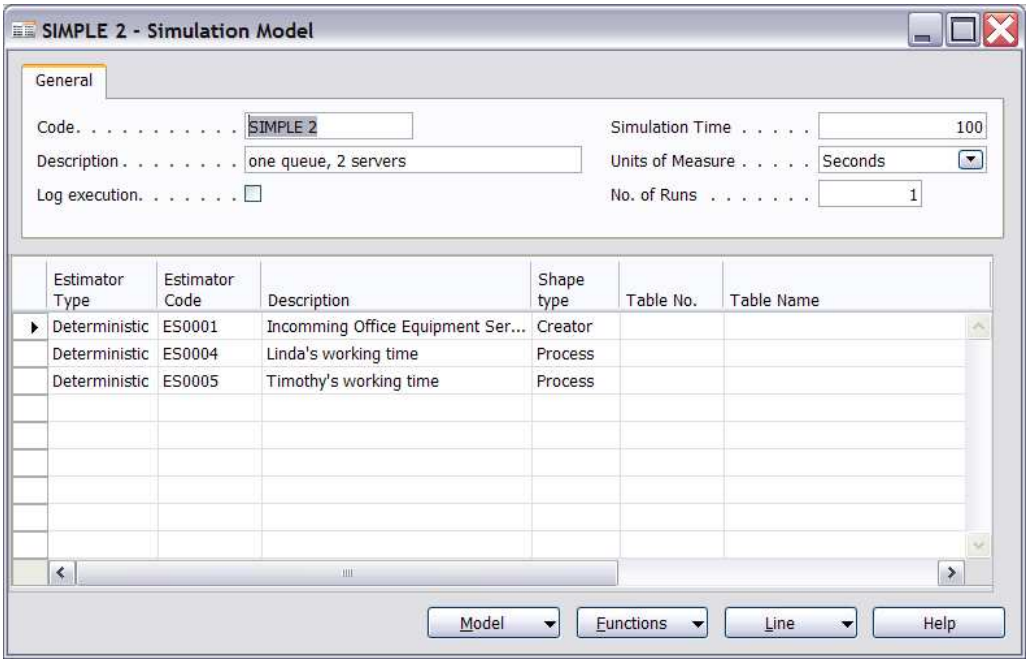


Figure 32 Simulation Model form

This form consists of two tables: the Simulation Model and the Simulation Model Estimator tables. The Simulation Model table will be use to set the general properties of the model, the number or runs that should be performed and the length of the simulation. The user can also specify the estimators that this model would use. These estimators are going to be use in the drawing so they are linked to some shapes in order to execute the simulation and obtain the information from the relevant estimator to generate random variates. It is worth mentioning that the model does not oblige the user to set up some estimators previous the creation of the model drawing, the estimators can be added or removed during the creation of a drawing. It would be the user responsibility to keep link a drawing shape to an estimator.

The tables that are part of the Simulation Model form are described below:

Field No.	Field Name	Data Type	Length	Description
1	Code	Code 20	20	The code of the model. This is the primary key field
2	Description	Text 80	80	The description of the model
4	Model	BLOB		A binary large object that holds the information of the model drawing
5	Simulation Time	BigInteger		The total length of the simulation
6	Units of Measure	Option		The unit of measure of the simulation time
7	Stop Based on Estimator Code	Code 20	20	A form of stopping a simulation based on the value of an estimator.
8	No. of Runs	Integer		The number of runs that the simulation would perform.
9	Stop After Created/Processed No.	Integer		The stopping criteria when the Stop Based on estimator code is filled.
10	Log execution	Boolean		Field that tells the simulation engine to log its execution steps.

Table 26 Simulation Model table fields

Field No.	Field Name	Data Type	Length	Description
1	Model Code	Code	20	The model code
2	Estimator Code	Code	10	The estimator code
3	Description	Text	80	The description of the estimator
4	Table No.	Integer		The table no. of the estimator
5	Table Name	Text	50	The table name of the estimator
6	Field No.	Integer		The field no. of the estimator in case it is of type Table-Field
7	Field Name	Text	50	The field name of the estimator
9	Value	Decimal		The value of the estimator in case it is deterministic.
11	Estimator Type	Option		The type of estimator.
12	Shape Type	Option		The shape type of the estimator

Table 27 Simulation Model Estimator table fields

The Simulation Model form allows the user to display the modeling drawing tool as well as running a simulation. To display the drawing tool he has to use the *Show Model* option from the *Model* menu button, as seen in figure 27. When this button is pushed, a call to the Simulation Management codeunit is executed.

The steps performed by this codeunit are as follows:

1. Check if a drawing was previously made, this is done by checking if the Model field is not empty.
2. If no drawing was made then call the NaviSim.Dll Initializer class with its Intialize method empty, insert an XML document with the model details otherwise and model estimators otherwise.
3. Since the NaviSim.dll has the DataReady events linked to Navision. Each time this events are triggered-which contains an XML document with the model details, the incoming XML would be stored in the *Model* binary large object field.

4. When receiving an XML document with the model details from the NaviSim.dll, this information will also be stored in the Simulation Model Details table. If some Router shape were used, this information will be stored in the Model Router Value table.

The structure of the XML documents that are send to and receive from the NaviSim.dll are shown below:

Model Detail XML , to send and receive:

```
<?xml version="1.0" encoding="utf-8" ?>
<ModelDetails>
 <Shape>
  <ShapeID>1</ShapeID>
  <ShapeName>Lindas process</ShapeName>
  <ShapeType>Router</ShapeType>
  <ConnectFromShapeID>0</ConnectFromShapeID>
  <ConnectToShapeID>1</ConnectToShapeID>
  <EstimatorCode>ES0001</EstimatorCode>
  <BasedonNextShapeQueue>No</BasedonNextShapeQueue>
  <ProbabilisticRouter>No</ProbabilisticRouter>
  <MinQueueValue>0</MinQueueValue>
  <RouterValues>
   <Line ConnecttoShapeID="2" Value="5" Decision="Minus" />
   <Line ConnecttoShapeID="3" Value="6" Decision="equal" />
   <Line ConnecttoShapeID="4" Value="7" Decision="otherwise" />
  </RouterValues>
 </Shape>
 <Shape>
  <ShapeID>1</ShapeID>
  <ShapeName>Lindas process</ShapeName>
  <ShapeType>Process</ShapeType>
  <ConnectFromShapeID>0</ConnectFromShapeID>
  <ConnectToShapeID>1</ConnectToShapeID>
  <EstimatorCode>ES0001</EstimatorCode>
  <BasedonNextShapeQueue>No</BasedonNextShapeQueue>
  <ProbabilisticRouter>No</ProbabilisticRouter>
  <MinQueueValue>0</MinQueueValue>
  <Shape In Connections>
     <ConnectFromShapeID>3</ConnectFromShapeID>
     <ConnectFromShapeID>4</ConnectFromShapeID>
      <ConnectFromShapeID>5</ConnectFromShapeID>
  </JoinerConnections>
 </Shape>
</ModelDetails>
```

Model Estimators, only to send:

```
<?xml version="1.0" encoding="utf-8" ?>
<Estimators>
    <Estimator code="ES0001" name="Incomming service orders"
  distribution="Exponential" Value="" />
    <Estimator code="ES0002" name="Incomming Sales orders"
  distribution="Deterministic" Value="2" />
<Estimator code="ES0003" name="Mary's working time" distribution="Exponential"
  Value="" />
<Estimator code="ES0004" name="Mark's working time" distribution="Exponential"
  Value="" />
</Estimators>
```

The Simulation Model details, Model Router Value and Model Shape In Connection tables are described below:

Field No.	Field Name	Data Type	Length	Description
1	Model Code	Code	20	The model code
2	Shape ID	Integer		The shape ID of the joiner shape
3	Connect From Shape ID	Integer		The ID of the shape the joiner is connected from.
4	Entity Waiting	Boolean		Field use when running a simulation. Tell that an entity is in waiting status.
5	Entity ID	Integer		The ID of the entity that is waiting.

Table 28 Model Shape In Connection table fields

Field No.	Field Name	Data Type	Length	Description
1	Model Code	Code	20	The model code
2	Shape ID	Integer		The ID of the Shape
3	Shape Name	Text	50	The Name of the Shape
4	Shape Type	Option		The Type of shape created, the options are:Creator,process,Router,joiner
9	Connect From Shape ID	Integer		The shape that the current shape is connected from
10	Connect To Shape ID	Integer		The shape that the current shape is connected to
11	Estimator Code	Code	10	The code of the estimator the shape is linked to
13	Idle	Boolean		This field is used when running a simulation it tells if the process is idle. Only relevant for process shapes.
15	Based on Next Shape Queue	Boolean		This field is only relevant for creator shapes, in order to create entities based on the queue of the next shape.
16	Probabilistic Router	Boolean		This field tells the system if the splitter is probabilistic or just a router
17	Min Queue Value	Decimal		The Minimum queue value of a process shape
18	Units to Create	Integer		The units to create by a creator shape
19	Create based on linked shape	Boolean		The creation event would be condition to the queue of the next shape

Table 29 Simulation Model Detail table field

Field No.	Field Name	Data Type	Length	Description
1	Model Code	Code	20	The model code
2	Shape ID	Integer		The ID of the router shape.
5	Probability	Decimal		The probability of choosing the next shape.
7	Connect to Shape ID	Integer		The shape that the router would route to

Table 30 Model Router Values table fields

4.2.3. Simulation Model Execution

The simulation model execution area consists of the following objects:

Object Type	Object Name
Table	Simulation Event List
Table	Shape Queue
Table	Entity Statistics
Table	Model Shape In Connection
Table	Model Router Value
Table	Simulation Model Result Line
Form	Simulation Model Results
Codeunit	Simulator Engine

Table 31 Objects of the model execution functionality

This section would describe the Navision design of the model execution methodology as it was explained in section 4.2.2. The execution functionality it is divided in three phases: Initiation, execution and termination. Each phase would be described in the following sections (To obtain an overview of the steps of each phase please refer to figure 11). Before explaining the steps involved in each section it is important to clarify the elements of the Simulation Event List table since it plays a fundamental role in the execution of a model.

Simulation Event List table

The Simulation event list is a fundamental repository for the engine and for the execution of the simulation. In this table all the future events that should be handled are registered.

The Simulation event list consists of the following fields:

Field No.	Field Name	Data Type	Length	Description
1	Time	BigInteger		The clock time of the event
2	Shape ID	Integer		The shape ID to which the event refers to
3	Event Type	Option		The type of event, the options are: Arrival, departure and termination.
4	Entity ID	Decimal		The entity ID
5	In Progess	Boolean		Flag field that tells that the current event is being analyzed.

Table 32 Simulation Event List table fields

The event list has 3 types of events: arrival, departure and termination. The **arrival** type event is use when an entity is created and is ready to be push through the system. The **departure** type event is use when an entity is ready to leave to the next available shape. The **termination** event is use to tell the simulation engine when should stop the execution of the current run. The *In Progress* field is use to keep track of the record that is currently in use by the simulation engine. The engine would read and write to this table through the whole execution of the simulation; therefore these transactions need to be of great speed. For this reason, the Simulation Event List would be a temporary table; temporary tables have the characteristic that all the reading and writing operations are handled in memory and no access to the database is done. This feature would allow the simulation engine to have a better performance in comparison to the one obtained by using normal tables.

The following sections would describe the different phases of the simulation engine in more details.

Initiation

The initiation phase deals mainly with the validation and setup steps necessary to perform a successful execution. The following data flow diagram illustrates the processes and tables involved in this process:

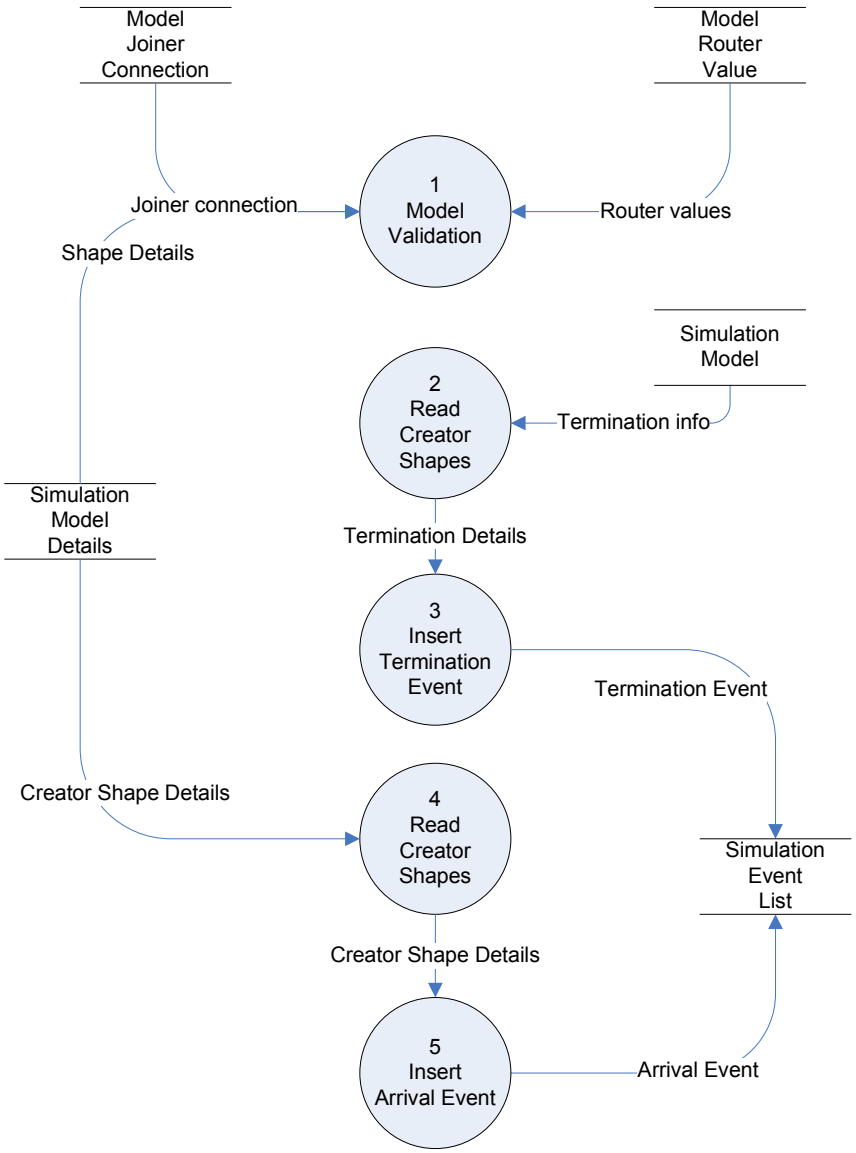


Figure 33 Simulation execution initiation steps

The first step in the initiation phase is validation of the model. The validation steps are the ones described in section 4.2.1. The initiation phase deals with initializing all the counter variables and tables.

If any errors are found the engine will stop processing further steps and will notify the user the errors found.

After a successful validation of the model the system initializes all the global variables that are going to be use in the model execution and in the different runs. Afterwards, the engine inserts the termination event on the Simulation Event List table by reading at the termination criteria specified in the Simulation Model table. Finally, it reads the creator type shapes from the Simulation Model Details table and insert the initial creation events on the event table.

When a creator type shape creates an event it also creates an entity. An entity is an individual case e.g. a service order request, an incoming customer, etc. This entity will have an ID or identification number that will ensure at the end of the simulation the possibility of calculating relevant statistics.

When all the above mentioned steps have been completed, the simulation engine is ready to commence the execution phase.

Execution

After the model has been validated the execution can occur. The execution follows the principles described in section 4.2.2. The following data flow diagram depicts the processes and tables involved in this phase:

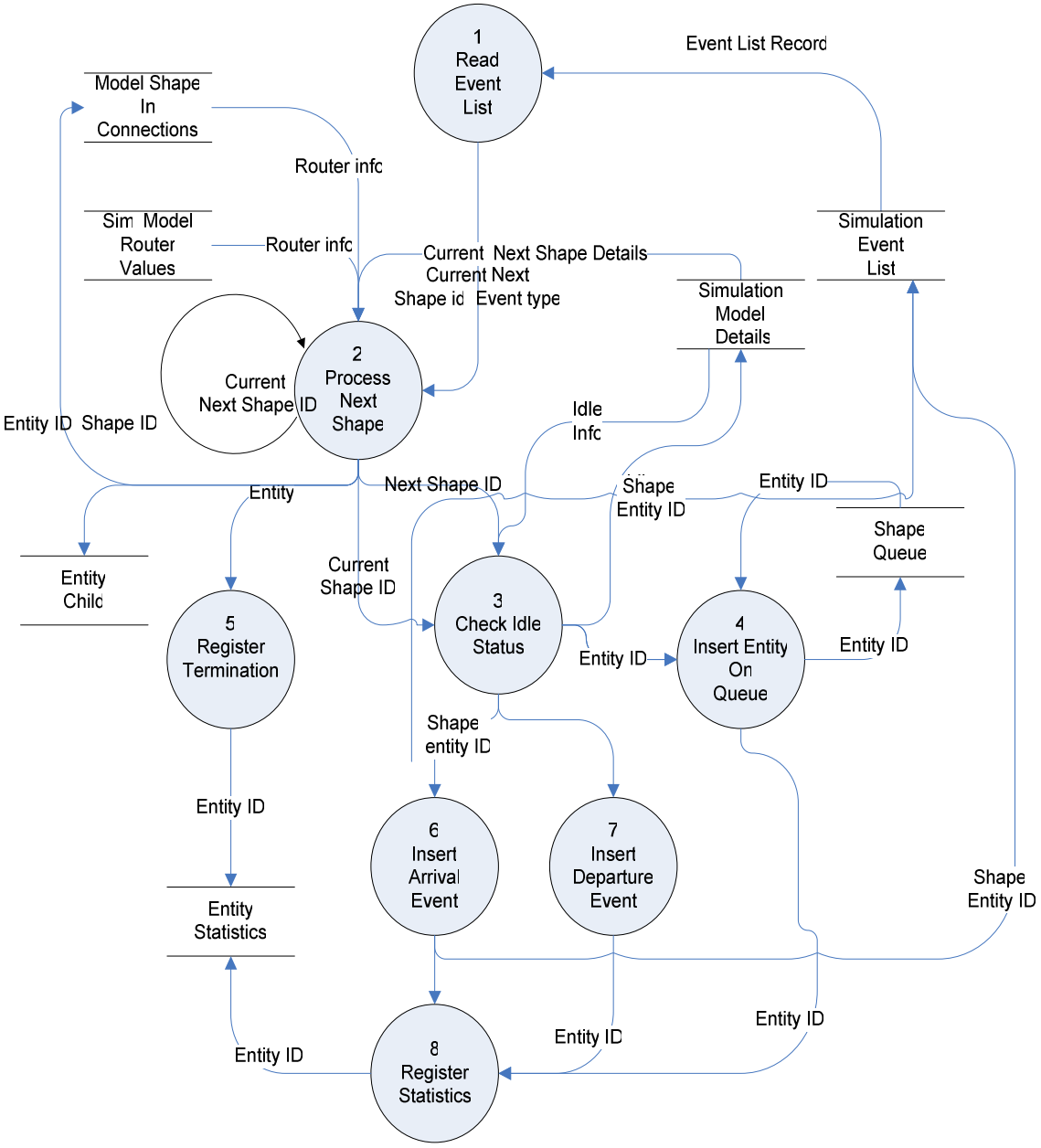


Figure 34 Simulation execution steps

The first step in the execution phase is reading the first record from the event list. The engine marks the retrieve record using the *In Process* field and checks what type of event, shape and entity are involved. Later it processes the event, deletes the record and moves to the next one until a termination event is found. The termination event only involves the registration of the disposal of the entity from the system and the registration of the event into the Entity Statistic table. The other types of events: departure and arrival would be described below.

Arrival Event

This type of event can only be generated by a Creator shape. The only shapes that can be connected to an Arrival event are: Processes, Routers, and Joiners. If the next shape is a process, the system checks if the process is idle (by reading the idle field from the Simulation Model Details table), if so, the process is set to busy and a departure event (based on the process estimator value) is generated and inserted in the event list. Otherwise, it inserts the entity on the process queue by inserting a record in the Shape Queue table. The Shape Queue table fields are described below:

Field No.	Field Name	Data Type	Length	Description
1	Shape ID	Integer		The id of the shape that this queue belongs. This field is part of the primary key.
2	Line No.	BigInteger		The line no. this field is part of the primary key.
3	Entity ID	Decimal		The ID of the entity in queue.
4	Connect from Shape ID	Integer		The connected shape for a Joiner shape

Table 33 Shape queue table fields

If the next shape found is a Joiner the system checks if there are any other entities waiting to move to the shape the Joiner is connected to. If this happens, the entity is put in the Joiner's queue. Otherwise, the entities are combined into one entity and this one is moved to the next shape connected. The way the system combines the entities is by filling the Entity Child table; the fields of the Entity Child Table are described below:

Field No.	Field Name	Data Type	Description
1	Entity ID	Decimal	The parent entity ID, this field is part of the primary key.
2	Child Entity	Decimal	The child entity ID. This field is part of the primary key.

Table 34 Entity Child table fields

If the next shape is a Router the system would calculate what would be the next shape to be routed according to the Router's values (by reading them from the Model Router Value table) and the operations described in section 4.2.1. When the next shape is found, which can only be a Joiner or a Process, the system performs the same steps that have been described for this type of shapes in this section.

Since this type of event is only generated by Creator shapes the next step would be to generate another arrival event. If the Creator's shape creation process is linked to another process (the field *Create based on link Shape* is true on the Model Details table) then no further action is taken. If the arrival event is concerned with the creation of a bulk of entities then the system would perform the operations described in this section for each entity created.

Departure Event

When this event occurs it means that an entity in a Process shape is ready to move to the next shape connected. There can only be four types of connected shapes: a Router, Terminator, Joiner or another Process type shape. If the next shape found is a Router, Joiner or a Process the system performs the same operations that were described in the section above. When a terminator type shape is found the system inserts the termination time on the entity statistics table.

Since the current shape is a Process the system checks if there are any other entities waiting in the shape's queue. If this happens an entity is retrieved in a first-in-first-out basis and a new departure event is generated. When a process has a *Minimum Queue Value* set an arrival event is generated if the queue falls below this value. If after retrieving an entity the queue happens to be empty then the Process is set to idle.

Statistics Registration

Each time an entity is: created, dispose from the system, enters or leaves a queue and enters or leaves a process on a simulation run. The system register this information in the Entity Statistic table, this table is later used to collect all the gathered statistics and generate the models results. The statistics that would be calculated based on this table would be described later in this section.

The fields that belong to this table and their description are listed in the following table:

Field No.	Field Name	Data Type	Length	Description
1	Entity ID	Decimal		The ID of the entity. This is a primary key field
2	Event Type	Option		The type of event, the options are: created, disposed, Enter Queue, Left Queue, Begin Process, End Process. This is a primary key field.
3	Time	Decimal		The simulation time of the event
4	Run No	Integer		The run no. of the simulation. This is a primary key field.
5	Shape ID	Integer		The id of the shape where the event happen.

Table 35 Entity Statistic table fields

Termination

The termination phase starts when all the simulation runs have been computed. In this phase the system reads the collected statistics and inserts the results in the Simulation Result Detail table.

On section 4.2.2 it was mentioned that some statistics should be collected on the model. The method in which each statistic is calculated is based on the values registered in the Entity Statistic table. The calculations are based by filtering on the *Event Type* field, the options of this field are: created, disposed, Enter Queue, Left Queue, Begin Process and End Process.

Each time a statistic is calculated is is inserted in the Simulation Model Result Detail table, the description of the table fields are shown below:

Field No.	Field Name	Data Type	Length	Description
1	Model Code	Code	20	The model code
2	Run No.	Integer		The Run No the statistic belongs to.
3	Shape ID	Integer		The shape the result relates to.
4	Shape Name	Text	250	The name of the shape
5	Result Item	Option		The result item, the options are: Avg. entity system time, Avg. entity queue time, Avg. processed time, Avg. idle time, Avg. busy time, Avg. process queue time, No. of processed entities, No. of created entities, No. of dispose entities.
6	Result Value	Decimal		The value of the statistic.

Table 36 Simulation Model Result Detail table fields

The statistics are grouped by shape ID in order to have statistics per shape giving more information to the user. The results are consolidated by the Simultion Model Result table. This table has sum index fields that automatically calculate the average based on some filtering done on the Simulation Model Result Detail table (SMRD). The fields and description of the fields on this table are shown below.

Field No.	Field Name	Data Type	Length	Description
1	Model Code	Code	20	The model the results belong to. Also is the primary key.
2	Avg. entity system time	Decimal		Average flow field that looks at the SMRD table filtering on result item = Avg. entity system time.
3	Avg. entity queue time	Decimal		Average flow field that looks at the SMRD table filtering on result item = Avg. entity queue time.
4	Avg. entity processed time	Decimal		Average flow field that looks at the SMRD table filtering on result item = Avg. processed time
5	Avg. process idle time	Decimal		Average flow field that looks at the SMRD table filtering on result item = Avg. idle time.
6	Avg. process busy time	Decimal		Average flow field that looks at the SMRD table filtering on result item = Avg. busy time.
7	Avg. process queue time	Decimal		Average flow field that looks at the SMRD table filtering on result item = Avg. process queue time.
8	Avq. No. of processed ents.	Decimal		Average flow field that looks at the SMRD table filtering on result item = No. of processed entities.

9	Avg. No. of created entities.	Decimal		Average flow field that looks at the SMRD table filtering on result item = No. of created entities.
10	Avg. No. of disposed entities	Decimal		Average flow field that looks at the SMRD table filtering on result item = No. of dispose entities.

Table 37 Simulation Model result table fields

The statistics and the calculation methods for each of them are described below:

Average time on the system: This is the average time an entity spends on the system. This statistic is calculated by the total time of an entity on the system divided by the total number of entities created.

Average time on queue for an entity: This is the average of the queue times spend on some process, is calculated as the sum of all the times spend on a queue divided by the total number of queue times collected

Average processed time for an entity: This is the average of the times an entity spent on a process, is calculated as the sum of all the times spend on a process divided by the total number of records collected.

Average total busy time for a process: This is the average time a process was busy in a simulation run, is calculated as the sum of all the process times for the entities processed by a Process shape.

Average idle time for a process: This is the average time a process was idle, would be calculated by deducting the total busy time by the total simulation tim*e*

Average process queue time: The average time an entity waits in the queue prior entering the process. Calculated by summing the times of all the entities that waited to enter a process shape divided by the total number of entities involved.

Average number of processed entities on a process: This is the average of the total number of processed entities on a process shape, is calculated by counting all the records for a process shape ID of type *End Process*.

Average number of created entities on a creator shape: This is the average number of entities that were created in a Creator shape, is calculated by counting all the records from the ES table of type *created*.

Average number of disposed entities: This is the average number of all the entities that dispose by a Terminator shape. This is calculated counting all the records of type d*isposed*.

PART III Implementation and Testing

5. Implementation

5.1. Implementation Plan

The previous chapters have described all the functionality that the simulator prototype would have. Since there are many tasks in hand and a limited time frame an implementation plan is needed. The following table presents such a plan. The sequence column shows the steps in ascending order in which the implementation should be carried out. The priority column gives an indication of what tasks should be carried out first in case of lack of time or independencies.

Main Feature	Sub Feature	Sequence	Priority	Status
NaviSim.dll				
	Navision Communication	22	2	Design only
	Shape reading and Validation	23	3	Design only
	Main Form	17	2	Design only
	Creator sub form	18	2	Design only
	Process sub form	20	2	Design only
	Router sub-form	21	2	Design only
	Data Set Storage	19	2	Design only
NaviMath.dll				Implemented
Simulation Model	Navision Tables and forms	2	1	Implemented
Simulation Execution				
	Tables and forms	3	1	Implemented
	Normal Creator shape and process	4	1	Implemented
	Special Creator shape and process	15	2	Not Implemented
	Router shape	10	2	Implemented
	Joiner Shape	11	2	Implemented
	Statistics	5	1	Implemented
Data Fitting				
	Tables and forms	1	1	Implemented
	K-S test	14	2	Partially
	Chi-Square test	16	2	Partially
	Temp Data Creation	6	1	Implemented

	Temp Time Data Creation	7	1	Implemented
	Frequency Table	12	2	Implemented
	Data Fitting Worksheet	13	3	Implemented
	Estimator Worksheet	24	3	Not Implemented
	Parameter Estimation	9	1	Partially

Table 38 Implementation Plan

5.2. NaviMath.dll

The implementation of this component is shown below. It consist of the INaviMath interface and two classes: NaviMath and RandomGen.

5.2.1. INaviMath Interface

```
using System;
using System.Runtime.InteropServices;
namespace NaviMath
{
            [Guid("DC51CE16-D799-49f6-86F2-BE9F41EEAA69")]
            public interface INaviMath
            {
                        double Exponential(double mean);
                        double Uniform(double StartNo,double EndNo);
                        double triangular(double a,double b,double
                          c);
                        void Normal(double mu,double sigmaSqr,ref
                          double X1,ref double X2);
                        double Weibull(double alpha, double beta);
                        double Exp(double number);
                        double Log(double number);
                        double Power(double number,double power);
                        double Sqrt(double number);
                        double GetRandom();

            }
}
```

5.2.2. NaviMath Class

This Class Implements the INaviMath interface and implements the functionality described in section 4.1.1.

```
using System;
using System.Runtime.InteropServices;
namespace NaviMath
{
    [Guid("16964EA9-4AA9-4a51-9C3A-34C441E12093")]
    [ClassInterface(ClassInterfaceType.AutoDual)]
    public class NaviMath:INaviMath
          {
                    private Random Rnd;

                    public double Exponential(double mean)
                    {
                              if (mean <= 0)
                              {
                                        return 0.0;
                              }
                              else
                              {
                                return   (-mean*Math.Log(
                                         RandomGen.
                                         NextRandNotZeroOne()));
                              }
                    }

                    public double Uniform(double StartNo,double
                      EndNo)
                    {
                      return StartNo + (EndNo - StartNo)*
                        RandomGen.NextRandomDouble();
                    }

                    public double triangular(double a,double
                      b,double c)
                    {
                              double Number,c2;
                              double RandomNo = RandomGen.
                                NextRandomDouble();
                              c2 = (c-a)/(b-a);
                              if (RandomNo <= c2)
                              {
                                  Number = Math.Sqrt(c2 *
                                    RandomNo);
```

```
            }
            else
            {
                Number = 1 - Math.Sqrt((1.0 -
                  c2) * (1.0 - RandomNo));
            }

            return a + (b-a) * Number;

}

public void Normal(double mu,double
  sigmaSqr,ref double X1,ref double X2)
{
            double Rnd1,Rnd2,log;
            Rnd1 = Rnd2 = 0.0;
            Rnd1 = RandomGen.
              NextRandomDouble();
            Rnd2 = RandomGen.
              NextRandomDouble();
            log = Math.Log(Rnd1);
            X1 = Math.Sqrt(-2.0*log)*
              Math.Cos(2.0*Math.PI*Rnd2);
            X2 = Math.Sqrt(-2.0*log)*
              Math.Sin(2.0*Math.PI*Rnd2);
            X1 = mu + sigmaSqr*X1;
            X2 = mu + sigmaSqr*X2;
            return;
}

public double Weibull(double alpha, double
  beta)
{
            return beta * Math.Pow(-
              1.0*Math.Log
            (RandomGen.NextRandNotZeroOne()
             ),1.0/alpha);
}

public double Exp(double number)
{
            return Math.Exp(number);
}

public double Log(double number)
{
            return Math.Log(number);
}
```

```
                public double Power(double number,double
                  power)
                {
                        return Math.Pow(number,power);
                }

                public double Sqrt(double number)
                {
                        return Math.Sqrt(number);
                }

                public double GetRandom()
                {
                        if (Rnd == null)
                        {
                                Rnd = new Random();
                        }
                        return this.Rnd.NextDouble();
                }
        }
}
```

5.2.3. RandomGen Class

This class is used to give random number generators and is used in the NaviMath class.

```
using System;

namespace NaviMath
{
        public class RandomGen
        {
                private static Random Rnd;

                public static double NextRandomDouble()
                {
                        if (Rnd == null)
                        {
                                Rnd = new Random();
                        }

                        return Rnd.NextDouble();
                }

                public static double NextRandNotZeroOne()
```

```
                    {
                                if (Rnd == null)
                                {
                                            Rnd = new Random();
                                }
                                double Num;
                                do
                                {
                                            Num = Rnd.NextDouble();
                                }
                                while ((Num == 0.0)||(Num ==
1.0));
                        return Num;
                    }
```

5.3. Data Fitting

This chapter describes the implementation done of some of the features of the data fitting functionality. The areas that were implemented can be seen on table 35 and will be described in the following sections. All this methods are located in the SimStatisticMgt codeunit.

5.3.1. Creation of Temp Data Table

The functionality concerning the estimator lies on the GenerateTempDataCalculations function. This function includes the implementation of the occurrence algorithm described in section 4.3.1. The implementation is simpler since it makes use of the duration type field. This type of fields makes possible the calculation of duration just by subtracting two values. The implementation is as follows:

```
GenerateTempDataCalculation(EstimatorCode : Code[20])

Variables

 Name                DataType    Subtype                Description
 SimEstimator        Record      Simulation Estimator
 RecordReference     RecordRef
 DatetimeMgt         Codeunit    Datetime Mgt.
 FieldReference      FieldRef
 LineNo              Integer
```

```
 RecordCounter       Integer
 Value               Decimal
 StartDate           Date
 EndDate             Date
 StartTime           Time
 EndTime             Time
 NextVal             Integer                          The next record
 TempDataCalculation  Record   Temp Data Calculation          Temp table
 TempTimeDataCalc     Record   Temp Time Data Calculation     Temp table
 LineNo               Integer
 TimeLineNo           Integer
```

```
SimEstimator.GET(EstimatorCode);
SimEstimator.TESTFIELD("Table No.");
RecordReference.OPEN(SimEstimator."Table No.");
RecordReference.CURRENTKEYINDEX(1);
IF NOT RecordReference.FIND('-') THEN
  ERROR('There are no records in the table the estimator is linked to');
RecordCounter := 0;
LineNo := 1;
CLEAR(TempDataCalculation);
CASE SimEstimator.Type OF
  //Table-data estimator
  1:BEGIN
    SimEstimator.TESTFIELD("Field No.");
    FieldReference := RecordReference.FIELD(SimEstimator."Field No.");
    REPEAT
      RecordCounter += 1;
      Value := FieldReference.VALUE;
      InsertTempDataRec(Value);
    UNTIL (RecordReference.NEXT = 0) OR (RecordCounter >=1000000);
  END;
  //Interval estimators
  2:BEGIN
    //checking that the fields of the estimator table contains a
    //value
    SimEstimator.TESTFIELD("Start Interval Date");
    SimEstimator.TESTFIELD("End Interval Date");
    CASE SimEstimator."Interval Calculation Type" OF
      SimEstimator."Interval Calculation Type"::"Date Occurrence":
        BEGIN
          generatetemptimedatacalc(estimatorcode);
          RecordReference.OPEN(database::"temp time data
           calculation");
          RecordReference.CURRENTKEYINDEX(1);
          FieldReference := RecordReference.FIELD(2);
          RecordReference.FIND('-');
          REPEAT
            FieldReference := RecordReference.FIELD(2);
            StartDate := FieldReference.VALUE;
```

```
      NextVal := RecordReference.NEXT;
      IF  NextVal <> 0 THEN BEGIN
        FieldReference := RecordReference.FIELD(2);
        EndDate := FieldReference.VALUE;
         //Inserts the value on the temp data calculation
                  table
        InsertTempDataRec(EndDate-StartDate);
        RecordCounter += 1;
      END;
    UNTIL (NextVal = 0) OR (RecordCounter >=1000000);
  END;
SimEstimator."Interval Calculation Type"::"Time Ocurrence":
  BEGIN
    generatetemptimedatacalc(estimatorcode);
    RecordReference.OPEN(database::"temp time data
                  calculation");
    RecordReference.CURRENTKEYINDEX(1);
    FieldReference := RecordReference.FIELD(2);
    RecordReference.FIND('-');
    REPEAT
      FieldReference := RecordReference.FIELD(2);
      StartDate := FieldReference.VALUE;
      FieldReference := RecordReference.FIELD(3);
      StartTime := FieldReference.VALUE;
      NextVal := RecordReference.NEXT;
      IF  NextVal <> 0 THEN BEGIN
        FieldReference := RecordReference.FIELD(2);
        EndDate := FieldReference.VALUE;
        FieldReference := RecordReference.FIELD(3);
        EndTime := FieldReference.VALUE;
         InsertTempDataRec(
            (DatetimeMgt.Datetime(EndDate,EndTime)
            -DatetimeMgt.Datetime(StartDate,StartTime))
            /3.6);
        RecordCounter += 1;
       END;
    UNTIL (NextVal = 0) OR (RecordCounter >=1000000);
  END;
SimEstimator."Interval Calculation Type"::"Date Duration":
  BEGIN
    generatetemptimedatacalc(estimatorcode);
    FieldReference := RecordReference.FIELD
        (SimEstimator."Start Date Field No.");
    FieldReference.SETRANGE(SimEstimator."Start Interval
        Date",SimEstimator."End Interval Date");
    RecordReference.FIND('-');
    REPEAT
      FieldReference := RecordReference.FIELD(
         SimEstimator."Start Date Field No.");
      StartDate := FieldReference.VALUE;
      FieldReference := RecordReference.FIELD(
        SimEstimator."End Date Field No.");
      EndDate := FieldReference.VALUE;
      InsertTempDataRec((EndDate-StartDate));
```

```
            RecordCounter += 1;
          UNTIL (RecordReference.NEXT = 0) OR
                        (RecordCounter >=1000000);
        END;
      SimEstimator."Interval Calculation Type"::"Time Duration":
        BEGIN
          SimEstimator.TESTFIELD("Start Date Field No.");
          SimEstimator.TESTFIELD("End Date Field No.");
          SimEstimator.TESTFIELD("Start Time Field No.");
          SimEstimator.TESTFIELD("End Time Field No.");
          SimEstimator.TESTFIELD("Start Interval Date");
          SimEstimator.TESTFIELD("End Interval Date");
          FieldReference := RecordReference.FIELD(
             SimEstimator."Start Date Field No.");
          FieldReference.SETRANGE(SimEstimator."Start Interval
             Date",SimEstimator."End Interval Date");
          RecordReference.FIND('-');
          REPEAT
            FieldReference := RecordReference.FIELD(
               SimEstimator."Start Date Field No.");
            StartDate := FieldReference.VALUE;
            FieldReference := RecordReference.FIELD(
                SimEstimator."End Date Field No.");
            EndDate := FieldReference.VALUE;
            FieldReference := RecordReference.FIELD(
               SimEstimator."Start Time Field No.");
            StartTime := FieldReference.VALUE;
            FieldReference := RecordReference.FIELD(
                SimEstimator."End Time Field No.");
            EndTime := FieldReference.VALUE;
            InsertTempDataRec((DatetimeMgt.Datetime
             (EndDate,EndTime) -
               DatetimeMgt.Datetime(StartDate,StartTime))/3.6);
            RecordCounter += 1;
          UNTIL         (RecordReference.NEXT = 0) OR
                        (RecordCounter >=1000000);
        END;
    END;
  END;
END;
```

5.3.2. Kolmogorov-Smirnov Test

The implementation of the Kolmogorov-Smirnov test follows the considerations and guidelines stated in section 2.2.4. The test was implemented only for the Normal, exponential and uniform distributions.

```
RunKSTest(EstimatorCode : Code[20]; TableNo : Integer;FieldNo :
           Integer;KeyIndex   :   Integer;Distribution   :   '   ,
           Exponential,Uniform,Normal,

Variables

 Name             DataType
 RecordReference  RecordRef
 FieldReference   FieldRef
 i                Integer
 FieldValue       Decimal
 iDivN            Decimal
 Dplus            Decimal
 MaxDplus         Decimal
 Dminus           Decimal
 MaxDminus        Decimal
 Result           Decimal
 Dn               Decimal
 TestStatistic    Decimal
 SqrtN            Decimal
 x                Decimal
 CValue           Decimal


CheckNaviMathVar;//Check if the NaviMath variable is created otherwise
                 //create it
RecordReference.OPEN(TableNo);
RecordReference.CURRENTKEYINDEX(KeyIndex);
RecordReference.FIND('-');
FieldReference := RecordReference.FIELD(FieldNo);
i := 1;
Dplus := 0;
Dminus := 0;
REPEAT
UNTIL RecordReference.NEXT = 0;
REPEAT
  iDivN := i/N;
  FieldValue := FieldReference.VALUE;
```

```
  CASE Distribution OF
    Distribution::Normal:
      BEGIN
        x := FieldValue - Mean/NaviMath.Sqrt(Variance);
        Result := Normal(x);
      END;
    Distribution::Exponential:
      Result := Exponential(FieldValue,Mean);
    Distribution::Uniform:
      Result := 1/(b-a);
  END;
  Dplus :=  iDivN - Result;
  IF Dplus >= MaxDplus THEN
    MaxDplus := Dplus;
  Dminus := Result - (i - 1)/N;
  IF Dminus >= MaxDminus THEN
    MaxDminus := Dminus;
  i += 1;
UNTIL RecordReference.NEXT = 0;
IF MaxDplus > MaxDminus THEN
  Dn := MaxDplus
ELSE
  Dn := MaxDminus;

SqrtN := NaviMath.Sqrt(N);
CASE Distribution OF
  Distribution::Normal:
    TestStatistic := (SqrtN - 0.01 + (0.85/SqrtN))*Dn;
  Distribution::Exponential:
    TestStatistic := (Dn - 0.2/N)*(SqrtN+0.26+0.5/SqrtN);
  Distribution::Uniform:
    TestStatistic := (SqrtN - 0.12 + (0.11/SqrtN)*Dn);
END;
//InsertTestResult(ECode,DistName,TestName,StatName,StatValue,Cvalue,Resul
t)
//0: ,1:Exponential,2:Uniform,3:Normal,4:Weibull
//0:Chi-Square,1:Kolmogorov-Smirnov
//0:X2,1:D
//0:Approved,1: Failed
CValue := KScValues(Distribution,Confidence);
IF TestStatistic > CValue THEN
  //pass test
  InserTestResult(EstimatorCode,Distribution,1,1,TestStatistic,CValue,0)
ELSE
  //fail test
  InserTestResult(EstimatorCode,Distribution,1,1,TestStatistic,CValue,1);
```

5.3.3. Chi-Square Test

The implementation of the Chi-Square test follows the considerations and guidelines stated in section 2.2.4. The test was partially implemented for the exponential and uniform distributions.

```
RunKSTest(EstimatorCode : Code[20];TableNo : Integer;FieldNo :
Integer;KeyIndex : Integer;Distribution : ' ,Exponential,Uniform,Normal,

Variables

 Name                 DataType   Subtype
 RecordReference      RecordRef
 FieldReference       FieldRef
 NoOfRecords          Integer
 Probability          Decimal
 Max                  Decimal
 Min                  Decimal
 Interval             Decimal
                                 Estimator Frequency
 EstimatorFreqTable   Record     Table
 k                    Integer
 i                    Integer
 Number               Decimal
 X2                   Decimal
 ExpectedValue        Decimal
 CValue               Decimal
 DF                   Integer
 s                    Integer
```

```
RunChiSquareTest(EstimatorCode : Code[20];TableNo : Integer;FieldNo :
Integer;KeyIndex : Integer;Distribution : ' ,Exponential,Uniform,
//Check if it is possible to do the test
CheckNaviMathVar;
RecordReference.OPEN(TableNo);
RecordReference.CURRENTKEYINDEX(KeyIndex);
n := RecordReference.COUNT;
IF n < 20 THEN
  //Is not possible to do the test
  EXIT;
//calculate probability and number of intervals to use
```

```
CASE TRUE OF
  n < 50:
    BEGIN
      Probability := 1/10;
      k := 10
    END;
  n < 100:
    BEGIN
     Probability := 1/15;
     k := 15
    END;
  n >= 100:
    BEGIN
      Probability := 5/n;
      k :=  n/5;
    END;
END;
k:=8;
Probability := 1/k;
//Calculate Frequency table
RecordReference.OPEN(TableNo);
RecordReference.CURRENTKEYINDEX(KeyIndex);
RecordReference.ASCENDING(TRUE);
RecordReference.FIND('-');
FieldReference := RecordReference.FIELD(FieldNo);
CASE Distribution OF
  Distribution::Normal: BEGIN
     s := 2;
     EXIT;//not implemented
  END;
  Distribution::Weibull: EXIT; // not implemented
  Distribution::Exponential: BEGIN
    BeginInterval := 0;
    Number := 1 - Probability;
    EndInterval := -Beta*NaviMath.Log(Number);
    s := 1;
  END;
  Distribution::Uniform:BEGIN
    BeginInterval := a;
    EndInterval := ((b-a)*Probability)+a;
    s := 2;
  END;
END;
EndInterval := ROUND(EndInterval,0.00000001,'>');
RecordReference.FINDFIRST;
CLEAR(EstimatorFreqTable);
i := 1;
REPEAT
  EstimatorFreqTable.INIT;
  EstimatorFreqTable."Estimator Code" := EstimatorCode;
  EstimatorFreqTable."From Value" := BeginInterval;
  EstimatorFreqTable."To Value":= EndInterval;
  FieldReference := RecordReference.FIELD(FieldNo);
  FieldReference.SETRANGE(BeginInterval,EndInterval);
```

```
  EstimatorFreqTable.Frequency := RecordReference.COUNT;
  EstimatorFreqTable.INSERT;
  i += 1;
  BeginInterval := EndInterval + 0.00000001;
  CASE Distribution OF
    Distribution::Normal:;
    Distribution::Weibull:;
    Distribution::Exponential: BEGIN
      Number := 1 - i*Probability;
      IF Number = 0 THEN BEGIN
        RecordReference.RESET;
        RecordReference.CURRENTKEYINDEX(KeyIndex);
        RecordReference.ASCENDING(TRUE);
        RecordReference.FIND('+');//finds the last record
        FieldReference := RecordReference.FIELD(FieldNo);
        EndInterval := FieldReference.VALUE;
      END ELSE
        EndInterval := -Beta*NaviMath.Log(Number);
    END;
    Distribution::Uniform:
      EndInterval := ((b-a)*i*Probability)+a;
  END;
    EndInterval := ROUND(EndInterval,0.00000001,'>');
UNTIL i > k ;

//Calculation of X2
EstimatorFreqTable.RESET;
EstimatorFreqTable.SETRANGE("Estimator Code",EstimatorCode);
EstimatorFreqTable.FIND('-');
ExpectedValue := n*Probability;
REPEAT
  X2       :=       X2       +       ((EstimatorFreqTable.Frequency       -
ExpectedValue)*(EstimatorFreqTable.Frequency                                -
ExpectedValue))/ExpectedValue
UNTIL EstimatorFreqTable.NEXT = 0;
//Check and insert results
DF := k - s - 1;
CValue := CScValues(Confidence,DF);
//InsertTestResult(ECode,DistName,TestName,StatName,StatValue,Cvalue,Resul
t)
//0: ,1:Exponential,2:Uniform,3:Normal,4:Weibull
//0:Chi-Square,1:Kolmogorov-Smirnov
//0:X2,1:D
//0:Approved,1: Failed
IF X2 >= CValue THEN
  //fail test
  InserTestResult(EstimatorCode,Distribution,0,0,X2,CValue,1)
ELSE
  //fail test
  InserTestResult(EstimatorCode,Distribution,0,0,X2,CValue,0);
```

5.3.4. Creation of Frequency table

The functionality concerning the creation of a frequency table was implemented in the CreateFreqTable function. The implementation is as follows:

```
CreateFreqTable(TableNo : Integer;FieldNo : Integer;KeyIndex :
           Integer;EstimatorCode : Code[20];IntervalNo : Integer)

RecordReference.OPEN(TableNo);
RecordReference.CURRENTKEYINDEX(KeyIndex);
RecordReference.ASCENDING(TRUE);
RecordReference.FIND('-');
FieldReference := RecordReference.FIELD(FieldNo);
Min := FieldReference.VALUE;
Min := ROUND(Min,1,'<');
RecordReference.FIND('+');
Max := FieldReference.VALUE;
Max := ROUND(Max,1,'>');
Interval := Max - Min;
Interval := ROUND(Interval/IntervalNo,0.001,'>');
RecordReference.FIND('-');
EstimatorFreqTable.SETRANGE("Estimator Code",EstimatorCode);
EstimatorFreqTable.DELETEALL;
REPEAT
  EstimatorFreqTable.INIT;
  EstimatorFreqTable."Estimator Code" := EstimatorCode;
  EstimatorFreqTable."From Value" := Min;
  EstimatorFreqTable."To Value":= Min + Interval;
  FieldReference := RecordReference.FIELD(FieldNo);
  FieldReference.SETRANGE(Min,Min+Interval);
  EstimatorFreqTable.Frequency := RecordReference.COUNT;
  EstimatorFreqTable.INSERT;
  Min := Min + Interval;
UNTIL Min >= Max;
```

5.4. Simulation Execution

The simulation execution area follows the principles and concepts described in sections 3.2.2 and 4.2.3. The implementation of this functionality was made on the SimulationEngine Codeunit.

5.4.1. Engine

The engine acts as the main distribution center for the rest of the execution functionality. It reads the events from the event list and routes to the relevant functionality based on the retrieved values. The engine lies on the OnRUN trigger in codeunit SimulationEngine.

```
OnRun()

Variables
 Name             DataType    Subtype           Length
 ModelCode        Code                               20
                              Simulation
 SimEventList     Record      Event List
 Window           Dialog
 Clock            Decimal
 LogExecution     Boolean
 RunNo            Integer
 NoOfRuns         Integer
 SimulationTime   BigInteger
 RandomNo         Decimal
 EntityID         Decimal
 StartTime        Time
 EndTime          Time


//Validation steps;
IF ModelCode = '' THEN
  ERROR('No model has been setup for simulation');
SimulationModel.GET(ModelCode);
SimulationModel.TESTFIELD("No. of Runs");
SimulationModel.TESTFIELD("Simulation Time");

//Init values for variables
StartTime := TIME;
```

```
EntityStat.DELETEALL;
LogExecution := SimulationModel."Log execution";
NoOfRuns := SimulationModel."No. of Runs";
SimulationTime := SimulationModel."Simulation Time";
SimModelResult.SETRANGE("Model Code",ModelCode);
SimModelResult.DELETEALL;
SimModResultDetail.SETRANGE("Model No.",ModelCode);
SimModResultDetail.DELETEALL;

//Start of simulation
FOR RunNo := 1 TO NoOfRuns DO BEGIN
  //Inititation step
  Clock := 0;
  EntityID[1] := 0;
  EntityID[2] := 0;
  ClearLog(ModelCode);
  CLEAR(SimEventList);
  SimEventList.DELETEALL;
  InsertLog(ModelCode,0,'Start of simulation...');
  ShapeQueue.DELETEALL;
  SetProcessShapesIdle(ModelCode);
  EndSimulation := FALSE;

  //Insert the termination event
  IF SimulationModel."Stop Based on Estimator Code" = '' THEN BEGIN
    EndingTime := ConvertToSeconds(SimulationModel."Units of Measure",
      SimulationModel."Simulation Time");
    InsertEvent(EndingTime,2,0,0,0);
  END;

  //Find all the "Creator" shapes and start creating init events
  SimModelDetail.RESET;
  SimModelDetail.SETRANGE("Model Code",ModelCode);
  SimModelDetail.SETRANGE("Shape Type", SimModelDetail."Shape
                                    Type"::Creator);
  IF SimModelDetail.FIND('-') THEN BEGIN
    REPEAT
      IF SimModelDetail."Units to Create" <> 0 THEN BEGIN
        FOR x := 1 TO SimModelDetail."Units to Create" DO BEGIN
          EntityID[1] := EntityID[1] + 1;
          InsertEntityOnQueue(SimModelDetail."Shape ID",EntityID[1],0,0);
        END;
        InsertEvent(0,0,SimModelDetail."Shape ID",EntityID[1],0);
      END ELSE;
        //Generate arrival events
        GenerateArrivalEvent(SimModelDetail);
    UNTIL SimModelDetail.NEXT = 0;
  END ELSE
    ERROR('Model %1 does not contain sufficient details to perform a
simulation',
      ModelCode);

  Window.OPEN('Run No: #1#####\@2@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@');
  //START of interation runs
```

```
  REPEAT
    Window.UPDATE(1,FORMAT(RunNo));
    SimEventList.RESET;
    SimEventList.SETCURRENTKEY(Time);
    IF SimEventList.FIND('-') THEN BEGIN
      //Fields to find the current event on the list to later delete it
      SimEventList.Inprogress := TRUE;
      SimEventList.MODIFY;
      Clock := SimEventList.Time;

      Bar := ROUND(Clock /EndingTime * 10000,1);
      Window.UPDATE(2,Bar);

      CASE SimEventList."Event Type" OF
        SimEventList."Event Type"::Arrival:
          BEGIN
            //Here we have to get the next shape and put it on a process
            //or a queue
            CurrentShapeID := SimEventList."Shape ID";
            EntityIDFromList[1] := SimEventList."Entity ID";
            IF SimEventList."Parent Entity" THEN
              EntityIDFromList[2] := 1
            ELSE
              EntityIDFromList[2] := 0;
            SetNextEvent(
              SimEventList."Shape ID",GetNextShapeID(CurrentShapeID),
              EntityIDFromList,SimEventList."Event Type");
            SimModelDetail.GET(ModelCode,CurrentShapeID);
            GenerateArrivalEvent(SimModelDetail);
          END;
        SimEventList."Event Type"::Departure:
          BEGIN
            CurrentShapeID := SimEventList."Shape ID";
            EntityIDFromList[1] := SimEventList."Entity ID";
            IF SimEventList."Parent Entity" THEN
              EntityIDFromList[2] := 1
            ELSE
              EntityIDFromList[2] := 0;
            SetNextEvent(
              SimEventList."Shape ID",GetNextShapeID(CurrentShapeID),
              EntityIDFromList,SimEventList."Event Type");
          END;
        SimEventList."Event Type"::Termination:
          EndSimulation := TRUE;
      END;
      //Delete the current event taken from the list
      SimEventList.RESET;
      SimEventList.SETRANGE(Inprogress,TRUE);
      SimEventList.FIND('-');
      SimEventList.DELETE;
    END ELSE
      EndSimulation := TRUE;
  UNTIL EndSimulation = TRUE;
END;
```

```
Window.CLOSE;
EndTime := TIME;
CalculateStatistics;
EntityStat.DELETEALL;
```

5.4.2. Next Event Method

The next event method is a recursive method that calls the next shape connected to the one being currently examined. Implements the free/busy, router and joiner mechanisms.

```
SetNextEvent(CurrShapeID : Integer;NextShapeID : Integer;CurrEntityID :
            ARRAY [2] OF Decimal;
            CurrEventType : 'arrival,departure,termination'

Variables
 Name              DataType  Subtype
                             Simulation Model
 NextShapeDetail   Record    Detail
                             Simulation Model
 CurrShapeDetail   Record    Detail
 NewEntityID       Decimal

//We get the next shape and based on the current event type
//we figure out what to do.
NextShapeDetail.GET(ModelCode,NextShapeID);
CASE NextShapeDetail."Shape Type" OF
   NextShapeDetail."Shape Type"::Process:BEGIN
     CASE CurrEventType OF
       //arrival events only check if the queue is empty if not put on
            queue
       CurrEventType::arrival:
         BEGIN
           IF NextShapeDetail.Idle THEN BEGIN
             //schedule departure
             GenerateDepartureEvent(NextShapeDetail,
                        CurrEntityID[1],CurrEntityID[2]);
             NextShapeDetail.Idle := FALSE;
             NextShapeDetail.MODIFY;
             InsertLog(ModelCode,Clock,
                STRSUBSTNO('Shape ID %1 has been set to busy',
                  FORMAT(NextShapeDetail."Shape ID")));
           END ELSE
             InsertEntityOnQueue(NextShapeID,0,CurrEntityID[1],
                        CurrEntityID[2]);//put entity on shape queue
         END;
       //departure events route to next event
       //and check if theres is something on queue otherwise wait.
       CurrEventType::departure:
         BEGIN
```

```
            SetNextEvent(CurrShapeID,NextShapeID,CurrEntityID,
             CurrEventType::arrival);
            //get element from the queue if any otherwise set the shape to
             //idle again.
            ReloadProcessShape(CurrShapeID);
            //Check minimum queue value and generate departure event
          END;
      END;
    END;
    //ROUTER: route to next shape
    NextShapeDetail."Shape Type"::Router:BEGIN
      IF NextShapeDetail."Probabilistic Router" THEN
        RandomNo := RANDOM(1001);
      SetNextEvent(
        CurrShapeID,GetNextShapeFromRouter(NextShapeID),
        CurrEntityID,CurrEventType::arrival);
    END;
    //JOINER: Wait until all the shapes connected has set an entity
    NextShapeDetail."Shape Type"::Joiner:BEGIN
      GetEntityIDFromJoiner(NextShapeID,CurrShapeID,
            CurrEntityID,NewEntityID);
      IF NewEntityID[1] <> -1 THEN
        SetNextEvent(
          CurrShapeID,NextShapeDetail."Connect To Shape ID",
            NewEntityID,CurrEventType::arrival);
    END;
    //TERMINATOR
    NextShapeDetail."Shape Type"::Terminator:BEGIN//termination
      CurrShapeDetail.GET(ModelCode,CurrShapeID);
      //Reload the process if the source shape was a process.
      IF    CurrShapeDetail."Shape    Type"    =    CurrShapeDetail."Shape
             Type"::Process
      THEN
        ReloadProcessShape(CurrShapeID);
      //Collect statistics on the entity
      InsertEntityStatistic(NextShapeDetail."Shape ID",CurrEntityID[1],
            CurrEntityID[2],1,Clock);
      InsertLog(ModelCode,Clock,
        STRSUBSTNO('entity %1 has terminated the run',
            FORMAT(CurrEntityID[1])));
    END;
  END;
```

5.4.3. Joiner implementation

The implementation of the Joiner shape is done in the GetEntityIDFromJoiner and follows the principles described in section 3.2.1

```
GetEntityIDFromJoiner(ShapeID : Integer;ConnectFromShapeID : Integer;
            InEntityID : ARRAY [2] OF Decimal;
            VAR NewEntityID : ARRAY [2] OF Decimal

Variables
 Name               DataType           Subtype
 EntityChild        Record             Entity Child
                                       Model Shape In
 ModelShapeInConn   Record             Connection
 TempNewEntityID    Decimal
 QueueEntityID      Decimal



//First Check if the joiner has a emtpy value otherwise put on queue.

GetEntityIDFromJoiner(ShapeID      :      Integer;ConnectFromShapeID      :
Integer;InEntityID : ARRAY [2] OF Decimal;VAR NewEntityID : ARRAY [2] OF D
//First Check if the joiner has a emtpy value otherwise put on queue.
ModelShapeInConn.GET(ModelCode,ShapeID,ConnectFromShapeID);
IF ModelShapeInConn."Entity ID" <> 0 THEN BEGIN
  //Put on queue

InsertEntityOnQueue(ShapeID,ConnectFromShapeID,InEntityID[1],InEntityID[2]
);
  NewEntityID[1] := -1;
END ELSE BEGIN
  ModelShapeInConn.RESET;
  ModelShapeInConn.SETCURRENTKEY("Entity ID");
  ModelShapeInConn.SETRANGE("Model Code",ModelCode);
  ModelShapeInConn.SETRANGE("Shape ID",ShapeID);
  ModelShapeInConn.SETFILTER("Connect              From              Shape
ID",'<>%1',ConnectFromShapeID);
  ModelShapeInConn.SETRANGE("Entity ID",0);
  IF NOT ModelShapeInConn.FIND('-') THEN BEGIN
    EntityID[1] := EntityID[1] + 1;
    EntityID[2] := 1;
    InsertLog(ModelCode,Clock,STRSUBSTNO('Entity ID %1 has been created on
Joiner no %2, The joiner has been flushed',
      EntityID[1],ShapeID));
    //Insert log
    ModelShapeInConn.RESET;
    ModelShapeInConn.SETRANGE("Model Code",ModelCode);
```

```
    ModelShapeInConn.SETRANGE("Shape ID",ShapeID);
    REPEAT
      IF ModelShapeInConn."Entity ID" <> 0 THEN BEGIN
        EntityChild."Parent Entity ID" := EntityID[1];
        EntityChild."Child Entity ID" := ModelShapeInConn."Entity ID";
        EntityChild.INSERT;
      END;
      //Reload from queue
      GetFirstFromShapeQueue(ShapeID,ConnectFromShapeID,QueueEntityID);
      IF QueueEntityID[1] = -1 THEN BEGIN
        ModelShapeInConn."Entity ID" := 0;
        ModelShapeInConn."Parent Entity" := FALSE;
      END ELSE BEGIN
        ModelShapeInConn."Entity ID" := QueueEntityID[1];
        IF QueueEntityID[2] = 1 THEN
          ModelShapeInConn."Parent Entity" := TRUE
        ELSE
          ModelShapeInConn."Parent Entity" := FALSE;
      END;
      ModelShapeInConn.MODIFY;
    UNTIL ModelShapeInConn.NEXT = 0;
    NewEntityID[1] := EntityID[1];
    NewEntityID[2] := EntityID[2];
  END ELSE BEGIN //put entity on the model shape in conexions to wait for
the arrival of another entity
    ModelShapeInConn.GET(ModelCode,ShapeID,ConnectFromShapeID);
    ModelShapeInConn."Entity ID" := InEntityID[1];
    IF InEntityID[2] = 0 THEN
      ModelShapeInConn."Parent Entity" := FALSE
    ELSE
      ModelShapeInConn."Parent Entity" := TRUE;
    ModelShapeInConn.MODIFY;
    NewEntityID[1] := -1;
  END;
END;
```

5.4.4. Router Implementation

The implementation of the Router shape is done in the GetNextShapeFromRouter and follows the principles described in section 3.2.1

```
GetNextShapeFromRouter(ShapeID : Integer) : Integer

Variables
 Name                    DataType  Subtype
                                   Simulation Model
 SimModelDetail          Record    Detail
                                   Model Router
 RouterValue             Record    Value
 ShapeQueue              Record    Shape Queue
```

		Simulation Model Detail
TempRec	Record	
Decision	Decimal	
Found	Boolean	
CummulativeProbability	Decimal	
QueueCount	Integer	
NullCounter	Integer	
NextShapeID	Integer	
MinQueue	Integer	
FirstTime	Boolean	

```
SimModelDetail.GET(ModelCode,ShapeID);
IF SimModelDetail."Probabilistic Router" THEN BEGIN
  Decision := (RandomNo-1)/1000;
  CummulativeProbability := 0;
  Found := FALSE;
  RouterValue.RESET;
  RouterValue.SETCURRENTKEY(Probability);
  RouterValue.SETRANGE("Model Code",ModelCode);
  RouterValue.SETRANGE("Shape ID",ShapeID);
  IF RouterValue.FIND('-') THEN
    REPEAT
       CummulativeProbability := CummulativeProbability +
            RouterValue.Probability;
       IF CummulativeProbability > Decision THEN
         Found := TRUE;
       IF NOT Found THEN
         RouterValue.NEXT;
    UNTIL Found;
  EXIT(RouterValue."Connect to Shape ID");
END ELSE BEGIN
  //Check entity with the lest amount of records or with the empty one
  RouterValue.RESET;
  RouterValue.SETRANGE("Model Code",ModelCode);
  RouterValue.SETRANGE("Shape ID",ShapeID);
  MinQueue := 0;
  QueueCount := 0;
  NullCounter := 0;
  FirstTime := TRUE;
  CLEAR(TempRec);
  IF RouterValue.FIND('-') THEN
     REPEAT
       ShapeQueue.RESET;
       ShapeQueue.SETRANGE("Shape ID",RouterValue."Connect to Shape ID");
       //Checks for shapes with no queue and inserts the found ones on a
       //Temp table
       IF NOT ShapeQueue.FIND('-') THEN BEGIN
         TempRec."Model Code" := ModelCode;
         TempRec."Shape ID" := RouterValue."Connect to Shape ID";
         TempRec.INSERT;
```

```
        NullCounter += 1;
      END ELSE BEGIN
        IF NullCounter = 0 THEN BEGIN
          QueueCount := ShapeQueue.COUNT;
          IF FirstTime THEN BEGIN
             MinQueue := QueueCount;
             NextShapeID := RouterValue."Connect to Shape ID";
             FirstTime := FALSE;
          END;
          IF MinQueue > QueueCount THEN BEGIN
            MinQueue := QueueCount;
            NextShapeID := ShapeQueue."Shape ID";
          END;
        END;
      END;
    UNTIL RouterValue.NEXT =0;
    IF NullCounter > 0 THEN BEGIN
       IF NullCounter = 1 THEN BEGIN
         TempRec.FIND('-');
         EXIT(TempRec."Shape ID");
       END ELSE BEGIN
         Decision := RANDOM(NullCounter);
         IF Decision = 1 THEN BEGIN
           TempRec.FIND('-');
           EXIT(TempRec."Shape ID");
         END ELSE BEGIN
             TempRec.NEXT(Decision);
             EXIT(TempRec."Shape ID");
         END;
       END;
    END ELSE
      EXIT(NextShapeID);
END;
```

- Make the possibility of linking model together or that a shape can be link to a model. In this way its going to be possible to link models together and create model of models.

6. Testing

This chapter describes the tests and testing methods performed on the different parts of the simulator prototype. The testing should be viewed as a first attempt in order to guarantee the correctness of the operations of the main parts of the application.

The chapter is divided in two sections that cover the tasks performed on the data fitting and simulation execution functionality respectively.

6.1. Data Fitting

The tests performed in the data fitting functionality are based on the data inserted on the Data Fitting Worksheet Line table. The methodology consists in the creation of data using the NaviMath methods that generate random number variates for different distributions. The columns of the worksheet were populated with data associated with different distributions. Afterwards, a frequency table was constructed for each of them. In this way the random number generators and the frequency table functionality could be tested.

The results of the tests can be depicted in the following graphs constructed from data generated for different distributions.

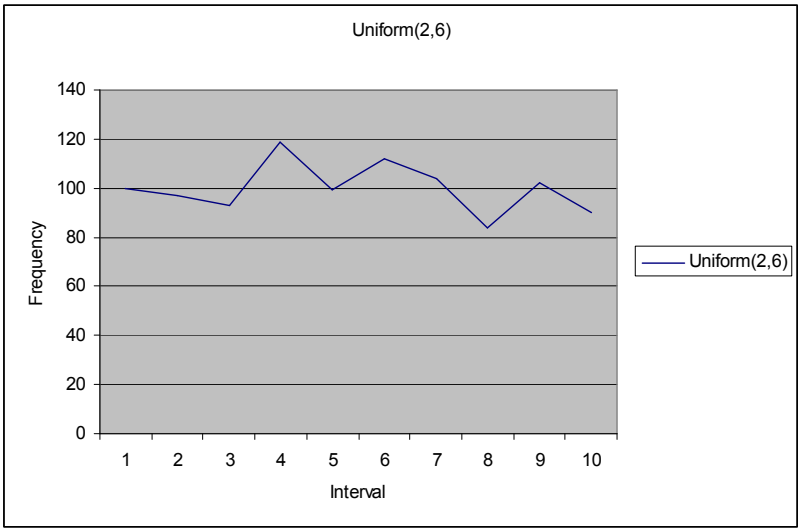


Figure 35 Uniform Distribution Frequency table result

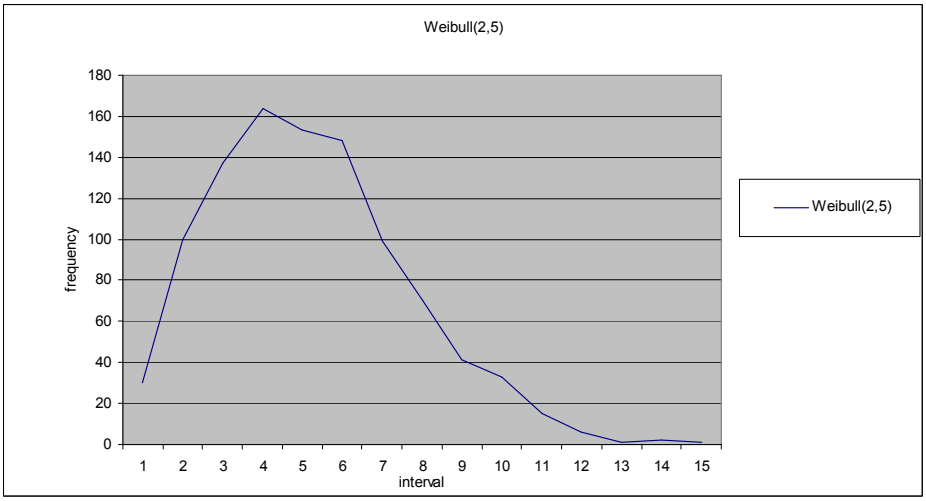


Figure 36 Weibull distribution frequency table result

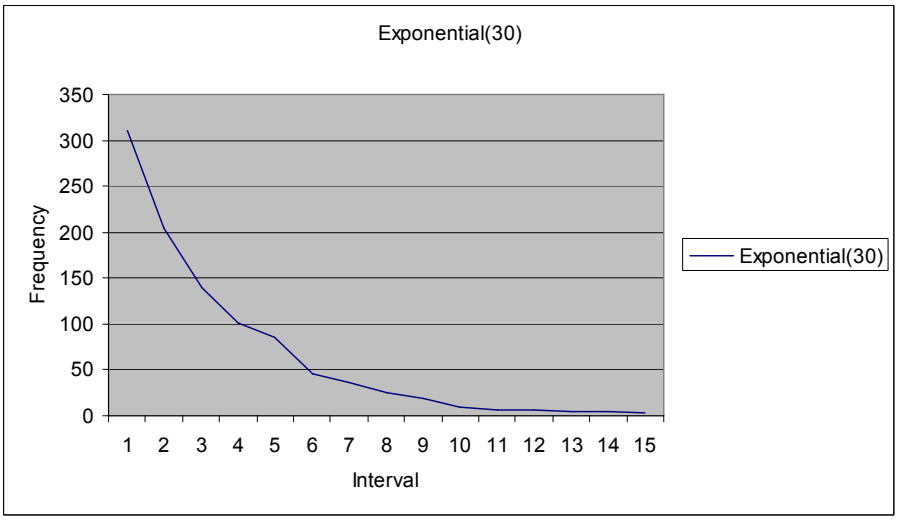


Figure 37 Exponential distribution frequency table result

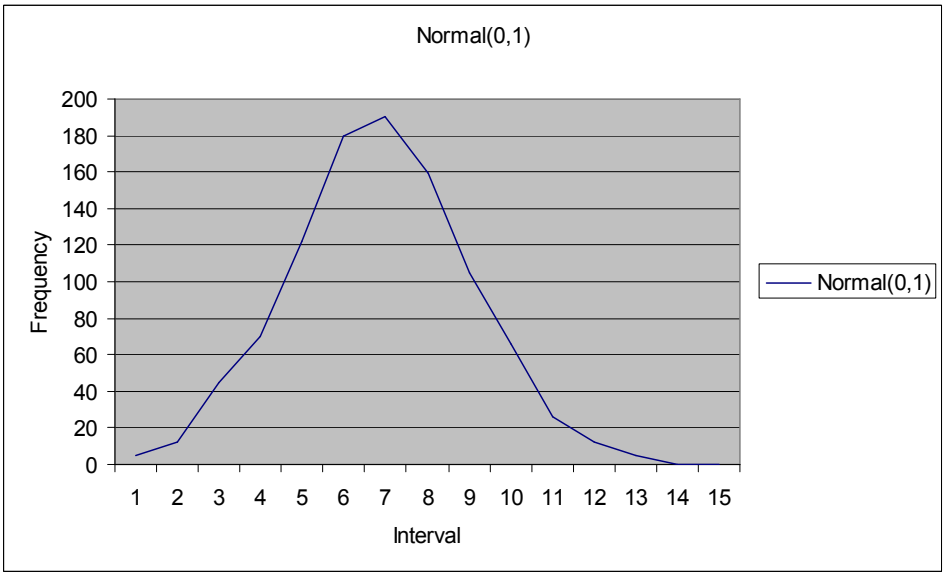


Figure 38 Normal distribution frequency table result

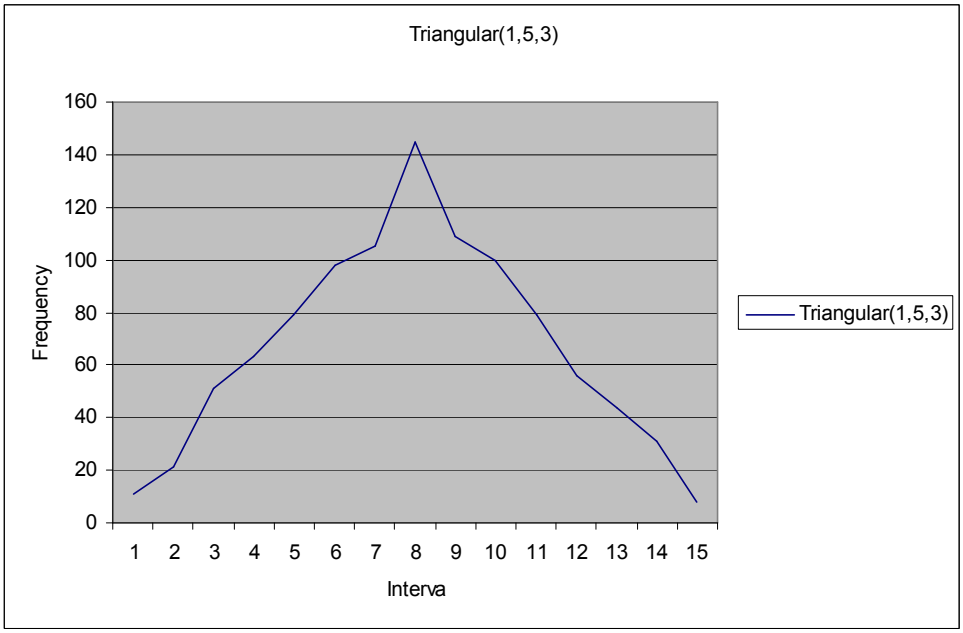


Figure 39 Triangular distribution frequency table

After confirming that the distributions and the frequency table were correctly developed, the interval estimators were tested. The testing consisted in creating a testing table that had interval data, its table fields are the ones that are sought after when performing interval calculations and are shown below:

Field No.	Field Name	Data Type
1	Key	BigInteger
3	Start Date	Date
4	Start Time	Time
5	End Date	Date
6	End Time	Time

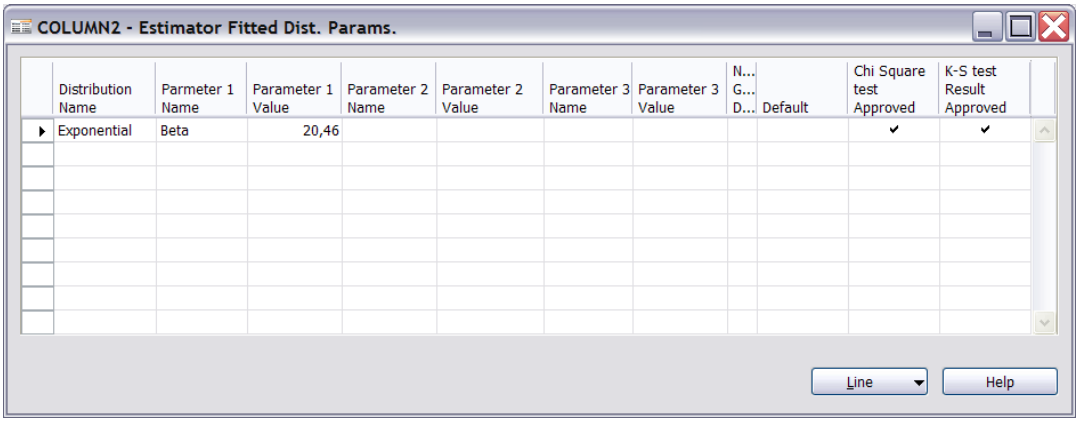
Table 39 Interval Testing table fields

The data for this table was generated from the worksheet line that contained the exponential distribution data points. Each point is a time interval, then, given a starting date and time (the start of an interval), the end date and time of this interval is simply generated by adding the values from the worksheet line to the starting date and time ones. Afterwards, the next record of the testing table has its start

interval values equal to the end ones of the previous record. When the test table was finally populated with some test data, the function for creating the Temp Data Calculation table was executed. The results inserted on the Temp Data Calculation table should be the same as the ones from the column of the worksheet line that contain the exponential points. The script used for creating the Testing Table is shown in appendix A.

6.1.1. Goodness-of-fit tests

The testing for the goodness of fit tests was simply performed by applying the test to a column of the Data Fitting Worksheet Line table that had exponentially distributed data; The result from this test are shown in the figures below, the data used for the test can be found in appendix 2.



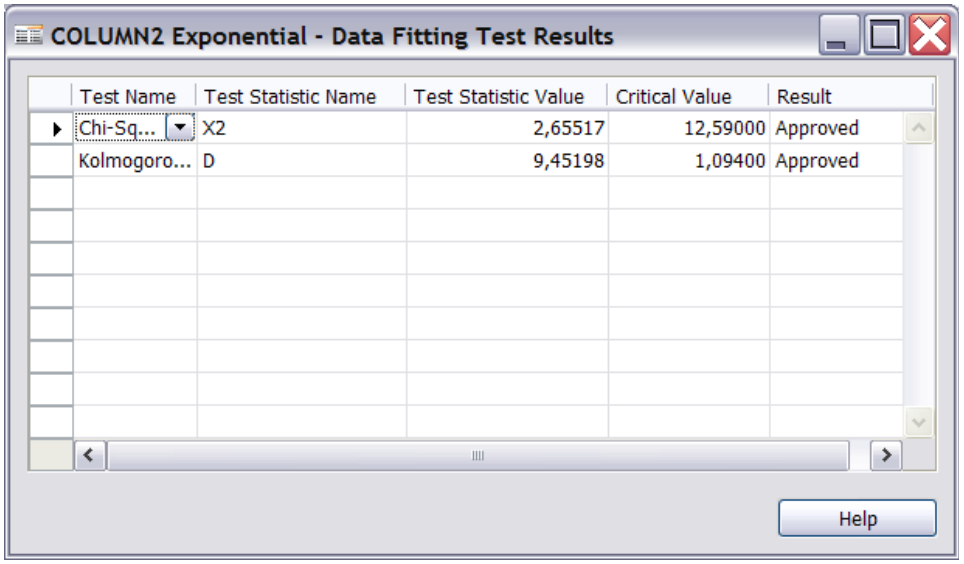


Figure 40, 40 Results of the data fitting performed in exponentially distributed data

6.2. Simulation Execution

The simulation execution testing was performed by looking at the log values from different models. The following models were created:

- Simple 1: one creator shape connected to a process and later connected to a terminator
- Simple 2: one creator shape connected to two sequential processes that are linked to a terminator.
- Router test: one creator shape connected to a router that is connected to two processes.
- Joiner test: two creator shapes connected to a router that is connected to a process.

All the estimators used in these tests were deterministic in order to alleviate the review process done on the simulation log. The creation shapes created one entity at a time and the process shape delayed them for two units of time. The results from parts of the logs can be found on appendix C to F.

For the router shape a small test script was conducted in order to test the router shape behavior. This test consisted in creating some sample data for a model in order to test the probabilistic and queue behavior of the shape. The scripts code can be found on appendix G.

7. Conclusions

In the introduction section of this project there were some questions outlined that were used as the source of inspiration to pursue this investigation. After the completion of the design and the development of the majority of the core functionality of the prototype, it is my belief that these questions have been answered and are given as the conclusions of this project.

Is simulation used in some ERP systems today and if so, in what areas?

Simulation is used to a certain degree in the majority of the top players in the ERP market. Nonetheless, only few vendors implement the functionalities presented in this project and all of them implement the simulation for specific (key?) areas like manufacturing and distribution. Also, all of these vendors target the corporate segment of the market. More importantly, the inclusion of discrete event simulation applications in ERP software for small and medium size companies has not been fully developed yet.

Is it possible to create a simulation module in the ERP system that I have been working on?

The development of a simulation application has been proved to be possible. All the components of a standard simulation application have been developed to some degree. It is important to mention that the simplicity of the application language and the built capabilities of C/Side took care of many complications that could arise when developing a simulator in a less restricted environment. However, if performance issues are found when executing large models, the developer would have a limited set of possibilities to overcome them.

An important suggestion for the application is to include the functionality present in the external components developed. This would make possible the usage of all the available business logic and functionality of the application allowing the creation of very complex models.

What would this system need? What functionality could it have and how could it use the potential that is already part of the ERP software?

By researching some of the simulation literature the needed components were gathered and later developed. The potential of the application was not fully used since the Navision application does not allow the interaction with external components inside the development environment. Nevertheless, the fact that the simulation application was a part of the ERP system allowed a continuous flow between the data analysis and the modeling and execution phases.

What do I think can be done to enhance the simulation application developed?

While developing the project new ideas for functionality or enhancement were thought of. Some of them were implemented while others were just outlined. The main point behind this creative process is that the fusion of simulation with a business application gives a lot of room for inspiration since simulation is a broad concept that can be applied in many areas which can affect the application as a whole.

The following list summarizes some features that could be use to enhance the current prototype:

Data Analysis

- Inclusion of more statistical distributions, especially discrete ones or user defined.
- A better usability for the statistical test results. For instance, a normal user would not be very familiar with terms like: chi-square test, Kolmogorov-smirnov, confidence intervals, statistical distributions and its parameters. The way to present these concepts to a "normal" user can be by using simpler terminology and wizards. For example;
    - Confidence intervals could be presented to the user in a wizard with a question like: How sure do you want to be on the similarities of the data and the test distributions?
    - When inserting parameters for the triangular distribution, instead of asking for the values for a, b and c the system could ask in a wizard: What is the most probable, minimum or maximum values for this estimator?

    - Not show any test results just give the user information about how sure he could be that the data fits a certain distribution.
- Include graphic capabilities to show the user how close the data is with the proposed distribution.
- If users are more knowledgeable in statistics, include the possibility of creating different kinds of statistical graphs. E.g. box plots, probability plots and histograms.

Estimators

- The current process of linking an estimator with the data available in the system presumes that the user has the knowledge of the table relationships and structure. This assumption is not valid for the standard user since the main interaction points he has with the application are the visible forms, menus and reports. A way to target this inconvenience would be to have a functionality linked to form fields that would allow the user to create estimators based on selections of fields performed on the form.
- Currently the user can use the filter capabilities for tables on the estimator card for setting some sort of attributes on the selected estimator. However, it would give more power to the user if he could insert any desired attributes to the estimators. A practical example would be that the router shape could “route” based on the created attributes.
- Make possible to export the data created in the temp data calculation table. In this way the data fitting can be use in an external program or send to another company for investigation. Create a communication interface in order for the external partner to send information that could be easily imported in order to link it to an estimator or create an estimator from.

Modeling

- The modeling tool could be better created as html. In this way it can be possible to create a webpage and use the modeling tool over the internet.
- Animation of the simulation process could help tracking the execution of the simulation.
- When a user creates a model and links the shapes to estimators, basically what he is doing is associating data from a table with logical processes and linking them together. This information could be use by the ERP system to establish links of functionality together, making the modeling process a tool

that the ERP system can use to adapt to the users behavior. Furthermore, the current relations of tables or any recorded step performed by a user could be use a source for creating models.

- There are several process statuses that could be implemented. Some of them are: failed, on setup, starved. An idea would be to create some sort of logic units that would allow the user to give shapes certain behavior linked to user-defined statuses.

7.3. Future Work

The simulation concept and simulation software has been around for many years, most probably, at the same time ERP applications started to be developed. The fact that discrete event simulation packages are not part of the offerings of many of today's ERP vendors, or the ones that do offer them are targeted to the high end of the ERP market, gives a strong indication that simulation is not a widely used tool nowadays.

In my opinion, the main reason for the slow movement of simulation applications into the ERP arena is given by the fact that the simulation process requires a great deal of knowledge in technical areas that is not present in the majority of the consumers that constitutes the ERP market as a whole. For this reason, I believe that simulation would gain a large number of advocates if considerable efforts were made in automating the technical operations performed in a simulation study, specifically, in the data analysis process. If this process is fully automated giving reasonable results, the simulation process would basically be a modeling exercise. This could certainly help the adoption of the concept since, for instance, users that previously rejected it because of its technical barriers would now embrace it since modeling techniques are relatively ease to learn and widely used. As an example we could consider the adoption and offerings on human workflow capabilities were the user basically models the hierarchy of their organizations.

For the above mentioned reasons I believe that the main area of study to focus for some future work would be in the automation of the data analysis process.

To conclude, I have investigated the possibilities for the inclusion of a simulation application in an ERP application. The emphasis has been made on the interaction between the data fitting and the modeling phases of simulation with ease of use as a key design criterion. In my opinion, the project has been quite successful, as the original hypotheses seem to be confirmed. Also within the limited time available it has been possible to develop a quite convincing simulation prototype.

Appendix A Script for creating Interval Data

```
Function InsertTestTable()

TestEstimator.DELETEALL;
DataFitWrkstLine.RESET;
LineNo := 1;
StartDate := 010105D;
StartTime := 000000T;
IF DataFitWrkstLine.FIND('-') THEN BEGIN
  REPEAT
    TestEstimator.Key := LineNo;
    TestEstimator."Start Date" := StartDate;
    TestEstimator."Start Time" := StartTime;
    Days := DataFitWrkstLine."Column 3" * 3.6;
    DTVar := DTMgt.Datetime(StartDate,StartTime);
    DTVar := DTVar + Days;
    TestEstimator."End Date" := DTMgt.Datetime2Date(DTVar);
    TestEstimator."End Time" := DTMgt.Datetime2Time(DTVar);
    StartDate := TestEstimator."End Date";
    StartTime := TestEstimator."End Time";
    TestEstimator.INSERT;
    LineNo += 1;
  UNTIL DataFitWrkstLine.NEXT = 0;
END;
```

Appendix B Exponential data used for testing the goodness-of-fit tests

Line No.	Column 2	Line No.	Column 2	Line No.	Column 2
2	23	36	53	70	11
3	7	37	29	71	55
4	62	38	38	72	83
5	3	39	58	73	6
6	13	40	21	74	32
7	5	41	19	75	10
8	7	42	15	76	21
9	4	43	21	77	2
10	21	44	32	78	0
11	6	45	49	79	32
12	2	46	9	80	1
13	20	47	20	81	29
14	2	48	29	82	13
15	14	49	0	83	4
16	28	50	1	84	1
17	5	51	37	85	27
18	29	52	0	86	16
19	56	53	6	87	56
20	14	54	1	88	7
21	1	55	13		
22	2	56	54		
23	31	57	16		
24	3	58	76		
25	15	59	51		
26	19	60	3		
27	18	61	19		
28	3	62	49		
29	1	63	5		
30	11	64	55		
31	9	65	1		
32	73	66	21		
33	10	67	23		
34	10	68	14		
35	3	69	6		

Appendix C Simple 1 model log

Line No.	Simulation Clock	Description	Date	Time
1	0	Start of simulation...	27-10-2005	21:05:06
2	0	entity 0 has been set for Termination at shape id 0 at time 100	27-10-2005	21:05:06
3	0	entity 1 has been set for Arrival at shape id 1 at time 1	27-10-2005	21:05:06
4	1	entity 1 has been set for Departure at shape id 2 at time 4	27-10-2005	21:05:06
5	1	Shape ID 2 has been set to busy	27-10-2005	21:05:06
6	1	entity 2 has been set for Arrival at shape id 1 at time 2	27-10-2005	21:05:06
7	2	entity 2 has been set on queue for shape 2	27-10-2005	21:05:06
8	2	entity 3 has been set for Arrival at shape id 1 at time 3	27-10-2005	21:05:06
9	3	entity 3 has been set on queue for shape 2	27-10-2005	21:05:06
10	3	entity 4 has been set for Arrival at shape id 1 at time 4	27-10-2005	21:05:06
11	4	entity 2 has been taken from the queue on shape ID 2	27-10-2005	21:05:06
12	4	entity 2 has been set for Departure at shape id 2 at time 7	27-10-2005	21:05:06
13	4	entity 1 has terminated the run	27-10-2005	21:05:06
14	4	entity 4 has been set on queue for shape 2	27-10-2005	21:05:06
15	4	entity 5 has been set for Arrival at shape id 1 at time 5	27-10-2005	21:05:06
16	5	entity 5 has been set on queue for shape 2	27-10-2005	21:05:06
17	5	entity 6 has been set for Arrival at shape id 1 at time 6	27-10-2005	21:05:06
18	6	entity 6 has been set on queue for shape 2	27-10-2005	21:05:06
19	6	entity 7 has been set for Arrival at shape id 1 at time 7	27-10-2005	21:05:06
20	7	entity 3 has been taken from the queue on shape ID 2	27-10-2005	21:05:06
21	7	entity 3 has been set for Departure at shape id 2 at time 10	27-10-2005	21:05:06
22	7	entity 2 has terminated the run	27-10-	21:05:06

			2005	
23	7	entity 7 has been set on queue for shape 2	27-10-2005	21:05:06
24	7	entity 8 has been set for Arrival at shape id 1 at time 8	27-10-2005	21:05:06
25	8	entity 8 has been set on queue for shape 2	27-10-2005	21:05:06
26	8	entity 9 has been set for Arrival at shape id 1 at time 9	27-10-2005	21:05:06
27	9	entity 9 has been set on queue for shape 2	27-10-2005	21:05:06
28	9	entity 10 has been set for Arrival at shape id 1 at time 10	27-10-2005	21:05:06
29	10	entity 4 has been taken from the queue on shape ID 2	27-10-2005	21:05:06
30	10	entity 4 has been set for Departure at shape id 2 at time 13	27-10-2005	21:05:06
31	10	entity 3 has terminated the run	27-10-2005	21:05:06
32	10	entity 10 has been set on queue for shape 2	27-10-2005	21:05:06
33	10	entity 11 has been set for Arrival at shape id 1 at time 11	27-10-2005	21:05:06
34	11	entity 11 has been set on queue for shape 2	27-10-2005	21:05:06
35	11	entity 12 has been set for Arrival at shape id 1 at time 12	27-10-2005	21:05:06
36	12	entity 12 has been set on queue for shape 2	27-10-2005	21:05:06
37	12	entity 13 has been set for Arrival at shape id 1 at time 13	27-10-2005	21:05:06
38	13	entity 5 has been taken from the queue on shape ID 2	27-10-2005	21:05:06
39	13	entity 5 has been set for Departure at shape id 2 at time 16	27-10-2005	21:05:06
40	13	entity 4 has terminated the run	27-10-2005	21:05:06
41	13	entity 13 has been set on queue for shape 2	27-10-2005	21:05:06
42	13	entity 14 has been set for Arrival at shape id 1 at time 14	27-10-2005	21:05:06
43	14	entity 14 has been set on queue for shape 2	27-10-2005	21:05:06

Appendix D Simple 2 model log

Line No.	Simulation Clock	Description	Date	Time
1	0	Start of simulation...	29-10-2005	17:45:51
2	0	entity 0 has been set for Termination at shape id 0 at time 100	29-10-2005	17:45:51
3	0	entity 1 has been set for Arrival at shape id 1 at time 1	29-10-2005	17:45:51
4	1	entity 1 has been set for Departure at shape id 2 at time 5	29-10-2005	17:45:51
5	1	Shape ID 2 has been set to busy	29-10-2005	17:45:51
6	1	entity 2 has been set for Arrival at shape id 1 at time 2	29-10-2005	17:45:51
7	2	entity 2 has been set on queue for shape 2	29-10-2005	17:45:51
8	2	entity 3 has been set for Arrival at shape id 1 at time 3	29-10-2005	17:45:51
9	3	entity 3 has been set on queue for shape 2	29-10-2005	17:45:51
10	3	entity 4 has been set for Arrival at shape id 1 at time 4	29-10-2005	17:45:51
11	4	entity 4 has been set on queue for shape 2	29-10-2005	17:45:51
12	4	entity 5 has been set for Arrival at shape id 1 at time 5	29-10-2005	17:45:51
13	5	entity 1 has been set for Departure at shape id 3 at time 7	29-10-2005	17:45:51
14	5	Shape ID 3 has been set to busy	29-10-2005	17:45:51
15	5	entity 2 has been taken from the queue on shape ID 2	29-10-2005	17:45:51
16	5	entity 2 has been set for Departure at shape id 2 at time 9	29-10-2005	17:45:51
17	5	entity 5 has been set on queue for shape 2	29-10-2005	17:45:51
18	5	entity 6 has been set for Arrival at shape id 1 at time 6	29-10-2005	17:45:51
19	6	entity 6 has been set on queue for shape 2	29-10-2005	17:45:51
20	6	entity 7 has been set for Arrival at shape id 1 at time 7	29-10-2005	17:45:51
21	7	Shape ID 3 has been set to idle again	29-10-2005	17:45:51
22	7	entity 1 has terminated the run	29-10-2005	17:45:51
23	7	entity 7 has been set on queue for shape 2	29-10-2005	17:45:51

24	7	entity 8 has been set for Arrival at shape id 1 at time 8	29-10-2005	17:45:51
25	8	entity 8 has been set on queue for shape 2	29-10-2005	17:45:51
26	8	entity 9 has been set for Arrival at shape id 1 at time 9	29-10-2005	17:45:51
27	9	entity 2 has been set for Departure at shape id 3 at time 11	29-10-2005	17:45:51
28	9	Shape ID 3 has been set to busy	29-10-2005	17:45:51
29	9	entity 3 has been taken from the queue on shape ID 2	29-10-2005	17:45:51
30	9	entity 3 has been set for Departure at shape id 2 at time 13	29-10-2005	17:45:51
31	9	entity 9 has been set on queue for shape 2	29-10-2005	17:45:51
32	9	entity 10 has been set for Arrival at shape id 1 at time 10	29-10-2005	17:45:51
33	10	entity 10 has been set on queue for shape 2	29-10-2005	17:45:51
34	10	entity 11 has been set for Arrival at shape id 1 at time 11	29-10-2005	17:45:51
35	11	Shape ID 3 has been set to idle again	29-10-2005	17:45:51
36	11	entity 2 has terminated the run	29-10-2005	17:45:51
37	11	entity 11 has been set on queue for shape 2	29-10-2005	17:45:51
38	11	entity 12 has been set for Arrival at shape id 1 at time 12	29-10-2005	17:45:51
39	12	entity 12 has been set on queue for shape 2	29-10-2005	17:45:51
40	12	entity 13 has been set for Arrival at shape id 1 at time 13	29-10-2005	17:45:51
41	13	entity 3 has been set for Departure at shape id 3 at time 15	29-10-2005	17:45:51
42	13	Shape ID 3 has been set to busy	29-10-2005	17:45:51
43	13	entity 4 has been taken from the queue on shape ID 2	29-10-2005	17:45:51
44	13	entity 4 has been set for Departure at shape id 2 at time 17	29-10-2005	17:45:51
45	13	entity 13 has been set on queue for shape 2	29-10-2005	17:45:51
46	13	entity 14 has been set for Arrival at shape id 1 at time 14	29-10-2005	17:45:51
47	14	entity 14 has been set on queue for shape 2	29-10-2005	17:45:51
48	14	entity 15 has been set for Arrival at shape id 1 at time 15	29-10-2005	17:45:51
49	15	Shape ID 3 has been set to idle again	29-10-	17:45:51

			2005	
50	15	entity 3 has terminated the run	29-10-2005	17:45:51
51	15	entity 15 has been set on queue for shape 2	29-10-2005	17:45:51
52	15	entity 16 has been set for Arrival at shape id 1 at time 16	29-10-2005	17:45:51
53	16	entity 16 has been set on queue for shape 2	29-10-2005	17:45:51
54	16	entity 17 has been set for Arrival at shape id 1 at time 17	29-10-2005	17:45:51
55	17	entity 4 has been set for Departure at shape id 3 at time 19	29-10-2005	17:45:51
56	17	Shape ID 3 has been set to busy	29-10-2005	17:45:51
57	17	entity 5 has been taken from the queue on shape ID 2	29-10-2005	17:45:51
58	17	entity 5 has been set for Departure at shape id 2 at time 21	29-10-2005	17:45:51
59	17	entity 17 has been set on queue for shape 2	29-10-2005	17:45:51
60	17	entity 18 has been set for Arrival at shape id 1 at time 18	29-10-2005	17:45:51
61	18	entity 18 has been set on queue for shape 2	29-10-2005	17:45:51
62	18	entity 19 has been set for Arrival at shape id 1 at time 19	29-10-2005	17:45:51
63	19	Shape ID 3 has been set to idle again	29-10-2005	17:45:51
64	19	entity 4 has terminated the run	29-10-2005	17:45:51
65	19	entity 19 has been set on queue for shape 2	29-10-2005	17:45:51
66	19	entity 20 has been set for Arrival at shape id 1 at time 20	29-10-2005	17:45:51
67	20	entity 20 has been set on queue for shape 2	29-10-2005	17:45:51
68	20	entity 21 has been set for Arrival at shape id 1 at time 21	29-10-2005	17:45:51
69	21	entity 5 has been set for Departure at shape id 3 at time 23	29-10-2005	17:45:51

Appendix E Joiner Test model log

Line No.	Simulation Clock	Description	Date	Time
1	0	Start of simulation...	29-10-2005	4:44:42
2	0	entity 0 has been set for Termination at shape id 0 at time 100	29-10-2005	4:44:42
3	0	entity 1 has been set for Arrival at shape id 1 at time 1	29-10-2005	4:44:42
4	0	entity 2 has been set for Arrival at shape id 2 at time 1	29-10-2005	4:44:42
5	1	entity 3 has been set for Arrival at shape id 1 at time 2	29-10-2005	4:44:42
6	1	Entity ID 4 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
7	1	entity 4 has been set for Departure at shape id 4 at time 5	29-10-2005	4:44:42
8	1	Shape ID 4 has been set to busy	29-10-2005	4:44:42
9	1	entity 5 has been set for Arrival at shape id 2 at time 2	29-10-2005	4:44:42
10	2	entity 3 has been set on queue for shape 3	29-10-2005	4:44:42
11	2	entity 6 has been set for Arrival at shape id 1 at time 3	29-10-2005	4:44:42
12	2	Entity ID 7 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
13	2	entity 7 has been set on queue for shape 4	29-10-2005	4:44:42
14	2	entity 8 has been set for Arrival at shape id 2 at time 3	29-10-2005	4:44:42
15	3	entity 6 has been set on queue for shape 3	29-10-2005	4:44:42
16	3	entity 9 has been set for Arrival at shape id 1 at time 4	29-10-2005	4:44:42
17	3	Entity ID 10 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
18	3	entity 10 has been set on queue for shape 4	29-10-2005	4:44:42
19	3	entity 11 has been set for Arrival at shape id 2 at time 4	29-10-2005	4:44:42
20	4	entity 9 has been set on queue for shape 3	29-10-2005	4:44:42
21	4	entity 12 has been set for Arrival at shape id 1 at time 5	29-10-2005	4:44:42
22	4	Entity ID 13 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42

23	4	entity 13 has been set on queue for shape 4	29-10-2005	4:44:42
24	4	entity 14 has been set for Arrival at shape id 2 at time 5	29-10-2005	4:44:42
25	5	entity 7 has been taken from the queue on shape ID 4	29-10-2005	4:44:42
26	5	entity 7 has been set for Departure at shape id 4 at time 9	29-10-2005	4:44:42
27	5	entity 4 has terminated the run	29-10-2005	4:44:42
28	5	entity 12 has been set on queue for shape 3	29-10-2005	4:44:42
29	5	entity 15 has been set for Arrival at shape id 1 at time 6	29-10-2005	4:44:42
30	5	Entity ID 16 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
31	5	entity 16 has been set on queue for shape 4	29-10-2005	4:44:42
32	5	entity 17 has been set for Arrival at shape id 2 at time 6	29-10-2005	4:44:42
33	6	entity 15 has been set on queue for shape 3	29-10-2005	4:44:42
34	6	entity 18 has been set for Arrival at shape id 1 at time 7	29-10-2005	4:44:42
35	6	Entity ID 19 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
36	6	entity 19 has been set on queue for shape 4	29-10-2005	4:44:42
37	6	entity 20 has been set for Arrival at shape id 2 at time 7	29-10-2005	4:44:42
38	7	entity 18 has been set on queue for shape 3	29-10-2005	4:44:42
39	7	entity 21 has been set for Arrival at shape id 1 at time 8	29-10-2005	4:44:42
40	7	Entity ID 22 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
41	7	entity 22 has been set on queue for shape 4	29-10-2005	4:44:42
42	7	entity 23 has been set for Arrival at shape id 2 at time 8	29-10-2005	4:44:42
43	8	entity 21 has been set on queue for shape 3	29-10-2005	4:44:42
44	8	entity 24 has been set for Arrival at shape id 1 at time 9	29-10-2005	4:44:42
45	8	Entity ID 25 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
46	8	entity 25 has been set on queue for shape 4	29-10-2005	4:44:42
47	8	entity 26 has been set for Arrival at shape id 2 at time 9	29-10-2005	4:44:42
48	9	entity 10 has been taken from the queue on shape ID 4	29-10-	

			2005	4:44:42
49	9	entity 10 has been set for Departure at shape id 4 at time 13	29-10-2005	4:44:42
50	9	entity 7 has terminated the run	29-10-2005	4:44:42
51	9	entity 24 has been set on queue for shape 3	29-10-2005	4:44:42
52	9	entity 27 has been set for Arrival at shape id 1 at time 10	29-10-2005	4:44:42
53	9	Entity ID 28 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
54	9	entity 28 has been set on queue for shape 4	29-10-2005	4:44:42
55	9	entity 29 has been set for Arrival at shape id 2 at time 10	29-10-2005	4:44:42
56	10	entity 27 has been set on queue for shape 3	29-10-2005	4:44:42
57	10	entity 30 has been set for Arrival at shape id 1 at time 11	29-10-2005	4:44:42
58	10	Entity ID 31 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
59	10	entity 31 has been set on queue for shape 4	29-10-2005	4:44:42
60	10	entity 32 has been set for Arrival at shape id 2 at time 11	29-10-2005	4:44:42
61	11	entity 30 has been set on queue for shape 3	29-10-2005	4:44:42
62	11	entity 33 has been set for Arrival at shape id 1 at time 12	29-10-2005	4:44:42
63	11	Entity ID 34 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
64	11	entity 34 has been set on queue for shape 4	29-10-2005	4:44:42
65	11	entity 35 has been set for Arrival at shape id 2 at time 12	29-10-2005	4:44:42
66	12	entity 33 has been set on queue for shape 3	29-10-2005	4:44:42
67	12	entity 36 has been set for Arrival at shape id 1 at time 13	29-10-2005	4:44:42
68	12	Entity ID 37 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
69	12	entity 37 has been set on queue for shape 4	29-10-2005	4:44:42
70	12	entity 38 has been set for Arrival at shape id 2 at time 13	29-10-2005	4:44:42
71	13	entity 13 has been taken from the queue on shape ID 4	29-10-2005	4:44:42
72	13	entity 13 has been set for Departure at shape id 4 at time 17	29-10-2005	4:44:42
73	13	entity 10 has terminated the run	29-10-2005	4:44:42

74	13	entity 36 has been set on queue for shape 3	29-10-2005	4:44:42
75	13	entity 39 has been set for Arrival at shape id 1 at time 14	29-10-2005	4:44:42
76	13	Entity ID 40 has been created on Joiner no 3, The joiner has been flushed	29-10-2005	4:44:42
77	13	entity 40 has been set on queue for shape 4	29-10-2005	4:44:42
78	13	entity 41 has been set for Arrival at shape id 2 at time 14	29-10-2005	4:44:42
79	14	entity 39 has been set on queue for shape 3	29-10-2005	4:44:42
80	14	entity 42 has been set for Arrival at shape id 1 at time 15	29-10-2005	4:44:42

Appendix F Router Test model log

Line No.	Simulation Clock	Description	Date	Time
1	0	Start of simulation...	29-10-2005	4:46:02
2	0	entity 0 has been set for Termination at shape id 0 at time 120	29-10-2005	4:46:02
3	0	entity 1 has been set for Arrival at shape id 1 at time 3	29-10-2005	4:46:02
4	3	The router 2 chose to route to shape no: 3	29-10-2005	4:46:02
5	3	entity 1 has been set for Departure at shape id 3 at time 7	29-10-2005	4:46:02
6	3	Shape ID 3 has been set to busy	29-10-2005	4:46:02
7	3	entity 2 has been set for Arrival at shape id 1 at time 6	29-10-2005	4:46:02
8	6	The router 2 chose to route to shape no: 3	29-10-2005	4:46:02
9	6	entity 2 has been set on queue for shape 3	29-10-2005	4:46:02
10	6	entity 3 has been set for Arrival at shape id 1 at time 9	29-10-2005	4:46:02
11	7	entity 2 has been taken from the queue on shape ID 3	29-10-2005	4:46:02
12	7	entity 2 has been set for Departure at shape id 3 at time 11	29-10-2005	4:46:02
13	7	entity 1 has terminated the run	29-10-2005	4:46:02
14	9	The router 2 chose to route to shape no: 4	29-10-2005	4:46:02
15	9	entity 3 has been set for Departure at shape id 4 at time 11	29-10-2005	4:46:02
16	9	Shape ID 4 has been set to busy	29-10-2005	4:46:02
17	9	entity 4 has been set for Arrival at shape id 1 at time 12	29-10-2005	4:46:02
18	11	Shape ID 3 has been set to idle again	29-10-2005	4:46:02
19	11	entity 2 has terminated the run	29-10-2005	4:46:02
20	11	Shape ID 4 has been set to idle again	29-10-2005	4:46:02
21	11	entity 3 has terminated the run	29-10-2005	4:46:02
22	12	The router 2 chose to route to shape no: 4	29-10-2005	4:46:02

23	12	entity 4 has been set for Departure at shape id 4 at time 14	29-10-2005	4:46:02
24	12	Shape ID 4 has been set to busy	29-10-2005	4:46:02
25	12	entity 5 has been set for Arrival at shape id 1 at time 15	29-10-2005	4:46:02
26	14	Shape ID 4 has been set to idle again	29-10-2005	4:46:02
27	14	entity 4 has terminated the run	29-10-2005	4:46:02
28	15	The router 2 chose to route to shape no: 4	29-10-2005	4:46:02
29	15	entity 5 has been set for Departure at shape id 4 at time 17	29-10-2005	4:46:02
30	15	Shape ID 4 has been set to busy	29-10-2005	4:46:02
31	15	entity 6 has been set for Arrival at shape id 1 at time 18	29-10-2005	4:46:02
32	17	Shape ID 4 has been set to idle again	29-10-2005	4:46:02
33	17	entity 5 has terminated the run	29-10-2005	4:46:02
34	18	The router 2 chose to route to shape no: 3	29-10-2005	4:46:02
35	18	entity 6 has been set for Departure at shape id 3 at time 22	29-10-2005	4:46:02
36	18	Shape ID 3 has been set to busy	29-10-2005	4:46:02
37	18	entity 7 has been set for Arrival at shape id 1 at time 21	29-10-2005	4:46:02
38	21	The router 2 chose to route to shape no: 3	29-10-2005	4:46:02
39	21	entity 7 has been set on queue for shape 3	29-10-2005	4:46:02
40	21	entity 8 has been set for Arrival at shape id 1 at time 24	29-10-2005	4:46:02
41	22	entity 7 has been taken from the queue on shape ID 3	29-10-2005	4:46:03
42	22	entity 7 has been set for Departure at shape id 3 at time 26	29-10-2005	4:46:03
43	22	entity 6 has terminated the run	29-10-2005	4:46:03
44	24	The router 2 chose to route to shape no: 3	29-10-2005	4:46:03
45	24	entity 8 has been set on queue for shape 3	29-10-2005	4:46:03
46	24	entity 9 has been set for Arrival at shape id 1 at time 27	29-10-2005	4:46:03
47	26	entity 8 has been taken from the queue on shape ID 3	29-10-2005	4:46:03
48	26	entity 8 has been set for Departure at shape id 3 at time 30	29-10-	4:46:03

			2005	
49	26	entity 7 has terminated the run	29-10-2005	4:46:03
50	27	The router 2 chose to route to shape no: 3	29-10-2005	4:46:03
51	27	entity 9 has been set on queue for shape 3	29-10-2005	4:46:03
52	27	entity 10 has been set for Arrival at shape id 1 at time 30	29-10-2005	4:46:03
53	30	entity 9 has been taken from the queue on shape ID 3	29-10-2005	4:46:03
54	30	entity 9 has been set for Departure at shape id 3 at time 34	29-10-2005	4:46:03
55	30	entity 8 has terminated the run	29-10-2005	4:46:03
56	30	The router 2 chose to route to shape no: 4	29-10-2005	4:46:03
57	30	entity 10 has been set for Departure at shape id 4 at time 32	29-10-2005	4:46:03
58	30	Shape ID 4 has been set to busy	29-10-2005	4:46:03
59	30	entity 11 has been set for Arrival at shape id 1 at time 33	29-10-2005	4:46:03
60	32	Shape ID 4 has been set to idle again	29-10-2005	4:46:03
61	32	entity 10 has terminated the run	29-10-2005	4:46:03
62	33	The router 2 chose to route to shape no: 4	29-10-2005	4:46:03
63	33	entity 11 has been set for Departure at shape id 4 at time 35	29-10-2005	4:46:03
64	33	Shape ID 4 has been set to busy	29-10-2005	4:46:03
65	33	entity 12 has been set for Arrival at shape id 1 at time 36	29-10-2005	4:46:03
66	34	Shape ID 3 has been set to idle again	29-10-2005	4:46:03
67	34	entity 9 has terminated the run	29-10-2005	4:46:03
68	35	Shape ID 4 has been set to idle again	29-10-2005	4:46:03
69	35	entity 11 has terminated the run	29-10-2005	4:46:03
70	36	The router 2 chose to route to shape no: 3	29-10-2005	4:46:03
71	36	entity 12 has been set for Departure at shape id 3 at time 40	29-10-2005	4:46:03
72	36	Shape ID 3 has been set to busy	29-10-2005	4:46:03
73	36	entity 13 has been set for Arrival at shape id 1 at time 39	29-10-2005	4:46:03

74	39	The router 2 chose to route to shape no: 3	29-10-2005	4:46:03
75	39	entity 13 has been set on queue for shape 3	29-10-2005	4:46:03
76	39	entity 14 has been set for Arrival at shape id 1 at time 42	29-10-2005	4:46:03
77	40	entity 13 has been taken from the queue on shape ID 3	29-10-2005	4:46:03
78	40	entity 13 has been set for Departure at shape id 3 at time 44	29-10-2005	4:46:03
79	40	entity 12 has terminated the run	29-10-2005	4:46:03
80	42	The router 2 chose to route to shape no: 3	29-10-2005	4:46:03

Appendix G Test Script for the Router Shape

Function TestNonEmptyQueueRouter()

```
SimModelDetail."Model Code" := 'TestRouter12345';
SimModelDetail."Shape ID" := 1 ;
SimModelDetail."Probabilistic Router" := FALSE;
SimModelDetail.INSERT;

FOR ShapeID := 2 TO 4 DO BEGIN
  ModelRouterValue."Model Code" := 'TestRouter12345';
  ModelRouterValue."Shape ID" := 1;
  ModelRouterValue."Connect to Shape ID" := ShapeID;
  ModelRouterValue.INSERT;
END;

ShapeQueue.DELETEALL;
EntityID := 0;
FOR ShapeID := 2 TO 4 DO BEGIN
  EntityID += 1;
  CASE ShapeID OF
    2:FOR x := 1 TO 20 DO BEGIN
        InsertEntityOnQueue(ShapeID,EntityID);
      END;
    3:FOR x := 1 TO 10 DO BEGIN
        InsertEntityOnQueue(ShapeID,EntityID);
      END;
    4:FOR x := 1 TO 5 DO BEGIN
        InsertEntityOnQueue(ShapeID,EntityID);
      END;
  END;
END;
SimEngine.SetModelCode('TestRouter12345');
MESSAGE('Next                    Shape                    selected:
'+FORMAT(SimEngine.GetNextShapeFromRouter(1)));
```

Function TestEmptyQueueRouter()

```
SimModelDetail.SETRANGE("Model Code",'TestRouter12345');
SimModelDetail.DELETEALL;
SimModelDetail."Model Code" := 'TestRouter12345';
SimModelDetail."Shape ID" := 1 ;
SimModelDetail."Probabilistic Router" := FALSE;
SimModelDetail.INSERT;
```

```
FOR ShapeID := 2 TO 4 DO BEGIN
  ModelRouterValue."Model Code" := 'TestRouter12345';
  ModelRouterValue."Shape ID" := 1;
  ModelRouterValue."Connect to Shape ID" := ShapeID;
  ModelRouterValue.INSERT;
END;
SimEngine.SetModelCode('TestRouter12345');
FOR x := 1 TO 1 DO BEGIN
  MESSAGE('Next                    Shape                    selected:
'+FORMAT(SimEngine.GetNextShapeFromRouter(1)));
END;
```

Function TestProbRouter()

```
SimModelDetail.SETRANGE("Model Code",'TestRouter12345');
SimModelDetail.DELETEALL;
SimModelDetail."Model Code" := 'TestRouter12345';
SimModelDetail."Shape ID" := 1 ;
SimModelDetail."Probabilistic Router" := TRUE;
SimModelDetail.INSERT;

FOR ShapeID := 2 TO 4 DO BEGIN
  ModelRouterValue."Model Code" := 'TestRouter12345';
  ModelRouterValue."Shape ID" := 1;
  CASE ShapeID OF
    2:ModelRouterValue.Probability := 0.125;
    3:ModelRouterValue.Probability := 0.325;
    4:ModelRouterValue.Probability := 0.550;
  END;
  ModelRouterValue."Connect to Shape ID" := ShapeID;
  ModelRouterValue.INSERT;
END;
SimEngine.SetModelCode('TestRouter12345');
FOR x := 1 TO 5 DO BEGIN
  Rnd := RANDOM(1001);
  MESSAGE(FORMAT((Rnd-1)/1000));
  SimEngine.SetRandomNo(Rnd);
  MESSAGE('Next                    Shape                    selected:
'+FORMAT(SimEngine.GetNextShapeFromRouter(1)));
END;

InsertEntityOnQueue(ShapeID : Integer;EntityID : Integer)
ShapeQueue.RESET;
ShapeQueue.SETRANGE("Shape ID",ShapeID);
IF ShapeQueue.FIND('+') THEN
  LineNo := ShapeQueue."Line No." + 1
ELSE
  LineNo := 1;
```

```
ShapeQueue.INIT;
ShapeQueue."Shape ID" := ShapeID;
ShapeQueue."Line No." := LineNo;
ShapeQueue."Entity ID" := EntityID;
ShapeQueue.INSERT;
```

References

[1] Law, A.M and Kelton, W. David, Simulation Modeling and Analysis Second Edition, 1991, McGraw Hill Inc.

[2] Banks, Jerry and Carson John S., Discrete-Event Simulation, Prentice-Hall Inc.

[3] David W Sabo, "Frequency Tables". http://www.math.bcit.ca/faculty/david_sabo/apples/math2441/section2/freqtables/freqtables.htm

[4] New for August 2002, Textile/Clothing Technology Corporation, 2005. Web Resource :http://www.techexchange.com/BreakingNews/arc/0802.html

[5] Business Intelligence marketing material, Sap AS., 2005. Web Resouce: http://www.sap.com/solutions/netweaver/businessintelligence/index.epx

[6] Supply Chain Management, SSA global technologies, inc, 2005. Web Resource: http://www.ssaglobal.com/solutions/scm/manscheduling.aspx

[7] Eric W. Weisstein. "Normal Distribution." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/NormalDistribution.html

[8] Eric W. Weisstein. "Central Limit Theorem." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/CentralLimitTheorem.html

[9] The Weibull Distribution, *NIST/SEMATECH e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/eda/section3/eda3668.htm, 2005.

[10] History of Navision, Accounting Software advisor, 2005 Web Resource.http://www.mbsadvisor.com/navision/history.htm

[11] ERP software, Wikipedia the On-line encyclopedia, 2005
Web Resource: http://en.wikipedia.org/wiki/ERP_software.
[12]Microsoft Business Solutions web site, 2005.
Web Resource: http://www.microsoft.com/BusinessSolutions
[13] Simulation, Sage, 2005.
Web Resource: http://www.sage.de/public2/smb/prodloes/office_line/
produktion_simulation.asp
[14] Peoplesoft Workforce Rewards, Oracle 2005,
Web Resource: http://www.oracle.com/applications/performance-management/
ent/module/wrkfrc_rewards.html
[15] Oracle BPEL Process Manager,
Web Resource: http://www.oracle.com/technology/products/ias/bpel/index.html