FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Avaliação de Configurações de Sistemas de Produção

Tiago Ramos Duarte Gomes
MIEEC 2008

U.PORTO MIEEC Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Faculdade de Engenharia da Universidade do Porto

FEUP

Avaliação de Configurações de Sistemas de Produção

Tiago Ramos Duarte Gomes

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Prof. Dr. Américo Lopes de Azevedo

Julho de 2008

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"Avaliação de Configurações de Sistemas de Produção"

foi aprovada em provas realizadas em 16/Julho/2008

o júri

Presidente Professor Doutor José António Rodrigues Pereira de Faria
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

Professora Doutora Anabela Carvalho Alves
Professora Auxiliar da Escola de Engenharia da Universidade do Minho

Professor Doutor Américo Lopes de Azevedo
Professor Associado da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Auto - Tiago Ramos Duarte Gomes

Faculdade de Engenharia da Universidade do Porto

Resumo

As actuais exigências de produtos de qualidade elevada a preços reduzidos num mercado global, obrigam a que, cada vez mais, qualquer sistema produtivo ou fabril tenha o máximo possível de eficiência em todas as suas fases (desenho, implementação e funcionamento). A simulação surge como uma ferramenta capaz de testar novas configurações do processo produtivo, explorar novas políticas de escalonamento de recursos e procedimentos operativos, sem comprometer tais recursos ou interromper o funcionamento normal do referido sistema. A simulação é, assim, uma tecnologia capaz de oferecer aos sistemas de produção actuais, a melhoria continua e optimização necessária nos mercados de hoje e amanhã.

O trabalho aqui desenvolvido encontra-se inserido num projecto que tem como objectivo desenvolver um conjunto de ferramentas de simulação, que permitam reduzir o tempo e custos de concepção e desenvolvimento de sistemas automáticos de movimentação e armazenamento de produtos em curso de fabrico. De facto, o projecto em questão propõe-se construir uma biblioteca de componentes de simulação de sistemas de logística interna, capaz de diminuir de forma substancial o esforço de desenvolvimento deste tipo de sistemas

Esta dissertação concentra-se na construção de dois componentes desta biblioteca: elementos que permitam criar, tanto a lógica como a animação de sistemas de movimentação automáticos industriais, baseados em tapetes com diferentes tipos de cruzamentos e ligações entre si.

Pretende-se obter, assim, uma ferramenta que combate, através da criação automática de modelos de simulação, os principais problemas inerentes à utilização das tecnologias de simulação: o tempo e o custo da construção de modelos, as dificuldades da sua validação e a sua pouca flexibilidade.

Abstract

The demand for products with high quality standards and low prices in today's global markets, puts much pressure on increasing efficiency in all stages of a manufacturing system's life cycle(design, implementation and operation). Simulation rises as a tool capable of testing new configurations of a manufacturing system, exploiting new resource usage polices and experimenting different operation procedures, without compromising resources or interrupting the manufacturing system's regular operation. Simulation is therefore, a technology capable of offering to manufacturing systems, the continuous improvement and optimization required in today's and tomorrow's markets.

The work developed here, is part of a project, that has it's purpose set on creating a set of simulation tools that will bring time gains and reduce conception and development costs of automatic movement and storage systems, existent in manufacturing systems. The main goal is to build a library of components that can adequately simulate internal logistics systems, as to reduce costs and efforts required for the development of such systems.

This dissertation will focus on the creation of two components for this library: elements that will allow to create both logic and animation of automatic movement systems, based in conveyor belts and different intersections formed in their different connections.

Consequently , the work developed here aims to fight, through automatic creation of simulation models, the challenges that keep simulation implementation in check: the time consumed and costs associated to the construction of simulation models, the difficulties of it's validation and low flexibility of models created.

Agradecimentos

Aos meus pais, por sempre me incentivarem e acreditarem em mim.

Aos meus amigos, sem os quais eu não teria chegado tão longe...vocês sabem quem são!

À minha namorada, pela sua paciência e carinho, mesmo nos meus piores momentos.

Aos professores Américo Lopes de Azevedo e José António Rodrigues Pereira de Faria, as suas orientações e paciência para comigo, nunca serão esquecidas.

E, por fim, ao INESC, e em especial ao Eng. Paulo Jorge da Rocha e Silva Sá Marques, cuja experiência na utilização do ARENA valeu por uma dúzia de livros.

A todos, os meus profundos agradecimentos, sem vocês este trabalho não existiria.

Índice

Lista de figuras

Lista de tabelas

Capítulo 1

Introdução

Este capítulo contextualiza o trabalho desenvolvido na temática em que este se insere: simulação de sistemas discretos. São apresentados os objectivos do trabalho assim como a metodologia seguida na sua realização.
Conclui-se o capítulo com explicação da estrutura e organização de todo o documento.

1.1 - Enquadramento do trabalho

Actualmente as empresas e organizações são confrontadas, diariamente, com uma competição crescente que é imposta pela globalização dos mercados, com uma procura cada vez mais exigente e cenários políticos e económicos muito instáveis e complexos. Para se manter competitiva, uma empresa tem de apresentar qualidade elevada nos seus produtos, oferecer preços competitivos, ter um tempo de resposta mínimo, ser flexível e estar constantemente a inovar, não só a nível de produto mas também ao nível dos processos de produção.

A dominância da procura relativamente à oferta, a progressiva orientação total ao cliente e as novas oportunidades à escala mundial trazidas pela evolução das novas tecnologias, obrigam a fabricar produtos cada vez mais complexos, de alta qualidade, produzidos em pequenas séries e com ciclo de vida cada vez mais curto. Surge então a grande meta dos sistemas produtivos actuais, a *World Class Manufacturing*: "O Fabrico por encomenda, com zero defeitos, custos mínimos, o menor tempo de resposta possível e entrega *just-in-time*. Ter capacidade de desenvolvimento rápido e lançamento de novos produtos que respondam integralmente às expectativas e requisitos do mercado" [1].

É assim fundamental, mais do que nunca, que as empresas procurem ter uma gestão eficiente da sua capacidade, recursos e processos. O aumento da eficiência e qualidade em todas as fases do ciclo de vida dos sistemas produtivos nunca foi tão importante. Para

aumentar esta eficiência é necessário minimizar tempos e custos de concepção, desenvolvimento e operação, criando assim a necessidade de ser capaz de analisar e avaliar cenários de operação de sistemas de produção antes de serem implementados. Além disso, a diminuição do tempo de vida dos produtos nos mercados de hoje obriga a que os sistemas de produção sejam altamente flexíveis, de modo a suportar constantes mudanças em períodos de tempo reduzido. Já não é suficiente ser capaz de responder às mudanças quando elas ocorrem [2]. Estas mudanças devem ser previstas e preparadas com antecedência.

Com a evolução do desempenho computacional das tecnologias de informação, a simulação surge como uma ferramenta capaz de testar novas configurações do processo produtivo e explorar novas politicas de escalonamento de recursos e procedimentos operativos sem comprometer recursos ou interromper o funcionamento normal do referido sistema. Para além de ser uma ferramenta de optimização, as capacidades de compressão ou expansão temporal (simular meses em minutos ou horas por exemplo) oferecidas pela simulação, dão repostas a questões do tipo "E se?", fazendo da simulação uma ferramenta de apoio à decisão no âmbito da gestão industrial. A simulação é assim uma tecnologia passível de ser utilizada como uma ferramenta que possibilita a melhoria continua, necessária à competitividade nos mercados de amanhã.

Melhoria continua e reengenharia de processos, juntamente com iniciativas de adopção de práticas em conformidade com referências normativas (como por exemplo ISO9001), têm motivado empresas e outras organizações a procurar maneiras de controlar, documentar e comunicar as suas operações. Gestores de organizações líder vêm nas tecnologias de simulação uma maneira de explorar alterações e alternativas de negócio antes de qualquer aplicação [3].

Aliás, em sistemas produtivos, a simulação já é usada para resolver questões como:

- Número e tipo de máquinas para determinado objectivo;
- Número, tipo e disposição de transportadores, tapetes e outro equipamento de suporte; no chão de fábrica;
- Localização e dimensionamento de *buffers* nas linhas de produção;
- Avaliação de novos equipamentos e investimentos;
- Avaliação de alterações nos volumes de produção;
- Análise de fluxo, performance e pontos de estrangulamento;
- Estudo de stocks e inventários;

No entanto, esta tecnologia ainda enfrenta diversos desafios para ser aceite como ferramenta no mundo dos sistemas produtivos. Actualmente, apesar de se lhe reconhecer um enorme potencial como ferramenta de suporte de tomada de decisões, o tempo e custo da

construção de modelos e sua validação por um lado e a pouca flexibilidade dos mesmos por outro, atentam contra a sua efectiva aplicação.

O trabalho realizado nesta dissertação centra-se precisamente na necessidade de se desenvolver soluções que automatizem o processo de criação de modelos de simulação, tendo em vista o aumento de eficiência global do processo de simulação em sistemas produtivos.

1.2 - Objectivos

Este trabalho está integrado num projecto mais vasto que tem como objectivo, entre outros, o desenvolvimento de um conjunto de componentes a utilizar em ferramentas de simulação, no apoio ao projecto e desenvolvimento de soluções de automação da logística interna de empresas.

Este projecto[1] tem como alvo de estudo sistemas de logística interna que permitem automatizar a movimentação e armazenamento de matéria prima e produtos em curso de fabrico num ambiente industrial.

Pretende-se assim reduzir significativamente o esforço de concepção e de desenvolvimento deste tipo de sistemas, através da criação de componentes que permitam simular o hardware dos sistemas de movimentação utilizados e uma biblioteca de componentes com funções de automação de sistemas logísticos, tendo em vista o seu reaproveitamento em construção futura de outras instalações.

Este trabalho tem por base outro projecto de simulação realizado pelo mesmo investigador[2]. Neste último era estudada a implementação de um sistema de movimentação de contentores com produtos em curso de fabrico numa empresa de calçado portuguesa.

A empresa onde se realizou o referido projecto de simulação, implementou um sistema de movimentação completamente automático, constituído por uma vasta rede de tapetes e torres elevatórias, que efectuam a interface entre os armazéns e as secções de fabrico, com o objectivo de responder as exigências impostas pelos mercados actuais.

Após a implementação deste sistema foi construído um modelo de simulação, utilizando uma versão anterior[3] da plataforma de simulação usada neste trabalho, a fim de testar o seu

---

[1] Trata-se do projecto SIMULOG promovido pelo INESC (Instituto de Engenharia de Sistemas e Controlo), proposto pelo Eng. Paulo Sá Marques

[2] "Simulação de um Sistema Automático de Logística Interna para a Industria do Calçado", dissertação submetida pelo Eng. Paulo Jorge da Rocha e Silva Sá Marques para satisfação parcial dos requisitos do grau de mestre em Automação, Instrumentação e Controlo no ano de 2007.

desempenho através de um conjunto de cenários criados pelo utilizador. Este projecto de simulação permitiu estudar a minimização dos custos de toda a movimentação de contentores com produtos em curso de fabrico.

O trabalho aqui apresentado sustenta-se, pois, neste modelo de simulação já desenvolvido previamente, tendo como objectivo a automação do seu processo de criação, visando reduzir o tempo usualmente consumido na criação de modelos de sistemas de movimentação autónomos e libertando o criador do modelo de actividades "pouco inteligentes", como preencher a mesma informação de forma repetitiva em elementos que surgem várias vezes no mesmo.

Após discussão com os responsáveis do desenvolvimento do projecto acima referido, foram definidos um conjunto de objectivos âmbito da tese aqui discutida:

- Construção de uma metodologia de inserção dos dados necessários à construção da lógica e animação de sistemas de movimentação autónomos (tapetes automáticos)
- Construção de uma ferramenta de criação automática da parte lógica dos modelos de simulação de sistemas de movimentação automáticos.

1.3 - Metodologia Utilizada

A metodologia seguida na realização desta dissertação consistiu em:

- Realizar um levantamento da fundamentação teórica e dos conceitos relativos à simulação de sistemas de produção, fazendo uma revisão de dissertações, artigos e de outros documentos que constam da literatura da especialidade.

- Estudo da linguagem de simulação ARENA, a tecnologia usada na construção dos diversos modelos que serviram de base ao trabalho proposto, mediante revisão da literatura e trabalho em cooperação com engenheiros experientes na utilização desta ferramenta.

- Estudo do sistema de produção sobre o qual foi realizado o modelo de simulação que serve de base ao trabalho desenvolvido nesta dissertação, através de documentação existente e contacto com os criadores do modelo.

---

[3] No projecto referido foi utilizado o ARENA versão 10.0, no trabalho desenvolvido no âmbito desta tese a versão utilizada foi a v11.0.

- Criação da ferramenta de construção automática do sistema de movimentação automático e respectiva verificação e validação.

- Testes e aplicação da mesma.

- Uma análise critica dos resultados conseguidos, avaliação do seu impacto, bem como do seu potencial de replicação, aplicação e desenvolvimento em futuros projectos.

1.4 - Estrutura e Organização da Dissertação

Esta dissertação encontra-se estruturada em 5 capítulos.

No capítulo 1 é exposto o enquadramento do trabalho, definidos os seus objectivos de forma clara, assim como a sua estrutura e a metodologia utilizada na sua realização.

No segundo capítulo é feita uma introdução ao conceito geral de simulação e são apresentadas as vantagens e desvantagens da sua aplicação. É feita também uma introdução aos diferentes tipos de ferramentas utilizadas na simulação hoje em dia, assim como são discutidas as fases de um projecto de simulação e as principais causas de insucesso da implementação de um tal projecto.

No capítulo 3 traça uma breve descrição da ferramenta de simulação utilizada na dissertação (a linguagem de simulação ARENA).

No quarto capítulo é feita uma breve explicação da lógica utilizada no modelo de simulação que serve de base ao trabalho proposto. Na sua sequência é discutida a informação necessária à criação automática dos modelos pretendidos e a criação dos mesmos.

No último capítulo é feita uma avaliação ao trabalho realizado. Nele são referidos as principais dificuldades encontradas na sua realização e feitas algumas considerações relativas ao futuro do trabalho e da ferramenta proposta.

Neste capítulo foi feito um levantamento dos factores que têm levado à crescente importância e utilização das tecnologias de simulação no panorama dos sistemas de produção, justificando assim a realização deste trabalho.

Foi dado a conhecer ao leitor os objectivos do trabalho realizado no âmbito desta dissertação assim como a metodologia utilizada na sua realização, juntamente com uma perspectiva da organização de todo o documento.

No capítulo seguinte pretende-se então aprofundar o estudo desta tecnologia, dando a conhecer ao leitor os conceitos em que a mesma se apoia e a que recorre alguns factores importantes relativos à sua implementação. Por fim, é feita uma análise dos diferentes tipos de tecnologia disponíveis para a realização de um projecto de simulação, com estas características.

Capítulo 2

Simulação: Conceitos e Fundamentos

Neste capítulo é apresentada uma definição de simulação e estabelecida a sua ligação com outros dois conceitos fundamentais ao seu entendimento: Sistema e Modelo. São apresentadas as vantagens e desvantagens da utilização desta tecnologia, causas de insucesso na sua implementação bem como passos para uma correcta aplicação da mesma. É dada também uma visão das tecnologias actualmente disponíveis e suas características, visto estas influenciarem em muito o sucesso de um projecto de simulação. Por fim, é proporcionada ao leitor uma discussão acerca dos desafios que se colocam neste domínio, bem como uma perspectiva sobre os futuros desenvolvimentos desta tecnologia.

2.1 - Introdução à simulação

O número de definições de simulação que se podem encontrar na literatura é elevado, o que levou à selecção das que consideramos mais relevantes e que são apresentadas de seguida, já que, de algum modo, sintetizam as principais ideias veiculadas na literatura sobre esta temática.

A palavra "Simulação" refere-se a uma vasta colecção de métodos e aplicações que copiam o comportamento de sistemas reais [4]. É uma técnica de Investigação Operacional e ferramenta utilizada para projecto e operação de sistemas complexos. A simulação pode, assim, ser considerada como o processo de construção de um modelo representativo de um sistema real, bem como da realização de experiências com esse modelo com o intuito de

conhecer melhor o seu comportamento e avaliar o impacto de estratégias alternativas e operações [5].

Segundo Banks, simulação é a imitação do comportamento, ao longo do tempo, de uma operação ou de um sistema do mundo real: Consiste em construir uma história artificial do sistema, bem como a observação do modo como essa história decorre, em consonância com as características de operação do sistema representado [6].

Segundo Shannon [7], a construção de modelos de simulação pode ser vista como uma metodologia experimental e com aplicabilidade que procura descrever o comportamento de um sistema, assim como prever o seu comportamento futuro.

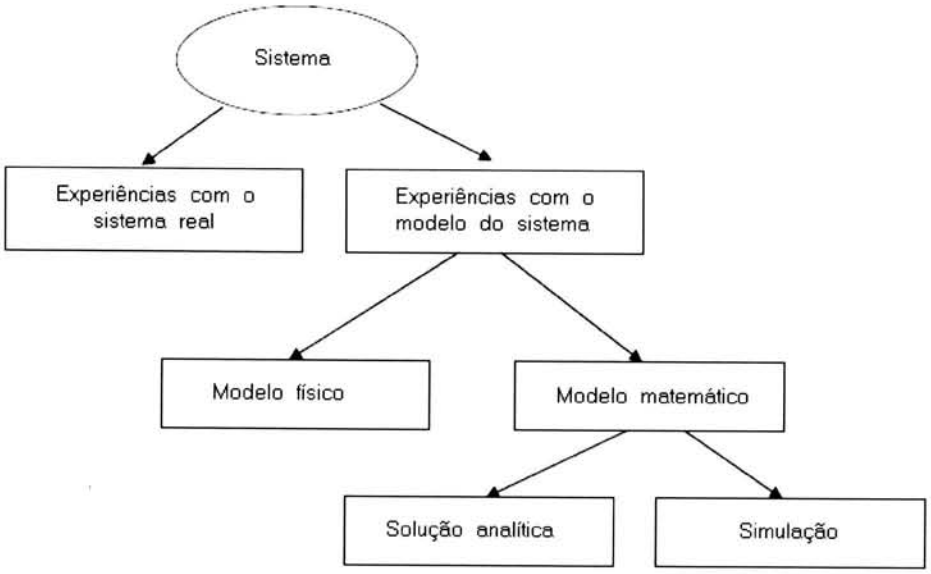


Figura 2.1 - Modelos usados no estudo de sistemas(adaptado de [8])

Vemos, pois, que, em geral, a definição de simulação está intrinsecamente ligada aos conceitos de sistema e de modelo. Sobre esta questão encontramos uma literatura muito fértil.

Um *sistema* pode ser entendido como um conjunto de entidades que interagem tendo em vista a concretização de determinado fim. Estas entidades podem ser pessoas, matérias-primas, objectos etc [9].

Brito e Teixeira [10] consideram que um *sistema* é todo e qualquer objecto relativamente ao qual se deseja realizar um determinado estudo, enquanto um *modelo* é uma representação do *sistema* sobre o qual se irá realizar o estudo. Ou seja, trata-se de uma aproximação do mundo real, onde o analista prescinde de alguns detalhes no seu

desenvolvimento. Muitas vezes o modelo precisa, apenas, de replicar o comportamento entre Entradas e Saídas. A qualidade do modelo deve ser avaliada pela maneira como as suas saídas confirmam as observações na realidade ou sistema real [11].

Segundo Anu Maria [2], um bom modelo é pois um compromisso equilibrado entre realismo e simplicidade. Este conceito, é referido como abstracção: o modelo será uma abstracção ou aproximação do sistema real. Daí que seja possível o facto de ser no modelo, e não no sistema real, que todas as acções são realizadas.

2.2 - Conceitos de simulação

Para melhor entender o trabalho desenvolvido é necessário ter noção de vários conceitos, inerentes a estudos envolvendo tecnologias de simulação.

2.2.1 Tipos de modelo

Os Modelos podem ser classificados nos termos a seguir indicados:

Estáticos versus Dinâmicos: Nos modelos estáticos o tempo não é um factor relevante; as mudanças de estado não envolvem o tempo uma vez que se trata de uma representação do sistema num momento particular.

Nos modelos dinâmicos o tempo influencia o comportamento do sistema: um modelo dinâmico de simulação representa um sistema que evoluí com o passar do tempo.

Contínuos versus Discretos: Nos modelos contínuos, as variáveis dependentes variam ao longo do tempo simulado, ou seja, o estado do sistema pode mudar continuamente.

Nos modelos discretos as alterações ocorrem em momentos separados no tempo, ou seja, as variáveis dependentes variam em espaços temporais específicos do tempo de simulação.

Podem existir modelos híbridos, onde temos ambos os casos ao mesmo tempo.

Determinístico versus Estocástico: Em modelos determinísticos, os valores dos parâmetros de entrada não são aleatórios, são constantes.

Nos modelos de simulação estocásticos, os valores de funcionamento do sistema são aleatórios.

Sempre que num modelo se entra em conta com flutuação dos valores de certas variáveis, o processo deixa de ser determinístico. Os modelos determinísticos são menos exigentes em termos computacionais que os estocásticos.

Os modelos criados no trabalho desenvolvido podem ser classificados como:

- Dinâmicos, pois representam sistemas que evoluem ao longo do tempo;

- Discretos, pois as alterações ocorrem em instantes bem definidos no tempo;

Relativamente à classificação entre determinístico ou estocástico, depende da parametrização que o utilizador quiser implementar, quando correr o modelo. Por norma, são colocadas variáveis aleatórios, tornando assim os modelos estocásticos.

2.2.2 Abordagens de simulação

Na modelação de sistemas discretos, os simuladores podem ser classificados como orientados ao processo, à actividade e ao evento. Existe ainda a abordagem das três fases.

Um simulador é *orientado às actividades* quando estão especificadas, ao máximo pormenor, a sequência de acções a realizar em cada uma dessas actividades. O funcionamento do sistema é visto como um encadeamento de actividades realizadas pelas diversas entidades. Todas as actividades são analisadas em todo os incrementos da simulação, mesmo que não sejam executada nesse espaço temporal.

Numa simulação *orientada a eventos*, a dinâmica do sistema é representada segundo um encadeamento de eventos, e a atenção centra-se nos instantes de simulação onde ocorrem transições de estado previstas no sistema [10]. São assim estudados os conjuntos de acções associadas a cada evento, acções que são executadas sempre que esse evento surge na simulação. Cada rotina de evento é executada de forma independente das outras e portanto permite saber o estado de qualquer entidade do sistema em qualquer instante de tempo.

Na simulação *orientada ao processo, a* dinâmica do sistema é representada através da descrição do fluxo das suas entidades. É feita a descrição do seu percurso desde a sua entrada no sistema até à sua saída do mesmo. A cada um destes percursos dá-se o nome de processo [10].

A *abordagem das três fases* (introduzida por Tocher na década de 60) pretende combinar a simplicidade da abordagem orientada às actividades com a eficiência da abordagem por eventos [10]. O nome desta abordagem deve-se ao algoritmo usado na sua implementação:

- Fase A - incrementa o relógio de simulação para o tempo associada ao próximo evento.
- Fase B - Executa todas as actividades do tipo 'B' (ou não condicionadas) cuja hora de ocorrência coincida com a hora actual de simulação. Estas são as actividades cujo instante de ocorrência pode ser pré-determinado.
- Fase C - Executa as actividades do tipo C(ou condicionadas) e que satisfaçam as suas condições. O algoritmo permanece nesta fase até deixarem de existir actividades nestas condições. Os instantes de ocorrência destas actividades não podem ser pré-determinados pois dependem de condições associadas à execução de outras actividades.

Qualquer uma destas abordagens aqui apresentadas proporcionam orientações, disciplinam a fase de concepção do modelo e facilitam a sua implementação computacional. No entanto, a maior parte dos autores considera que a elaboração de modelos de simulação é essencialmente uma actividade criativa, que não deve ser demasiado condicionada pelo uso desta classificação, desde que se mantenha o rigor da adequação da simulação à realidade que se pretende simular.

A plataforma criadora dos modelos de simulação utilizada nesta dissertação[4], e que será objecto de estudo mais profundo no capítulo seguinte, utiliza a técnica do próximo evento (técnica que actualiza o modelo quando existe uma mudança no seu estado), para representar os estados do sistema, ao longo do período durante o qual decorre a sua evolução.

2.2.3 Terminologia Comum

Aqui encontra-se referida a terminologia mais comum associada a simulação e que será utilizada ao longo deste trabalho:

Entidades - São objectos dinâmicos na simulação, isto é, elementos que se movem, mudam de estado, afectam e são afectados por outras entidades e pelo estado do sistema, influenciando o seu desempenho e resultado [4].

Sem entidades nada aconteceria na simulação. Cada entidade tem um ciclo de vida onde estados activos e passivos se alternam. Consoante a sua permanência no sistema, as entidades podem ser classificadas como permanentes, se permanecem sempre dentro do sistema, ou temporárias se o abandonam ao fim de um determinado tempo de simulação. As entidades são caracterizadas pelos seus atributos.

Atributos - São características associadas a entidades que individualizam as mesmas. Um atributo é uma característica comum a todas as entidades, cujo valor especifico pode ou não diferir de entidade para entidade [4].

Recursos - As entidades de um sistema, normalmente, competem entre si para "agarrar" um recurso. Um recurso pode ser descrito como um elemento do sistema que fornece serviços. Uma entidade apodera-se de um recurso, quando este lhe é necessário e se encontra disponível, libertando-o quando não necessita mais dele [12]. Um recurso pode ter capacidade de servir uma ou mais entidades ao mesmo tempo e uma entidade pode

[4] Arena version 11.0, Copyright© 2006 Rockwell Automation Technologies Inc.

necessitar de múltiplos recursos simultaneamente. Um recurso pode ter vários estados como ocupado, bloqueado ou disponível.

Evento - Ou acontecimento, como também é conhecido, é uma ocorrência instantânea que produz uma mudança no estado do sistema. O sistema mantém o seu estado até a ocorrência do próximo evento. Representa uma transição no sistema, que pode alterar atributos e variáveis, e não uma acção no mesmo.

Actividades - Uma actividade é uma operação ou conjunto de operações que alteram o estado de uma ou mais entidades. As entidades realizam actividades que fazem o sistema evoluir através dos seus vários estados.

Filas - São locais de espera no fluxo do modelo onde entidades ficam retidas. Estas filas estão associadas a lógicas ou regras que gerem acessos ou saídas de entidades em processos ou sistemas.

Estado do sistema - É como que uma fotografia do sistema, num dado instante de tempo. É definido em função do estado das entidades que o constituem e dos valores dos atributos das mesmas.

Relógio de simulação - Elemento que contém o valor actual do tempo de simulação. Na linguagem de simulação ARENA usada neste trabalho, o tempo não flúi continuamente mas sim de evento em evento.

2.3 - Projecto de simulação

Um estudo com simulação é um exercício complexo e exigente em termos de tempo e recursos, desde a construção dos modelos até à análise dos resultados obtidos. É importante ter uma noção das vantagens e desvantagens associadas a este tipo de estudos, assim como noções de como implementar um projecto deste tipo e estar atento às típicas causas de insucesso dos mesmos.

2.3.1 Vantagens/Desvantagens

As vantagens da simulação são muitas e encontram-se defendidas numa grande quantidade de autores.

Por exemplo:

- Testar novas configurações do processo produtivo sem compromisso nos recursos, evitando custos [6] e eliminando o risco financeiro e físico de mexer no sistema.

- Explorar novas politicas de escalonamento dos recursos, procedimentos operativos, regras de decisão, estruturas organizacionais, fluxos de informação, sem interromper o funcionamento do sistema [7].
- Identificar pontos de estrangulamento em linhas de produção ou no fluxo de materiais, optimizar o seu funcionamento e aumentar a eficiência [7].
- Permitir à gestão analisar a disponibilidade de todos os recursos e sua alocação [13].
- Estudar um sistema com um largo horizonte temporal (compressão e expansão temporal) [6].
- Conhecer melhor o sistema e identificar quais as variáveis que realmente influenciam o seu desempenho mostrando, assim, como o sistema funciona realmente, contrapondo com a maneira como as pessoas pensam que funciona [12].
- Perceber o "porquê?" de algo acontecer no sistema [6] [7]. Durante o próprio desenvolvimento dos modelos os responsáveis são obrigados a entenderem o papel de cada componente do sistema e quais as possíveis interacções entre eles [14].
- Testar o comportamento do sistema perante situações inesperadas e permitir análise do tipo "what if" [7].
- Simular sistemas complexos com elementos estocásticos que não conseguem ser descritos perfeitamente com modelos matemáticos ou problemas demasiado complexos para serem resolvidos através de modelos analíticos [14].
- Analisar investimentos e fiabilidade do sistema [15].
- Promover a compreensão do sistema e o consenso entre participantes no projecto de simulação [6].
- Simular vários cenários que podem ser testados e comparados rapidamente, ou seja, a replicação precisa de experiências, podendo-se assim testar alternativas diferentes do sistema [14].
- Especificar requisitos para um sistema [6].
- Separar parâmetros controláveis de não controláveis e estudar o seu impacto no desempenho no sistema [14].
- Resolver e testar (formalmente) problemas normalmente resolvidos através de regras intuitivas [13].

Em suma, parece haver consenso na literatura da especialidade sobre o facto de a simulação permitir uma importante melhoria na tomada de decisão e consequentes aumentos de eficácia e reduções de custos num sistema produtivo.

No entanto, os mesmos autores referem-se também a algumas desvantagens da simulação, tais como:

- Um modelo de um sistema complexo pode ter custos elevados e levar demasiado tempo a ser desenvolvido, especialmente em casos onde a os dados e informação são de difícil obtenção e fraca qualidade.
- A simulação não gera resultados fiáveis se as entradas não estiverem adequadamente simuladas [14].
- Pode haver uma confiança injustificada nos resultados [14].
- Cada execução da simulação estocástica produz apenas estimativas dos parâmetros analisados [14].
- A simulação não é uma técnica optimizante: não fornece soluções óptimas para os problemas em estudo; só testa alternativas dadas pelo utilizador [14].
- Os modelos não são reutilizáveis em outros sistemas. Cada modelo de simulação é único [13].
- A construção de modelos requer conhecimentos e experiência. Os modelos criados variam de criador para criador [6].
- Os resultados podem ser difíceis de interpretar [6].
- Demasiado tempo gasto no desenho, recolha de dados e informação, execução e análise dos modelos para poder serem considerados em processos de tomada de decisão.
- Exige elevados níveis de conhecimento ao nível de linguagem e simulação e/ou linguagens de programação [12].
- Exige um conhecimento profundo do sistema a modelar [7].
- Se o modelo não for uma representação válida do sistema, pouca informação útil se pode tirar do resultado [12].
- A simulação depende muito dos recursos computacionais disponíveis.

Este elenco de desvantagens ou limitações, chama a atenção para o facto de ser necessário um importante trabalho prévio à utilização de simulação em termos de construção do sistema, sob pena de invalidar os resultados.

2.3.2 Fases de um projecto

Existem várias fases, cada uma com as suas características e problemas, que devem ser consideradas no desenvolvimento de um projecto de simulação. Segundo Averill M.Law os seguintes passos devem ser seguidos pelos construtores de modelos [16]:

- Formulação do problema - Todo o trabalho de simulação começa com a definição dos objectivos: Porque é que estamos a estudar este sistema, a que questões pretendemos responder? Este passo é extremamente importante e o criador dos modelos de simulação deve ter o cuidado de se assegurar que o problema está

devidamente compreendido. Soluções apropriadas para problemas mal formulados terão pouco significado porque não respondem às reais necessidades do utilizador

Sendo assim, algumas das perguntas a efectuar antes de começar um projecto de simulação serão:

- O que deve conter o modelo de simulação?
- Qual o tempo e recursos disponíveis para o estudo?
- Qual o nível de detalhe necessário?
- Existem restrições físicas, tecnológicas ou legais na operação do sistema?
- Essas restrições podem ser alteradas?
- Os procedimentos do sistema estão bem definidos?
- Como são tomadas as decisões?
- Existem dados disponíveis?
- Como são recolhidos estes dados?
- Qual o tipo de animação necessário?
- Quem vai verificar e validar o modelo, e como o fará?
- Quais as saídas necessárias?
- Quão geral ou específico deve ser o modelo?
- Quem vai fazer a análise de resultados?
- Que critérios vão ser utilizados na análise?
- Quantos e quais os cenários devem ser considerados?

- Recolha de dados e informação e construção de um documento conceptual - Nesta fase o criador do modelo deve-se preocupar em recolher informação acerca do sistema e seus processos operativos, assim como os dados necessários, para poder simular correctamente todos os parâmetros do modelo. Deve depois elaborar um documento conceptual onde descreve aquilo que vai assumir na criação do modelo e informações relativas aos dados recolhidos.
- Validar o documento conceptual - O documento realizado no passo 2. deve ser validado por todos os intervenientes do projecto de simulação, incluindo gestores e futuros utilizadores da ferramenta. Caso seja detectado algum erro ou incoerência o documento tem de ser corrigido antes de passar ao passo seguinte.
- Programar o Modelo - Com base na informação do documento realizado nos passos anteriores, há que construir o programa, através da ferramenta ou linguagem de programação escolhida para construir o modelo e verificação do mesmo. Como verificação entenda-se o processo de análise que permite confirmar se o modelo construído está de acordo com os parâmetros inicialmente estabelecidos.
- Validação do Modelo programado - Caso exista um sistema real, os resultados da simulação devem ser comparados com dados de saída do sistema existente. Caso não

haja um sistema real, os dados devem ser analisados em termos de razoabilidade, para ver se são consistentes com o que era esperado do sistema. Este é o processo que pode assegurar que o modelo é uma representação correcta da realidade.

- Planeamento, realização e análise de experiências - Para cada experiência devem ser definidos vários parâmetros importantes, como quantas vezes deve o modelo ser corrido, qual o seu "aquecimento" e quanto tempo deve ser simulado. Devem ser analisados os resultados e verificar se mais experiências devem ser realizadas.
- Documentação e apresentação de resultados - Devem ser documentados os modelos conceptuais, assim como as descrições do programa criado e os resultados do estudo efectuado. A apresentação deve incluir uma animação do modelo e a discussão dos procedimentos de construção e validação, de maneira a promover a compreensão e a credibilidade dos modelos criados.

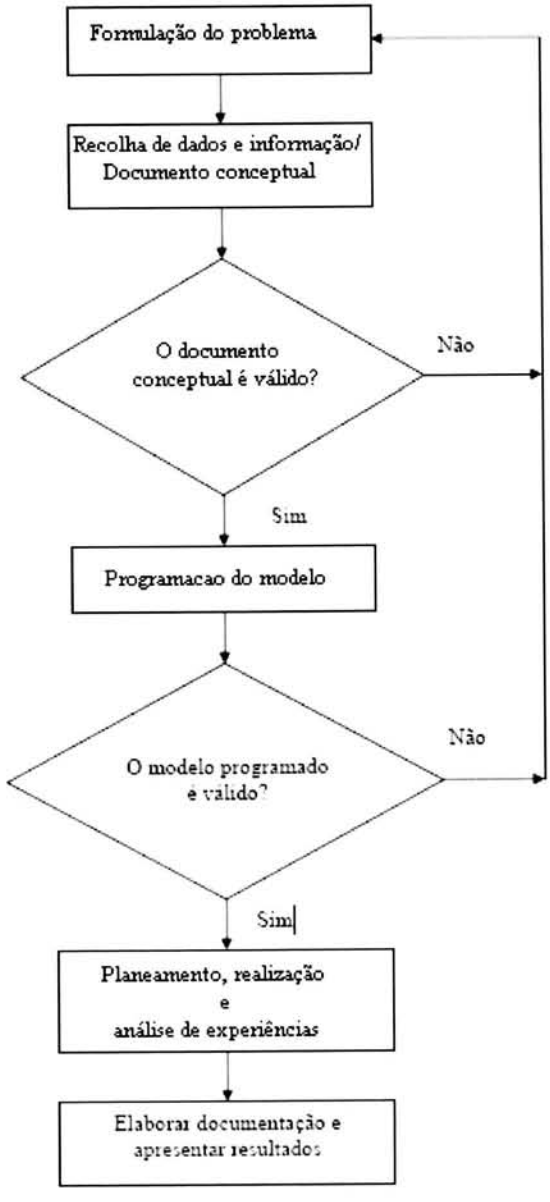


Figura 2.2 - Os sete passos de abordagem a um projecto de simulação (adaptado de [16])

Para finalizar, refira-se no entanto que, segundo Sadowski [17], é um erro pensar que a simulação se limita a percorrer uma sequência de passos como a descrita em cima. A autora defende que estes passos devem ser repetidos à medida que o projecto vai avançando e o modelo vai sendo construído.

De facto no trabalho realizado foi esta a filosofia seguida, sendo a verificação, validação e análise feitas repetidas vezes ao longo da construção dos diversos automatismos e à medida que novas funcionalidades iam sendo acrescentadas.

2.3.3 Causas de insucesso de projectos de simulação

À maneira de alerta, e no sentido de estarmos despertos para a possibilidade de uma ocorrência, são apresentadas algumas causas mais comuns de insucesso no desenvolvimento da simulação, que são referidas em [13], [7] e [14]:

- Falha na definição de objectivos claros no inicio do estudo ou desenvolvimento da simulação.
- Nível de detalhe não apropriado para o que se pretende, quer seja de mais ou de menos.
- Falta de comunicação entre os responsáveis pela construção do modelo e os responsáveis do sistema simulado durante o desenvolvimento da simulação, o que pode levar a deficiente compreensão dos objectivos.
- Interpretações erradas por parte dos criadores do modelo acerca do sistema a simular.
- Falha de compreensão dos resultados da simulação por parte dos responsáveis.
- Escolha de *software* ou linguagem inapropriada, demasiada complexidade ou documentação inadequada.
- Utilização de animações inadequadas ou incorrectas.
- Modelos inválidos.
- Maus geradores de números aleatórios e distribuições probabilísticas incorrectas, fazendo com uma simulação incorrecta do comportamento real do sistema
- Tempo de simulação inadequado.
- Uso de indicadores chave de performance ou medidas de desempenho (KPI's) inadequados
- Executar simulação apenas uma vez e considerar resultados como válidos e correctos.

2.4 - Tecnologias e Ferramentas de Simulação

Escolher a ferramenta de simulação adequada para determinado problema pode ser uma tarefa difícil. A escolha da ferramenta errada pode comprometer qualquer projecto de

simulação, aumentado em muito os tempos de modelação e criando custos adicionais para o projecto.

Como ferramentas a usar no âmbito da simulação, é classicamente feita uma distinção entre linguagens de programação, linguagens de simulação e simuladores (levado ao extremo um simulador puro não necessitaria de programação). No entanto muitos fornecedores de linguagens de programação e simulação têm vindo a implementar capacidades semelhantes aos dos simuladores e vice-versa, tornando a categorização das ferramentas disponíveis bastante complicada [18].

Linguagens de programação genéricas - as linguagens de programação genérica oferecem grande flexibilidade, devido à quantidade de recursos que as mesmas oferecem. No entanto, requerem do utilizador conhecimentos de programação e de simulação profundos. Exemplos de linguagens genéricas são Pascal, C,C++, FORTRAN e Java.

Linguagens específicas de simulação - São linguagens desenhadas especificamente para a modelação de sistemas e têm um interface entre o programador e a linguagem de simulação. Estas trazem uma série de facilidades para criar o programa do modelo conceptual do sistema. Um exemplo deste tipo de linguagens é a linguagem SIMAN. (que serve de base à ferramenta utilizada neste trabalho o ARENA). Outros exemplos são: GPSS, ECSL, DYNAMO, SLAM, SIMSCRIPT, Simple++ e MODSIM II.

Algumas das características desejáveis nas linguagens de simulação são [13]:

- Flexibilidade de modelação;
- Facilidade para o desenvolvimento e rastreabilidade do modelo;
- Execução rápida do modelo;
- Disponibilidade em várias plataformas;
- Capacidades de animação;
- Capacidades estatísticas;
- Relatórios de resultados;

Dentro desta classe de ferramentas de simulação, podemos ainda mencionar os Geradores Automáticos de Código de Simulação - *Software* onde o utilizador apenas introduz dados através de um conjunto de perguntas efectuadas pela ferramenta. CAPS, DRAFT, TESS e HOCUS são nomes de alguns exemplos deste tipo de ferramenta.

Os simuladores são ferramentas de simulação para aplicações particulares.

Os modelos são construídos através de menus, janelas de diálogo ou gráficos e as suas principais vantagens são a facilidade de aprendizagem e o manuseamento. Em contrapartida,

oferecem menor flexibilidade que as linguagens de simulação. Alguns exemplos são: Witness, FlexSIM, Promodel ou FACTOR/AIM

Podem ser identificados 2 tipos de simuladores: os genéricos e os específicos (desenhados para uma determinada aplicação).

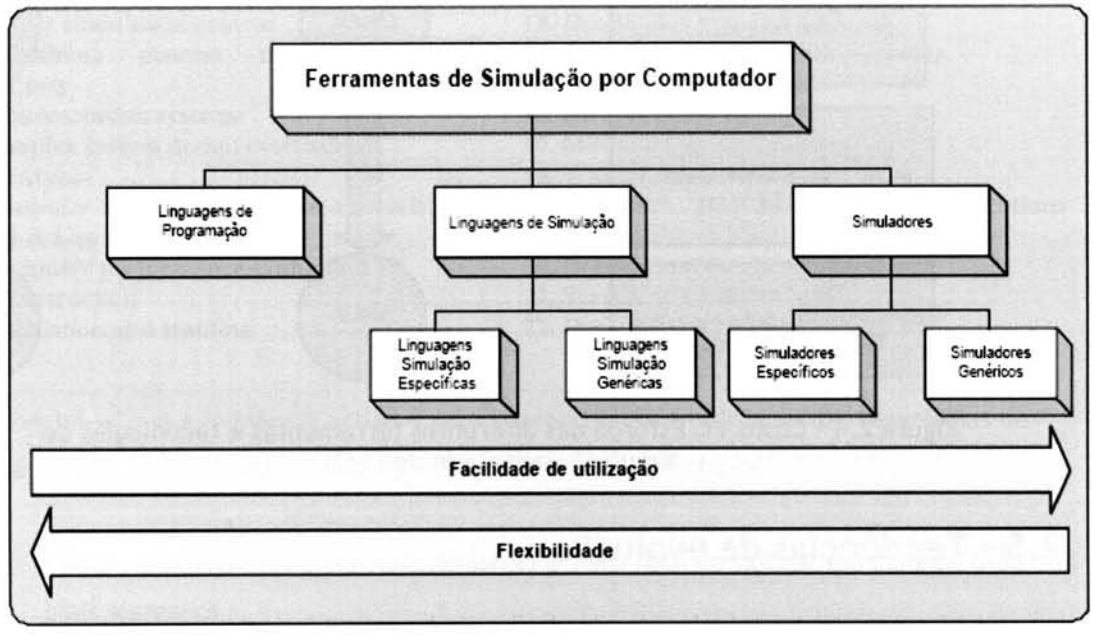


Figura 2.3 - Flexibilidade Vs. Facilidade de utilização(adaptado de [13])

A utilização de linguagens de programação oferece uma maior flexibilidade, visto que as restrições colocadas a quem está a programar o modelo são mínimas. No entanto, exigem mais tempo e esforço no desenvolvimento.

As Linguagens de simulação trazem as suas características próprias para o desenvolvimento de modelos de simulação, facilitando assim o trabalho do programador. No entanto, a ferramenta torna-se mais dispendiosa e apesar das facilidades acrescidas, por norma, apenas especialistas ou informáticos conseguem criar modelos de forma eficaz.

A utilização de simuladores é significativamente mais simples graças aos interfaces amigáveis que apresentam. Hoje em dia os simuladores oferecem um ambiente integrado, com a possibilidade de descrever o modelo, controlar a simulação, visualizar estatísticas, com capacidades de animação e tratamento de dados. A sua aprendizagem é mais fácil assim como o seu manuseamento, contudo é na mesma necessária alguma formação relativa à ferramenta e por norma, estas não oferecem a flexibilidade inerente às linguagens de programação. Geralmente, quanto mais fáceis de usar, mais limitam a criatividade [19]. Outro factor adverso à utilização de simuladores mais potentes é o custo do *software*, que aumenta geralmente consoante a capacidade que oferece.

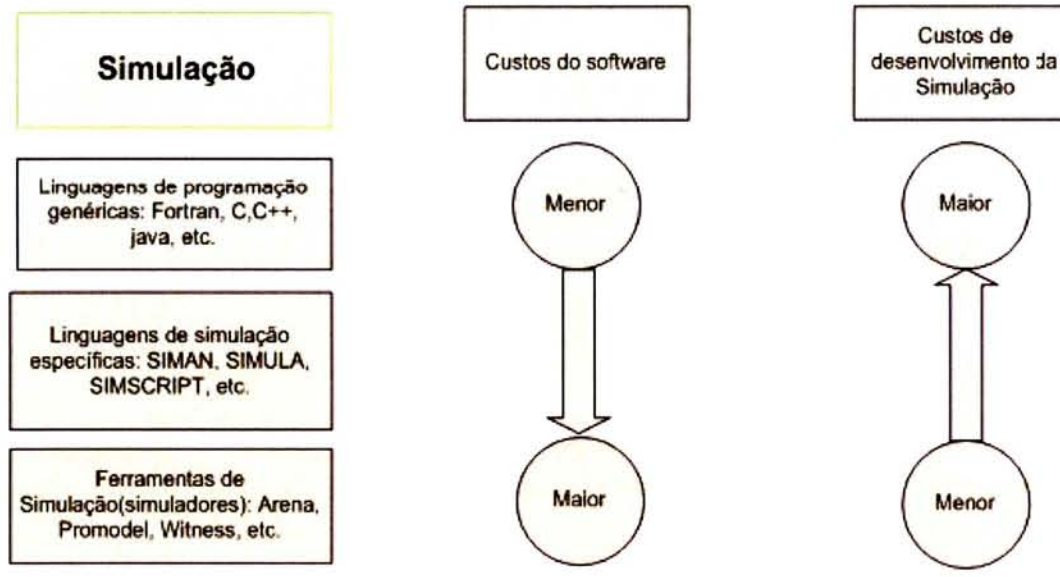


Figura 2.4 - Custo Vs. Esforço das diferentes ferramentas e tecnologias de simulação(adaptado de [14])

2.5 - Tendências de evolução

Actualmente, as tecnologias de simulação ainda encontram alguma resistência na sua utilização por parte de organizações e empresas. Carley Jurishica [20] aponta algumas razões para tal:

- É novo e diferente - Muitas empresas nunca usaram simulação antes e é comum existir um sentimento de desconfiança e resistência à mudança.
- A simulação é falível. Qualquer projecto de simulação não é imune a falhas, desde modelos não validados até projectos que demoram tempo de mais, causando potenciais prejuízos; representam riscos que empresas e organizações temem assumir.
- Cara e morosa. A simulação requer um investimento que não tem retorno imediato, desde tempo gasto em formação até ao tempo necessário à construção de um bom modelo.
- Falta de informação. Muitos gestores e responsáveis de decisão desconhecem o que é a simulação e as suas potencialidades.

Apesar, disso graças à sua flexibilidade, versatilidade e poder, a simulação pode ser utilizada em quase qualquer tipo de sistema e é já usada em vários campos bastantes distintos.

Existem no mercado pacotes de simulação já preparados para os seguintes tipos de aplicação [21]:

1. Air traffic control and space systems
2. Supply chain management
3. Business process reengineering and workflows
4. Transportation systems
5. Complex system design evaluation
6. Aerospace
7. Computer and communication networks
8. Oil & Gas
9. Computer performance evaluation
10. Construction
11. Education and training
12. Financial modeling
13. Health care systems
14. Parcels & parcel handling(queue)
15. Manufacturing systems
16. De-bottlenecking
17. Military / combat systems
18. What if scenarios
19. Satellite and wireless communications systems
20. Robotic and mechanical systems
21. Service systems
22. Decision and risk analysis

Conclui-se que a tendência actual é para a importância e utilização das tecnologias de simulação aumentar, devido, nomeadamente aos seguintes factores:

- Existe tecnologia cada vez mais poderosa e mais fácil de utilizar em campos cada vez mais alargados.
- "A complexidade está a crescer mais rápido do que a nossa habilidade de lidar com a mesma" [22]. Sistemas cada vez mais complexos necessitam de tecnologias que ofereçam soluções para novos problemas.

Há cada vez mais interesse em realizar estudos de simulação em grandes empresas/organizações e a tendência para a simulação começar a ser utilizada para a tomada de decisão em tempo real, acentua-se. No entanto, segundo vários autores e investigadores do ramo, o futuro da simulação terá de passar pela ultrapassagem dos seguintes desafios:

- Aumentar o seu poder e flexibilidade juntamente com o incremento da sua facilidade de utilização. Uma das barreiras a ultrapassara para um crescimento na utilização mais ampla de tecnologias de simulação é a complexidade das suas ferramentas.
- A partilha de modelos dentro do contexto empresarial, onde a integração das tecnologias de simulação com a Internet será fundamental. A Internet trás a hipótese de ter on-line bases de dados com informação acerca de sistemas, processos e produtos de uma organização ou empresa. Estas bases de dados poderão ser acedidas por membros da organização devidamente autorizados, permitindo trabalho colaborativo com modelos de simulação que possibilitem melhorias na tomada de decisões.

- O desenvolvimento da simulação ser distribuída, com o objectivo de diminuir tempos de processamento muito grandes, resultantes de modelos altamente complexos com grande número de resultados.
- O conceito de modelos pré construídos ou componentes de modelos que podem ser usados no desenvolvimento de outros modelos. A ideia é seleccionar esses componentes a partir de uma biblioteca e usá-los directamente. Por exemplo, gerar um modelo representativo de toda uma cadeia de abastecimento a partir da junção de modelos genéricos - pré construídos - das instalações em causa, dos centros de distribuição e de transporte genéricos. O objectivo será construir cada componente apenas uma vez, verificar a sua operacionalidade e torná-lo disponível numa biblioteca para ser usado em diversas aplicações. (É neste contexto que o trabalho desta tese se insere)
- O desenvolvimento de standards que acelerem os processos de modelação e reduzam os custos da criação de modelos, juntamente com formatos que permitam guardar os modelos e transferir dados entre os mesmos.

Refira-se, que relativamente aos sistemas de produção, há estudos (*Integrated Manufacturing Technology Roadmapping* 1998) que indicam as tecnologias de simulação como uma das tecnologias com mais potencial para melhorar produtos, processos, reduzir ciclos de desenvolvimento e produção de produtos, assim como custos dos mesmos.

Temos pois que há uma perspectiva convergente ou consensual que aponta no sentido de que, em geral, a simulação deve ser usada quando a solução do problema é muito cara, ou mesmo impossível de realizar, através de intervenção directa no sistema ou modelo físico, e/ou quando os problemas são demasiado complexos para tratamento analítico (modelos matemáticos impossíveis de resolver ou demasiado complexos).

À medida que os sistemas produtivos aumentam de complexidade, também aumentam a complexidade dos problemas inerentes aos mesmos. Melhorias nos *software*'s de simulação, têm vindo a reduzir o tempo de desenvolvimento do modelo de simulação assim como novas animações gráficas, promovendo um melhor entendimento e uso de simulação, tanto por engenheiros como por gestores.

Hoje em dia, graças à evolução dos computadores pessoais e ferramentas disponíveis, é relativamente simples implementar um modelo conceptual num computador. De facto, a utilização do computador pessoal, tanto pela elevada capacidade de armazenamento de dados que permite, como pelo excelente poder de cálculo que põe à disposição do analista, deu uma nova importância à simulação e contribui para a viabilizar.

Podem haver várias razões para realizar experiências de simulação, mas são todas realizadas com o intuito de tomar melhores decisões.

Melhorias em tomadas de decisão conduzem a aumentos de eficácia e custos reduzidos num sistema produtivos. Assim, a principal razão para utilização de técnicas de simulação: o apoio que estas fornecem à tomada de decisão [23].

Num mundo cada vez mais competitivo, a simulação tornou-se portanto numa metodologia de resolução de problemas indispensável quer para engenheiro, quer para gestores de topo [7].

2.6 - Conclusão

Neste capítulo visou proporcionar ao leitor um conhecimento aprofundado relativamente ao conceito de simulação. De igual modo, apresentou definições, terminologia, vantagens, desvantagens, causas de insucesso e passos para uma implementação correcta desta tecnologia.

Foi feita uma análise ao tipo de ferramentas disponíveis para a realização de um projecto de simulação e feito um levantamento das perspectivas futuras desta tecnologia. De seguida vamo-nos focar na plataforma escolhida para o desenvolvimento do trabalho realizado nesta tese, o ARENA.

Capítulo 3

Plataforma de simulação

Neste capítulo, é feita uma breve apresentação da aplicação utilizada no desenvolvimento dos automatismos criados para o processo de simulação de sistemas de movimentação.

É descrita a sua estrutura e são discutidas as tecnologias e ferramentas por detrás das suas potencialidades, assim como é fornecida uma descrição dos componentes utilizados nos modelos criados.

É explicada a importância da integração VBA/ARENA no contexto do trabalho realizado, terminando como um breve olhar sobre alguns componentes não utilizados mas que fazem parte do leque de opções oferecidos por esta ferramenta.

3.1- O software de simulação

Como já referido nos capítulos anterior, este trabalho foi feito com base num outro já desenvolvido[5], por alguns dos responsáveis do projecto onde o trabalho aqui desenvolvido se integra. Sendo esse o caso, não houve escolha da ferramenta de simulação, estando o ARENA escolhido "à priori". Isto não retira em nada o mérito à ferramenta, como vamos analisar de seguida.

[5] "Simulação de um Sistema Automático de Logística Interna para a Industria do Calçado", dissertação submetida pelo Eng. Paulo Jorge da Rocha e Silva Sá Marques para satisfação parcial dos requisitos do grau de mestre em Automação, Instrumentação e Controlo no ano de 2007.

Listamos aqui algumas das características importantes que foram tidas em conta na escolha do ARENA como ferramenta de simulação:

- Requisitos do sistema:
  - Sistema operativo, RAM.
- Construção do modelo:
  - Construção gráfica, possibilidade de utilização de uma linguagem de programação, *Debug* em *run-time*.
- Distribuição de entrada, desenho experimental, suporte para análise de saída
  - Optimização, reutilização do código, partilha do modelo, ferramentas de suporte e custos pela ferramenta;
  - Análise de custos, misto discreto/contínuo.
- Animação
  - Vista em tempo real, exportar animação, compatibilidade, animação3D, Importação de desenhos CAD
- Suporte/Formação
  - Suporte/linha de suporte, grupo de discussão na área, Cursos de formação, formação, formação num site, consultadoria.
- Informação sobre preços
  - Standard, versões para estudantes.

E "Também por ser um *software* genérico, com enorme potencial e por estar disponível no INESC Porto, foi a aplicação escolhida para a realização da simulação deste trabalho" [14].

Num estudo realizado pela Rockwell *Software*, a criadora do ARENA, podemos ver que na conferência sobre simulação discreta 'Winter Simulation Conference WSC 2006', dos mais de 300 artigos apresentados, quarenta e oito por cento, quase metade, mencionavam o ARENA, o que dá a medida de sua importância entre os especialistas da matéria.

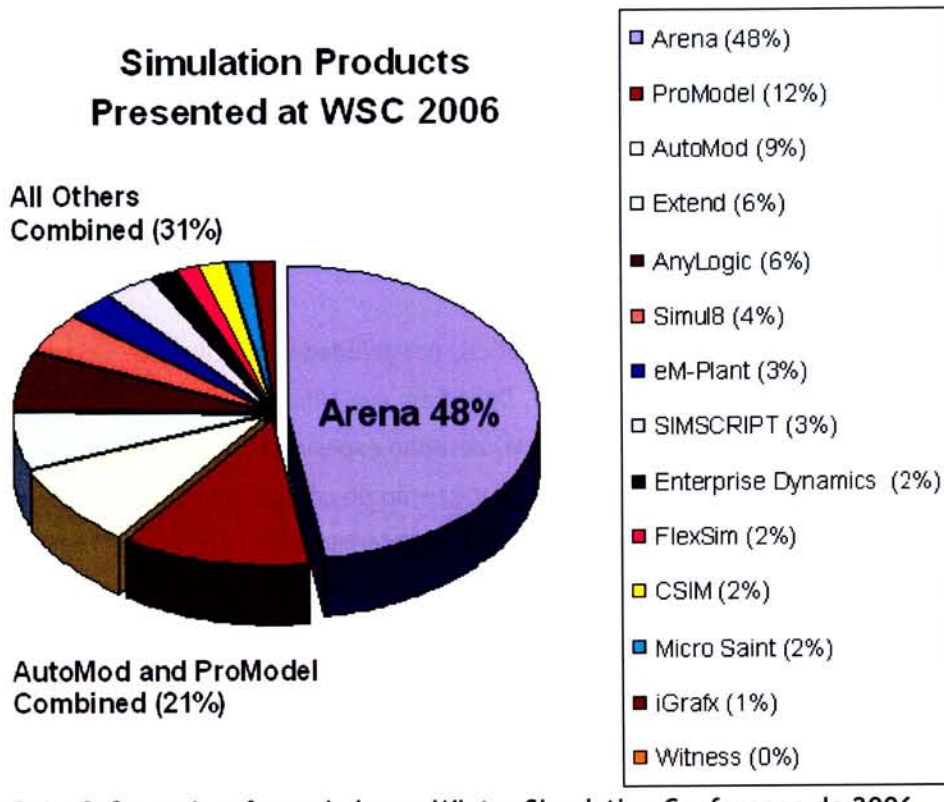


Figura 3.1 - *Software*'s referenciados na Winter Simulation Conference de 2006

De facto, o ARENA une os recursos de uma linguagem de simulação, referidos anteriormente, à facilidade de uso de um simulador, num ambiente gráfico integrado, que contém todos os recursos para modelação de sistemas, desenho do modelo, animação das entidades, análise estatística, análise de resultados e integração de diferentes linguagens de uso geral (VisualBasic, C, C++) [4].

3.2- Níveis de modelação do software

O ARENA é uma linguagem de simulação orientada ao objecto, permitindo o desenvolvimento do modelo de simulação de forma gráfica e oferecendo diferentes níveis de flexibilidade: de facto, o ARENA permite ao utilizador responder a problemas específicos, oferecendo a possibilidade de programar algoritmos de decisão ou aceder a informação externa de outras aplicações, através da programação em linguagens de programação genéricas (VisualBasic, C e C++). Este nível de programação é o mais baixo da escala hierárquica, e é também aquele que permite o maior nível de flexibilidade.

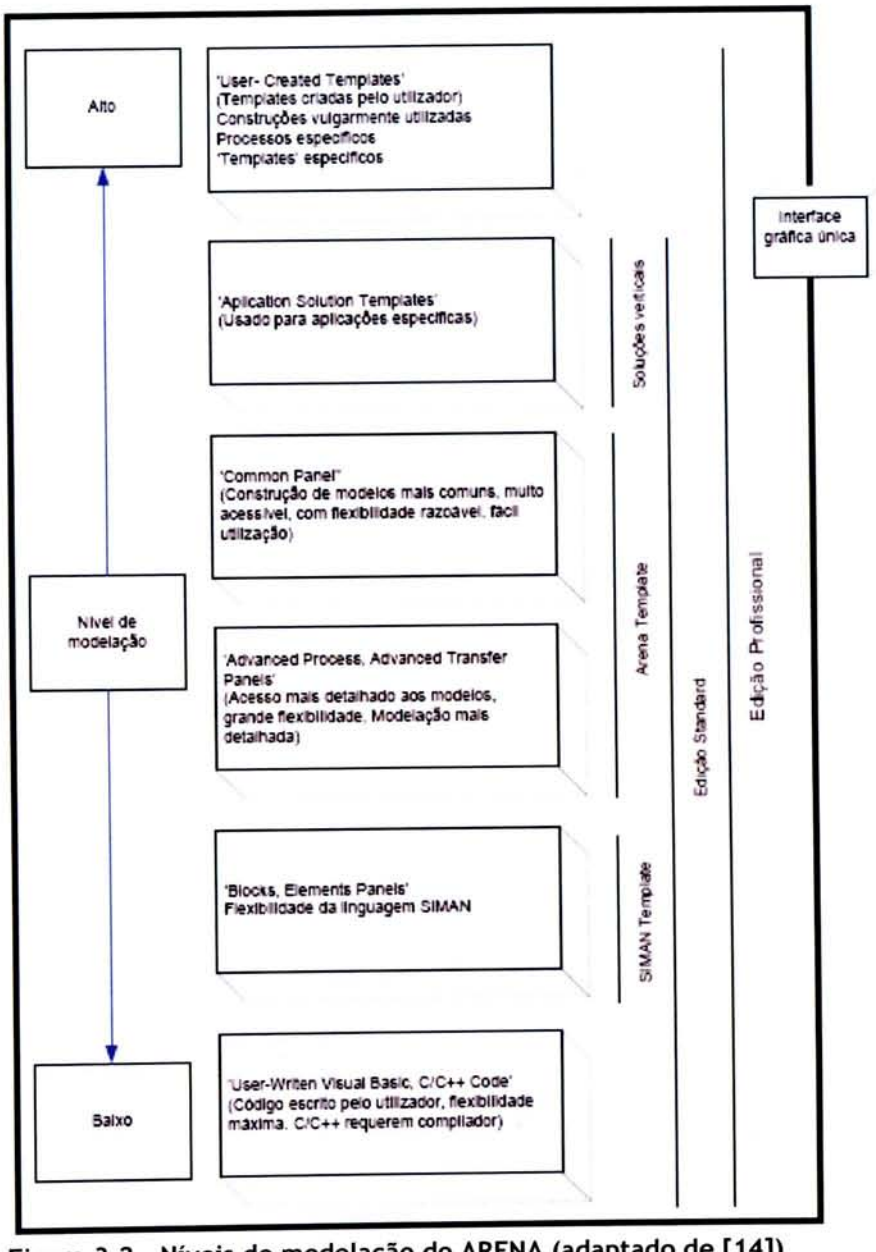


Figura 3.2 - Níveis de modelação do ARENA (adaptado de [14])

De facto, o ARENA tem uma arquitectura hierárquica na modulação, isto é, os módulos são definidos utilizando outros módulos. Temos assim que modelos são construídos juntando blocos lógicos, através de um editor gráfico de alto nível. Estes blocos, chamados de módulos, são formados a partir de componentes construídos em linguagem SIMAN e definem os componentes dos sistemas a simular, tais como maquinas, operadores ,etc. Isto permite criar uma ferramenta com conceitos e terminologia não ligada à programação, oferecendo uma ligação ao sistema real, tornando-se mais acessível a um conjunto de utilizadores com menos conhecimentos técnicos de programação [24]. Este tipo de arquitectura oferece vantagens, porque os módulos que representam subconjuntos ou conjuntos de processos podem ser desenvolvidos e verificados uma vez, e posteriormente poderão ser utilizados para desenvolver módulos de processos de mais alto nível hierárquico [14].

Assim, nos níveis mais baixos, encontramos os módulos presentes nos painéis "Blocks" e "Elements". Aqui temos acesso a todas as potencialidades da linguagem SIMAN (De notar que mesmo sendo o modelo construído graficamente, ou seja, a um nível mais elevado, o ARENA gera automaticamente todo o modelo SIMAN). Os níveis acima oferecem menos flexibilidade, mas permitem a construção dos modelos com menos esforço por parte do utilizador

É assim que, no nível mais elevado de modelação, o ARENA permite que o utilizador crie as suas próprias bibliotecas de objectos, "User-Created Templates".

Um "template" do ARENA consiste num painel ou um conjunto de painéis que os analistas utilizam no desenvolvimento de uma aplicação, sistema, ou classes de sistemas. Num "template panel" estão incluídos dois tipos de blocos (dados e lógicos) [14].

Vemos pois que, usando esta linguagem, o utilizador é assim capaz de criar ferramentas para responder a necessidades especificas. Ainda mais, o resultado pode assim ser guardado para uso futuro e para partilha com qualquer utilizador de ARENA.

Este trabalho insere-se num projecto desenvolvido neste nível mais alto, criando assim uma ferramenta capaz de responder a necessidades especificas dos responsáveis pelo projecto, relativas à construção de modelos de sistemas de logística interna. Este *template* pretende incorporar sistemas de armazenamento e movimentação, sendo que esta dissertação relata apenas o desenvolvimento dos automatismos referentes ao sistema de movimentação.

3.3- Animação e Lógica

Existem duas partes básicas do ARENA: a lógica e a animação. A primeira é constituída pelos blocos mencionados acima que montam todo o sistema a ser modelado. A segunda é à parte da animação gráfica. Tendo os blocos da parte lógica pretendidos, são anexados a estes desenhos, para conseguir a visualização do sistema real durante a evolução do tempo na simulação. É importante entender que estas duas partes estão relacionadas mas são desenvolvidas separadamente, sendo que sem a lógica não é possível haver animação.

3.3.1 Lógica

Os modelos são então construídos ligando blocos entre si, formando um tipo de fluxograma que vai interagindo com as entidades à medida estas seguem o seu percurso, descrevendo assim a lógica do processo da maneira mais visual possível. Apesar de existir esta componente visual nos módulos, a mesma não define nem pouco mais ou menos as

potencialidades de animação do ARENA. Esta descrição modular apenas é referente à lógica do modelo.

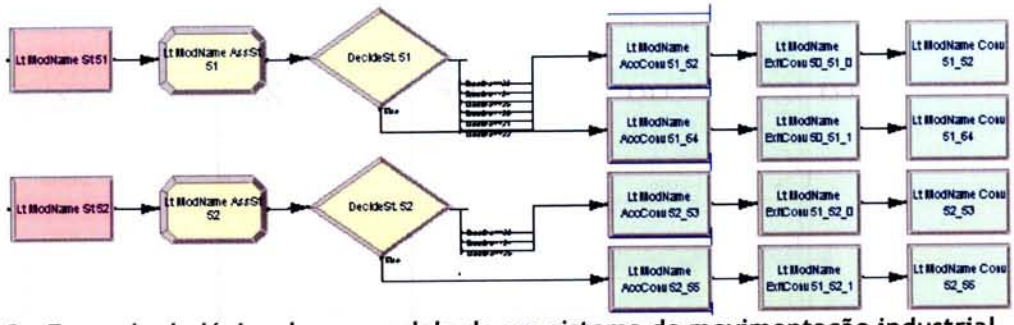


Figura 3.3 - Exemplo de lógica de um modelo de um sistema de movimentação industrial

É necessário referir que existem 2 tipos de módulos: módulos lógicos e módulos de dados. Estes últimos não fazem parte do fluxograma descritivo da lógica do modelo, mas definem especificações de elementos presentes no mesmo.

Os módulos contêm parâmetros cujo o preenchimento é necessário. Estes podem ser inseridos numa tabela (módulos de dados) ou numa caixa de diálogo que aparece através de duplo toque(módulos lógicos), utilizando o rato no respectivo ícone

Após a criação do modelo graficamente, o Arena gera automaticamente todo o modelo SIMAN, que é usado para correr a simulação. É possível construir modelos sem escrever código de qualquer tipo, ou seja, não obriga os programadores a conhecer linguagens de programação genéricas ou a linguagem de simulação SIMAN.

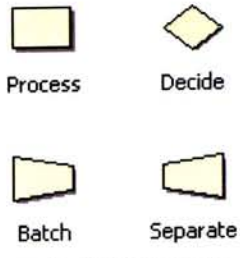


Figura 3.4 - Exemplo de ícones de módulos lógicos

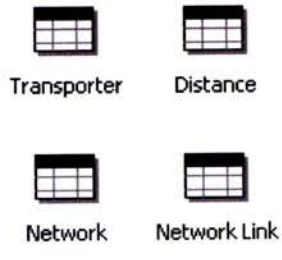


Figura 3.5 - Exemplo de ícones de módulos de dados

O utilizador tem então acesso a um conjunto de bibliotecas que contém diversos módulos. Vários painéis podem estar a ser utilizados em simultâneo na aplicação. A versão "Standard" do ARENA fornece quatro painéis genéricos: "Basic Process", "Advanced Process" , "Advanced Transfer" e "Flow Process"; juntamente com um painel "Reports" que dá acesso a um conjunto de dados estatísticos, referentes a entidades, filas de espera e outros

elementos presentes no modelo e ainda um painel que dá acesso a um mapa que permite a navegação no modelo desenhado.

Figura 3.6 - Os paneis genéricos disponíveis na versão standard do *software*

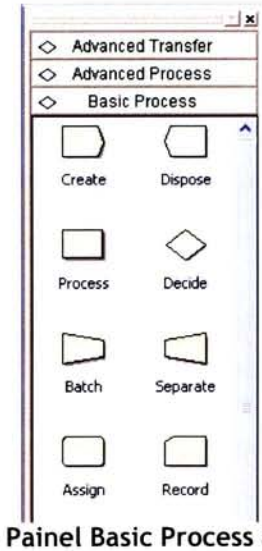


Figura 3.7 - Painel Basic Process aberto para ver os diversos módulos disponíveis

3.3.2 Animação

A animação mostra, pois, ao longo do tempo de execução o fluxo das entidades no sistema. Facilita assim a percepção do que se passa relativamente ao desempenho do sistema, em detalhes que seriam de dificilmente perceptíveis através de analise estatística.

A utilização de animação na criação de um modelo contribui para:

- Realizar a verificação e validação de modelos;
- Promover uma melhor compreensão dos resultados;
- Comunicar e demonstrar os resultados obtidos no estudo realizado;
- Promover credibilidade das conclusões obtidas;
- Destruir barreiras de desconfiança em relação a aplicação de tecnologias de simulação;

O ARENA dispõe ainda de bibliotecas de ícones ou permite a importação dos mesmos de outras aplicações externas, para criar melhores animações:

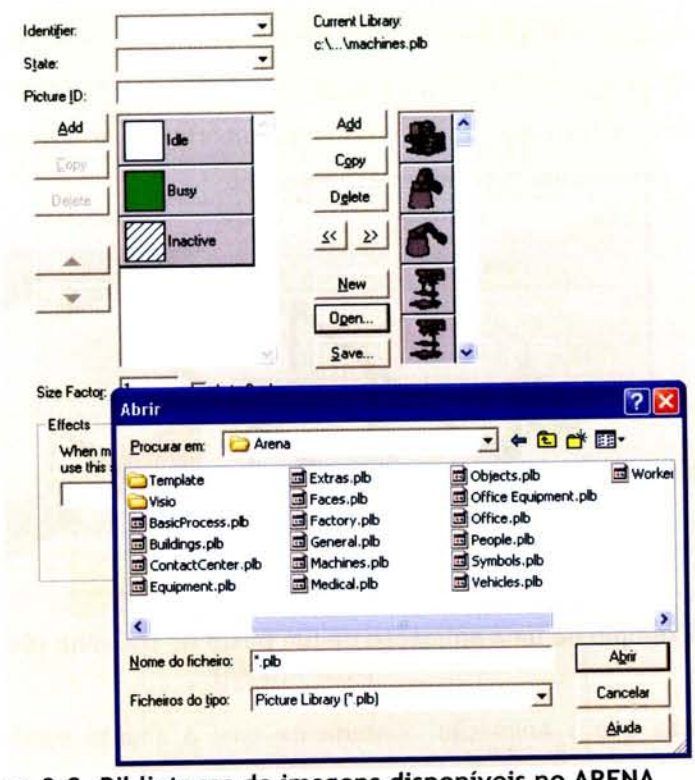


Figura 3.8 -Bibliotecas de imagens disponíveis no ARENA

Segundo Rohrer [25], a psicologia estuda e distingue a maneira como as pessoas processam informação, de forma consciente e pré-consciente. O sistema de processamento de informação visual humana é involuntário e pré-consciente, libertando assim as nossas capacidades de processamento de informação consciente para resolver problemas. Mas se a habilidade de analisar um problema visualmente é diferente de pessoa para pessoa, gráficos e animação podem ajudar pessoas que normalmente não seriam capazes de visualizar um problema muito complexo, a "ver" melhor o problema.

A animação traz pois ajudas preciosas no que toca à verificação e validação do modelo criado.

De facto e como já foi visto no capítulo anterior, na etapa de verificação, o criador do modelo observa se o modelo programado está de acordo com o idealizado (documento conceptual). Ou seja, neste passo, o criador do modelo não procura erros conceptuais, só procura erros relativos à programação do modelo e a animação é, como já referido, importante no processo de verificação, pois mostra o encadeamento dos eventos à medida que vão acontecendo.

A validação é o processo de determinar se o modelo desenvolvido é uma representação adequada do sistema real. Quando o sistema real ainda não existe (por exemplo testar configurações de *layout* alternativas), validar o modelo pode ser bastante complicado. Aqui o

grafismo e animação adquirem uma importância ainda maior. Tipicamente, a validação de um projecto de simulação envolve várias pessoas, possivelmente de áreas profissionais diferentes e a animação é, nestes casos, uma ferramenta importante de comunicação e demonstração do resultados, promovendo a compreensão de todos.

Figura 3.9 - Exemplo de uma animação de um posto de trabalho (Exemplo disponibilizado pelo ARENA)

Temos ainda que a Animação, juntamente com a análise estatística, é um elemento fundamental para determinar a qualidade de um modelo. Há, no entanto, por vezes resistência à utilização de gráficos e animação em simulação: há quem veja estas como uma coisa desnecessária ou que não acrescenta valor à simulação em si, esquecendo-se da importância de transmitir correctamente as ideias presentes no modelo e resultados obtidos, de forma clara.

3.4- Componentes utilizados

Vejamos agora uma descrição dos módulos utilizados ao longo do trabalho realizado:

- **Do painel *Basic Process*:**

Create- O módulo *Create* é o módulo criador das entidades que surgem no modelo. Estes blocos são os pontos iniciais do sistema. As entidades são os elementos que vão seguir o fluxo do modelo, sofrendo operações e fazendo funcionar processos.

Neste bloco é possível definir o tipo de entidade a criar, os intervalos de tempo em que são criadas e as quantidades de entidades a criar. O tipo de entidade é um atributo que permite agrupar e identificar um grupo de entidades e este parâmetro pode ser usado em decisões lógicas no modelo.

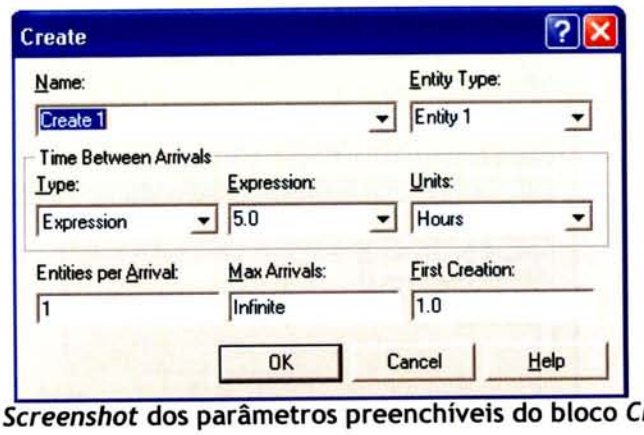


Figura 3.10 - *Screenshot* dos parâmetros preenchíveis do bloco *Create*

Assign - O módulo *Assign* permite dar a uma entidade que o atravesse um atributo especifico ou alterar um já existente.

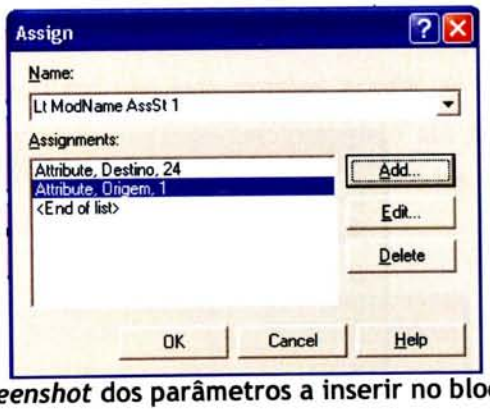


Figura 3.11 - *Screenshot* dos parâmetros a inserir no bloco *Assign*

Decide - Este módulo permite a tomada de decisão no fluxo das entidades no sistema. Ou seja o módulo *Decide* é fundamental na simulação dos cruzamentos desenvolvidos neste trabalho. Permite decisões do tipo binária (verdadeiro ou falso) ou então do tipo múltipla. As condições podem ser do tipo probabilístico (definindo uma % de probabilidade para cada ramo de saída do bloco), através de atributos (onde uma entidade com determinado atributo segue determinado ramo) ou de expressões (por exemplo, se uma fila de espera de um bloco do modelo estiver cheia então seguir um determinado caminho) ou ainda através de variáveis do processo.

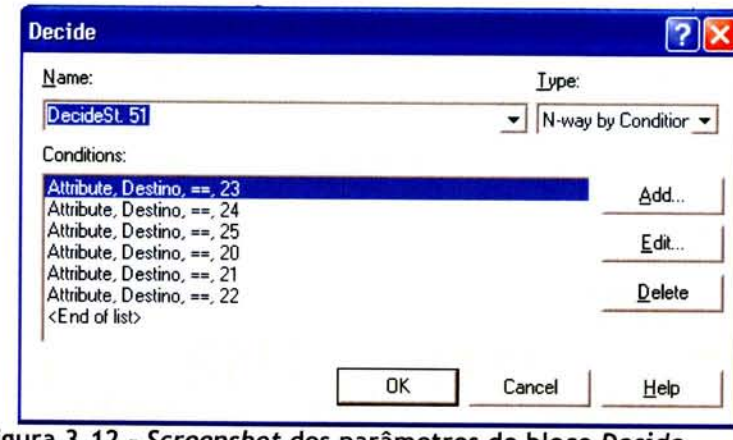


Figura 3.12 - *Screenshot* dos parâmetros do bloco *Decide*

Dispose - Se o módulo *Create* é o inicio, então o módulo *Dispose* é o fim do modelo. Este módulo retira do modelo as entidades que a ele chegam. Sem este módulo as entidades não desapareceriam do sistema, acumulando infinitamente. Este módulo encontra-se nas estações destino final.

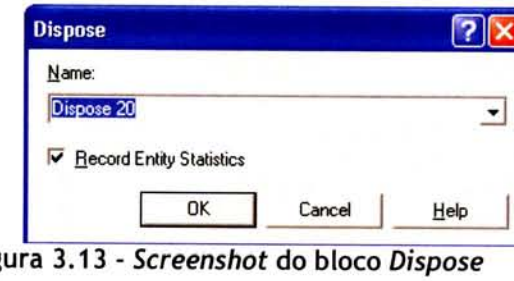


Figura 3.13 - *Screenshot* do bloco *Dispose*

- **Do painel *Advanced Transfer*:**

Station - Este módulo define uma estação. Esta estação corresponde a um local físico ou lógico onde ocorre o processamento. A entidade vai deslocar-se entre estações. As entidades criadas no modelo são transportadas de estação em estação. Não existe ligação entre estações na parte lógica do modelo, apenas na animação. De facto este módulo assume principal importância na animação do modelo.

A estação permite que todos os processos que não envolvam a movimentação física da entidade sejam agrupados num único bloco. Por exemplo, simular uma secção de uma fábrica onde a entidade seria alvo de várias acções, em diversos postos de trabalho, sem sair da mencionada secção: o bloco *Station* permite agrupar essas diversas acções num mesmo bloco.

Para realizar a movimentação física da entidade é necessário utilizar as várias hipóteses para transporte admitidas pelo ARENA.

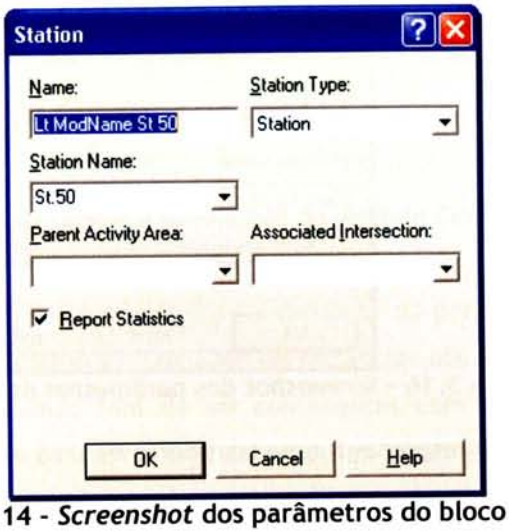


Figura 3.14 - *Screenshot* dos parâmetros do bloco *Station*

Access - O módulo *Access* é responsável pela alocação de determinado número de células de um tapete a uma entidade para a deslocação entre estações. Só após a posse destas é que a caixa pode ser movimentada. Caso não seja possível aceder ao tapete em questão, a entidade ficará numa fila de espera associada a este módulo até ser possível o acesso a tapete desejado. Aqui é necessário identificar o número de células necessário, o tapete a aceder, o tipo de fila e o seu nome.

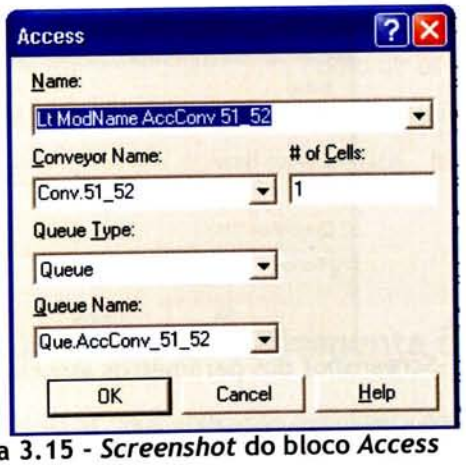


Figura 3.15 - *Screenshot* do bloco *Access*

Exit - O módulo *Access* gere a ocupação de um tapete mas para este ser libertado e voltar a ficar disponível para uso por outras entidades é necessário utilizar o módulo *Exit*. Este módulo liberta as células ocupadas por uma entidade num tapete. Só é possível fazer *Exit* de um tapete se tiver havido um *Access*.

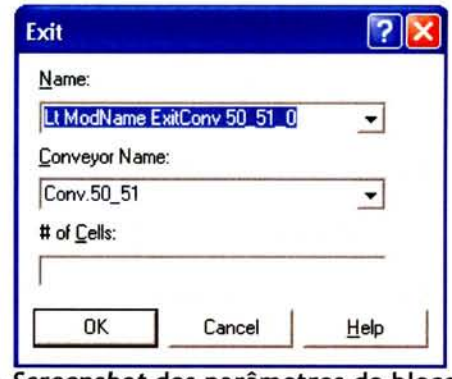


Figura 3.16 - *Screenshot* dos parâmetros do bloco *Exit*

Convey - É o módulo responsável pelo transporte de uma entidade entre estações. Neste módulo é especificado o tapete a utilizar e estação destino do mesmo. Este módulo de fluxo está intrinsecamente ligado ao módulo de dados *Conveyor*, onde estão definidas as características do tapete utilizado no transporte. *Convey* é portanto a ordem para transportar enquanto o módulo *Conveyor* é o tapete utilizado nesse mesmo transporte. O tempo gasto neste transporte depende da velocidade (definida no módulo *Conveyor)* e da distância entre estações (definida no módulo *Segment* associado ao *Conveyor)*. Para este transporte ser efectuado, primeiro a entidade precisa de atravessar um bloco *Access*, de modo a ter acesso ao tapete utilizado.

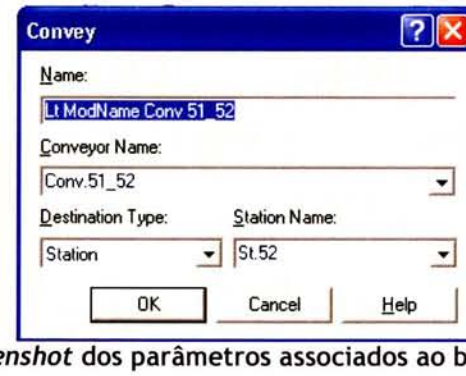


Figura 3.17 -*Screenshot* dos parâmetros associados ao bloco *Convey*

Conveyor - Este bloco define o tapete responsável pelo transporte das entidades do sistema entre estações. Aqui é possível definir um tapete como acumulativo ou não acumulativo. Tapetes não acumulativos aquando de uma paragem mantêm todas as entidades com espaçamento entre elas constante. No caso do acumulativo quando há uma paragem, as entidades param encostando uma às outras. É necessário definir a velocidade, tamanho de célula e segmentos associados (ligação com o módulo de dados *Segment)* com uma unidade de comprimento comum.

Conveyor - Advanced Transfer

	Name	Segment Name	Type	Velocity	Units	Cell Size	Max Cells Occupied	Accumulation Length	Initial St
1	Conv.1_50	Seg.1_50	Accumulating	1	Per Second	1	1	1	Active
2	Conv.2_55	Seg.2_55	Accumulating	1	Per Second	1	1	1	Active
3	Conv.48_50	Seg.48_50	Accumulating	1	Per Second	1	1	1	Active
4	Conv.58_59	Seg.58_59	Accumulating	1	Per Second	1	1	1	Active
5	Conv.50_51	Seg.50_51	Accumulating	1	Per Second	1	1	1	Active

Figura 3.18 - *Screenshot* da tabela a preencher do módulo *Conveyor*(um módulo de dados)

Segment - O módulo *Segment* é necessário na definição do percurso de uma entidade. É definido por uma estação inicial e um conjunto de distâncias até às próximas estações. As distâncias definidas neste módulo têm de ser consistentes com as do módulo *Conveyor*. Pormenor importante é o facto de os segmentos serem unidireccionais, só admitindo o percurso da estação inicio para as próximas.

Segment - Advanced Transfer

	Name	Beginning Station	Next Stations
1	Seg.1_50	st.1	1 rows
2	Seg.2_55	st.2	1 rows
3	Seg.48_50	st.48	1 rows
4	Seg.58_59	st.58	1 rows
5	Seg.50_51	st.50	1 rows
6	Seg.55_56	st.55	1 rows

Figura 3.19 - *Screenshot* da tabela de dados correspondente ao módulo *Segment*(módulo de dados)

Os módulos acima descritos foram os utilizados para construir os modelos de simulação de sistemas de movimentação industriais. Para criar automaticamente um modelo estes módulos têm de ser eles criados e preenchidos de forma automática. Isto é feito recorrendo a programação VBA.

3.5- Programação na utilização da ferramenta de simulação

O *software* de simulação ARENA é uma aplicação Windows, completamente compatível com outros "*software*'s" Windows, tais como o processador Word, folha de cálculo Excel, aplicações de *software* CAD e Oracle. Permite importar Viso Flowcharts e traduzir os mesmos em módulos ARENA, assim como importar vídeos, Bitmaps ou desenhos do Clipart, de forma a obter uma animação melhor e mais personalizada.

O ARENA faz uso de duas tecnologias Windows criadas para propósitos de integração:

- "ActiveX Automation" (a tecnologia anteriormente conhecida como OLE Automation) que permite a integração com outros programas que suportam esta tecnologia. Exemplos de programas que detêm esta tecnologia são Microsoft Office, AutoCAD e

Visio. Na criação desta integração é possível usar as linguagens de programação genéricas C++, Visual Basic ou Java para controlar o ARENA.

- "Visual Basic for Applications" ou VBA. O VBA é um componente do ARENA licenciado pela Microsoft e é a mesma linguagem de programação que se encontra presente nas aplicações Microsoft Office. Dentro do ARENA existe um editor de Visual Basic que permite ao utilizador escrever código directamente no ARENA, tendo o utilizador acesso aos objectos, métodos, propriedades e eventos do mesmo. O código escrito é guardado no mesmo ficheiro em que é guardado o modelo com toda a lógica, dados e animação. O VBA pode ser usado quando o projecto modelo está a ser lido, executado, terminado, ou enquanto as entidades flúem através dos módulos do modelo do ARENA. O editor permite ao utilizador correr o programa passo-a-passo, permitindo assim a detecção de erros e visualização de todas as variáveis, inclusive as utilizadas na linguagem SIMAN, da simulação resultando numa maior facilidade de desenvolvimento de automatismos no ARENA.

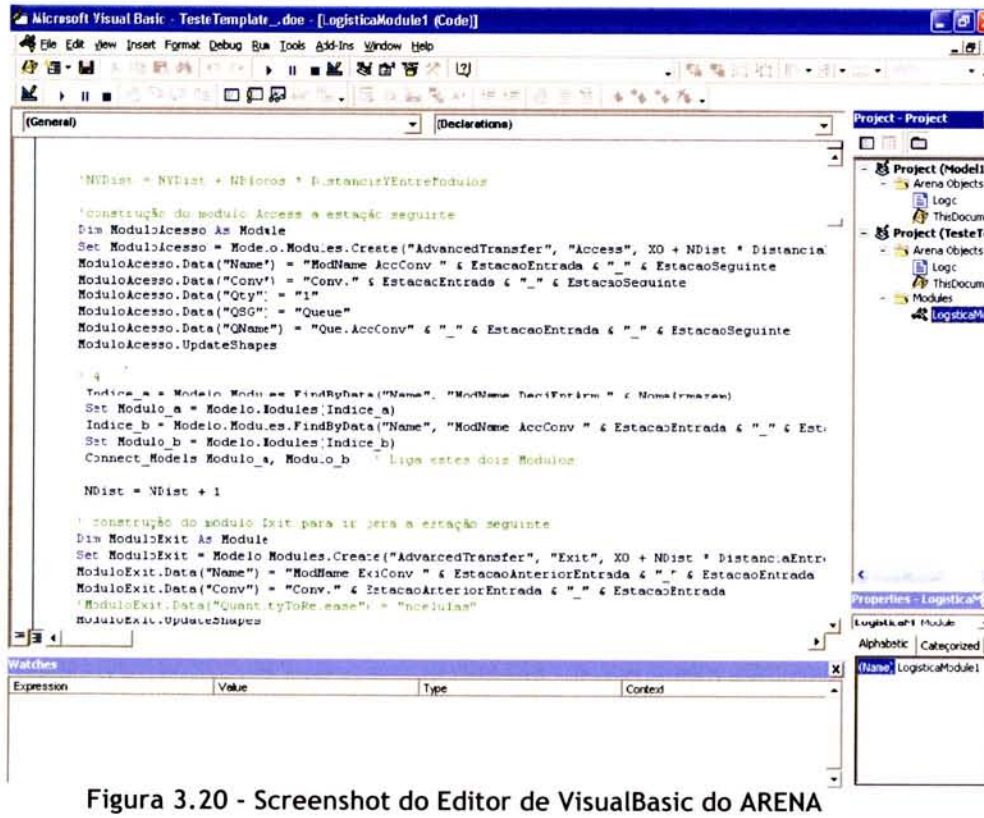


Figura 3.20 - Screenshot do Editor de VisualBasic do ARENA

A utilização de VBA e do editor interno ao ARENA foram fundamentais no desenvolvimento dos automatismos criados nesta ferramenta. Quando a solução não se encontra nas ferramentas e componentes fornecidos no ARENA, é permitido ao utilizador criar as suas soluções através de programação em VBA no editor acima mencionado. Não existindo nenhum mecanismo disponibilizado pelo ARENA para a criação de automatismos na

criação de modelos, estes foram criados utilizando as facilidades de programação oferecidas em VBA.

Ao longo do trabalho realizado foram utilizadas as potencialidades do VBA recorrendo a duas "estratégias" diferentes:

- Criando um 'form', no editor disponibilizado, que, ao pressionar de um botão, executa o código VBA e respectiva criação automática do modelo de simulação.

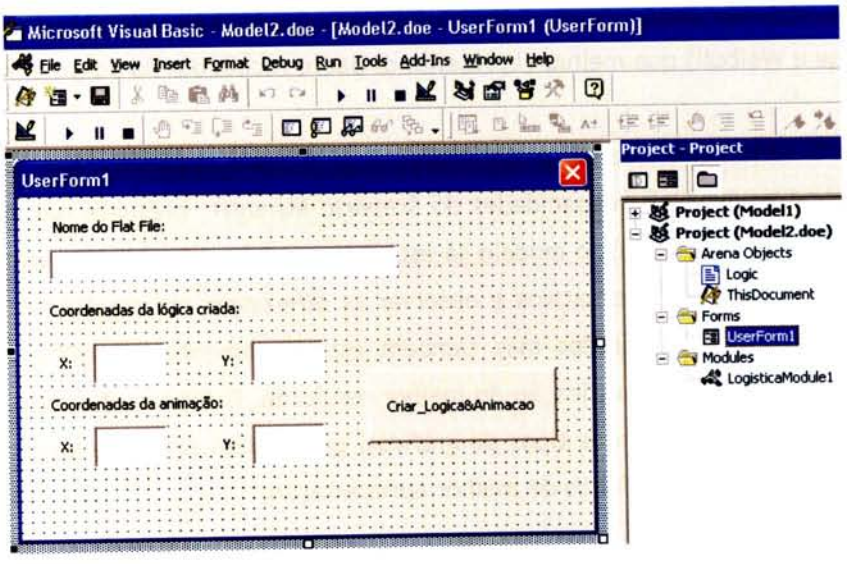


Figura 3.21 - Suporte para criação de 'forms' oferecida pelo editor de VBA do ARENA

- Fazendo uma entidade de um modelo de simulação atravessar um módulo criado pelo utilizador. A entidade faz então "disparar" a execução do código VBA e consequente criação automática dos modelos.

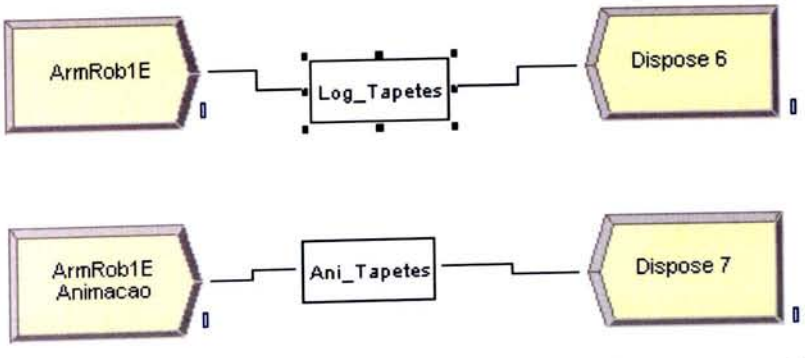


Figura 3.22 - Blocos ou módulos criados no âmbito deste trabalho que executam VBA associado

3.6 - Componentes adicionais da plataforma escolhida

Vimos assim que o ARENA fornece assim, ao utilizador, um ambiente para o desenvolvimento do modelo de simulação, executa as simulações e gera relatórios, apresentando os resultados das experimentações.

Inclui ainda uma componente chamado *Input Analyzer*. Esta ferramenta é utilizada para auxiliar o analista na determinação da distribuição de probabilidade (Beta, Contínua, Discreta, Erlang, Exponencial, Gamma,Johnson, Lognormal, Normal, Poisson, Triangular, Uniforme e Weibull) que melhor representa os dados recolhidos no sistema real.

Outra ferramenta standard do ARENA é o *Process Analyser*. Este componente veio substituir e aumentar as capacidades do *Scenario Manager*, presente nas versões anteriores do ARENA. É utilizada após o modelo já estar introduzido e permite ao utilizador avaliar um conjunto de cenários alternativos e observar as diferenças nos resultados. Estes resultados não são apenas apresentados mas também estatisticamente avaliados pelo ARENA, de maneira a acelerar a identificação do melhor resultado. O *Process Analyser* permite também ao utilizador criar diferentes tipos de gráficos, que podem ser copiados ou imprimidos para outras aplicações, para efeitos de melhor apresentação.

Através do menu *Tools* é possível iniciar uma versão do *software* Optquest, que permite optimizar as soluções obtidas nos modelos realizados. Este *software* retira informação do modelo, incluindo os parâmetros de controlo do modelo e coloca-a na sua base de dados, onde executa uma optimização através do seu próprio interface. Após gerada a solução, esta é enviada de novo para o ARENA.

O ARENA, para além de gravar *macros* para ajudar os utilizadores a perder menos tempo no desenvolvimento e criação dos modelos, permite também gravar *AVI's*, fazendo vídeos seja da criação do modelo, seja do funcionamento do mesmo.

3.7 - Conclusão

Neste capítulo foi feita uma breve introdução ao funcionamento do ARENA, sua estrutura, os componentes que foram utilizados no desenvolvimento deste trabalho e apresentadas as suas potencialidades, justificando assim a escolha desta ferramenta para o trabalho em mãos. Foi feita também uma análise da importância da integração VBA/ARENA que permite a criação de automatismos no processo de criação de modelos de simulação, terminando com uma análise a outros componentes deste *software* apesar de não tão importantes no trabalho em questão.

Seguidamente vamos descrever o trabalho realizado com esta ferramenta, no âmbito da criação de automatismos para simulação de sistemas de movimentação industriais.

Capítulo 4

Automação na construção de modelos de simulação

Este capítulo inicia-se com uma introdução aos sistemas físicos, simulados nos modelos criados de forma automática. São discutidos os dados necessários à automação da construção dos modelos pretendidos, assim como o método de inserção dos mesmos.

De seguida, é feita uma análise à lógica e animação, resultante do dos componentes criados, terminando o capítulo com a integração dos componentes resultantes do trabalho desenvolvido, no template do projecto onde este trabalho se insere.

A criação automática de modelos ou parte de um modelo envolvendo sistemas de movimentação industrial foi conseguida através da criação de um "macro-componente", ou neste caso dois, do ARENA. Este componente é acedido através de um 'template' no ARENA, colocado num ficheiro de modelo (ficheiro.doe), como qualquer outro bloco, e quando atravessado por uma entidade faz disparar a criação de um modelo complexo, utilizando os componentes standard do ARENA, a partir de informação inserida pelo utilizador.

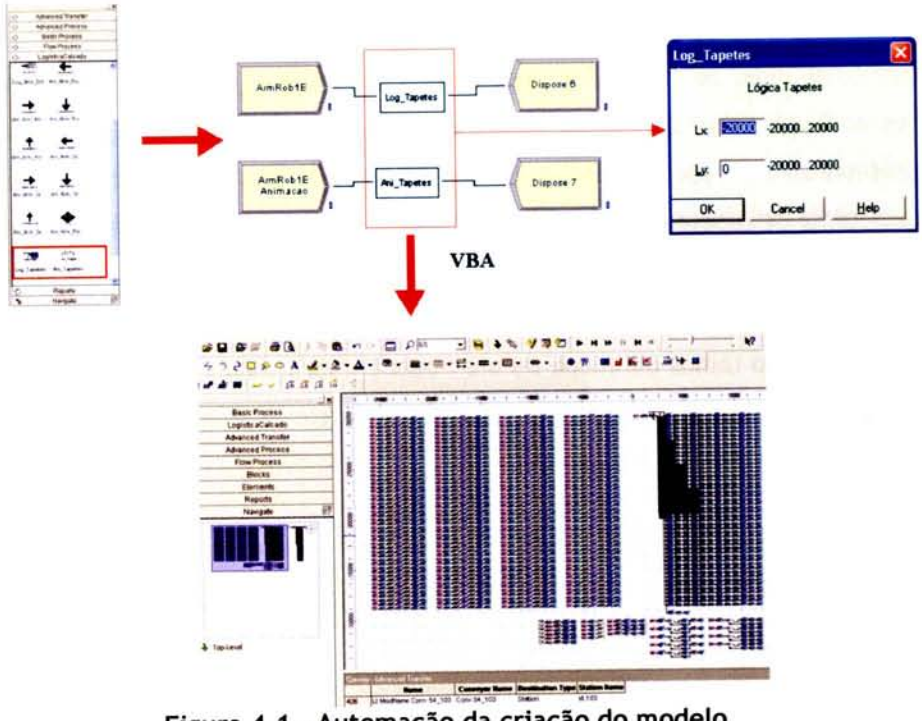


Figura 4.1 - Automação da criação do modelo

Mas para enquadrar o trabalho desenvolvido, primeiro é preciso saber o que se pretende criar, ou seja, a que problema se pretende responder.

4.1 - O Sistema físico a modelar

De maneira a ter uma melhor compreensão dos modelos que pretendemos criar, de forma automática, fez-se aqui uma breve descrição do tipo de sistemas físicos que estamos a tentar simular.

O objecto do estudo do projecto inicial, em que este trabalho se insere, consiste em sistemas de armazenamento e movimentação automáticos, baseados num caso industrial real de uma empresa de calçado em Portugal. Após implementação de todo o sistema foi criada uma simulação, de forma a testar o seu desempenho. Durante o desenvolvimento do estudo de simulação deste sistema, foram notadas as enormes vantagens que uma ferramenta capaz de acelerar e melhorar a eficiência do processo de criação de modelos poderia trazer, tanto para criadores de modelos, como para clientes dos mesmos.

O sistema físico em causa é composto por um sistema de movimentação automático e por um sistema de armazenamento intermédio do produto em curso de fabrico.

O sistema de armazenamento está dividido em quatro armazéns. Os quatro armazéns, tal como todo o sistema de tapetes, encontra-se a uma altura de cerca de três metros, permitindo que toda a movimentação e o armazenamento sejam realizados a essa altura, não interferindo com o funcionamento normal do chão de fábrica.

O sistema automático de transporte é composto por uma rede de tapetes, a uma altura de 3 metros do chão de fábrica, e torres elevatórias que ligam estes tapetes aos diversos armazéns e secções (o '*layout*' da empresa está dividido em secções, e o produto sofre alterações nas diversas secções conforme uma gama operatória previamente definida).

Este sistema de tapetes, juntamente com o sistema elevatório que sobe e baixa o produto até às secções, foi construído com configuração modular e escalável, de modo a adaptar-se a qualquer '*layout*' fabril.

Na figura abaixo temos um mapa de um '*layout*' fabril, onde podemos ver assinalados, a vermelho as secções, a branco os armazéns e a amarelo os elementos com mais interesse neste trabalho, o sistema de tapetes automáticos.

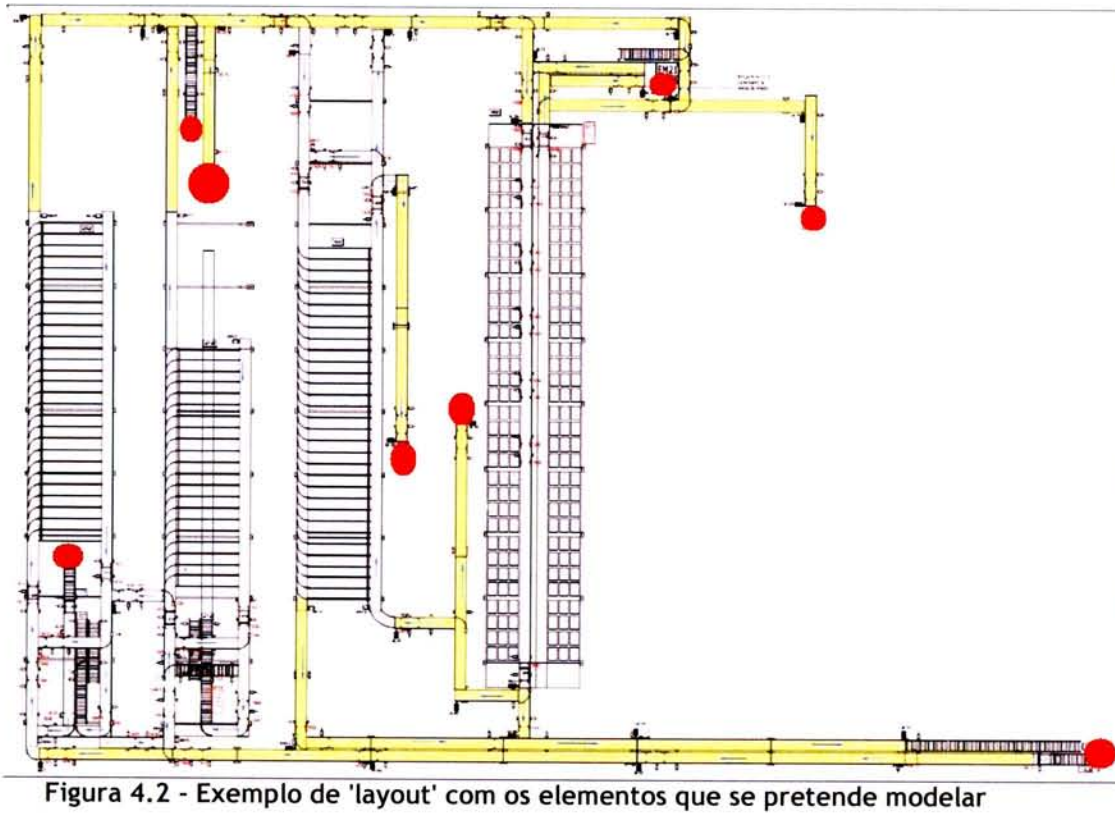


Figura 4.2 - Exemplo de 'layout' com os elementos que se pretende modelar

Vamos focar agora o elemento sobre o qual este trabalho se baseia, os tapetes. Como já foi referido, os objectivos deste trabalho foram definidos como:

- Construção de uma metodologia de inserção dos dados necessários à construção da lógica e animação de sistemas de movimentação industriais (tapetes automáticos).
- Construção de uma ferramenta de criação automática da parte lógica dos modelos de simulação de sistemas de movimentação automáticos.

Os tapetes do sistema físico do qual se realizou o estudo de simulação transportam o produto (em caixas) de secções para armazéns, armazéns para secções e de armazéns para

armazéns. Os tapetes são unidireccionais, com velocidade constante de 1m/s e nos locais onde existe a necessidade de cruzamentos , as mudanças de direcção são feitas por cancelas pneumáticas. No sistema existe um PLC responsável pela movimentação dos tapetes. O sistema de gestão operacional da empresa comunica com o PLC responsável pela movimentação, enviando as gamas operatórias, as configurações e recebe relatórios. Toda a informação de movimentação relativa às caixas com produto está neste PLC que gere a movimentação dos tapetes.

Figura 4.3 - Exemplo de um tapete automático

É pois para estes sistemas que se pretende criar, automaticamente, modelos de simulação.

4.2 - O sistema modelado

Vejamos, agora, como criar estes modelos. O primeiro passo para criar o modelo do sistema de tapetes é decompor o sistema nas suas diversas partes, ou segmentos de tapetes e ligações entre eles.

Todo o segmento é definido por 2 *estações*, uma de entrada e uma de saída, e uma direcção (visto os tapetes serem unidireccionais).

Figura 4.4 - Um tapete simples

Figura 4.5 - As duas estações e direcção que o descrevem

Mas ao longo dos sistemas de tapetes existem cruzamentos entre eles, surgindo assim *estações* onde chegam ou de onde partem vários segmentos. As *estações* podem ser classificadas como:

Figura 4.6 -Estações nEnS

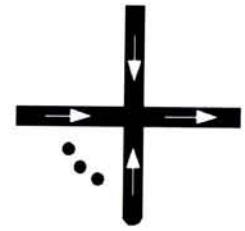


Figura 4.7 - Estações nE1s

Figura 4.8 - Estações 1EnS

- As ligações do tipo nE1S são pontos do sistema de movimentação onde existem 'n' caminhos de entrada e apenas um de saída: é necessário gerir a chegada de diversas entidades, através dos 'n' caminhos ou origens distintas, ao mesmo ponto do sistema.
- Uma ligação do tipo 1EnS consiste num ponto onde o caminho de chegada é único mas a saída tem 'n' alternativas: aqui é necessária uma decisão pois o caminho vai divergir, consoante o destino da entidade a ser transportada nos tapetes.
- Um cruzamento do tipo nEnS une a complexidade das estações nE1S com a das estações 1EnS tendo que gerir várias origens e vários destinos possíveis num só ponto. De notar que no caso mais simples também podemos ter um segmento simples 1E1S.
- *Estações origem*, pontos de entrada de entidades no sistema.
- E *estações destino*, pontos as entidades saem do sistema.

Com estas *estações* é possível descrever um sistema de movimentação. Para garantir a compreensão da descrição dos sistemas é apresentado um exemplo abaixo, onde um sistema de tapetes é descrito da maneira proposta:

Figura 4.9 - Exemplo de sistema de tapetes de um *'layout'* fabril

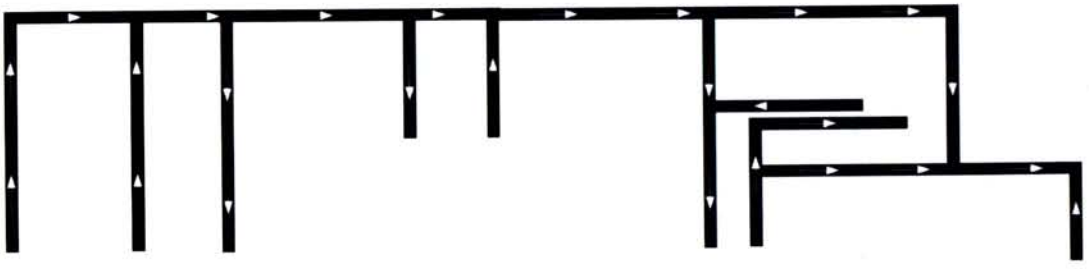


Figura 4.10 - Os diversos tapetes do sistema de movimentação presente no *'layout' fabril*

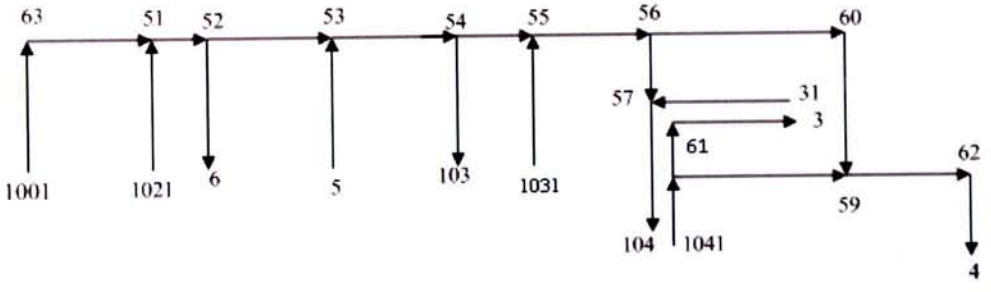


Figura 4.11 - A decomposição dos tapetes em estações e direcções

Teríamos então um sistema constituído pelas ligações:

- 1E1S (uma Entrada e uma Saída) - Estações 63,60 e 62
- 2E1S (dois caminhos de chegada e um caminho de saída) - Estações 51,53,55,57 e 59
- 1E2S (uma entrada e duas saídas) - Estacões 52,54,56 e 61
- Estações Origem -1001,1021, 5,1031,31 e 1041
- Estações Destino - 6,103,104,3 e 4

Obtém-se assim uma primeira descrição da constituição do sistema. O passo seguinte será desenvolver uma maneira de o utilizador descrever todos os parâmetros necessários, de maneira a garantir a simulação correcta de todo o sistema. Modelando o comportamento destas estações e suas ligações, conseguiremos modelar o comportamento de um sistema de tapetes industriais.

4.3 - Dados necessários à criação automática

Como já foi referido, um dos objectivos deste trabalho consistiu em criar uma metodologia de inserção dos dados necessários à criação automática de lógica e animação dos sistemas de movimentação autónomos.

Foi pois, tendo em vista o desenvolvimento de uma ferramenta capaz de construir segmentos, de tamanho e percursos variáveis, de um sistema de movimentação baseado em tapetes, que foi questionado quais os dados necessários, quer para descrever completamente o sistema, quer para inserir os mesmos.

Para saber que informação é necessária inserir no programa, é necessário conhecer previamente que módulos são usados na constituição dos modelos e os comportamentos das diversas estações a modelar. Tendo estes em mente, foi proposto que os dados necessários seriam, para cada estação:

- O número (único) identificador da estação (Ex: 51,62,..);
- O tipo de estação em questão (Ex:1E2S,2E1S);
- Estação ou estações de entrada;
- Estação ou estações de saída;
- O nome identificador dos *tapetes* que fazem a ligação com as estações de entrada e de saída;
- O comprimento e velocidade desses tapetes
- No caso de ser uma estação com múltiplas saídas e que necessita de um encaminhamento (e consequentemente um bloco *decide* do ARENA), é necessário distinguir quais as estações para o ramo *'yes'* (comportamento descrito no sub capítulo 3.4).

A forma mais fácil de compreender a informação pretendida é por observação de um exemplo:

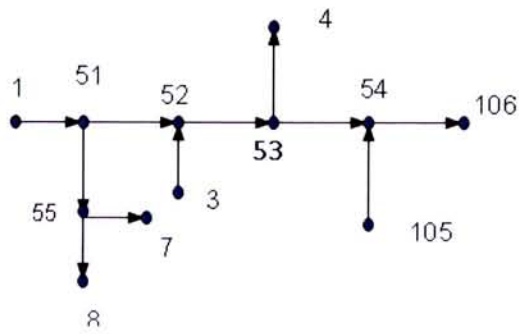


Figura 4.12 - Um pequeno sistema de tapetes

Tabela 4.1 - Primeira proposta relativa ao conjunto de dados necessários à criação automática dos modelos de simulação pretendidos

Tapetes									
Estação	Tipo	Origem1	Origem2	Dest.p1	Dest.p2	Dest.p3	Dest.p4	Est.dst(y)	Est.dst(n)
51	1E2S	1		7	8			55	52
52	2E1S	51	3					53	
53	1E2S	52		4				4	54
54	2E1S	53	105					106	
55	1E2S	51		7				7	8

Tabela 4.2 - Primeira proposta relativa ao conjunto de dados necessários à criação automática dos modelos de simulação pretendidos (continuação)

Conv1	Comp1	Vel1	Conv2	Comp2	Vel2
51_55	X	X	51_52	X	X
52_53	X	X			
53_4	X	X	53_54	X	X
54_106	X	X			
5_7	X	X	5_8	X	X

A **estação 51** é um cruzamento do tipo 1E2S, tendo uma estação de entrada e duas estações de saída (ver figura 4.12).

Seguindo a tabela 4.1 e sua continuação, a tabela 4.2, vemos que as primeiras colunas descrevem o número e tipo de estação.

A coluna seguinte "Orig1" diz respeito à sua entrada, a estação de entrada 1. A coluna "Orig2" não se encontra preenchida, pois só existe uma entrada no cruzamento.

Em cruzamentos do tipo 1E2S, assim, a decisão é tomada em função do destino.

É observável na figura 4.12 que a estação segue dois caminhos bem distintos, consoante o seu destino final. Caso o destino final seja a estação 4 ou a 106, a estação a seguir terá de ser a estação 52. Caso o destino final seja a estação 7 ou a estação 8, a estação a seguir será a estação 55. Um caso para o ramo "*Yes*", a coluna "Est.dst(y)", e um caso para o ramo "No", a coluna "Est.dst(N)". Neste caso a estação escolhida para o ramo '*Yes*' foi a estação 55.

As quatro colunas Dest.p1, Dest.p2, Dest.p3 , Dest.p4, dizem respeito aos destinos finais acessíveis através da estação colocada no ramo "*Yes*". Estas estações são colocadas nas condições do bloco *Decide*. Caso chegue uma entidade com o atributo Destino igual a uma destas estações a entidade é encaminhada pelo ramo "*yes*", caso contrário é encaminhado pelo ramo "*no*" do bloco *Decide*.

Como destino final entendem-se os pontos de saída de sistema acessíveis a partir desta estação: 4,7,8 e 106.

Caberá, pois, ao utilizador olhar para o desenho, compreendê-lo e preencher correctamente as colunas que definem o encaminhamento do cruzamento, ou seja quais os destinos para cada ramo.

As colunas seguintes dizem respeito aos tapetes utilizados nas 2 saídas deste cruzamento e suas propriedades, velocidade e comprimento.

A **estação 52** é um caso diferente da anterior, pois trata-se de um cruzamento do tipo 2E1S.

As duas colunas que se seguem indicam-nos as duas entradas do cruzamento, a estação 51 e a estação 3.

As 4 colunas seguintes não se encontra preenchidas visto que, na lógica de um cruzamento deste tipo, o encaminhamento baseia-se na estação de onde veio e não no seu destino final. Daí não haver necessidade de avaliar os diferentes destinos finais possíveis.

Havendo só uma saída possível do cruzamento, só é necessário preencher a coluna "Est.dst(y)". Aqui a decisão a tomar é de quem tem acesso ao tapete de saída.

Nas ultimas colunas só há necessidade de preencher a informação referente a um único tapete, o correspondente à única saída do sistema.

Esta primeira estrutura de dados provou não ser aplicável. A quantidade de informação é exagerada e requer um conhecimento vasto da simulação para poder ser descrito desta maneira por qualquer utilizador. Por exemplo as colunas "conv1" e "conv2", que indicam o nome dos tapetes são um exemplo de informação desnecessária. Basta saber a estacão origem e estacões destino para dar um nome adequado aos tapetes. Esta primeira versão ofereceria vantagens na programação a efectuar (toda a informação necessária para o preenchimento dos módulos que compõe a lógica do sistema está presente de forma directa) mas a custo de ser pouco perceptível, morosa e complicada para qualquer utilizador.

Uma grande lacuna nesta primeira versão é a ausência de informação relativa à construção da animação dos tapetes, que também deve ser construída de forma automática.

Após algumas experiências e estudo da informação necessária à construção de um modelo funcional, chegou-se à segunda versão do conjunto de dados necessários à criação automática, uma versão bastante mais simples que a proposta inicialmente.

Tabela 4.3 - Segunda e proposta final de estrutura de dados necessária à criação automática dos modelos

Tapetes				
Estação Origem	Estação Destino	Comprimento do Tapete	Velocidade do Tapete	Direcção
1	51	1	1	d
51	52	1	1	d
51	55	1	1	b
55	7	1	1	d
55	8	1	1	b
3	52	1	1	c
...	...	...	...	...

Estações origem e destino: número identificador(único) da estação

Comprimento: comprimento do tapete em metros

Velocidade: Velocidade a que o tapete transporta as caixas em metros por segundo

Direcção: A direcção do segmento.

As direcções possíveis são:

- Cima - c
- Baixo - b
- Direita -d
- Esquerda - e

Foi definida aqui uma restrição importante. Por questões de programação da lógica e animação, existe a restrição de um máximo de 4 ligações por estação, visto que só são admitidas 4 direcções (restrição acordada com o responsável pelo projecto onde este trabalho se insere).

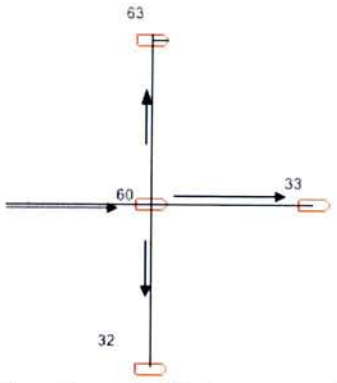


Figura 4.13 - Exemplo das 4 direcções admitidas nos modelos dos sistemas de movimentação criados

Passamos a ter então a admitir apenas as seguintes possibilidades de tipos de estações: Origem, Destino, 1E1S, 1E2S, 1E3S, 2E1S, 3E1S e 2E2S.

A partir desta informação, é possível descrever todo o sistema e preencher todos os módulos da lógica do sistema, assim como obter uma animação do mesmo.

4.4 - Inserção dos dados necessários

Sabendo a informação necessária à criação automática dos modelos pretendidos, passamos para o problema de como inserir os dados. Foi acordado com os responsáveis pelo projecto onde os componentes desenvolvidos neste trabalho teriam que ser integrados, que o acesso à informação seria através de um '*flat file*'. Por '*flat* file' entende-se um ficheiro de texto sem formatação nenhuma associada. Esta estrutura de dados permitiu testar as diferentes partes constituintes da biblioteca objectivo, sob construção em locais distintos, com a mesma informação, contribuindo em muito para a integração e funcionamento dos mesmos.

Também foi colocada a hipótese de colocar esta informação numa base de dados, mas foi decidida a opção do '*flat file*' por questões de universalidade e evitar a utilização de outra tecnologia.

Este '*flat file*' consiste num simples ficheiro de texto .txt com a estrutura de informação discutida no sub capítulo anterior e descrita na tabela 4.3.

Figura 4.14 - Exemplo de ficheiro '*flat file*' usado para descrever um sistema de tapetes

4.5 - Criação automática dos modelos

Estando definida a introdução dos dados do sistema, foi agora possível criar toda a lógica e animação do modelo, de forma automática e através do uso de código VBA. Este código é executado quando uma entidade atravessa os blocos criados neste trabalho, como explicado no inicio deste capítulo. São estes os blocos, ou "macro-componentes", que se pretendem parte integrante de uma biblioteca facilitadora da criação dos modelos pretendidos.

O '*flat file*' é lido e a informação guardada em *arrays* que vão ser chamados diversas vezes ao longo do código:

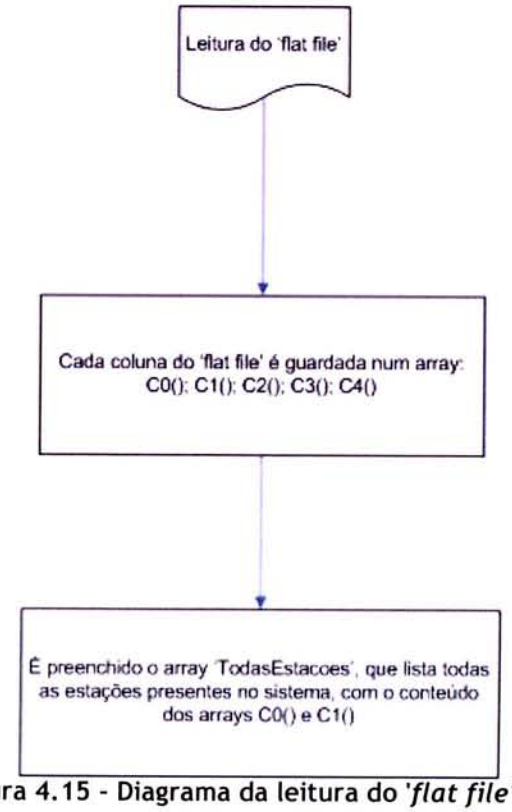


Figura 4.15 - Diagrama da leitura do '*flat file*'

As estações são distinguidas nos vários tipos discutidos no sub capítulo 4.2 , recorrendo a duas funções criadas em VBA:

- **EstSeg()** que utiliza como parâmetro de entrada um número identificador de estação e retorna com os números das estações que servem de saída à mesma.
- **EstAnt()** que utiliza como parâmetro de entrada um número identificador de estação e retorna as estações que servem de entrada ou origem da mesma.

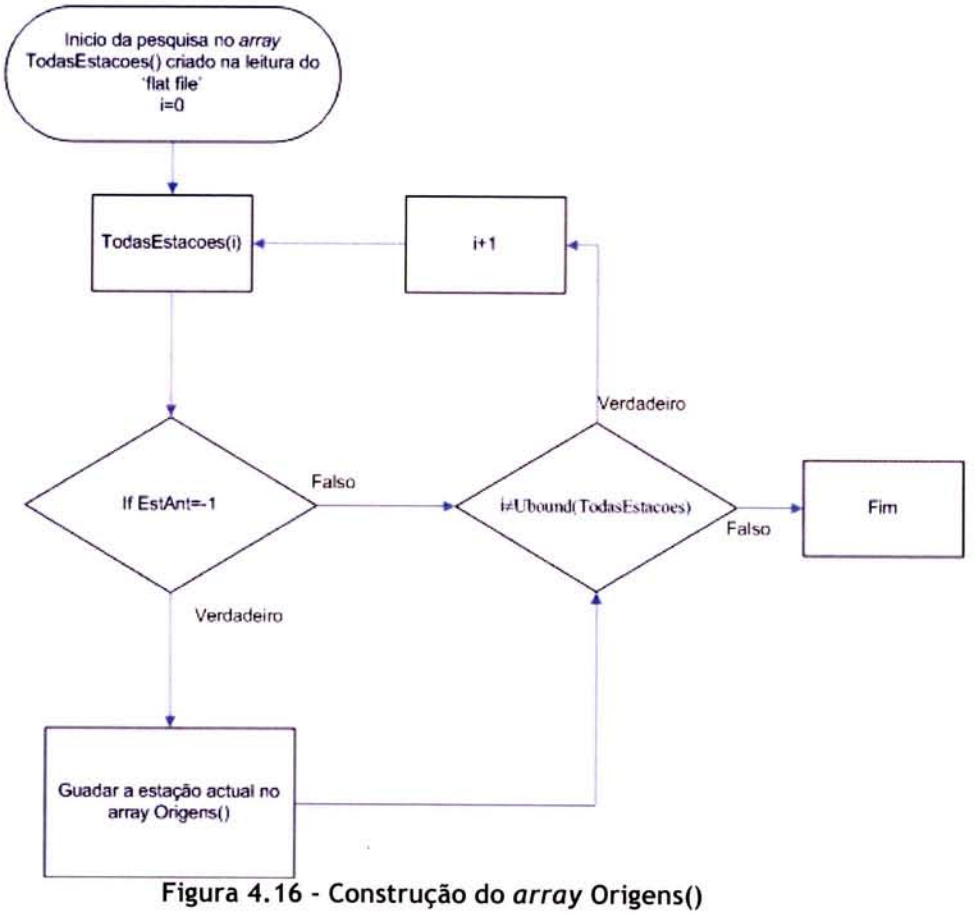


Figura 4.16 - Construção do *array* Origens()

Assim, as Estações que não tem outras estações como entrada (EstAnt=-1), são guardas num *array* Origens() onde todas as Estações Origem são guardadas (figura 4.16). As que não tem outras estações como saída (EstDest=-1), são Estações Destino e como tal guardadas no *array* Destinos(). Utilizando o comando UBound() do VBA, podemos testar o tamanho de um *array*, isto permite-nos ver o numero de saídas e entradas de cada estação, colocando cada tipo de estação num *array* próprio (ver figura 4.17). Estes *arrays* são construídos sequencialmente e mais tarde chamados na construção da parte lógica do modelo.

As estações 2E2S têm um ciclo de leitura que identifica este tipo de estação dentro da função que cria a sua lógica, não sendo guardadas num *array* especifico.

Figura 4.17 - Distinção das diferentes estações e respectivos *arrays*

4.5.1 Lógica criada

Foi detectado outro requisito básico na construção do modelo no ARENA, ainda não mencionado: as coordenadas da disposição dos módulos no mapa visualizador do ARENA. Estas coordenadas são necessárias na criação de todos os módulos através de código VBA, visto que não existe o utilizador a fazer o habitual *drag&drop* dos módulos no local pretendido.

A solução para esta questão foi permitir ao utilizador indicar as coordenadas do primeiro bloco e uniformizar as distâncias entre blocos e entre diferentes tipos de estações. Assim o utilizador só precisa de definir as coordenadas do primeiro bloco a ser criado. Estas coordenadas são pedidas num *form* que surge após '*double click*' em qualquer um dos "macro-componentes" criados (lógica dos tapetes ou animação dos tapetes).

Figura 4.18 - *form* do componente da lógica do sistema de movimentação que queremos modelar

Figura 4.19 - *form* da animação do sistema de movimentação que se pretende modelar

A parte lógica dos modelos pretendidos é então criada através de uma função VBA que chama outras sete funções. Estas funções constroem os diversos tipos de estação, identificados pelos *arrays* criados aquando a leitura do '*flat file*'. Este tipo de solução, a nível de programação, permitiu testar os diferentes tipo de estação separadamente, o que foi importante nos vários testes de integração com outros componentes da biblioteca, com os quais, os componentes aqui desenvolvidos neste trabalho, têm de estar integrados.

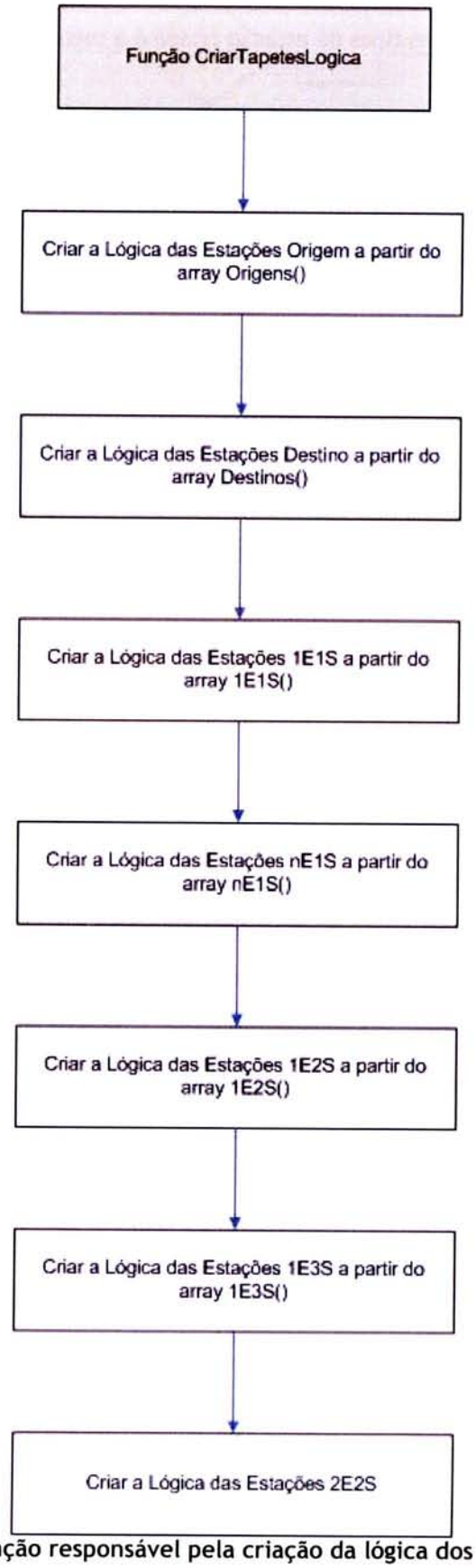


Figura 4.20 - Função responsável pela criação da lógica dos tapetes

A lógica dos diferentes tipos de estação criada é a seguinte;

Estação Origem:

A construção de estações deste tipo é bastante linear. A lógica para implementar este tipo de estações no ARENA consiste em:

- Um bloco *Create*, gerador de entidades;
- Um bloco *Station*, a estação em questão;
- Um bloco *Assign* que escreve um atributo Origem com o nome da estação e um atributo Destino com o número da estação final do percurso pretendido;
- Um bloco *Access* que dá acesso a um tapete para o transporte;
- E um bloco *Convey* que executa o transporte após ter tido acesso ao tapete.

Figura 4.21 -Lógica de uma estação origem

Estação Destino:

A lógica necessária consiste em:

- Um bloco *Station* da estação em questão;
- Um bloco *Exit* que liberta o tapete que transportou a entidade até esta estação;
- E finalmente um bloco *Dispose* que retira a entidade do modelo.

Figura 4.22 - Exemplo de lógica criada para uma estação destino do sistema

Este tipo de estação foi desenhado para admitir também mais do que um tapete de chegada. Nesses casos a lógica da estação passa a ser:

- Um bloco *Station;*
- Um bloco *Decide* que lê o atributo Origem e liberta o tapete correspondente. Aqui é necessário colocar nas condições do módulo *Decide*, tendo n Origens possíveis, coloca n-1 Origens para o ramo "*yes*" e a restante colocada no ramo "*no*";
- Um bloco *Exit* para cada uma das origens e respectivos tapetes;
- E um bloco *Dispose* que retira as entidades do modelo.

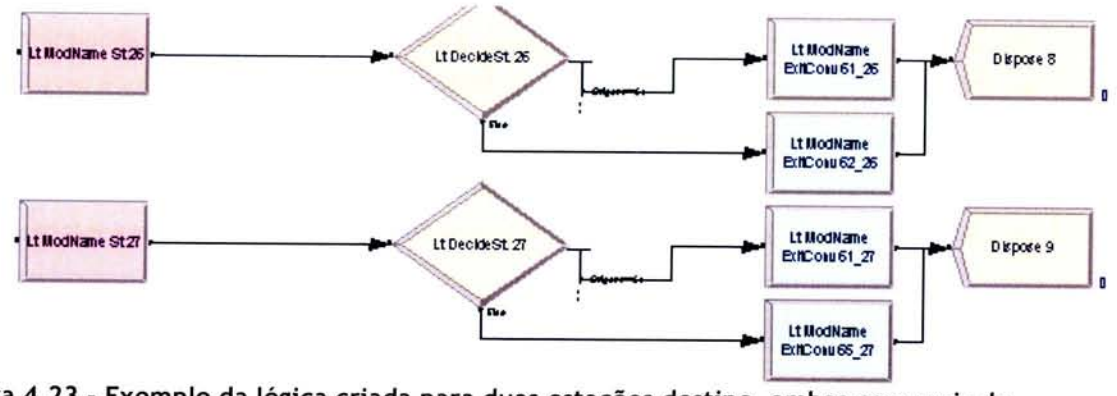


Figura 4.23 - Exemplo da lógica criada para duas estações destino, ambas com mais do que uma estação de entrada

Estação 1E1S:

Esta estação é um caso especial da lógica nEnS. No entanto, uma vez que é significativamente mais simples e surgindo um número de vezes francamente maior que a estação 2E2S, é construída separadamente. Esta estação necessita de:

- Uma estação ou módulo '*station*';
- Um módulo *Assign* que escreve esta estação como a ultima visitada (escrita do número da estação no atributo Origem);
- Um módulo de *Access* ao tapete a seguir;
- Após garantir acesso ao tapete através do módulo *Access* liberta o tapete que trouxe a entidade até esta estação através de um bloco *Exit;*
- E um módulo *Convey* que executa o transporte da entidade no tapete acedido através do bloco *Access.*

Figura 4.24 - Exemplo da lógica criada para uma estação 1E1S

Estação nE1S:

Esta estação já é mais exigente computacionalmente. Aqui, para modelar correctamente o sistema, é necessária a existência de um bloco *Decide*. Este módulo analisa um atributo Origem da entidade e liberta o respectivo tapete. No entanto, isto só é feito se for garantido o acesso ao tapete de saída desta estação. Deste modo, a lógica construída é constituída por:

- Um módulo *Station*;
- Um bloco *Access* que gera o acesso ao tapete de saída da estação;

- Um bloco *Decide* que lê o atributo Origem e liberta o tapete correspondente. Aqui é necessário colocar nas condições do módulo *Decide*, n-1 Origens para o ramo "*Yes*" e a última no ramo "*No*";
- Os diversos blocos *Exit* correspondentes aos diversos tapetes de entrada na estação;
- Um módulo *Assign* para escrever no atributo Origem o número desta estação (que é a ultima por onde passou) ;
- E um módulo *Convey* do tapete já acedido que é o único de saída deste tipo de estação.

Considerações: Apesar de no máximo só haver 3 estações devido às direcções admissíveis, na animação o algoritmo responderia para *n*(um número inteiro) estações de entrada.

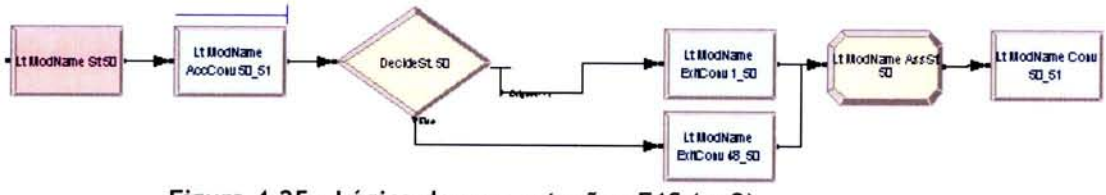


Figura 4.25 - Lógica de uma estação nE1S (n=2)

Estação 1E2S:

Esta estação é mais complexa na sua construção devido ao preenchimento automático do bloco decide.

É avaliado o número de estacões destino acessíveis por cada um dos dois caminhos de saída, sendo que para isso é necessário correr algoritmos que percorrem os diferentes percursos possíveis até aos seus destinos finais. O caminho com menor número de destinos vai ser acedido pelo ramo "*Yes*" do bloco *Decide* e esses mesmos destinos colocados nas diferentes condições desse bloco. Assim, a entidade que tiver no seu atributo Destino alguma dessas estações é encaminhada pelo ramo "*Yes*" e respectiva estação. As entidades com qualquer outro destino final seguem pelo ramo "*No*" e consequentemente pela outra estação. A lógica implementada consiste pois em:

- Uma estação;
- Um módulo *Assign* que escreve esta estação como última origem por onde a entidade passou;
- O bloco *Decide* que decide o encaminhamento seguindo o algoritmo acima mencionado;
- Um bloco *Access*, *Exit* e *Convey* para cada um dos tapetes de saída da estação

Considerações: Como o nome de todos os blocos tem de ser distinto, é necessário um índice no módulo *Exit*, para o ARENA considerar 2 módulos que saem do mesmo tapete de entrada. Seria possível juntar os dois num só bloco *Exit*, mas seria necessário fazer nova bifurcação na sua saída para aceder aos dois *Conveys* distintos.

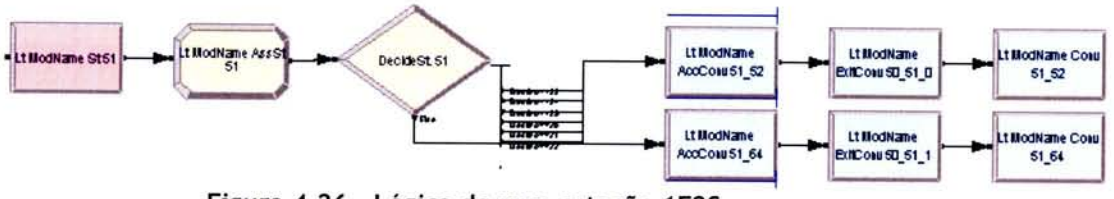


Figura 4.26 - Lógica de uma estação 1E2S

Estação 1E3S:

Nesta estação o preenchimento automático da informação do módulo *Decide* é ainda mais complexo. De facto, é necessário colocar duas estações no ramo "*Yes*", juntamente com as diversas estações destino acedidas por cada uma das estações imediatamente a seguir, e assegurar as ligações entre lista de estacões resultante e tapetes de saída correspondentes.

A sua lógica:

- Um bloco *Station;*
- Um bloco *Assign* que regista esta estação como ultima percorrida;
- Um bloco *Decide* que faz o encaminhamento da entidade consoante o atributo Destino da entidade;
- E um bloco *Access* , *Exit* e *Convey* para cada um dos tapetes de saída da estação.

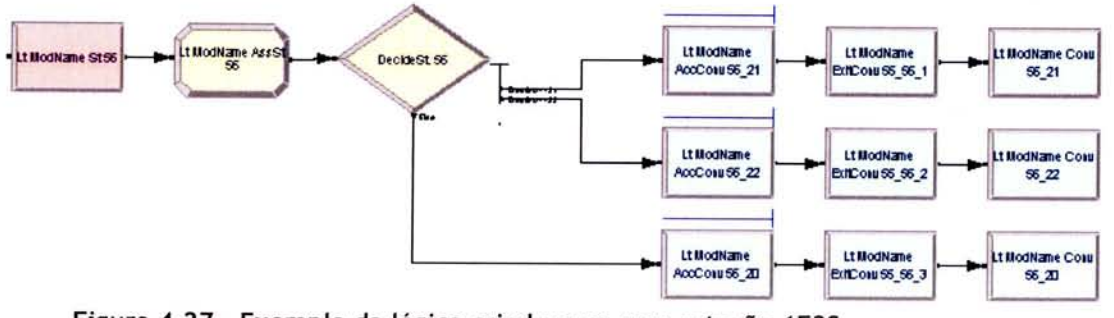


Figura 4.27 - Exemplo da lógica criada para uma estação 1E3S

Estações 2E2S:

Esta estação junta a complexidade de estações com mais do que uma entrada e mais do que uma saída simultaneamente. Precisa, primeiro, de analisar o Atributo Destino para realizar o encaminhamento correcto da entidade e tentar tomar passe do tapete apropriado. Após posse do tapete, o segundo bloco *Decide* analisa o atributo Origem e liberta o respectivo

tapete de chegada. Por fim, verifica novamente o atributo *Destino* e acede ao bloco *Convey* correspondente.

Esta estação é construída com a lógica:

- Um bloco *Station;*
- Dois módulos *Decide* que fazem o encaminhamento segundo o Destino da entidade;
- Dois blocos *Access* correspondentes aos dois tapetes de saída;
- Três blocos *Dispose* para detectar erros nos blocos *Decide* (devido à complexidade desta estação);
- Um bloco *Decide* que encaminha a entidade para o respectivo *Exit* do tapete de entrada, a partir do atributo Origem;
- Dois blocos *Exit* para cada um dos tapetes de entrada;
- E dois blocos *Convey* para realizar o transporte da entidade pela estação de saída indicada ao seu destino final.

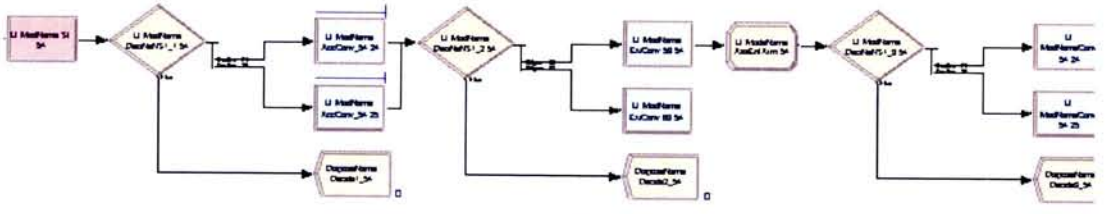


Figura 4.28 - Lógica de uma estação 2E2S

A criação da lógica acima descrita, de forma automática, assegura a simulação de mapas de sistemas de movimentação autónomos, num tempo muito reduzido quando comparado com o de sua criação manual.

4.5.2 Animação criada

A lógica garante a correcta simulação do modelo, mas oferece uma componente visual pouco clara. Para realizar a parte visual da simulação é necessário criar uma animação. A animação constrói os dois componentes fundamentais à visualização do fluxo da entidade no sistema de movimentação autónomo: Estações e Segmentos.

A animação é criada utilizando a informação do "*flat file*", analisando segmentos (definidos em cada linha do ficheiro), e sua direcção. Com essa informação o ARENA cria os elementos visuais *Station* e *Segment*, onde este reconhece as propriedades associadas aos *Segment* configuradas na parte da lógica. A animação não pode ser executada isoladamente, visto que esta reflecte o que acontece na lógica: se não existirem entidades a serem criadas, encaminhadas e depois eliminadas, não existe nada para ver.

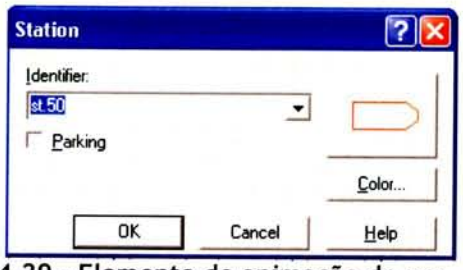


Figura 4.29 - Elemento de animação de um bloco *Station*

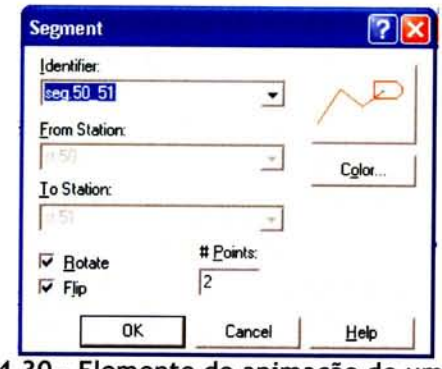


Figura 4.30 - Elemento de animação de um bloco *Segment*

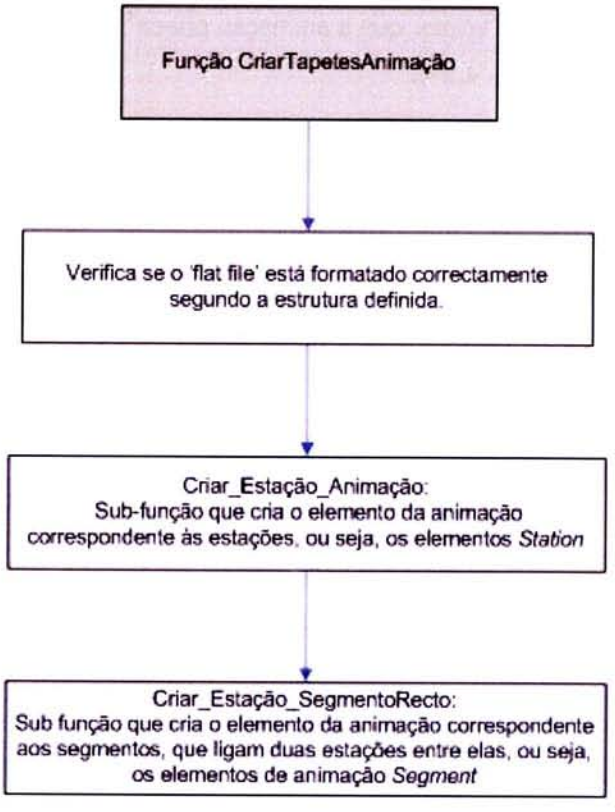


Figura 4.31 - Função criadora da animação do sistema de tapetes

O resultado é uma animação simples, mas capaz de mostrar o fluxo das entidades numa réplica visual do sistema que queríamos simular.

Figura 4.32 - Imagem do sistema de tapetes a modelar

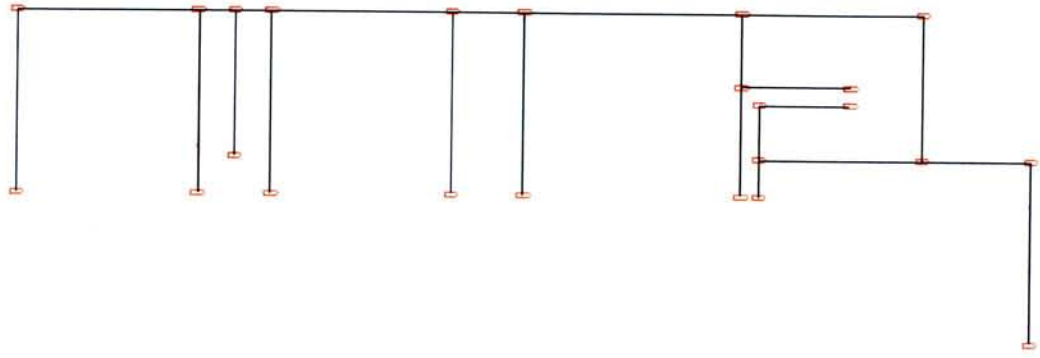


Figura 4.33 - Animação resultante do processo de criação automática do modelo

Podemos ver então nas figuras acima, que a animação criada automaticamente (figura 4.32) reflecte o sistema físico (figura 4.31).

4.6 - Integração dos componentes criados

A construção dos componentes foi sempre condicionada à futura integração dos mesmos numa biblioteca de componentes ou *template*, relativo ao projecto onde esta tese se insere.

Deste modo todos os módulos foram definidos com parâmetros que garantem a integração dos sistemas de movimentação autónomo (tapetes) com os outros componentes dos sistemas de logística interna (os vários tipos de armazém presentes na biblioteca e outros elementos futuros a desenvolver). Os componentes da animação e lógica dos tapetes podem, assim, ser chamados dento de um modelo que descreve um sistema complexo de logística interna, contendo diversos armazéns ligados por um sistema de tapetes. Os componentes criados garantem a simulação do sistema, simulando o fluxo de entidades entre tapetes e armazéns.

Podemos observar no *template* abaixo a contribuição do trabalho desenvolvido nesta dissertação presente nos 2 componentes assinalados a vermelho.

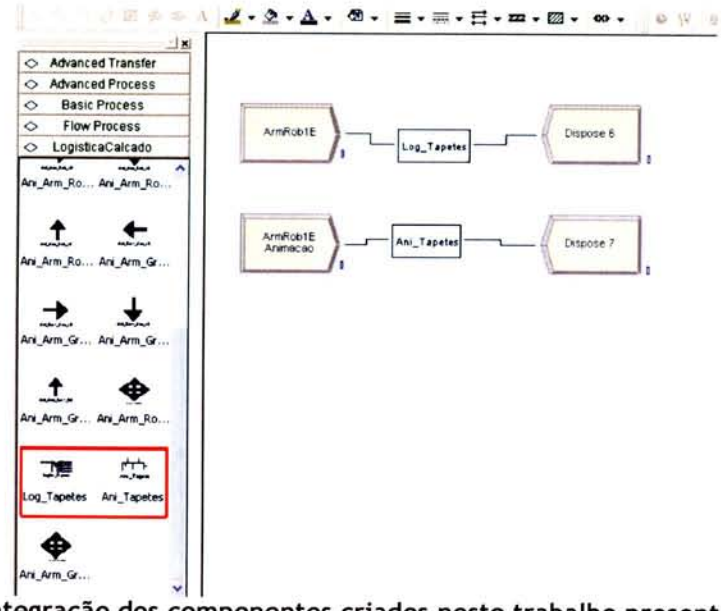


Figura 4.34 - Integração dos componentes criados neste trabalho presentes no *template* LogisticaCalçado

Os componentes podem ser então usados para criar sistemas complexos envolvendo diferentes armazéns ligados por sistemas de movimentação autónomos.

Exemplificando esta integração, criamos um mapa de tapetes com um armazém (figura 4.34):

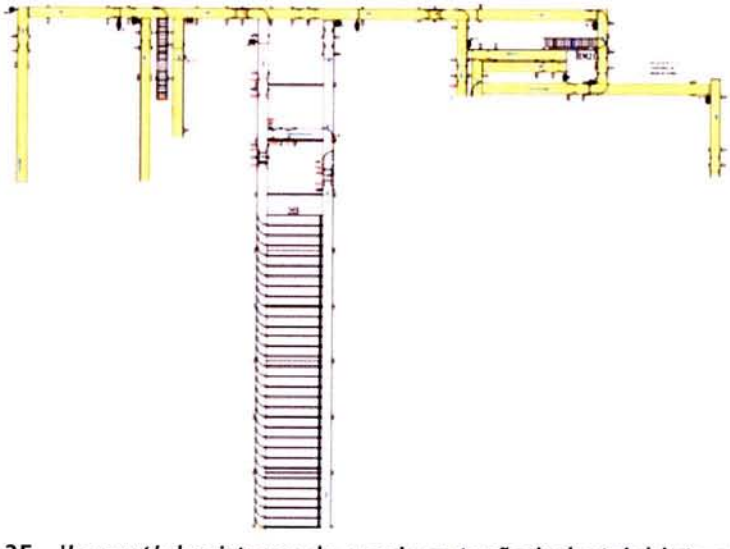


Figura 4.35 - *'Layout'* de sistema de movimentação industrial integrado com um armazém

Obtendo como resultado um modelo complexo , criado de forma automática, e representativo do sistema físico considerado (figura 4.35).

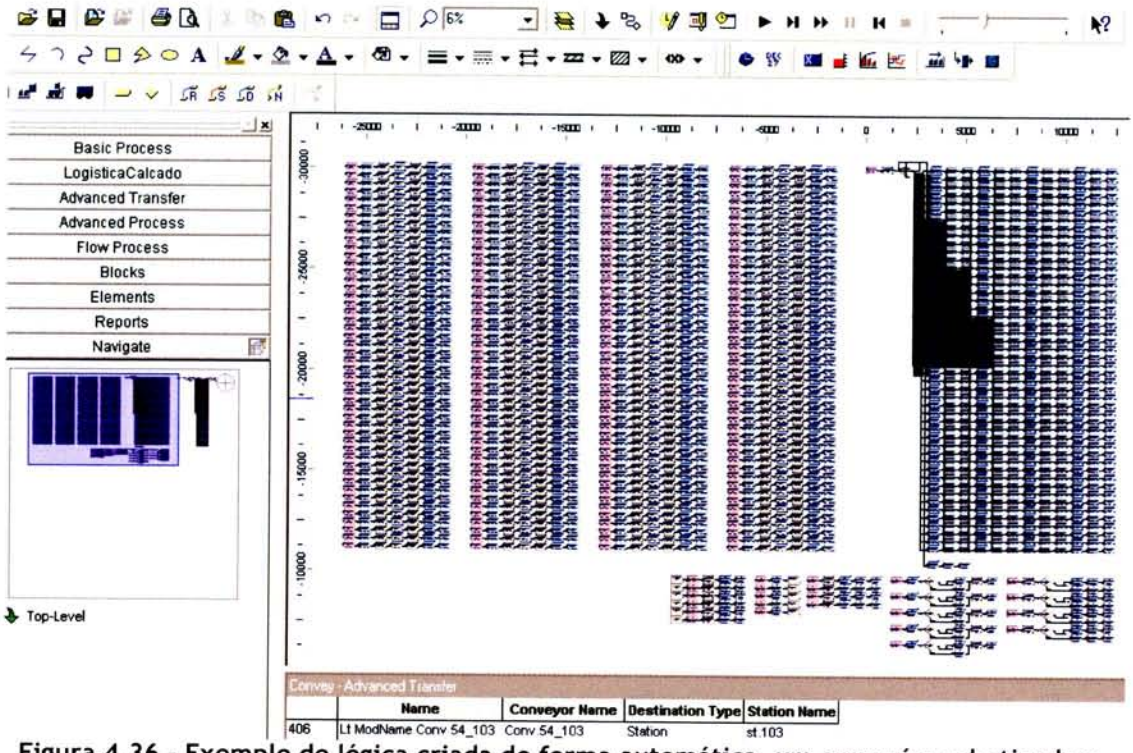


Figura 4.36 - Exemplo de lógica criada de forma automática, um armazém robotizado e um sistema de movimentação ligado ao mesmo.

4.7 - Conclusão

Foram construídos e integrados com sucesso num *template* dois componentes para uma biblioteca que pretende reduzir significativamente o esforço de concepção e de desenvolvimento de sistemas de logística interna. Estes componentes permitem, através da introdução de alguns dados por parte do utilizador, a criação automática de modelos de simulação de um conjunto de tapetes, permitindo-lhe avaliar o desempenho dos fluxos de uma entidade nesse sistema.

De seguida, serão discutidos os ganhos trazidos por estes componentes assim como as suas limitações. Serão apresentadas, também, outras possibilidades e melhoramentos futuros a que este trabalho pode fornecer a base.

Capítulo 5

Conclusões

5.1 - Problemática e objectivos

O objectivo principal do trabalho realizado nesta dissertação consistiu no estudo e desenvolvimento de elementos que possibilitam a construção de modelos de simulação de sistemas de movimentação, baseados em tapetes unidireccionais, no ambiente de simulação ARENA.

Normalmente, este tipo de sistemas é feito "à medida" de cada situação aumentando assim os custos do seu desenvolvimento e concepção. O desenvolvimento de bibliotecas de simulação com componentes flexíveis o suficiente para simular um grande número de sistemas de movimentação, servindo assim de apoio ao projecto e desenvolvimento deste tipo de soluções de automação de sistemas de logística interna, seria uma ferramenta de alto valor. Em empresas com sistemas deste tipo já implementados a sua optimização passa muitas vezes pela escolha óptima do *'layout'*. A simulação permite então manipular quantidade, disposição, velocidade e comprimento dos tapetes sem risco e com custos menores.

Estando este trabalho associado a um tal projecto de criação de uma biblioteca de componentes que permitam simular o hardware de sistemas de movimentação, foram definidos alguns objectivos mais precisos, relativos à construção dos componentes em questão:

- Construção de uma metodologia de inserção dos dados necessários à construção da lógica e animação de sistemas de movimentação autónomos (tapetes automáticos)
- Construção de uma ferramenta de criação automática da parte lógica dos modelos de simulação de sistemas de movimentação automáticos.

Estes componentes reduzem o esforço de concepção e de desenvolvimento de modelos de simulação deste tipo de sistemas de movimentação, eliminando assim tempo gasto a preencher informação repetitiva e eliminando o erro humano (nomes errados) proveniente do preenchimento de dezenas de módulos (centenas para modelos muito grandes). No caso especifico dos componentes desenvolvidos também elimina o trabalho de definir o encaminhamento nos diversos cruzamentos, sendo este feito de forma automática. Sistemas que necessitariam de dezenas ou centenas de blocos, e consequente preenchimento individual para serem modelados, são assim construídos em minutos e não horas.

A criação automática do modelo é feita através da integração VBA /ARENA. No entanto, encontrar informação acerca daquilo que é realizável ou de como é realizável não é fácil. Apesar de o ARENA fornecer *'smart files'* exemplificativos de algumas das suas potencialidades, não fornece, por exemplo, informação de como aceder aos diversos parâmetros dos módulos construídos. Mais informação acerca do funcionamento de algumas "ajudas" do ARENA como o *'Auto-connect'* também poderia ser fornecida (um exemplo onde esta é problemática, é nos blocos *Decide*, onde o seu funcionamento altera consoante o parâmetro *'Type'*). Foram, assim, encontradas diversas dificuldades ao longo da programação VBA/ARENA, que seriam superadas com bastante dificuldade, não fosse a experiência de utilização do ARENA por parte dos responsáveis pelo projecto onde este trabalho se insere.

A simulação de sistemas de movimentação tem um papel importante na melhoria de produtividade de uma empresa, visto que esta depende da sua eficiência e eficácia. Os modelos criados pela biblioteca e consequentemente os componentes para ela desenvolvida vão permitir estudar aspectos relevantes de um sistema de movimentação (estratégias de encaminhamento, filas de espera, etc.), permitindo uma noção do seu desempenho, quer através dos resultados estatísticos do ARENA, quer através das animações criadas. Os modelos resultantes poderão ser utilizados como ferramentas de apoio à decisão para analisar e avaliar o desempenho de sistemas de logística interna de empresas de diversos sectores.

5.2 - Principais contribuições do trabalho

Este trabalho debruçou-se sobre a criação de dois componentes de uma biblioteca desenvolvida para o ambiente de simulação ARENA. Estes componentes permitem ganhos de tempo substanciais, visto que o utilizador apenas precisa de criar um "*flat file*", de fácil preenchimento, com uma breve descrição do sistema. De facto, comparando o tempo de preenchimento do ficheiro em questão, com o tempo que demora a dispor os vários módulos e preencher todos os seus parâmetros no ARENA, as diferenças são enormes: e quanto maior for o modelo, maiores serão os ganhos.

Também à medida que a complexidade do sistema a modelar e, consequentemente, o número de módulos presentes no modelo aumenta, aumenta a probabilidade de erro humano

no preenchimento dos mesmos. Sendo que basta um erro para comprometer todo o modelo, também aqui as ferramentas de criação automática trazem ganhos de eficácia enormes ,visto a quase inevitabilidade de pequenas incoerências e erros de preenchimento.

Por fim, a presença de cruzamentos traz complexidade à criação de modelos visto ser necessário definir e fazer a gestão dos diversos caminhos. Aqui, o utilizador não precisa de verificar estes encaminhamentos: eles são preenchidos automaticamente pelos componentes criados.

Outra contribuição é a definição da informação necessária à simulação deste tipo de sistemas, a estrutura geral do "*flat file*" criado, que permite uma simulação adequada de um sistema de movimentação autónomo.

Espera-se então que estes componentes, inseridos numa biblioteca, sejam ferramentas flexíveis que ofereçam facilidade de concepção de modelos de sistema de movimentação automáticos e, como tal, sejam um incentivo à utilização de simulação em casos de implementação e desenvolvimento de sistemas deste tipo.

Também a simulação visual criada pretende facilitar a comunicação entre responsáveis de decisão e criadores de modelos.

Estes componentes poderão, pois, ser usados em vários projectos futuros e servir de primeiro passo para a criação de ferramentas futuras, uma vez que, pela sua simplicidade, apresentam-se como sendo facilmente integráveis em projectos de simulação.

5.3 - Limites e trabalho futuro

Devido ao tempo limitado e ao período de tempo de familiarização com o VBA e funcionamento do ARENA, o resultado final está ainda longe de ser um produto acabado.

Algumas possibilidades de desenvolvimento futuro, são:

Os tapetes poderão vir a ter um número indefinido de células. Ou seja, no trabalho aqui desenvolvido, o sistema, não aceita mais do que uma entidade a viajar no tapete, visto que todos têm apenas uma célula e todos os módulos reclamam uma célula para si. De facto, esta informação poderia vir a ser considerada na estrutura do '*flat file*'.

Um requisito de programação imposto pelos responsáveis do projecto, limitou o número de estações, relativas aos sistemas de tapetes em 50 estações, com ligações múltiplas. De facto, foi definido que o número identificador destas estações, teria de ser um número entre 50 e 100, sendo os outros números reservados para *estações origem, estações destino* ou outros componentes, garantindo a integração de todo o trabalho realizado com os restantes componentes do *template*. Esta limitação deveria ser superada, desenvolvendo um artificio que permita aceitar n estações, de maneira a possibilitar a criação de maiores e mais complexos sistemas de movimentação.

Devido às direcções admitidas para questões de animação, o número máximo de ligações possíveis numa estação é de 4 ligações. A lógica poderia ir mais além, mas devido à necessidade de ter uma versão testada, funcional e integrada no projecto em questão, tal não foi possível. Assim sendo, sugere-se que o sistema venha a implementar uma estação com encaminhamento para n estações de saída.

A interacção entre os componentes criados e o utilizador não é particularmente fácil. Tendo sido usado um *'flat file'* como método de inserção de dados, permitindo a integração rápida dos diferentes componentes realizados neste trabalho com os criados pelos responsáveis do projecto inicial, não foi colocado ênfase suficiente na criação de elementos que permitam uma interacção mais *user friendly*. Há, portanto, neste ponto muito espaço para melhoria e desenvolvimento mais cuidado.

Outras possibilidades de desenvolvimento futuro para este trabalho:

- Algoritmos de controlo, estratégias inteligentes de controlo para optimização da gestão de fluxos de entidades.
- Simulação de tapetes bi-direccionais
- Outros algoritmos de preenchimento dos blocos *Decide* responsáveis pelo encaminhamento de entidades no cruzamento.
- Considerar gamas operatórias no sistema, fazendo as entidades percorrer múltiplos destinos (por exemplo ir à estação 104, depois à 107 e finalmente à 108)

Referências

[1] A. L. Azevedo, "Integração Empresarial - Contexto," Faculdade de Engenharia da Universidade do Porto, 2007.

[2] Maria and Anu, "Introduction to modelling and simulation," in *Proceedings of the 1997 Winter Simulation Conference*, 1997.

[3] D. Sadowski, V. Bapat, and G. Drake, "The ARENA product family: Enterprise modeling solutions," in *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[4] W. D. Kelton, R. P. Sadowski, and D. T. Sturrock, *Simulation with Arena*. McGraw-Hill Professional, 2003.

[5] R. G. Ingals, "Introduction to Simulation," in *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[6] J. Banks, "Introduction to Simulation," in *Proceedings of the 2000 Winter Simulation Conference*, 2000.

[7] R. E. Shannon, "Introduction to the art and science of simulation," in *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[8] M. E. DINIZ, "UM MODELO DE APOIO A TOMADA DE DECISÃO PARA SOLUÇÃO DO PROBLEMA DE BALANCEAMENTO DE LINHA DE MONTAGEM: ESTUDO DE CASO EM UMA MANUFATURA ELETRÔNICA," 2005.

[9] W. D. Kelton and A. M. Law, *Simulation Modeling and Analysis* . McGraw Hill Higher Education, 2000.

[10] A. E. Brito and J. F. Teixeira, *Simulação por Computador Fundamentos e Implementacão de Código em C e C++*. 2001.

[11] P. J. Sánchez, "Fundamentals of simulation modelling," in *Proceedings of the 2007 Winter Simulation Conference*, 2007.

[12] L. C. R. N. P. Ferreira, "Geração Automática de Modelos de Simulação de uma Linha de Produção na Indústria Electrónica," Universidade do Minho; Escola de Engenharia; Departamento de Produção e Sistemas, 2003.

[13] A. F. d. O. Paiva, "Geração Automática de Modelos de Simulação de uma Linha de Produção na Industria Têxtil," Universidade do Minho; Escola de Engenharia, Departamento de Produção e Sistemas, 2005.

[14] P. J. d. R. e. S. Sá Marques, "Simulação de um Sistema Automático de Logística Interna para a Indústria de Calçado," Faculdade de Engenharia da Universidade do Porto, 2007.

[15] A. M. Law and M. G. McComas, "Simulation of manufacturing systems," in *Proceedings of the 1997 Winter Simulation Conference*, 1997.

[16] A. M. Law, "How to build valid and credible simulation models," in *Proceedings of the 2006 Winter Simulation Conference*, 2006.

[17] D. A. Sadowski, "Tips for successful practice of simulation," in *Proceedings of the 2007 Winter Simulation Conference*, 2007.

[18] J. Banks, "Software for Simulation," in *Proceedings of the 1993 Winter Simulation Conference*, 1993.

[19] M. J. Marcelino and T. Mendes, "Estratégias e ferramentas para construção de programas educativos de simulação," Centro de Informática e Sistemas - Universidade de Coimbra.

[20] C. R. Standridge, D. A. Finke, C. Jurishica, D. M. Ferrin, and C. M. Harmonosky, "What i wish they would have thaught me(or that i would have better remembered!) in school," in *Proceedings of the 2007 Winter Simulation Conference*, 2007.

[21] E. ABU-TAIEH and A. E. SHEIKH, "Commercial simulation packages: A comparative study," *International Journal of Simulation*, vol. 8, no. 2.

[22] J. O. Henriksen, "Taming The Complexity Dragon," in *Proceedings of the 2006 Winter Simulation Conference*, 2006.

[23] P. Klingstam and P. Gullander, "Overview of simulation tools for computer-aided production engineering," *Computers in Industry*, no. 38, p. 173-186, 1999.

[24] J. E. Hammann and N. A. Markovitch, "Introduction to ARENA," in *Proceedings of the 1995 Winter Simulation Conference.*, 1995.

[25] M. W. Rohrer, "Seeing is believing: The importance of visualization in manufacturing simulation," in *Proceedings of the 2000 Winter Simulation Conference*, 2000.

[26] V. Kachitvichyanukul, "Simulation environment for the new millenium (Panel)," in *Proceedings of the 2001 Winter Simulation Conference*, 2001.

[27] R. G. Ingalls, "Introduction simulation," in *Proceedings of the 2001 Winter Simulation Conference*, 2001.

[28] J. Banks, "Selecting simulation software," in *Proceedings of the 1991 Winier Simulation Conference*, 1991.

[29] C. McLean, C. Harrell, P. M. Zimmerman, and R. F. Lu, "Simulation standards: Current status, needs, and future directions," in *Proceedings of the 2003 Winter Simulation Conference*, 2003.

[30] J. Banks, "The future of simulation software: A panel discussion," in *Proceedings of the 1997 Winter Simulation Conference*, 1997.

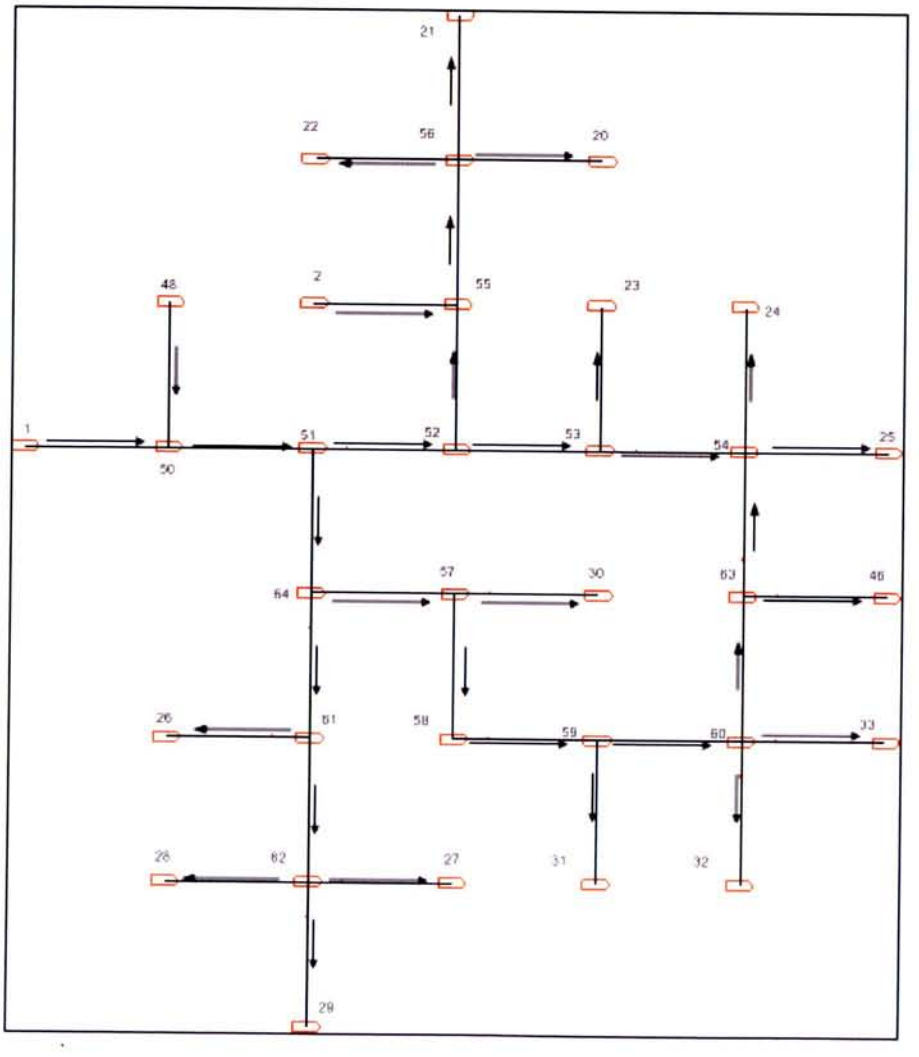
[31] F. Chance, J. Robinson, and J. Fowler, "Supporting manufacturing with simulation: Model design, development and deployment," in *Proceedings of the 1996 Winter Simulation Conference*, 1996.

ANEXOS

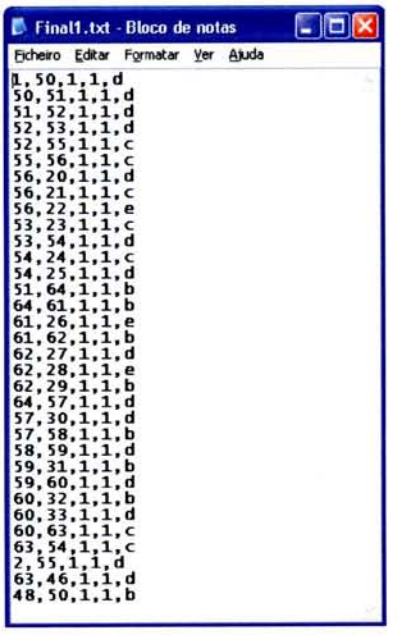
ANEXO A - MANUAL DE UTILIZADOR

Este secção destina-se a dar um exemplo prático dos passos que um utilizador da ferramenta deve efectuar para a criação de um sistema de movimentação automático.

1. O primeiro passo consistirá sempre em decompor o sistema a implementar em estações e cruzamentos. Isto não é necessariamente feito num documento formalmente construído; bastaria apenas um desenho à mão. O objectivo é ter informação necessária para escrever o "*flat file*".

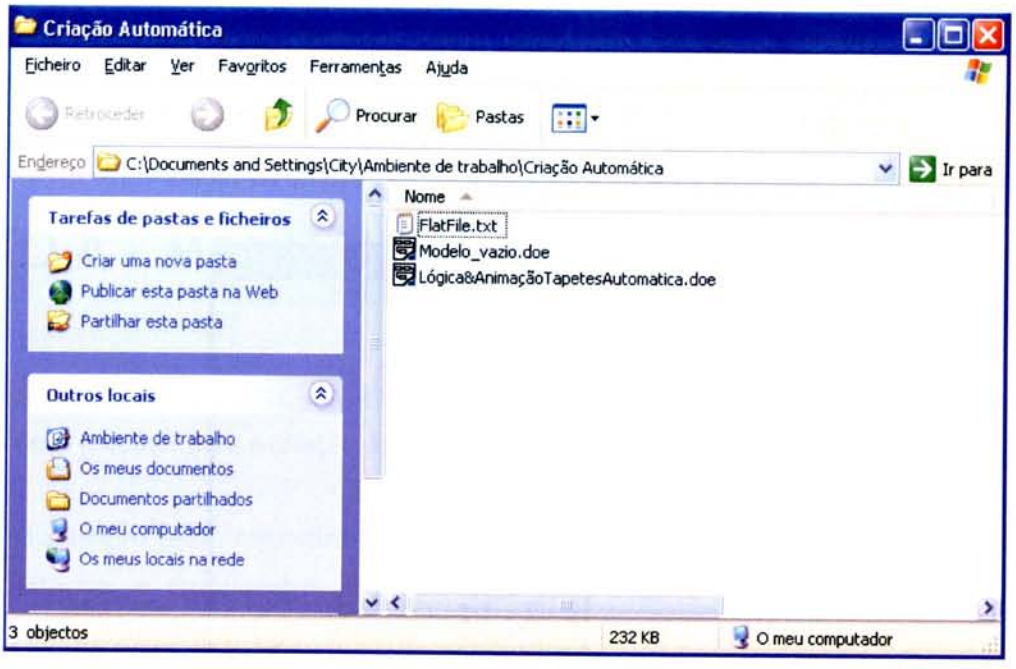


2. No passo seguinte o utilizador necessita de introduzir os diversos segmentos (os números das estações origem e estação destino, velocidade, comprimentos e direcções) num ficheiro .txt, o designado "*flat file*". Este deve cumprir a estrutura discutida no sub capítulo 4.3 e descrito na tabela 4.3.

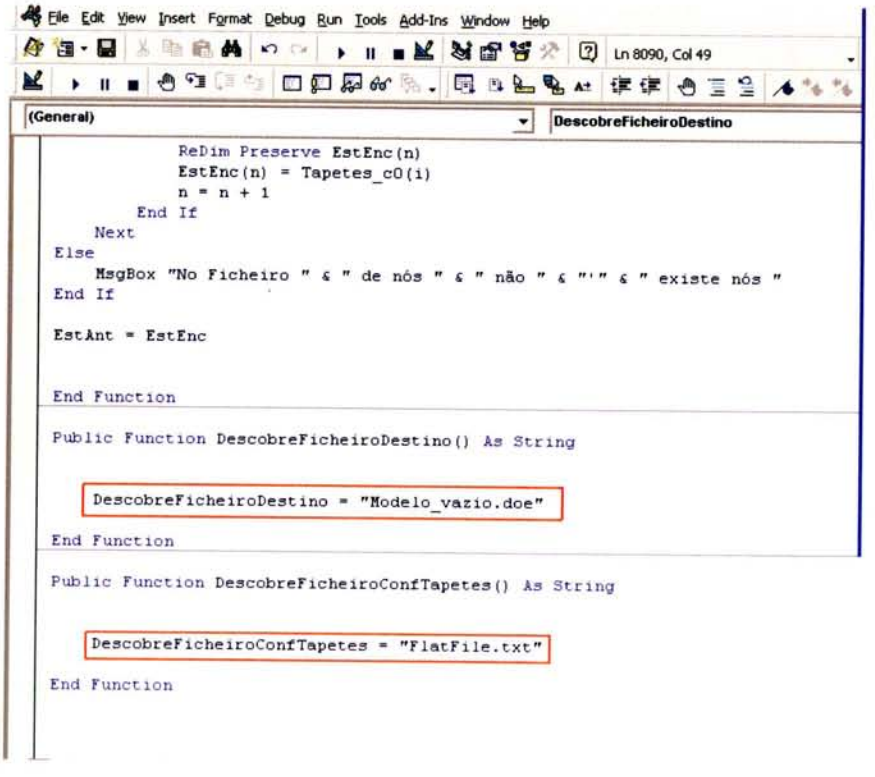


3. Este ficheiro deve ser colocado na mesma pasta do ficheiro .doe desenvolvido. Os ficheiro que devem estar presentes nesta pasta são:

- O ficheiro **Lógica&AnimaçãoTapetesAutomatica.doe** desenvolvido;
- O "Flat File"descritivo do sistema com o nome FlatFile.txt;
- E um ficheiro do tipo .doe do ARENA onde irá ser feita a criação da lógica e animação do sistema que se pretende simular.

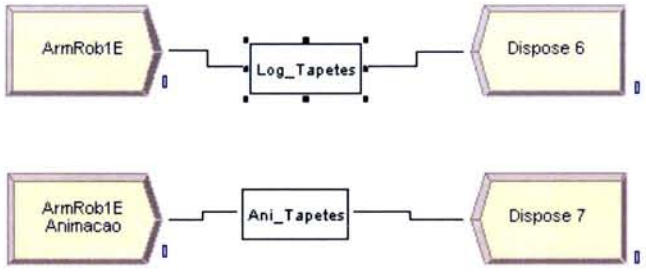


Caso seja necessário mudar o nome de um destes ficheiros é necessário aceder ao VB do ficheiro **Lógica&AnimaçãoTapetesAutomatica.doe** através do editor do ARENA onde as ultimas linhas de código devem ser editadas.

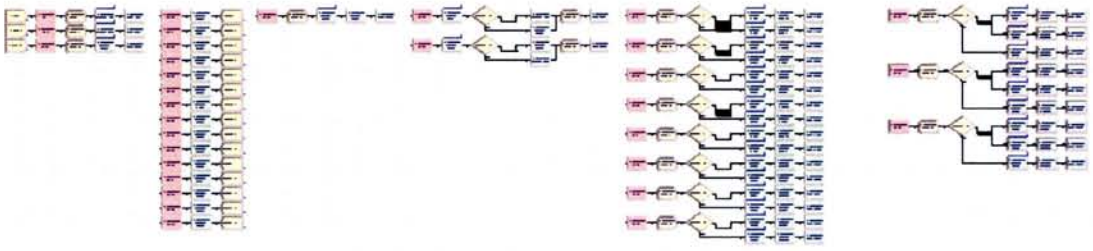


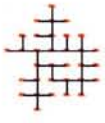
Após aberto o ficheiro **Lógica&AnimaçãoTapetesAutomatica.doe** no ARENA podemos, através de um "*double click*" em qualquer um dos dois componentes, indicar as coordenadas onde queremos começar a criar o modelo(lógica e animação com coordenadas separadas e

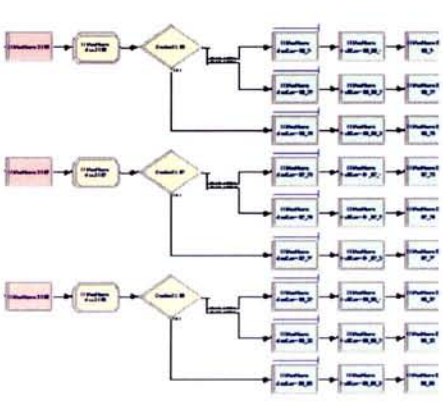
por pré-definição 0). Estas coordenadas dizem respeito ao ficheiro **Modelo_vazio.doe** onde por pré-definição é criado automaticamente o modelo de simulação.

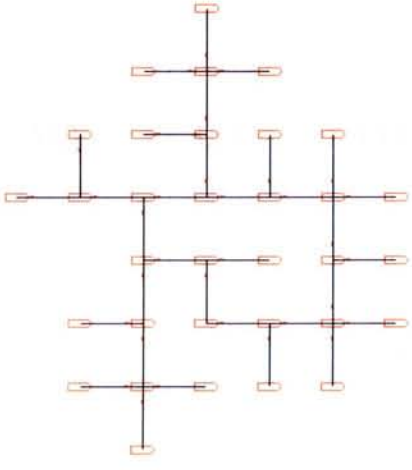


O modelo, nas suas duas partes separadas, é assim criado, no mesmo ficheiro .doe.

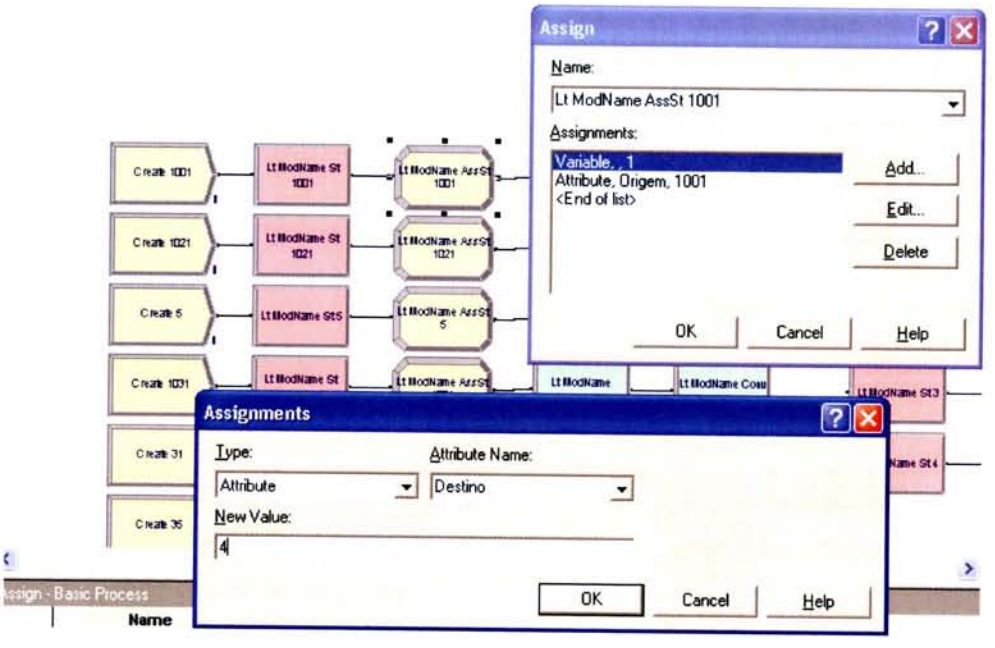








O passo final será o utilizador manipular os módulos *Create* e *Assign* para controlar o que simular. No módulo *Create* o utilizador decide os parâmetros da criação de entidades do sistema (Nome, tipo, quantidades e frequência). Nos módulos *Assign* o utilizador precisa de atribuir um destino as entidades criadas. Exemplificando:



Depois de garantir a criação de algumas entidades e lhes indexar destinos, o modelo está pronto a ser corrido, bastando carregar "*play*". A velocidade da simulação pode ser controlada com as teclas.

ANEXO B - EXEMPLO DE CÓDIGO

```
' '*************************************************************************************************
'Funcao que cria a lógica de estacoes com 1 Entrada e 1 saida _
'*************************************************************************************
Public Function Segmentos(EsteModelo As Model, ByVal X1 As Long, ByVal Y1 As Long)

Dim DistanciaEntreModulos, DistanciaYEntreModulos, NDist, NYDist As Integer
Dim i, j, n As Integer
Dim Indice_a As Long
Dim Modulo_a As Module
Dim Indice_b As Long
Dim Modulo_b As Module

Set Modelo = EsteModelo

'**********************Definicao das distancias entre os blocos*******************************
'*************************************************************************************************
DistanciaEntreModulos = 800
DistanciaYEntreModulos = 400

Dim PontosDestino() As Integer
Dim PontosOrigem() As Integer
Dim EstacaoActual As Integer

If (Not simples) <> -1 Then

For i = 0 To UBound(simples)

   NDist = 0

   EstacaoActual = simples(i)
   PontosOrigem = EstAnt(EstacaoActual)
   PontosDestino = EstSeg(EstacaoActual)

   '*************************************************************************************************
   'Construcao do módulo station
   '*************************************************************************************************
   Dim ModuloStation As Module
   Set ModuloStation = EsteModelo.Modules.Create("AdvancedTransfer", "Station", X1
+ NDist * DistanciaEntreModulos, Y1 + NYDist * DistanciaYEntreModulos)
   ModuloStation.Data("Name") = "Lt ModName St " & EstacaoActual
   ModuloStation.Data("Station Type") = "Station"
   ModuloStation.Data("Statn") = "st." & EstacaoActual
   ModuloStation.UpdateShapes

   NDist = NDist + 1

   '*************************************************************************************************
   ' construção do módulo assign
   '*************************************************************************************************
   Dim ModuloAssign As Module
   Set ModuloAssign = EsteModelo.Modules.Create("BasicProcess", "Assign", X1 + NDist
* DistanciaEntreModulos, Y1 + NYDist * DistanciaYEntreModulos)
```

```
            ModuloAssign.Data("Name") = "Lt ModName AssSt " & EstacaoActual
            ModuloAssign.Data("Type(1)") = "Attribute"
            ModuloAssign.Data("AName(1)") = "Origem"
            ModuloAssign.Data("Value(1)") = EstacaoActual
            ModuloAssign.UpdateShapes

            Indice_a = EsteModelo.Modules.FindByData("Name", "Lt ModName St " &
EstacaoActual)
            Set Modulo_a = EsteModelo.Modules(Indice_a)
            Indice_b = EsteModelo.Modules.FindByData("Name", "Lt ModName AssSt " &
EstacaoActual)
            Set Modulo_b = EsteModelo.Modules(Indice_b)
            Connect_Models Modulo_a, Modulo_b
            NDist = NDist + 1

            '*************************************************************************************************
            'Construcao do modulo Acesso
            '*************************************************************************************************
            Dim ModuloAcesso As Module
            Set ModuloAcesso = EsteModelo.Modules.Create("AdvancedTransfer", "Access", X1 +
NDist * DistanciaEntreModulos, Y1 + NYDist * DistanciaYEntreModulos)
            ModuloAcesso.Data("Name") = "Lt ModName AccConv " & EstacaoActual & "_" &
PontosDestino(0)
            ModuloAcesso.Data("Conv") = "Conv." & EstacaoActual & "_" & PontosDestino(0)
            ModuloAcesso.Data("Qty") = "1"
            ModuloAcesso.Data("QSG") = "Queue"
            ModuloAcesso.Data("QName") = "Que.AccConv" & "_" & EstacaoActual & "_" &
PontosDestino(0)
            ModuloAcesso.UpdateShapes


            '*************************************************************************************************
            'Liga os 2 móulos anteriores
            '*************************************************************************************************

            Indice_a = EsteModelo.Modules.FindByData("Name", "Lt ModName AssSt " &
EstacaoActual)
            Set Modulo_a = EsteModelo.Modules(Indice_a)
            Indice_b = EsteModelo.Modules.FindByData("Name", "Lt ModName AccConv " &
EstacaoActual & "_" & PontosDestino(0))
            Set Modulo_b = EsteModelo.Modules(Indice_b)
            Connect_Models Modulo_a, Modulo_b
            NDist = NDist + 1

            '*************************************************************************************************
            'Construcao do modulo Exit
            '*************************************************************************************************
            Dim ModuloExit As Module
            Set ModuloExit = EsteModelo.Modules.Create("AdvancedTransfer", "Exit", X1 + NDist
* DistanciaEntreModulos, Y1 + NYDist * DistanciaYEntreModulos)
            ModuloExit.Data("Name") = "Lt ModName ExitConv " & PontosOrigem(0) & "_" &
EstacaoActual
            ModuloExit.Data("Conv") = "Conv." & PontosOrigem(0) & "_" & EstacaoActual
            ModuloExit.UpdateShapes

            Indice_a = EsteModelo.Modules.FindByData("Name", "Lt ModName AccConv " &
EstacaoActual & "_" & PontosDestino(0))
            Set Modulo_a = EsteModelo.Modules(Indice_a)
            Indice_b = EsteModelo.Modules.FindByData("Name", "Lt ModName ExitConv " &
PontosOrigem(0) & "_" & EstacaoActual)
```

```
            Set Modulo_b = EsteModelo.Modules(Indice_b)
            Connect_Models Modulo_a, Modulo_b

            NDist = NDist + 1

            '*****************************************************************************************
            'Construção do módulo Convey
            '*****************************************************************************************
            Dim ModuloConvey As Module
            Set ModuloConvey = EsteModelo.Modules.Create("AdvancedTransfer", "Convey", X1
+ NDist * DistanciaEntreModulos, Y1 + NYDist * DistanciaYEntreModulos)
            ModuloConvey.Data("Name") = "Lt ModName Conv " & EstacaoActual & "_" &
PontosDestino(0)
            ModuloConvey.Data("Conv") = "Conv." & EstacaoActual & "_" & PontosDestino(0)
            ModuloConvey.Data("SG") = "Station"
            ModuloConvey.Data("Station") = "St." & PontosDestino(0)
            ModuloConvey.UpdateShapes

            Indice_a = EsteModelo.Modules.FindByData("Name", "Lt ModName ExitConv " &
PontosOrigem(0) & "_" & EstacaoActual)
            Set Modulo_a = EsteModelo.Modules(Indice_a)
            Indice_b = EsteModelo.Modules.FindByData("Name", "Lt ModName Conv " &
EstacaoActual & "_" & PontosDestino(0))
            Set Modulo_b = EsteModelo.Modules(Indice_b)
            Connect_Models Modulo_a, Modulo_b

            '*****************************************************************************************
            'cria os módulos conveyor e segment anterior. Este módulo é só de dados
            '*****************************************************************************************
            Dim ModuloSegment As Module

            Indice_a = EsteModelo.Modules.FindByData("Name", "Seg." & EstacaoActual & "_" &
PontosDestino(0))
            If Indice_a = 0 Then

            Set ModuloSegment = EsteModelo.Modules.Create("AdvancedTransfer", "Segment",
0, 0)  ' 0,0 porque é so de dados
            ModuloSegment.Data("Name") = "Seg." & EstacaoActual & "_" & PontosDestino(0)
            ModuloSegment.Data("BegStation") = "st." & EstacaoActual
            ModuloSegment.Data("NextStation(1)") = "st." & PontosDestino(0)
               For n = 0 To UBound(Tapetes_c2)
                     If Tapetes_c0(n) = EstacaoActual And Tapetes_c1(n) = PontosDestino(0)
Then
                        ModuloSegment.Data("Length(1)") = Trim(Str(Int(Tapetes_c2(n))))
                        Exit For
                     Else
                     End If
               Next
               ModuloSegment.UpdateShapes

            Else
            End If

            Indice_a = EsteModelo.Modules.FindByData("Name", "Conv." & EstacaoActual & "_"
& PontosDestino(0))
            Set Modulo_a = EsteModelo.Modules(Indice_a)
            Modulo_a.Data("Segment") = "Seg." & EstacaoActual & "_" & PontosDestino(0)
            Modulo_a.Data("Type") = "Accumulating"
               For n = 0 To UBound(Tapetes_c3)
                  If Tapetes_c0(n) = EstacaoActual And Tapetes_c1(n) = PontosDestino(0) Then
```

```
                    Modulo_a.Data("Vel") = Trim(Tapetes_c3(n))
                    Exit For
                Else
                End If
            Next
        Modulo_a.Data("Units") = "Per Second"
        Modulo_a.Data("CellSize") = "1"
        Modulo_a.Data("MaxPerEnt") = "1"
        Modulo_a.Data("EntSize") = "1"
        Modulo_a.Data("MaxPerEnt") = "1"
        Modulo_a.Data("Status") = "Active"

        NYDist = NYDist + 1

        If X1 + NDist * DistanciaEntreModulos > 25000 Then
        MsgBox "Desenho de tapetes perto dos limites permitidos... experiemente outras
coordenadas"
                Exit For
            End If

        If Y1 + NYDist * DistanciaYEntreModulos > 25000 Then
                X1 = X1 + NDist * DistanciaEntreModulos + DistanciaEntreModulos
                NYDist = 0
        Else
        End If

    Next

    ReDim Preserve Posicao(1)

    Posicao(0) = X1 + NDist * DistanciaEntreModulos
    Posicao(1) = Y1 + NYDist * DistanciaYEntreModulos

    Else
    End If

    End Function
```

Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL
www.fe.up.pt