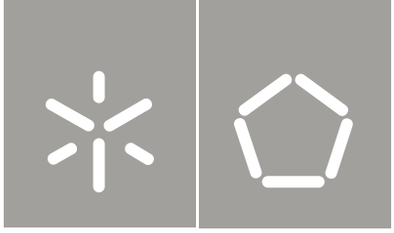
Universidade do Minho
Escola de Engenharia

Bruno Lourenço Silva Ferreira

A Aprendizagem da Simulação Através dos Diagramas Ciclo de Atividades – Uma Ferramenta de Modelação

Outubro de 2012

A Aprendizagem da Simulação Através dos Diagramas Ciclo de Atividades – Uma Ferramenta de Modelação
Bruno Lourenço Silva Ferreira
UMinho | 2012



Universidade do Minho
Escola de Engenharia

Bruno Lourenço Silva Ferreira

A Aprendizagem da Simulação Através dos Diagramas Ciclo de Atividades – Uma Ferramenta de Modelação

Tese de Mestrado
Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação do
Professor Doutor Guilherme A. B. Pereira
Professor Doutor Ricardo J. Machado

Outubro de 2012

Agradecimentos

A realização deste trabalho de dissertação só foi possível graças ao apoio direto, e indireto, de um conjunto de pessoas, em relação às quais desejo expressar o meu mais profundo agradecimento.

Em primeiro lugar desejo agradecer aos meus orientadores, Professores Guilherme A. B. Pereira e Ricardo J. Machado, pelo seu empenhamento e pelas críticas e sugestões com que permitiram enriquecer este trabalho.

Ao Professor Luís M. S. Dias que, apesar de não ter participado diretamente neste trabalho, sempre me desafiou para implementar uma solução desta natureza.

Aos meus professores e colegas de departamento que sempre me motivaram e se mostraram interessados no trabalho desenvolvido.

Aos funcionários do Bar de Engenharia II, cuja simpatia e boa disposição permitiram manter um ânimo constante.

A todos aqueles com que, como colegas da parte curricular do mestrado, ou de laboratório, tive o prazer de trabalhar, ou conviver.

A um conjunto muito restrito de amigos que sempre me apoiaram e souberam desculpar/compreender a minha ausência.

Finalmente, à Sandra, à minha afilhada, sobrinhos, pais, irmãos, tios, primos e avós, pelo carinho e pelo apoio sempre conselheiro e consolador.

Resumo

A metodologia de ensino e aprendizagem da simulação para estudantes universitários, nomeadamente no que concerne à simulação discreta, não tem dado o desejável relevo à compreensão dos conceitos-base da simulação — antes se concentra nas questões de sintaxe associadas às ferramentas de simulação utilizadas, nos quais se incluem ferramentas comerciais de simulação e linguagens de programação genéricas. Esta é tida como uma das razões para que a simulação não tenha ainda atingido os níveis de utilização que o seu inegável potencial de aplicabilidade exigiria.

Este trabalho pretende, por isso, explorar formas de reforçar o entendimento dos conceitos basilares da simulação, apoiadas no esforço de modelação — Diagramas de Ciclo de Atividades. Assim, pretende-se desenvolver uma ferramenta que exija aos alunos esse conhecimento de modelação e que, através dos modelos construídos, consiga construir o programa de simulação, que quando executado permitirá analisar os resultados da simulação.

Uma vez que se desconhece a exequibilidade da construção da ferramenta a que nos propomos, e porque a atividade é um conceito, historicamente, pouco explorado pelos fabricantes de ferramentas de simulação, para justificar o investimento no desenvolvimento de uma ferramenta desta natureza, proceder-se-á neste trabalho à construção de um primeiro protótipo.

Depois de enquadrado o problema que se pretende resolver com o desenvolvimento da ferramenta, e de definidos os objetivos para o protótipo, proceder-se-á descrição de cada um dos esforços de desenvolvimento. A validade da solução apresentada será posta à prova através da resolução de três casos de estudos, normalmente utilizados no ensino de simulação.

Abstract

Teaching Simulation Basics Through Activity Cycle Diagrams – An Activity World View Modeling Tool

The methodology of learning and teaching simulation for college students, namely in regard to the discrete simulation, has not given the desired emphasis to the understanding of the basic concepts of simulation – instead it concentrates on questions of syntax, associated to the tools employed for simulation, in which are included commercial tools for simulation and generic programming languages. This is regarded as one of the reasons why simulation has not yet achieved the levels of use that its undeniable potential of applicability would demand.

This work therefore seeks to explore ways of reinforcing the understanding of the core concepts of simulation, supported on the effort of modeling – Activity Cycle Diagrams. Thus, it is intended the development of a tool that demands from the students, that modeling knowledge, and through the constructed models, be able to build the simulation program.

Since one does not know the feasibility of the construction of the tool that we propose, and because the activity is a concept historically poorly explored by the manufacturers of simulation tools, to justify the investment on the development of a tool of this nature, it will be undertaken, in this work, the construction of the first prototype.

Once framed the problem to be solved with the development of the tool, and the definition of the objectives for the prototype, it will be described every one of the efforts of development. The validity of the solution presented will be put to the test by the resolution of three case studies, normally used in the teaching of simulation.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CAPS	*Computer Aided Programing of Simulations*
CSV	*Comma-Separated Values*
DCA	Diagramas Ciclo de Atividades
DSR	*Design Science Research*
ECSL	*Extended Control and Simulation*
FIFO	*First In First Out*
GPL	*General Purpose Language*
SDK	*Software Development Kit*
SI	Sistemas de Informação
SPL	*Special Purpose Language*
VB	*Visual Basic*
VBA	*Visual Basic for Applications*
VBS	*Visio Basic for Simulations*

1. Introdução

1.1 Enquadramento

A simulação como processo de modelação de sistemas e de condução de experiências [Shannon, 1998] tem-se tornado numa abordagem prática indispensável para a resolução dos mais variados problemas que atualmente engenheiros, cientistas e gestores enfrentam [Altiok *et al.*, 2001].

Esta técnica já faz parte do quotidiano de disciplinas como a Investigação Operacional, Engenharia Industrial, Ciências Empresariais [Goldsman, 2007], e Sistemas de Informação (SI) [April, Better, Glover, Kelly, & Laguna, 2006; Turban, Sharda, & Delen, 2010], sendo cada vez mais as áreas que ao nível aplicacional recorrem a esta técnica.

Apesar de conhecidas as potencialidades desta técnica na resolução de problemas, o seu uso, quando comparado com o seu potencial, é ainda surpreendentemente

baixo [Ståhl *et al.*, 2003]. Para que a utilização desta técnica se generalize, será necessário cativar aqueles que são considerados os potenciais interessados em simulação (alunos de áreas de gestão e engenharia), e rever a forma como esta disciplina é lecionada [Born, 2003].

Na abordagem prática do ensino introdutório, desta disciplina, são normalmente utilizadas ferramentas comerciais de simulação, e linguagens de programação genéricas que poderão ser complementadas com bibliotecas próprias de simulação.

A utilização de ferramentas comerciais poderá ser uma alternativa à falta de conhecimentos prévios de programação, por parte de alunos de áreas não tecnológicas. No entanto, esta abordagem não está isenta de riscos podendo estar a limitar o ensino à aprendizagem da utilização destas ferramentas, que sendo orientadas por interesses comerciais e desenhadas para serem utilizadas por especialistas, não evidenciam os fundamentos em que a simulação se alicerça.

A dificuldade de compreensão, por leigos, dos modelos criados nas ferramentas comerciais é um problema que surge normalmente associado ao elevado grau de sofisticação dessas ferramentas, e está a fazer com que, na comunicação entre cliente e especialista, um mesmo sistema tenha de ser representado em formalismos e suportes diferentes [Luís M S Dias, Rodrigues, & Pereira, 2005]. Numa primeira fase, os modelos são desenhados num formato facilmente compreendido pelo cliente. Posteriormente e depois de aprovados, os modelos são traduzidos e representados no formalismo de representação utilizado pela ferramenta de simulação, que varia de acordo com a ferramenta.

O elevado grau de sofisticação das ferramentas comerciais e dos formalismos de representação, orientados à automatização dos modelos que estas ferramentas utilizam, poderão afetar igualmente o ensino, principalmente nas disciplinas introdutórias, onde os conhecimentos de simulação ainda se estão a constituir.

De entre os formalismos de representação existentes, aquele que devido à sua simplicidade, riqueza sintática e semântica e que, por ser orientado à compreensão do sistema a simular, mais se adequa ao ensino de simulação é baseado no conceito de atividade [Luís M S Dias, Pereira, & Rodrigues, 2006; Kang & Choi, 2011], sendo designado por Diagrama Ciclo de Atividades (DCA). Apesar das primeiras ferramentas de simulação se terem baseado no conceito de atividade este paradigma tem sido, historicamente, pouco explorado pelos fabricantes de ferramentas de simulação [Nance, 1995].

Um outro aspeto, relativo ao ensino, que também tem sido discutido é sobre a inclusão, ou exclusão, da temática de construção de programas de simulação nos currículos de simulação. A exclusão dessa temática poderá favorecer o ensino aos alunos de cursos não tecnológicos, no entanto, poderá comprometer a manutenção e o aparecimento de novas ferramentas de simulação, uma vez que os novos especialistas deixam de estar habilitados para essas tarefas [Ståhl, 2000].

As particularidades do ensino desta disciplina, e o desajuste das ferramentas comerciais utilizadas no ensino, têm alimentado algumas discussões sobre qual a melhor forma de ensino e sobre as características que as ferramentas, para o ensino de simulação, deverão possuir. Ingolf Ståhl, uma das vozes ativas na discussão desta problemática, elaborou um estudo onde identificou os requisitos que este tipo de ferramentas deverá incluir.

Conhecidos os riscos que poderão advir da utilização de ferramentas comerciais no ensino de simulação e considerando que o paradigma que mais se adequa ao ensino continua a ser o menos explorado, pretende-se, através da construção de um protótipo, verificar se é possível construir uma ferramenta que, sendo baseada no conceito de atividade, possa ser usada no ensino de simulação e que possa justificar um investimento, futuro, em ferramentas desta natureza.

Uma vez que se pretende determinar a exequibilidade da construção de uma ferramenta futura para o ensino de simulação, para mitigar o risco da sua não exequibilidade, foram utilizadas ferramentas de desenvolvimento rápido e tecnologias de fácil integração. No desenvolvimento do protótipo foram considerados os requisitos identificados e caracterizados por Ståhl [2000].

1.2 Objetivos e Resultados Esperados

O objetivo principal deste trabalho consiste em dar resposta à questão de investigação: "Será possível construir uma ferramenta, baseada no conceito de atividade, para o ensino introdutório de simulação?".

A resposta à questão de investigação será determinada através do desenvolvimento de um protótipo, inicial, da ferramenta que deverá ser capaz de resolver três casos de estudo normalmente utilizados no ensino introdutório de simulação: máquinas semiautomáticas, agência bancária e bar.

Além de ser capaz de resolver os casos de estudo supracitados, de uma forma geral, o protótipo deverá possuir as seguintes funcionalidades:

1. Permitir a construção de modelos de simulação, através de DCA;
2. Com base na interpretação dos modelos construídos, gerar de forma automática o programa de simulação correspondente;
3. Permitir aumentar, modificar ou corrigir o programa de simulação gerado;
4. Executar o programa de simulação gerado, aumentado, modificado ou corrigido pelo utilizador, mostrando os resultados da simulação.

1.3 Abordagem Metodológica

Sendo este um projeto de natureza científica, surge a necessidade de recorrer a abordagens de investigação metodológicas, que credibilizem todo o trabalho realizado. Essa necessidade é ainda maior no domínio dos SI nas situações em que o resultado do trabalho realizado se poderá confundir com a aplicação de um conjunto de normas, e boas práticas, ao desenvolvimento de produto novo, ou já existente.

O cariz multidisciplinar dos SI exige métodos de investigação próprios. Isto porque, a investigação neste domínio debruça-se, não só, com os aspetos tecnológicos ou sociais dos sistemas, mas também, com os fenómenos que resultam da sua interação. A *Design Science Research* (DSR) [Peffers, Tuunanen, Rothenberger, & Chatterjee, 2008] é uma metodologia que tem vindo a ser desenvolvida, e que surgiu para unificar, e ajustar, as diferentes abordagens utilizadas na condução de projetos desta natureza.

A escolha da DSR para abordagem metodológica deste projeto deveu-se fundamentalmente a três aspetos: (1) tempo disponível para a execução do projeto, (2) facilidade de aplicação aos SI, e (3) adequação da metodologia ao objetivo do projeto. A duração estipulada para o projeto fez com que apenas fossem consideradas as abordagens, conhecidas, como sendo de curta duração — o que excluiu, logo à partida, a *Action Research*. O fator tempo limitou, também, a escolha de abordagens que tivessem atividades bem definidas, ou se ajustassem aos SI. Em virtude destes critérios sobressaiu a DSR. Além de responder aos dois critérios já referidos, o objetivo dessa metodologia confunde-se com o objetivo do próprio projeto (terceiro fator), que na terminologia dessa mesma metodologia, é a criação de um artefacto, cuja instanciação (protótipo) pretende resolver um problema existente.

1.3.1 *Design Science Research*

De acordo com Peffers *et al.* [2008], a DSR é uma metodologia de investigação, na qual a resposta a questões relevantes, e o contributo ao corpo científico de conhecimento, é efetuado através da construção de artefactos inovadores. O mesmo autor refere que, o fundamento principal desta metodologia é que o conhecimento para a compreensão e resolução de problemas advém da criação e aplicação de artefactos, que podem ser novos, ou versões melhoradas, de constructos, modelos, métodos, instanciações, ou teorias de conceção já existentes. Conforme se poderá observar na Figura 1.1, a metodologia adotada é composta por seis atividades (que designaremos por fases), e quatro pontos de entrada.

Fases da Metodologia

Na primeira fase é efetuada a (1) identificação do problema, mais especificamente a questão de investigação, sendo também demonstrada a pertinência da sua investigação. A segunda fase corresponde à (2) definição dos objetivos pretendidos para o artefacto.

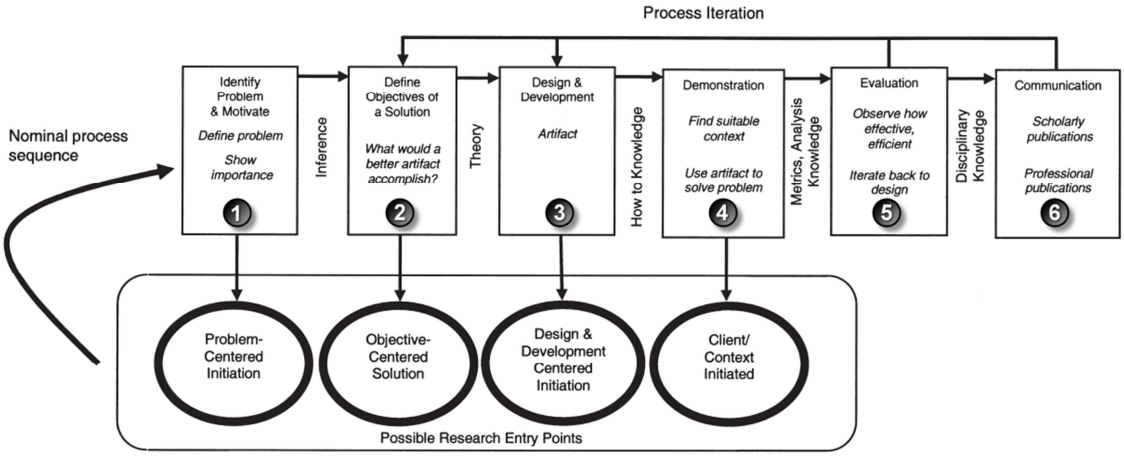


Figura 1.1: Modelo de processos da metodologia DSR, adaptado de Peffers *et al.* [2008]

Os objetivos serão inferidos através do problema identificado na fase anterior. A terceira fase é onde se inicia o processo de (3) conceção e implementação do artefacto, na qual se incluem a definição dos requisitos a implementar, a conceção

da arquitetura, e a implementação do artefacto propriamente dito. Os requisitos deverão ser definidos através dos objetivos explicitados na fase anterior. Estando o artefacto implementado, efetuar-se-á, na quarta fase, (4) a demonstração da utilidade da solução, ou seja, verificar-se-á se o artefacto resolve uma, ou mais, instâncias do problema. Na quinta fase será (5) avaliada a qualidade do artefacto produzido, ou seja, com base nos resultados obtidos na fase anterior, será avaliada a eficácia e a eficiência com que o artefacto resolve o problema a que se propôs. Se os resultados não forem satisfatórios, poderá ser necessário voltar à terceira fase. A última fase da metodologia corresponde à (6) divulgação da resposta ao problema, através do artefacto produzido.

Pontos de Entrada na Metodologia

Conforme já referido, existem quatro pontos, possíveis, de entrada na metodologia. O ponto de entrada deverá coincidir com o objetivo da própria investigação. Foram considerados quatro objetivos diferentes: (1) resposta a um problema existente; (2) resposta a uma necessidade da indústria; (3) aplicação de um artefacto existente, a outro domínio de aplicação; (4) reengenharia do processo, em o que se pretende é dar um cariz científico a uma solução usada na prática. Ao contrário do que acontece com os três primeiros pontos de entrada, o último ponto, será executado de forma decrescente, ou seja, o processo iniciará na quarta fase e terminará na primeira.

1.3.2 Instanciação da Metodologia

Conforme apresentado na Figura 1.2, e em virtude do tempo disponibilizado para a concretização deste trabalho de dissertação, apenas foram efetuadas as quatro primeiras fases da metodologia. Pretendendo este dar resposta a um problema existente, o ponto de entrada coincidiu com a primeira fase da metodologia.

Pré-Dissertação		**Dissertação**		**Pós-Dissertação**	
Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Fase 6

Figura 1.2: Distribuição das fases ao longo do processo de dissertação

Na primeira fase (1) foi efetuada a caracterização do problema, tendo também sido demonstrada a sua relevância, assim como a necessidade de soluções para o problema. Na segunda fase (2) foram identificados os requisitos que o protótipo deverá satisfazer. A conceção e a implementação do protótipo foram executadas ao longo da (3) terceira fase. Na quarta (4), e última fase deste trabalho, procedeu-se à demonstração do funcionamento do protótipo.

Em virtude do tempo disponibilizado e do calendário de dissertação, não foi possível avaliar o protótipo em contexto de sala de aula, ficando essa tarefa, assim como a comunicação dos resultados, para trabalho futuro.

Convém referir que esta metodologia foi complementada com o recurso a outras técnicas. Nas duas primeiras fases, foram aplicadas técnicas de pesquisa bibliográfica; na terceira fase, foram utilizadas técnicas da Engenharia de Software; e na quarta fase, para a avaliação do protótipo, foram utilizados casos de estudo.

1.4 Estruturação do Documento

Este documento encontra-se estruturado em cinco capítulos. No primeiro capítulo, já apresentado e no qual esta secção se insere, é efetuado o enquadramento do trabalho, sendo também referidos objetivos e os resultados que se pretendem alcançar. A abordagem metodológica, e as suas fases, são também descritas neste capítulo.

O segundo capítulo está reservado para a revisão da literatura, que se encontra dividida em três partes. Na primeira parte será efetuada uma introdução à temática da simulação, a que se seguirá o estudo dos formalismos de representação, e que culminará com a análise da problemática do ensino de simulação, de onde se extraíram os requisitos que as ferramentas de simulação deverão implementar.

No terceiro capítulo, dedicado inteiramente ao desenvolvimento do protótipo, é explicada a estratégia seguida, sendo também descritos cada um dos esforços de desenvolvimento do protótipo. Neste capítulo é também referida a estrutura interna do protótipo, assim como a sua lógica de funcionamento.

O protótipo é colocado à prova no quarto capítulo onde, através da utilização de três casos de estudo, é demonstrado o seu funcionamento. Além da demonstração, neste capítulo, explica-se o código fonte do programa de simulação gerado, e avalia-se os resultados apresentados pelo protótipo.

No quinto e último capítulo é efetuada uma análise ao trabalho realizado, onde, além de uma avaliação geral ao trabalho desenvolvido, serão tecidas algumas considerações sobre os contributos técnicos e científicos do trabalho. O capítulo encerrará com o trabalho futuro, onde será perspetivado o trabalho ainda por realizar.

2. Conceitos Fundamentais de Simulação

2.1 Simulação

Num mundo cada vez mais competitivo a simulação tem-se tornado numa metodologia indispensável para a resolução de problemas, permitindo o estudo, análise e avaliação de situações complexas, que de outra forma não seria possível [Shannon, 1998]. Esta disciplina, quando devidamente aprendida, é uma ferramenta bastante poderosa [Ingalls, 2008], ajudando as organizações a reduzir custos, aumentar a qualidade e a produtividade, assim como a diminuir o *time--to-market* [Zhou, Son, & Chen, 2004].

A simulação faz já parte do arsenal da Investigação Operacional, Engenharia Industrial e Ciências Empresariais [Goldsman, 2007], tendo nos últimos tempos aparecido também ligada aos SI, mais concretamente aos Sistemas de Suporte à Decisão [Turban *et al.*, 2010] e aos Sistemas de *Business Process Management* [April *et al.*, 2006].

Segundo Khoshnevis [1994] e Luís M S Dias [2005], ao nível aplicacional, são cada vez menos as áreas que não recorrem à simulação, sendo ainda menos aquelas que não poderão beneficiar com a sua utilização. Os mesmos autores referem que esta técnica é utilizada em áreas como as Comunicações, Finanças, Serviços, Saúde, Serviços Hoteleiros e Hospitalares, Transportes, Sistemas de Produção e Manufatura, Extração de Recursos Naturais, Agricultura e Pecuária, Geração de Energia, e outras.

Shannon [1998] define Simulação como o processo de modelação de sistemas reais e da condução de experiências, com o propósito de perceber o comportamento de um sistema, ou de avaliar estratégias para a sua operacionalização. Os termos "modelo" e "sistema" são componentes chave da sua definição, designando o primeiro, a representação de um grupo de objetos ou ideias; e o segundo, um grupo ou coleção de elementos, inter-relacionados, que cooperam em prol de um objetivo pré-determinado.

A simulação é também um meio privilegiado para a observação de sistemas em operação, uma vez que nos permite estudar situações, mesmo que não sejamos capazes de as experimentar diretamente no sistema real [Shannon, 1998], por este ainda não existir; por ser difícil, moroso ou dispendioso; por questões de segurança, ou até mesmo de legalidade [K. Preston White & Ingalls, 2009].

A simulação destaca-se dos métodos de análise matemáticos e analíticos por serem mais fáceis de compreender (pelos decisores), e por serem mais credíveis — uma

vez que o seu comportamento foi comparado com um sistema real, ou por este exigir menos simplificações da realidade que outros modelos [Turban *et al.*, 2010].

Apesar de ser possível simular o comportamento de sistemas manualmente (pelo menos em teoria), na maioria das situações é utilizado o computador. Esta situação justifica-se uma vez que a velocidade dos computadores atuais permitem explorar todo o espetro de possibilidades; simular meses, ou até anos de funcionamento, em apenas alguns instantes de processamento e obter respostas em tempo útil [Pidd, 1992] — ao longo deste trabalho, o termo simulação referir-se-á sempre à simulação baseada/apoiada em computadores.

Nance [1995], Kelton, Sadowski, e Sadowski [2001] afirmam que a simulação tem vindo a desenvolver-se desde os anos 50, tendo desde então passado por várias gerações. Os princípios em que a simulação assenta são fáceis de compreender e consistem na modelação do sistema a analisar; que depois será codificado, dando origem a um programa de simulação; que será executado e cuja execução imitará o comportamento do sistema; dando origem a um relatório [Pidd, 1992]. — O programa deverá estar codificado de forma a facilmente ser possível manipular as variáveis que definem as restrições do sistema, para que o analista, com base na análise dos relatórios produzidos, possa testar vários cenários.

2.1.1 Vantagens e Desvantagens da Simulação

A simulação, como método de análise de sistemas, apresenta várias vantagens mas não está isenta de limitações. Chase, Jacobs, e Aquilano [2005] apresentam uma lista daquelas que são as razões, normalmente aceites, pelas quais se deverá ponderar a utilização desta técnica. As razões, agrupadas por vantagens e desvantagens, são as seguintes:

Vantagens

- A modelação leva normalmente a um melhor entendimento do sistema.
- O tempo na simulação é bastante célere; anos de operação poderão ser simulados em alguns instantes de processamento.
- A simulação não interfere com as atividades em execução, no sistema real.
- A simulação é bastante mais geral que os modelos matemáticos, podendo ser usada quando estes não se adequem.
- A simulação poderá ser utilizada como ferramenta didática para formar gestores, supervisores, engenheiros e colaboradores.
- Os modelos de simulação representam melhor a realidade, do que os matemáticos.
- A simulação pode ser usada para analisar condições transientes, algo que as análises matemáticas raramente permitem.
- Existem no mercado várias ferramentas de simulação preparadas para lidar com vários tipos de problemas.
- A simulação responde a questões do tipo "e se… ?".

Desvantagens

- Apesar do tempo e esforços despendidos na construção do modelo de simulação, não é garantido que o modelo retorne respostas viáveis.
- Uma vez que a simulação envolve a repetição de várias sequências, cuja ocorrência é definida aleatoriamente, não existe forma de provar que a performance de um modelo seja totalmente sólida. Apesar de pouco provável, um modelo aparentemente bem definido e estável, poderá rebentar.
- Dependendo do sistema, a construção do modelo de simulação pode demorar desde 1 hora até 100 anos de trabalho. A modelação de sistemas,

mais complexos, pode ser altamente dispendiosa e demorar bastante tempo.

- Por se basear na aleatoriedade, a simulação poderá ser menos exata que os modelos de análise matemática. A execução de modelos de simulação mais complexos poderá exigir bastante tempo de processamento.
- Apesar do desenvolvimento das suas técnicas, a simulação carece ainda de abordagens padronizadas. Modelos de um mesmo sistema, quando construídos por diferentes indivíduos, poderão diferir bastante.

2.1.2 Classificações de Simulação

De acordo com a taxionomia proposta por Sulistio, Yeo, e Buyya [2004] e conforme apresentado na Figura 2.1, a simulação poderá ser classificada segundo três propriedades, nomeadamente (1) a presença de tempo, (2) a base do valor tempo e o (3) comportamento.

Presença de Tempo

A presença de tempo analisa se a simulação incorpora, ou não, o fator tempo. As simulações que incluam o fator tempo dizem-se dinâmicas, enquanto aquelas que não o incluem se designam por estáticas. As simulações que não incorporam o tempo como fator são também designadas por Simulações de Monte Carlo [Banks, John S. Carson, Nelson, & Nicol, 2010].

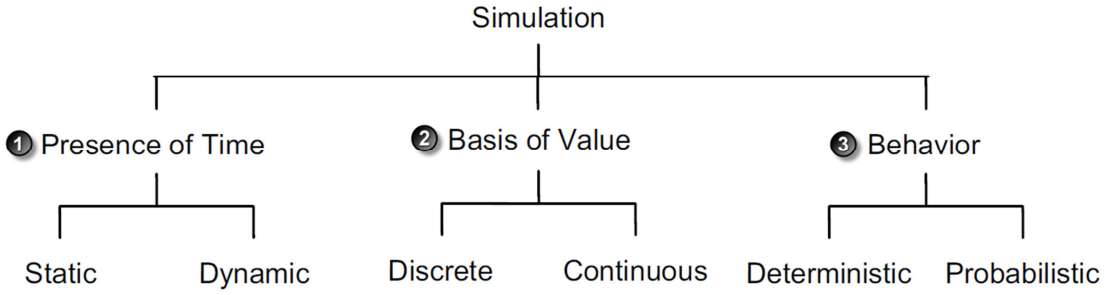


Figura 2.1: Taxinomia de Simulação, adaptado de Sulistio *et al.* [2004]

Base do Valor Tempo

A Base do Valor Tempo, presente apenas nas simulações que incorporem o tempo como fator, diz respeito à forma como o tempo avança na simulação. Na forma contínua, o tempo avança como se de um relógio se tratasse, ou seja, se é suposto a simulação demorar mil unidades de tempo, será analisada cada uma dessas unidades — mesmo que nada, ou quase nada, aconteça nesse período. Na forma discreta são os acontecimentos que definem a forma como o tempo avança, ou seja, se estiverem programados dois acontecimentos para ocorrerem respetivamente nos instante zero e quinhentos, depois do primeiro ocorrer o relógio avança imediatamente para o instante quinhentos, ignorando os instantes em que nada acontece.

Comportamento

Esta propriedade distingue a simulação pela previsibilidade do seu comportamento, ou seja, da ocorrência de acontecimentos. O comportamento poderá ser determinístico, quando a simulação, por não possuir variáveis aleatórias, ser totalmente previsível; ou estocástico, quando a simulação possui algum grau de incerteza, conhecendo-se apenas a probabilidade de determinado acontecimento ocorrer [Banks *et al.*, 2010; Pidd, 1992].

Neste trabalho iremos apenas focar-nos na simulação com uma presença de tempo dinâmica, cuja base de valor é discreta e cujo comportamento é estocástico (ou probabilístico). Este tipo de simulação é também conhecido na literatura por Simulação de Acontecimentos Discretos[1].

[1] Tradução adotada para *Discrete Event Simulation*

2.1.3 Conceitos Elementares de Simulação

Apesar de existirem vários paradigmas na simulação discreta de acontecimentos, conforme será abordado mais à frente, a estrutura básica é partilhada pela maioria das ferramentas, independentemente da complexidade da ferramenta, assim como do paradigma em que se baseiam. Os componentes estruturais das ferramentas de simulação incluem: entidades, atividades, acontecimentos, recursos, filas de espera, variáveis globais, variáveis de sistema, gerador de números aleatórios, organizador de dados estatísticos e executivo de simulação. Ao longo da descrição dos componentes serão referidas ideias de Pidd [1992], Shannon [1998], Carson [2004] e Ingalls [2008].

Entidades

As entidades são os elementos do sistema, alvo da simulação, que podem ser identificados e processados individualmente e de que são exemplo clientes de instituições, máquinas, ou qualquer outro objeto que mude de estado ao longo da simulação. As entidades possuem características próprias (atributos), que são únicas para a entidade e são essenciais para perceber o desempenho e a função das entidades envolvidas na simulação.

Do ponto de vista da sua permanência no sistema, as entidades poderão designar-se por permanentes ou temporárias, conforme permaneçam ou abandonem o sistema. Em simulação as entidades tanto podem processar, como ser processadas.

Atividades

As atividades são processos lógicos cuja função é atrasar o processamento de entidades por um determinado período de tempo (n) — constante ou gerado aleatoriamente. Quando uma atividade é executada ocorre um acontecimento (início do atraso), sendo também calculado e agendado o seu fim. O fim da atividade despoletará também um acontecimento (fim do atraso), sendo a sua

execução agendada para ocorrer n unidades de tempo depois do instante atual de simulação. O processo de atendimento de um cliente, ao balcão de uma instituição bancária, poderá ser representado por uma atividade deste tipo, em que o tempo de atraso representa a duração do atendimento ao cliente.

Fila de espera

As filas de espera são lugares onde as entidades aguardam, durante um período de tempo não especificado. As filas de espera são normalmente utilizadas quando uma entidade, para ser processada, está à espera que um determinado recurso fique disponível, ou que uma dada condição se verifique.

Instruções lógicas

As instruções lógicas permitem alterar o estado do sistema, através da manipulação das variáveis de sistema, ou através da definição de condições. Num sistema que simula o atendimento ao balcão, e em que existem dois funcionários para o mesmo serviço, este tipo de atividade permite redirecionar a entidade cliente, para um determinado funcionário aplicando para isso um critério predefinido.

Recursos

Os recursos são objetos intrínsecos do sistema, indispensáveis às atividades para o processamento das entidades e cuja capacidade é normalmente finita, representando uma restrição do sistema. Funcionários e máquinas são exemplos típicos de recursos em simulação.

Variáveis Globais

As variáveis globais são variáveis que estão acessíveis a qualquer momento, e a todo o modelo de simulação. Estas variáveis armazenarão os valores dos fenómenos que se pretendem analisar, e cuja manipulação permitirá determinar a eficácia do modelo.

Variáveis de Sistema

Além das variáveis globais deverão também existir variáveis internas, ou de sistema, que permitam controlar o estado atual do sistema. A variável que armazena a hora atual de simulação, vulgarmente designada por relógio, é exemplo de uma variável de sistema.

Gerador de Valores de Variáveis Aleatórias

Um gerador de valores de variáveis aleatórias é uma rotina computacional que permite a geração de valores aleatórios de acordo com várias distribuições estatísticas (ex.: normal, uniforme, triangular, exponencial, etc.).

Calendário

O calendário representa a lista de eventos que estão agendados para ocorrer no futuro. Em cada simulação, deverá existir apenas um calendário de acontecimentos futuros que deverá estar ordenado, ascendente, pela hora da execução agendada.

Executivo de Simulação

O executivo, motor de simulação, ou programa de controlo, é o mecanismo responsável pela gestão do calendário, mais concretamente pelo agendamento e ordenação de acontecimentos, assim como pelo avanço do relógio de simulação.

Organizador de Dados Estatísticos

O organizador de dados estatísticos monitoriza a evolução do estado dos objetos do sistema (ex.: entidades, recursos, filas de espera), para a produção de estatísticas. As estatísticas produzidas são normalmente contagens, médias simples e médias ponderadas; e são normalmente apresentadas quando a simulação termina — dito de outra forma, quando o tempo de simulação é atingido. A execução da simulação, por mais que uma vez, permitirá também calcular o desvio padrão e o intervalo de confiança.

2.1.4 Paradigmas e Ferramentas Clássicas de Simulação

De uma forma geral as simulações poderão ser executadas através de linguagens específicas de simulação[2] (SPL), linguagens de programação genéricas[3] (GPL), e através de rotinas de simulação que poderão ser incorporadas em linguagens de programação genéricas [Nance, 1995].

As linguagens específicas de simulação, por já incorporarem as rotinas e as funcionalidades necessárias para a execução de simulações são a ferramenta de eleição, sendo possível obter resultados muito mais rapidamente do que com outras ferramentas, uma vez que os mecanismos necessários já se encontram implementados.

Uma das formas utilizadas para a classificação dos SPL apoia-se no conceito de paradigma[4]. Segundo Nance [1995], este conceito surgiu pela primeira através de Lackner [1962], como forma de designar a abstração utilizada pelos analistas na modelação de sistemas. Os paradigmas que, pela sua história, são considerados clássicos, focam-se em diferentes aspetos do sistema, nomeadamente acontecimentos, processos e atividades [Carson, 2004].

Paradigma Baseado em Acontecimentos

Na modelação de sistemas segundo este paradigma, o analista deverá primeiro identificar todos os instantes que afetam o estado do sistema, definindo depois, para cada acontecimento, a lógica que descreve a consequência de cada acontecimento [Carson, 1993, 2004]. As consequências poderão ser a alteração do valor de variáveis de sistema; a execução de acontecimentos que estavam à espera

[2] Tradução adotada para *Simulation Programming Language*
[3] Tradução adotada para *General Purpose Language*
[4] Tradução adotada para *World View*

que uma determinada condição se verificasse; e o agendamento da execução de acontecimentos [Carson, 1993].

Paradigma Baseado em Processos

Este paradigma permite representar o comportamento de um sistema do ponto de vista das entidades que se movem num sistema e são alvo de processos [Carson, 1993]. Um processo é uma sequência ordenada de acontecimentos, atividades e atrasos que definem o fluxo das entidades no sistema [Carson, 1993]. A chegada do cliente à instituição bancária atrás referida representa um acontecimento; o tempo que aguardou representa um atraso; o atendimento representa uma atividade; e o funcionário representa um recurso.

Paradigma Baseado em Atividades

Este paradigma permite ao analista definir a lógica do sistema focando-se nas atividades que afetam o estado das entidades existentes no sistema [Carson, 2004]. Recorrendo a um exemplo do mesmo autor, uma das atividades que ocorre numa instituição bancária é o atendimento a clientes. Quando essa atividade termina, o funcionário verifica se existem clientes na fila para iniciar nova atividade. Se existirem, retira o primeiro cliente da fila, altera o seu estado para "ocupado" e inicia a nova atividade. Se não existirem, altera o estado para "parado", indicando que o funcionário se encontra parado.

Segundo Luís M S Dias [2005] e de acordo com o trabalho elaborado por Nance [1995], onde são apresentadas as ferramentas que marcaram a evolução da simulação no período compreendido entre 1955 e 1985, o paradigma de atividades foi o menos explorado, o que contrasta com as potencialidades desse paradigma no ensino da simulação.

De acordo com Pidd [1992, 2004] em Luís M S Dias *et al.* [2006], e com Kang e Choi [2011], o paradigma baseado no conceito de atividade é aquele que devido à

sua simplicidade, riqueza semântica e sintática, mais se adequa à aprendizagem da simulação. Segundo Paul [1993] esta abordagem era mais popular no Reino Unido — o que poderá, em parte, explicar o número diminuto de ferramentas assentes neste paradigma.

2.2 Diagramas Ciclo de Atividades

A modelação, em simulação, tenta representar os componentes de um sistema, e as suas interações, num nível de detalhe que se considere suficiente para atingir os objetivos, e responder às questões que orientam o estudo [Carson, 1993]. Esta é uma das primeiras tarefas a realizar num projeto de simulação e onde é consumida a maior parte do tempo [Luís M S Dias, 2005]. Conforme já foi referido neste documento, existem vários paradigmas nos quais a modelação se poderá basear e que poderão influenciar o formalismo de representação a adotar. Normalmente utilizam-se Fluxogramas, DCA, ou outros diagramas similares.

Apesar das ferramentas atuais de simulação virem equipadas com interfaces sofisticados de programação visual e modelação, a utilização de diagramas, mesmo que em papel, continua a ser algo comum [Luís M S Dias, Pereira, & Rodrigues, 2002]. Isto deve-se, não só ao elevado grau de sofisticação das ferramentas, que apenas permitem a sua compreensão por especialistas; como à falta de uma linguagem comum, orientada à simulação, que permita uma melhor compreensão do sistema, que apoie a construção do programa e que facilite a comunicação com o cliente [Luís M S Dias *et al.*, 2005; Zee & Vorst, 2007]. Segundo um estudo efetuado por Hlupic [2000], a facilidade de modelação é a característica mais apreciada nas ferramentas de simulação.

De entre os formalismos de representação utilizados os DCA são os diagramas que, segundo os autores Luís M S Dias *et al.* [2006] e Kang e Choi [2011], mais se adequam ao ensino de simulação, em virtude da sua simplicidade e riqueza sintática e semântica.

De acordo com Rodrigues [1996] a importância destes diagramas reside em três aspetos: (1) compreensão do sistema, (2) construção da versão programada do modelo, (3) e validação do modelo. O esforço de construção destes diagramas

obriga à compreensão detalhada do mecanismo de funcionamento do sistema, facilitando a estruturação desse conhecimento. A utilização de uma linguagem de simulação para a elaboração de um programa de simulação é mais fácil após o exercício prévio de construção de um DCA. Sendo estes diagramas uma representação pictórica do modelo, é mais fácil entender a sua lógica, comunicá-la e discuti-la. — Ao longo da explicação destes diagramas serão utilizadas ideias de Hutchinson [1975], Pidd [1992] e Rodrigues [1996].

2.2.1 Formalismos de Representação

Os formalismos, a que estes diagramas obrigam, são bastante simples e fáceis de entender. Conforme apresentado na Figura 2.2, as atividades deverão ser representadas por retângulos, e as filas de espera por círculos. As setas que cada símbolo possui, representam, respetivamente, os estados antecedentes (origem) e os consequentes (destino).

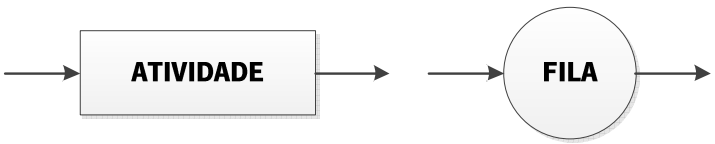


Figura 2.2: Notação gráfica dos DCA, adaptado de Paul [1993]

Para qualquer tipo de entidade, deverá ser mantida uma alternância entre atividades e filas de espera, nem que para isso seja necessário criar filas de espera fictícias. Uma fila de espera fictícia é uma fila onde o tempo de permanência de uma entidade é nulo.

2.2.2 Conceitos Básicos

Qualquer sistema pode ser percecionado como sendo constituído por entidades. As entidades, por sua vez, podem ser agrupadas em diferentes tipos, normalmente referidos como classes. Num sistema, uma entidade pode encontrar-se num estado

passivo (quando se encontra numa fila de espera), ou num estado ativo (quando envolvida numa atividade).

Para que uma determinada atividade possa ter início, é necessário que as entidades que vão estar envolvidas na atividade estejam disponíveis. Esta disponibilidade refere-se não só ao número de entidades necessárias, como também ao tipo e aos seus atributos. Reunidas as condições para que a atividade possa iniciar, é possível calcular a sua duração e o instante de tempo em que a atividade irá estar concluída. Ao contrário do que acontece com as atividades, o tempo durante o qual uma entidade permanece numa fila é desconhecido, só podendo ser determinado através de simulação.

Admitindo que a atividade Preparação, representada na Figura 2.3, exige uma entidade do tipo Máquina e uma entidade do tipo Operador, a atividade apenas se poderá iniciar quando existirem, respetivamente, nas filas Parada e Livre uma entidade do tipo Máquina e uma entidade do tipo Operador — a essas filas chamamos de Filas Antecedentes. Dito de outra forma, a Preparação apenas poderá iniciar quando existir uma Máquina Parada e um Operador Inativo.

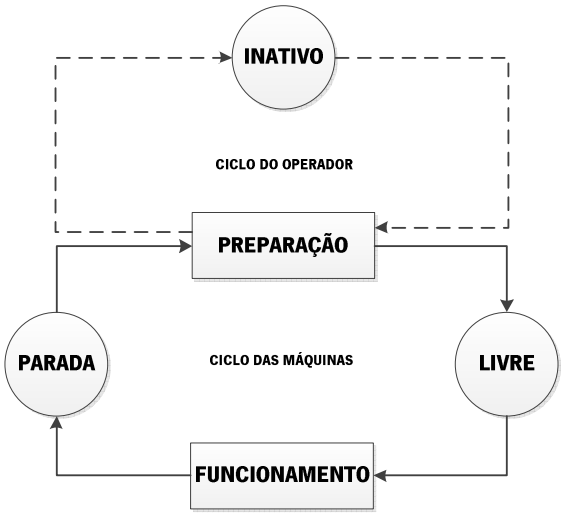


Figura 2.3: Diagrama de atividades de uma máquina semiautomática

O instante de tempo em que a Preparação irá estar concluída pode nessa altura ser calculado, somando-se, ao valor corrente do relógio, um valor característico da duração da atividade. Quando o relógio atinge o instante de conclusão da atividade, as entidades são colocadas nas Filas Consequentes. A Máquina será colocada na fila Livre e o Operador voltará à fila Inativo.

2.2.3 Processo de Construção

O processo de construção destes diagramas inicia-se com a identificação dos diferentes tipos de entidades que intervêm no sistema. Depois, para cada entidade, é representado, num diagrama, a transição de estados de cada entidade. A sobreposição de todos os diagramas produz o diagrama completo do sistema — o diagrama que está representado na Figura 2.3, corresponde ao diagrama completo do sistema que passaremos a exemplificar.

Consideremos um sistema onde intervêm dois tipos de entidades: Máquinas semiautomáticas e Operadores. Existem no sistema três Máquinas, que estão Paradas, e um Operador que está Inativo. Porque se tratam de máquinas semiautomáticas, para que estas possam entrar em Funcionamento, é necessário que o Operador faça previamente a sua Preparação — uma atividade que demora três unidades de tempo. Terminadas as dez unidades de tempo do Funcionamento da Máquina, esta fica Parada, aguardando Preparação para o próximo trabalho. Na Figura 2.4 está representado o diagrama da entidade Máquina, e na Figura 2.5 o da entidade Operador.

Conforme podemos observar no diagrama da entidade Máquina, existem duas atividades, Preparação e Funcionamento, e duas filas de espera Parada e Livre. Esta última fila é fictícia uma vez que a Máquina irá entrar em Funcionamento, mal esta fique Livre. — A introdução desta fila permite cumprir com a convenção de alternância entre atividades e filas.

Figura 2.4: Diagrama da entidade Máquina

No diagrama da entidade Operador temos apenas uma atividade, Preparação, e uma fila de espera, Inativo.

Figura 2.5: Diagrama da entidade Operador

A sobreposição dos dois diagramas irá dar origem ao diagrama completo do sistema, já apresentado na Figura 2.3. Nele é possível observar que a interação entre as entidades Máquinas e Operador ocorre na atividade Preparação. No diagrama, podemos também observar que a atividade Funcionamento, ao contrário do que acontece com a Preparação, apenas necessita de uma entidade — Máquina. Quando isto acontece dizemos que estamos perante uma *bound activity*.

2.2.4 Executivo de Simulação

Se pedirmos a um iniciado, em simulação, que escreva um programa que simule o comportamento de um sistema, onde interajam entidades, é provável que este comece por identificar as entidades, definindo depois, para cada entidade, o seu

ciclo de vida. — Se repararmos, estes coincidem com os passos necessários para a construção de um DCA. O problema surge quando este tem que decidir a forma como irá construir o programa. Existem várias formas para lidar com este problema, no entanto todas elas se baseiam na adoção de um executivo, ou motor, de simulação, que esteja de acordo com o paradigma de abstração adotado.

Independentemente do paradigma adotado, estes têm em comum o facto de produzirem um programa com uma estrutura de três níveis. No primeiro nível, insere-se a lógica do próprio executivo de simulação; no segundo, a lógica referente às interações (atividades) das entidades; e no último nível, um conjunto de operações, a que o nível anterior poderá recorrer, para a execução de operação comuns, como a geração de valores aleatórios, produção de relatórios, organização de estatísticas, entre outras. Porque os níveis 1 e 3 poderão já estar implementados, o programador terá apenas que se preocupar com a codificação das atividades (nível 2).

Regras de Execução

O executivo de simulação, baseado no paradigma de atividades, pode ser implementado, através do cumprimento de três regras básicas:

1. Examinar cada atividade, e verificar se esta pode iniciar, isto é, se as filas de espera precedentes contêm o número de entidades necessárias à atividade. Neste caso, e considerando uma simulação manual, movimentam-se as entidades para o retângulo que representa a atividade. É então calculado o instante de conclusão que também é inscrito na atividade. Após consideração de todas as atividades prossegue-se para a Regra 2.
2. Examinar os tempos de conclusão de cada atividade e selecionar o mais próximo como o novo tempo assumido pelo relógio. Prosseguir para a Regra 3,

a não ser que o valor do relógio tenha ultrapassado a duração estipulada para a simulação.

3. Examinar cada uma das atividades, terminando as que tiverem um instante de conclusão igual ao valor corrente do relógio. Movimentar as entidades dos retângulos que representam as atividades nessas condições, para as filas de espera consequentes. Prosseguir para a Regra 1.

Simulação Manual

Iniciada a simulação, com as condições iniciais apresentadas na Figura 2.6, por aplicação da Regra 1, verifica-se que a Preparação pode iniciar, uma vez que o Operador está Inativo e existem Máquinas Paradas, a aguardar Preparação. Demorando a Preparação 3 unidades de tempo, podemos imediatamente definir o seu fim para o instante 3 (instante atual + duração da atividade).

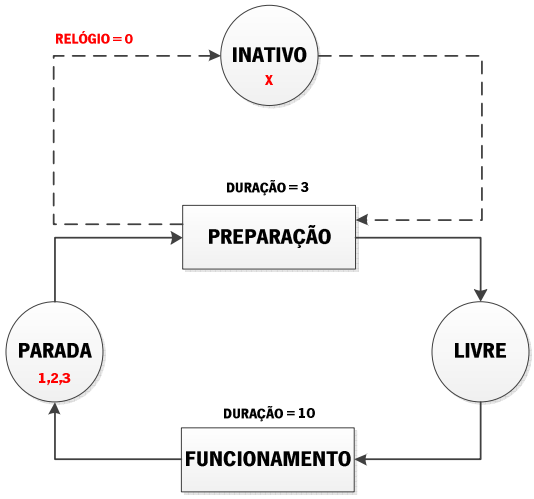


Figura 2.6: Condições iniciais da simulação manual

Desloca-se o Operador, designado pelo símbolo X, para a Preparação, anotando-se o identificador da Máquina e o instante de conclusão da atividade (Figura 2.7). A operação Funcionamento não pode iniciar, visto não existir nenhuma Máquina Livre, passa-se então à Regra 2.

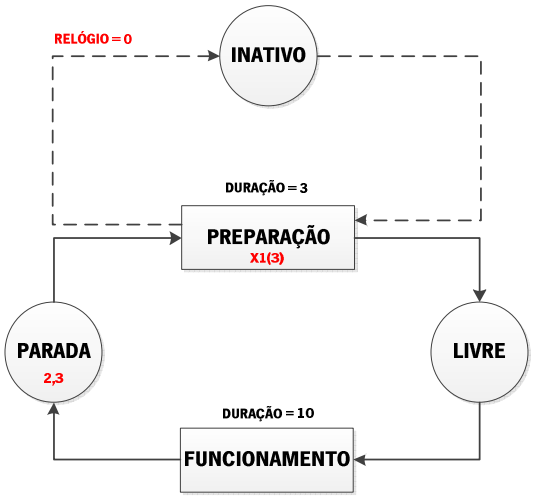


Figura 2.7: Estado da simulação no instante de tempo zero

O instante de tempo mais próximo da conclusão de alguma atividade é o instante 3. O relógio assume esse valor e passa à execução da Regra 3. Só a Preparação termina quando o relógio tem o valor 3. O Operador deve ser retirado da Preparação passando a estar Inativo, e a Máquina 1 passará a estar Livre. Voltamos à Regra 1 ainda com o relógio igual a 3 e verificamos que a atividade Preparação pode iniciar sobre a Máquina 2, e a atividade Funcionamento sobre a Máquina 1, com os instantes de conclusão de 6 e 13 respetivamente (Figura 2.8).

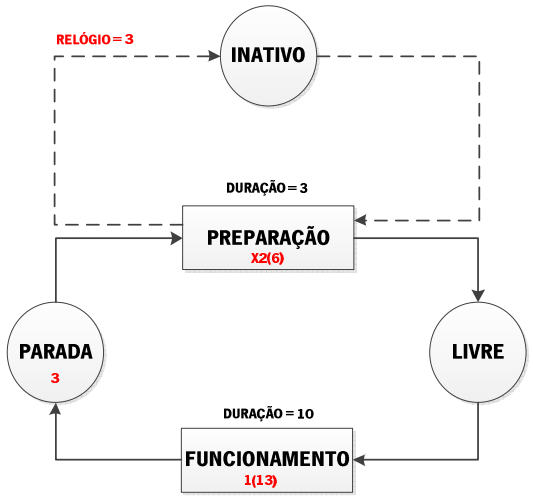


Figura 2.8: Estado da simulação no instante de tempo três

A Regra 2 leva-nos a avançar o relógio para o instante 6. A aplicação da Regra 3 termina a Preparação da Máquina 2 fazendo transitar esta Máquina para a fila Livre e libertando o Operador para a fila Inativo. Ainda com o relógio igual a 6, a Regra 1 leva-nos a iniciar a operação de Preparação sobre a Máquina 3, envolvendo o operador, com a conclusão no instante 9, e também o início da atividade de Funcionamento sobre a Máquina 2, com conclusão marcada para o instante 16 (Figura 2.9).

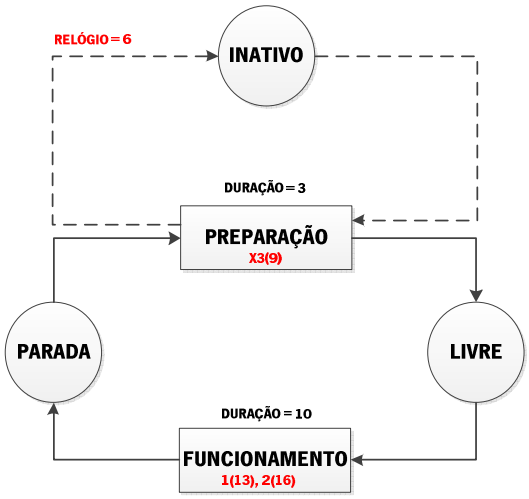


Figura 2.9: Estado da simulação no instante de tempo seis

A simulação prossegue com a aplicação sucessiva das regras até o relógio atingir o valor estipulado para o fim da simulação (Figura 2.10 e Figura 2.11).

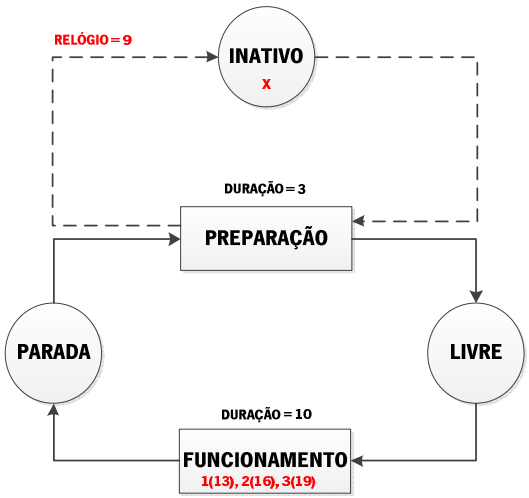


Figura 2.10: Estado da simulação no instante de tempo nove

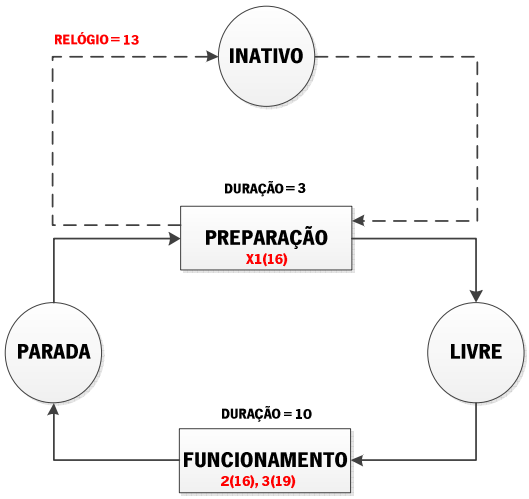


Figura 2.11: Estado da simulação no instante de tempo treze

A Tabela 2.1 apresenta o resumo da simulação até instante de tempo 29, o instante que iremos considerar como sendo o fim da simulação.

Tabela 2.1: Quadro resumo da simulação

Tempo	**Preparação**		**Funcionamento**	
	Nº Trabalho	*Instante de Fim*	*Nº Trabalho*	*Instante de Fim*
0	+P1	3		
3	-P1 +P2	6	+F1	13
6	-P2 +P3	9	+F2	16
9	-P3		+F3	19
13	+P4	16	-F1	
16	-P4 +P5	19	-F2 +F4	26
19	-P5 +P6	22	-F3 +F5	29
22	-P6		+F6	32
26	+P7	29	-F4	
29	-P7 +P8	32	-F5 +F7	39

No instante de tempo 0 tem início o primeiro trabalho de preparação (P1), cujo fim está previsto ocorrer no instante 3. No instante 3 termina o trabalho P1, é iniciado o segundo trabalho de preparação (P2) e o primeiro de funcionamento (F1) com o fim previsto, respetivamente, para os instantes 6 e 13. O símbolo "+" representa o início, enquanto o símbolo "-" representa o fim de uma atividade.

2.3 A Problemática do Ensino de Simulação

Segundo Altiok *et al.* [2001], face aos problemas de larga-escala que atualmente os engenheiros, cientistas e gestores enfrentam, a simulação tem-se tornado, possivelmente, na única abordagem prática para os analisar. O mesmo autor refere que a simulação teve um crescimento bastante acentuado nas últimas décadas, tanto ao nível dos aspetos teóricos e práticos, como ao nível dos aspetos educacionais e comerciais.

No entanto, apesar das vantagens que esta técnica apresenta, é surpreendente que ainda muitas escolas de gestão e algumas de engenharia, não tenham já adotado a simulação nos seus currículos [Ståhl, 2000]. Na opinião de autores como Born [2003] e Ståhl *et al.* [2003], a generalização do uso de simulação está longe de acontecer, assim como o seu ensino está longe de ser o mais correto.

Para que a utilização desta técnica se generalize, será necessário cativar aqueles que são considerados os potenciais interessados em simulação (alunos de áreas de engenharia e gestão que, por temerem a programação, se negam à simulação), dotando-os com os conhecimentos necessários para que estes comprendam as potencialidades e as limitações desta técnica [Ståhl, 2000] — o que não é compatível com o método de ensino geralmente adoptado, que se foca geralmente apenas nos aspetos teóricos, ou apenas nos práticos [Schriber & Brunner, 2008].

O reconhecimento das potencialidades da utilização desta técnica, na resolução de problemas complexos, cedo fez agitar a comunidade académica, sendo possível encontrar literatura sobre esta problemática a partir dos anos 70 [Nance, 2000].

De uma forma geral a literatura que aborda a problemática do ensino de simulação pode ser decomposta em três partes, de acordo com a predominância do seu foco. (1) Na primeira parte inclui-se a literatura que discute as competências que os profissionais desta área deverão possuir. (2) Na segunda, inclui-se a

literatura que versa sobre o balanceamento ideal entre as abordagens de ensino teóricas e práticas. (3) Incluindo-se na terceira parte, a literatura que discute a construção de um curriculo próprio de simulação. — Sendo o propósito deste trabalho a construção de uma ferramenta para o ensino de simulação, apenas será explorada a literatura referente ao balanceamento entre as abordagens.

2.3.1 Balanceamento entre Abordagens de Ensino

A academia depara-se atualmente com o difícil problema que é decidir quais os tópicos a focar no ensino de simulação. O principal problema prende-se com a escolha dos tópicos a focar numa disciplina semestral, tendo em consideração não só as competências com que se pretende dotar os alunos, como aquelas que estes já possuem em virtude dos seus diferentes percursos [Altiok *et al.*, 2001].

Ståhl [2000] agrupa os principais interessados em três grupos: (1) alunos de áreas de informática, que possuem geralmente um vasto conhecimento em programação e conhecimentos consideráveis de estatística; (2) alunos de outras áreas de engenharia, que possuem conhecimentos razoáveis de estatística e não sendo utilizadores experientes, podem ter frequentado disciplinas de programação; (3) alunos de áreas de gestão, que geralmente não possuem conhecimentos de programação e os conhecimentos de estatística são limitados. De acordo com Lorenz e Schriber [1996] pode ser considerado um outro grupo, onde serão incluídos os alunos oriundos do mercado de trabalho.

No ensino da simulação existem fundamentalmente duas abordagens: uma enfatiza os aspetos teóricos, e a outra, os aspetos práticos. Os tópicos normalmente abordados no ensino desta disciplina são a geração de valores de variáveis aleatórias, a análise de variáveis e resultados, os paradigmas de modelação e a verificação e validação de modelos — de seguida, de acordo com as

ideias defendidas por Altiok *et al.* [2001], será caracterizada cada uma das abordagens.

Na abordagem teórica esses tópicos podem ser lecionados com alguma profundidade, sem que para isso seja necessário recorrer a alguma ferramenta de simulação, ou de outro género. Esta abordagem ensina aos alunos a metodologia correta a empregar numa simulação, dotando-os também com alguns conhecimentos teóricos. No entanto, não os dota com os conhecimentos necessários para simular sistemas complexos, nem para lidar com situações de pressão, fruto de *deadlines* apertadas para a entrega de trabalhos — o que se considera ser uma exigência do mercado. Além do mais, os conhecimentos de modelação adquiridos poderão possuir algumas lacunas, levando-os a produzir modelos inválidos, ou demasiado simplistas.

Na abordagem prática, os fundamentos teóricos são brevemente focados, sendo a maior parte do tempo despendida em laboratório onde, com a ajuda de ferramentas de simulação (tipicamente comerciais), são lecionados e exercitados os detalhes de modelação. Nesta abordagem os alunos adquirem um vasto conhecimento da utilização destas ferramentas, assim como da modelação de situações complexas — uma vez que nesta abordagem existe normalmente a entrega de trabalhos práticos, os alunos poderão sentir alguma pressão, fruto de prazos de entrega apertadas. Nance [2000] e Ståhl *et al.* [2003] sugerem alguma prudência na utilização desta abordagem, pois uma demasiada atenção no modelo, ou na ferramenta, poderá fomentar algumas lacunas ao nível da validação e verificação do modelo, assim como da interpretação dos resultados.

Pelo que se pôde apurar na literatura, não existe uma fórmula mágica para o balanceamento destas duas abordagens. Existe no entanto o relato de experiências, que se revelaram de sucesso, e o reconhecimento que os fundamentos

teóricos são necessários não só para a modelação de sistemas complexos, como para garantir que os futuros programadores destas ferramentas conhecem os princípios básicos em que estas assentam.

Apesar de aqui não ter sido explorado o perfil ideal do profissional de simulação, é referido na literatura que este deverá possuir conhecimentos de programação, não só para que este possa criar uma primeira versão de um programa de simulação (justificando assim aos gestores de topo o investimento num projeto de maior envergadura), como para que em situações mais urgentes possa alterar programas já existentes [Ståhl *et al.*, 2003].

Porque se considera que as ferramentas comerciais existentes não são adequadas para o ensino desta disciplina, seja pelo seu custo, complexidade de utilização ou por não evidenciarem os princípios básicos desta disciplina, é possível também encontrar na literatura algumas discussões sobre as características que as ferramentas para o ensino de simulação deverão possuir.

2.3.2 Ferramentas para o Ensino de Simulação

Ao contrário do que acontece com as ferramentas comerciais, onde existe alguma tradição na publicação de *rankings* (dos quais Luís M. S. Dias, Pereira, Vik, e Oliveira [2011] são intervenientes) é difícil encontrar na literatura uma lista de ferramentas, didáticas, para o ensino da simulação. Pelo que se pôde apurar, com a análise do discurso daquelas que são as vozes ativas neste domínio, quando se recorre a ferramentas específicas de simulação, essas são comerciais. As ferramentas comerciais, por se focarem demasiado na resolução dos problemas do mercado, não são adequadas para o ensino.

Na análise da literatura apenas foi possível encontrar duas ferramentas, atuais, que se consideram didáticas, a *General Purpose Simulation System* (Ståhl, Henriksen, Born, e Herper [2011]) e o *Visio Basic for Simulations* (Pereira, Dias,

e Ferreira [2009]; Pereira, Dias, e Rocha [2009]). É de salientar que nenhuma dessas ferramentas se baseia no paradigma de atividades — o que não será de estranhar, se considerarmos o número diminuto de ferramentas desse tipo. A primeira baseia-se no paradigma de processos e a segunda no paradigma de acontecimentos.

Uma ferramenta que, apesar de se encontrar atualmente descontinuada, merece algum destaque é a *Computer Aided Programing of Simulations* (CAPS). De acordo com Kang e Choi [2011], esta foi a primeira ferramenta cuja abordagem pretendia facilitar a escrita de programas de simulação. Esta ferramenta, através de um diálogo interativo com o utilizador, pretendia capturar o modelo de um sistema, representado através de DCAs, gerando depois o programa de simulação correspondente em *Extended Control and Simulation Language* (ECSL) [Clementson, 1986].

Conforme apresentado em Clementson [1986] e referido por Luís M S Dias [2005], o utilizador identifica quais as entidades envolvidas no sistema a simular, descreve sequencialmente o nome dos estados (passivos e ativos) de cada uma das dessas entidades, fornece as expressões das distribuições que determinam a duração de cada atividade e fornece informação sobre a lógica do sistema [Luís M S Dias, 2005].

2.3.3 Requisitos desejados para as Ferramentas de Ensino

Uma das vozes ativas na discussão da problemática do ensino de simulação é Ingolf Ståhl, um académico com mais de 30 anos de experiência, que em 2007 já tinha lecionado simulação a mais de 7000 alunos [Ståhl, 2007], tendo também liderado o desenvolvimento do Micro-GPSS — uma ferramenta para o ensino de simulação [Herper & Ståhl, 1999].

Ståhl foi também autor de um levantamento exaustivo onde são referidos as características que as ferramentas, para o ensino de simulação, deverão apresentar [Ståhl, 2000]. Esse levantamento encontra-se, agrupado em oito categorias, no Anexo 1

3. Desenvolvimento do Protótipo

3.1 Estratégia e Considerações Iniciais

Na busca da resposta à questão de investigação, o objetivo principal deste trabalho consistiu no desenvolvimento do protótipo de uma ferramenta para o ensino, introdutório, de simulação. De acordo com as funcionalidades pretendidas, decidiu-se estruturar os esforços de desenvolvimento em três partes: (1) edição gráfica, (2) geração automática, e (3) execução do programa de simulação.

Na primeira parte foram englobados os esforços relativos à construção da estrutura para a representação de diagramas, no formalismo DCA. Na segunda parte, os esforços para a validação dos diagramas e para a geração automática do diagrama global e do programa de simulação correspondente. E na terceira parte,

os esforços referentes à depuração e à execução do programa gerado, ou alterado pelo utilizador.

Uma vez que o protótipo pretende determinar a exequibilidade da construção de uma ferramenta com as características especificadas no Capítulo 1, para mitigar o risco da sua não exequibilidade, e porque o tempo disponibilizado para a construção do protótipo era apenas de 3 meses, optou-se por utilizar ferramentas de desenvolvimento rápido e tecnologias de fácil integração.

A facilidade de obtenção e a familiarização do autor com as ferramentas da Microsoft favoreceram a utilização das ferramentas e tecnologias deste fabricante, tendo colocado as ferramentas *open-source* em segundo plano.

Dado que uma das funcionalidades pretendidas consistia em permitir a construção de modelos de simulação no formalismo DCA, e dado que a Microsoft possui no seu catálogo de produtos uma ferramenta de modelação, uma das decisões que teve de se tomar, foi se iríamos desenvolver de raiz um editor gráfico, ou se iriamos integrar um já existente. Qualquer uma das opções possuía vantagens e inconvenientes.

O desenvolvimento de raiz garantia um maior controlo e independência do que a integração, no entanto iria exigir um maior esforço de implementação que se iria refletir no tempo de execução do projeto. Das opções disponíveis, pelas razões referidas no início deste capítulo, optou-se por conceber o protótipo em género de *add-on* para que este pudesse ser integrado no Visio 2010, a ferramenta de modelação da Microsoft.

Uma outra opção possível seria a integração do protótipo numa ferramenta comercial de simulação, como por exemplo o Arena da Rockwell. No entanto,

porque se considera que as ferramentas comerciais são demasiado opacas para o ensino introdutório de simulação, essa opção foi desde cedo descartada.

Refira-se também que a implementação do protótipo foi realizada na linguagem Visual Basic .Net (VB), através do Microsoft Visual Studio 2010 (Figura 3.1).

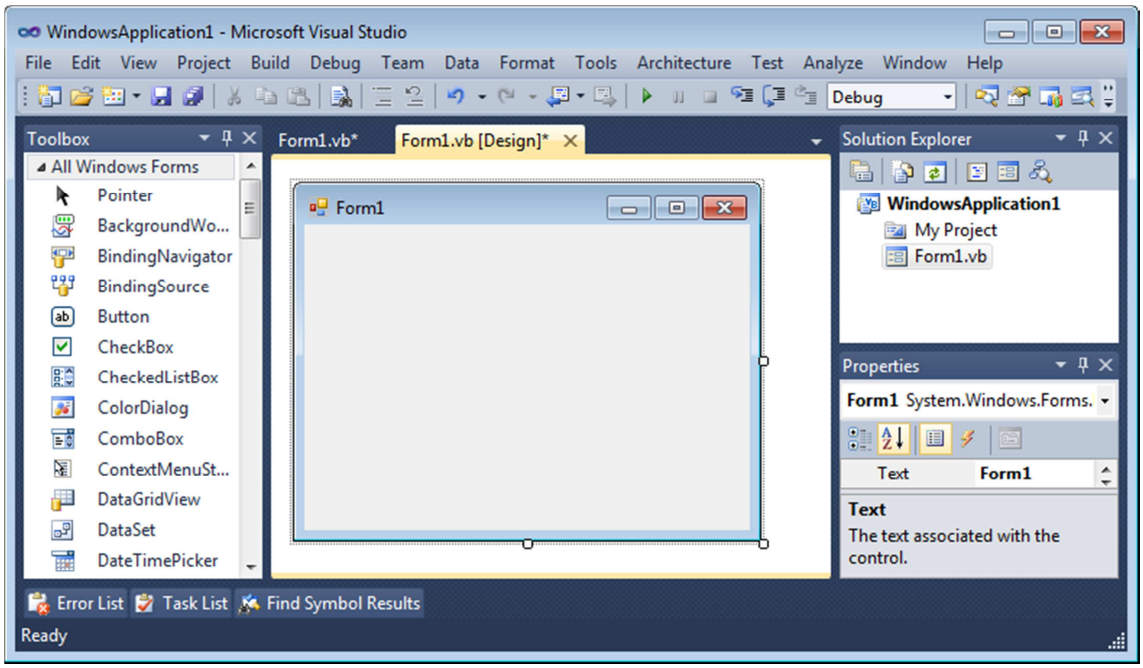


Figura 3.1: Microsoft Visual Studio 2010

3.2 Edição Gráfica

Como se optou pela integração do protótipo, no Visio, não foi necessário lidar com a complexidade inerente à construção, de raiz, de um novo editor gráfico para a representação de modelos. No entanto, foi necessário perceber o funcionamento do Visio, a estrutura dos seus documentos, as primitivas disponibilizadas para a sua manipulação e a forma de construção de novos formalismos. Compreendidos estes aspetos, foi necessário pensar nos mecanismos que iriam ser adicionados ao Visio para que este passasse a incorporar as funcionalidades pretendidas para o protótipo.

Funcionamento do Visio

Conforme se poderá observar na Figura 3.2, os documentos do Visio são organizados em páginas (1), cada página representa um diagrama diferente (2), mas do mesmo género. Um documento poderá ter um ou mais *stencils* (3), sendo que os *stencils* estão associados ao género do diagrama que se pretende construir. As *shapes* (4), que compõem os *stencils*, apenas podem ser arrastadas para os diagramas, ou seja, para qualquer uma das páginas do documento.

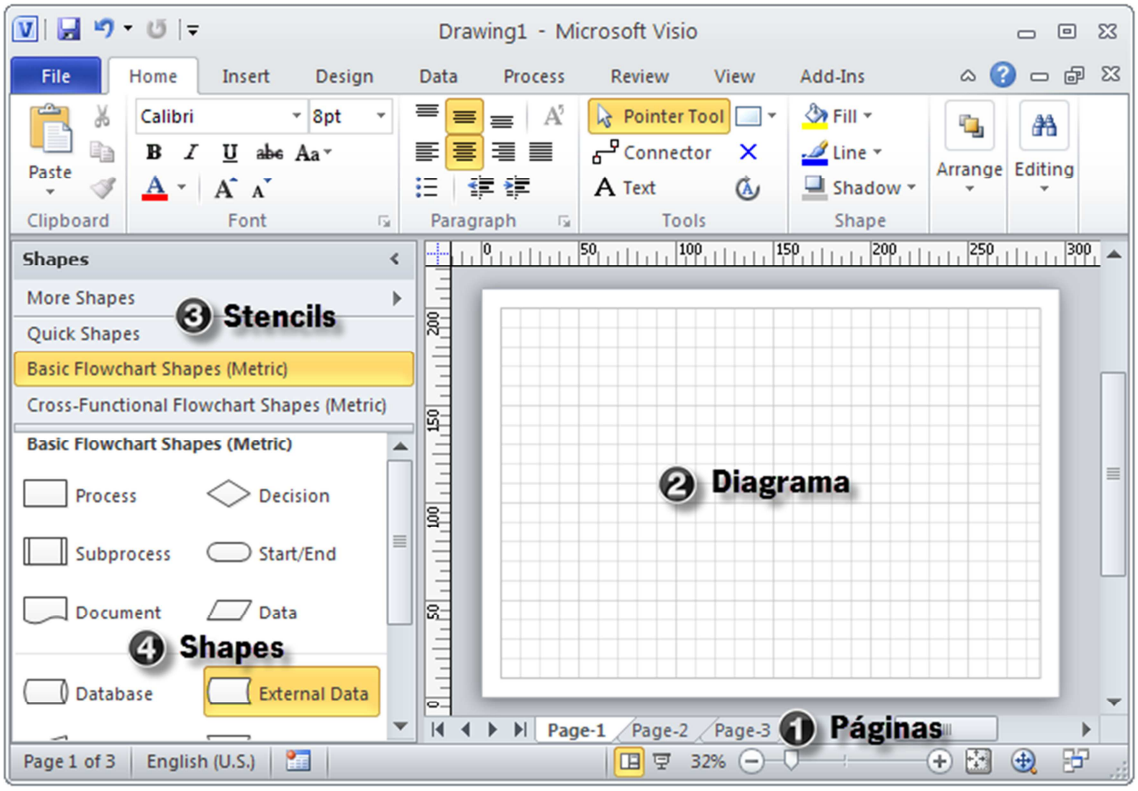


Figura 3.2: Janela principal do Visio 2010

Depois de instalado, o Visio surge equipado com vários *stencils*, que se encontram agrupados em categorias. No entanto, é possível criar novos *stencils* e novas *shapes*.

Manipulação do Visio

Para a manipulação, programática, dos documentos do Visio, a Microsoft coloca à disposição dos programadores informáticos o Visio 2010 *Software Development Kit*

(SDK). Este SDK é composto por bibliotecas computacionais que, quando referenciadas no Visual Studio, permitem a manipulação de todos os objetos que compõem o Visio. Além das bibliotecas computacionais, o SDK inclui alguns exemplos de programação, e uma vasta documentação onde são descritas todas as primitivas colocadas à disposição para a manipulação do Visio.

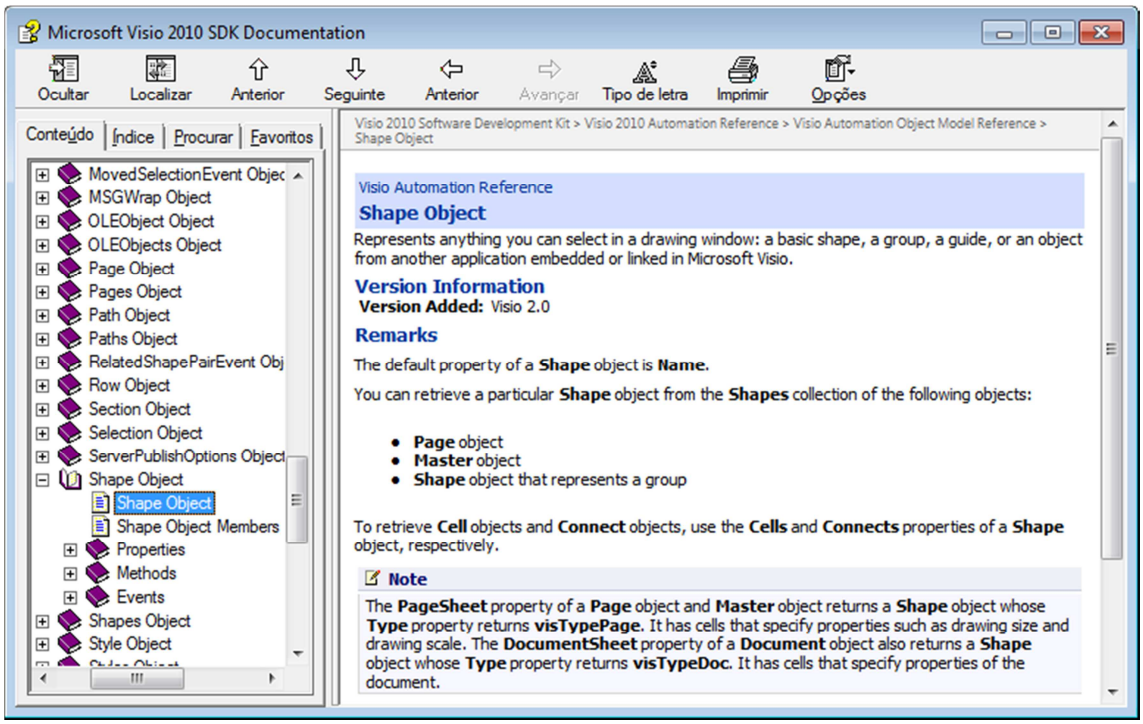


Figura 3.3: Documentação do SDK

Estrutura interna do Visio

Apesar de internamente representarem classes de objetos diferentes, documentos, páginas e *shapes* têm em comum um conjunto de características herdadas da classe *shape*. Essas características podem ser acedidas através dos atributos *documentSheet*, *pageSheet*, ou *shapeSheet* dos objetos, sendo que cada atributo contém, respetivamente, informação relativa ao documento, página, ou *shape*. Em cada um dos atributos referidos é possível armazenar estruturas de dados próprias.

Decisões de implementação

De acordo com os objetivos do protótipo e face à estrutura e ao funcionamento do Visio, além da construção do *stencil* e das *shapes* próprias do formalismo DCA, decidiu-se que:

1. As entidades seriam representadas em páginas diferentes;
2. As entidades fictícias, que modelam o mecanismo "porta" seriam também representadas em páginas diferentes, devendo o seu nome obedecer ao formato *x_door*, em que *x* designaria o nome da entidade que iria utilizar a porta;
3. Uma das páginas do documento não iria representar uma entidade, mas sim o diagrama global. Esse diagrama seria gerado, de forma automática, pela ferramenta, caso os diagramas individuais não apresentassem erros de construção. Os erros de construção deveriam ser assinalados;
4. Os dados referentes às entidades (ex.: variáveis locais) seriam armazenados no *pageSheet*, e os dados referentes a toda a simulação (ex.: variáveis globais), no *documentSheet*. Uma vez que não existe forma de aceder diretamente a esses atributos, deveria ser criado um interface gráfico para que o utilizador pudesse interagir com as propriedades da entidade e da simulação;
5. Para agilizar o processo de construção dos diagramas, seria construído um "assistente" que com base no preenchimento de alguns campos, deveria criar as páginas que representam as entidades.

Stencil e *Shapes*

Uma vez que os DCA ainda não fazem parte das categorias de diagramas que acompanham o Visio, para que fosse possível criar diagramas nesse formalismo, foi necessário construir um novo *stencil*, onde foram representadas as *shapes* que compõem o formalismo (Figura 3.4). Uma vez que o Visio permite ligar todas as *shapes*, através do seu ligador predefinido, o *dynamic connector*, não foi necessário adicionar ao *stencil* uma *shape* para representar o conector.

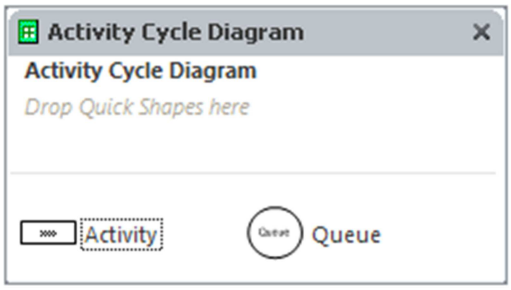


Figura 3.4: *Stencil* do formalismo DCA

Uma vez que as *shapes* designam objetos com características distintas, tendo em vista a geração do programa de simulação, foram adicionados vários atributos a cada uma das *shapes* do *stencil* (Tabela 3.1).

Tabela 3.1: Descrição dos atributos que compõem as *shapes*

Atributo	*Shape*	Descrição
Assignment	*Activity*	Instrução, do tipo atributivo, a executar aquando do fim da atividade.
Duration	*Activity*	Define a duração, característica, da atividade.
Condition	*Dynamic Connector*	Quando definida, informa que o caminho indicado pelo ligador, apenas deverá ser seguido quando a condição definida se verificar.
Name	*Activity, Queue*	Nome utilizado para identificar cada uma das atividades, ou filas.
Priority	*Activity*	Quando no mesmo instante de tempo se puder iniciar mais que uma atividade, este atributo define aquela que deve iniciar primeiro. Quanto mais pequeno o valor, maior será a prioridade.
PriorityField	*Queue*	Nome do atributo utilizado, nas filas do tipo *Priority*, para determinar a entidade a ser removida.
Quantity	*Queue, Dynamic Connector*	Quantidade inicial de entidades na fila; Número de entidades a serem removidas após a conclusão da atividade.
Type	*Queue*	Filosofia utilizada na remoção de entidades.

Métodos auxiliares para a manipulação do Visio

A manipulação do Visio foi sem dúvida a atividade que mais esforço de desenvolvimento exigiu. Isto deveu-se não à falta de documentação, pois ela

existia e era suficiente, mas sim ao facto dos eventos disponibilizados pelo SDK não serem suficientes para capturar toda a interação do utilizador com o Visio. Essa insuficiência obrigou à criação de um conjunto de métodos e eventos que se encontram implementados na classe *VisioEventsHandler* conforme se poderá observar na Figura 3.5 e na Figura 3.6.

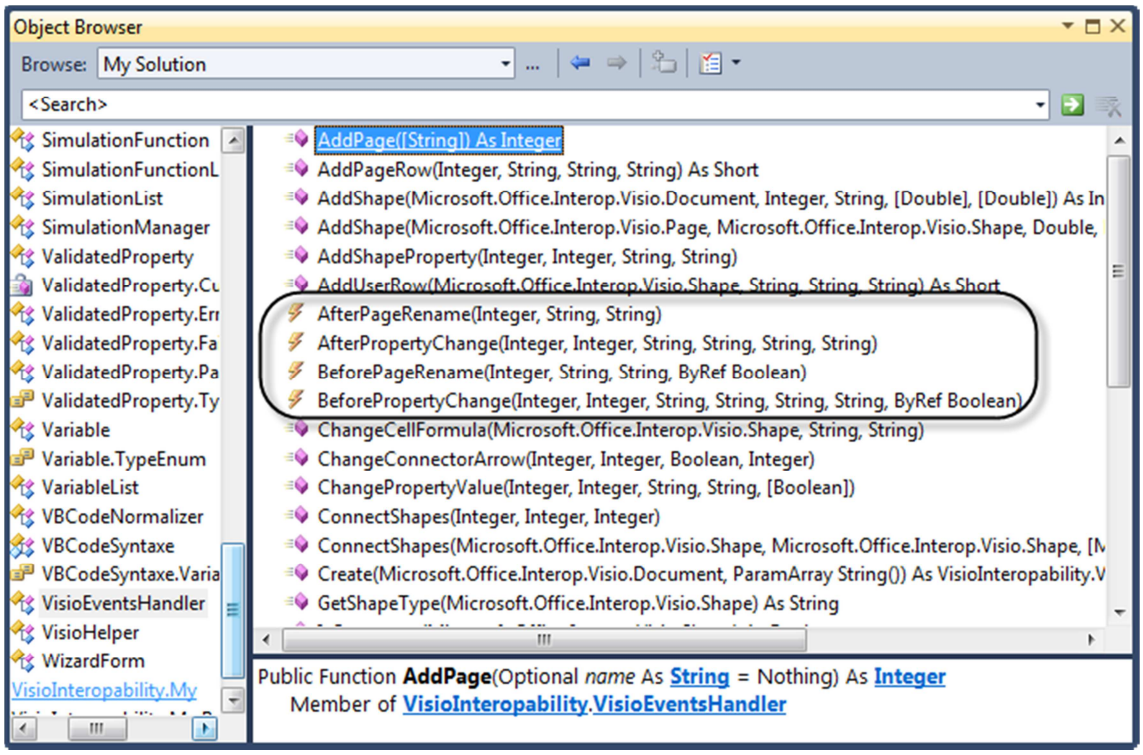


Figura 3.5: Métodos e eventos criados para capturar a interação do utilizador

```
Private Sub Application_CellChanged(Cell As Microsoft.Office.Interop.Visio.Cell)
  Handles Application.CellChanged
  If IsWatchedDocument(Cell.Document)
  AndAlso Not Application.IsInScope(scopeID)
  AndAlso Cell.Section = Visio.VisSectionIndices.visSectionProp
  AndAlso Cell.Column = 0 AndAlso Cell.Shape.MasterShape IsNot Nothing
  AndAlso IsOfWatchedType(Cell.Shape)
  AndAlso ShapesList.Contains(Cell.Shape.ID) Then
     Dim shape As Visio.Shape = Cell.Shape
     Dim cancel As Boolean = False
     Dim newValue As String = Cell.ResultStr("")
     Dim oldValue As String = Nothing
     If Not PropertiesList.ContainsKey(Cell.RowName) Then
        PropertiesList.Add(Cell.RowName, "")
     End If
     oldValue = PropertiesList(Cell.RowName)
     If newValue <> oldValue Then
        RaiseEvent BeforePropertyChange(shape.ContainingPageID, shape.ID,
          GetShapeType(shape), Cell.RowName, oldValue, newValue, cancel)
        If cancel Then
```

```
            ChangePropertyValue(shape.ContainingPageID, shape.ID, Cell.RowName,
              oldValue)
          Else
            PropertiesList(Cell.RowName) = newValue
            RaiseEvent AfterPropertyChange(shape.ContainingPageID, shape.ID,
              GetShapeType(shape), Cell.RowName, oldValue, newValue)
          End If
        End If
    End If
End Sub
```

Figura 3.6: Método criado para capturar os eventos *After* e *BeforePropertyChange*

Para agilizar a construção do protótipo foi também criada uma outra estrutura que permite, através da simples invocação dos métodos implementados, efetuar operações que de outra forma exigiriam várias linhas de código. Essa estrutura, designada por *VisioHelper*, foi implementada de forma genérica e independente, para poder ser utilizada noutros projetos que também recorram ao Visio, e encontra-se apresentada na Figura 3.7.

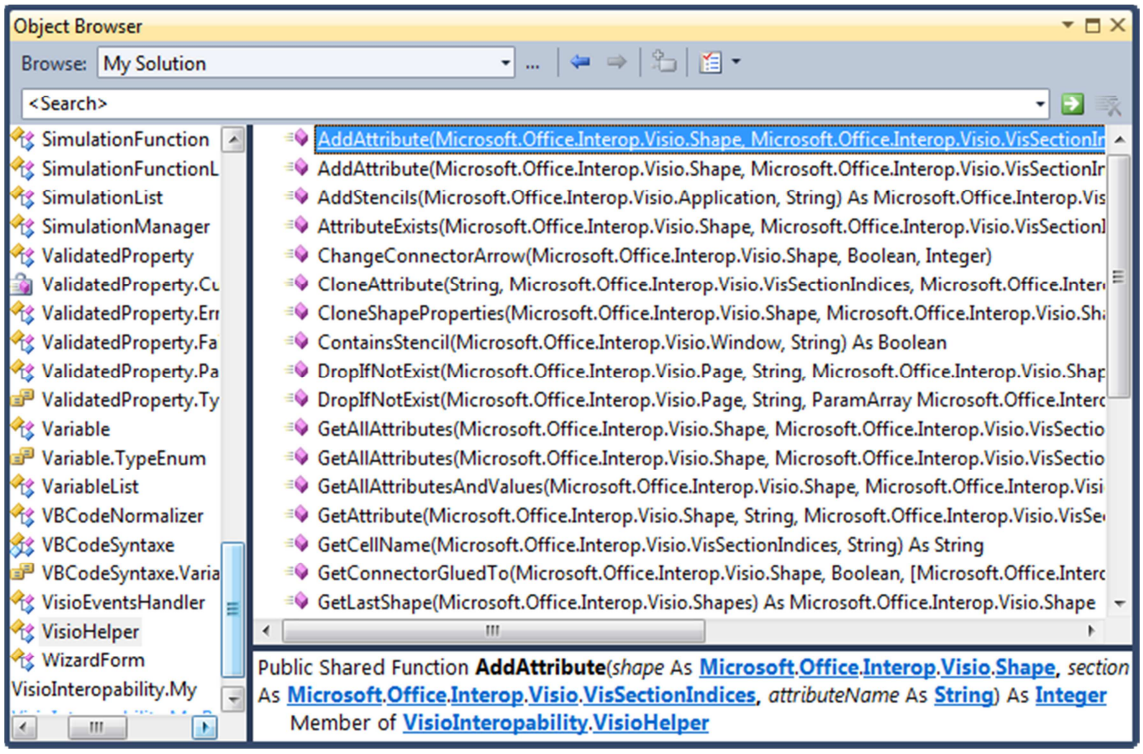


Figura 3.7: Métodos criados para agilizar a construção do protótipo

3.3 Geração Automática

Nesta secção englobam-se os esforços relativos à validação dos diagramas que compõem a simulação; os esforços relativos à geração automática do diagrama global, onde se encontra representado o ciclo de todas as entidades; e os esforços relativos à geração do programa de simulação, e que consiste na tradução do diagrama global da simulação para a linguagem de programação adotada, que neste caso é o Visual Basic for Applications (VBA).

Validação dos diagramas da simulação

Como pudemos já verificar no Capítulo 2, a construção dos DCA deverá obedecer a algumas regras que deverão ser satisfeitas para que se possa proceder à construção do diagrama global, que representa o ciclo de todas as entidades que compõem a simulação. Para garantir que os diagramas, que representam o ciclo das entidades, estão bem construídos foi criado um mecanismo, interativo, que informa o utilizador dos erros que cada diagrama possui (Figura 3.8).

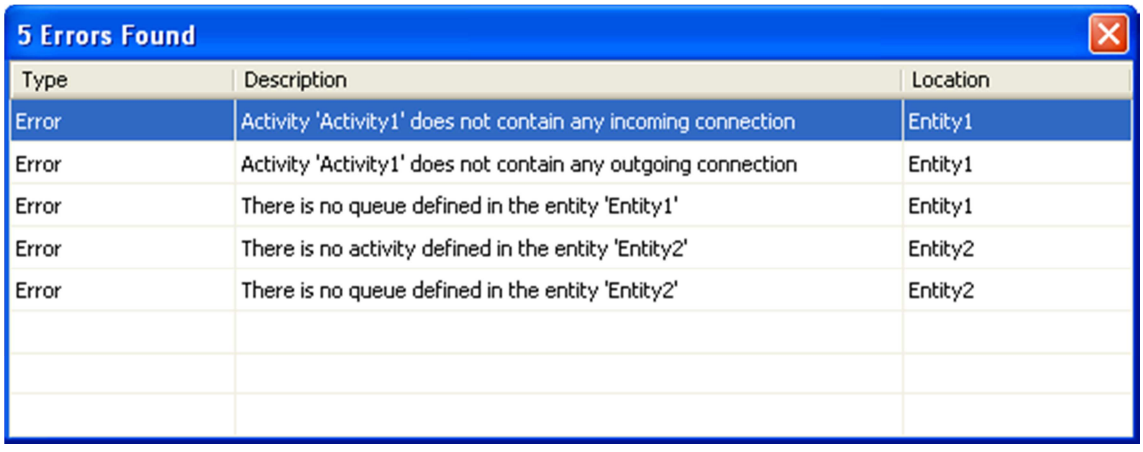


Figura 3.8: Interface gráfico do mecanismo de validação de erros dos diagramas

Uma vez que o mecanismo implementado informa o utilizador dos erros encontrados em cada diagrama, além de garantir que estão reunidos os pressupostos para a geração do diagrama global, este permite que os utilizadores

menos familiarizados com estes diagramas possam compreender melhor as regras de construção deste tipo de diagramas. Na Figura 3.9 está apresentado um dos procedimentos utilizados para a validação dos diagramas das entidades.

```
Private Sub CheckActivities(entity As Entity)
    If entity.Activities.Count > 0 Then
        For Each a As Activity In entity.Activities
            Dim incoming As ConnectedItem() = ConnectedItem.GetConnectedTo(a,
ConnectedItem.DirectionEnum.Incoming)
            Dim outgoing As ConnectedItem() = ConnectedItem.GetConnectedTo(a,
ConnectedItem.DirectionEnum.Outgoing)

            If incoming IsNot Nothing Then
                If incoming.Length > 1 Then
                    AddError(String.Format("Activity '{0}' contains {1} incoming
connections instead of 1", a.Name.Value, incoming.Length), entity)
                ElseIf Not TypeOf incoming(0).Item Is Queue Then
                    AddError(String.Format("Activity '{0}' is incoming connected
to an activity instead a queue", a.Name.Value), entity)
                End If
            Else
                AddError(String.Format("Activity '{0}' does not contain any
incoming connection", a.Name.Value), entity)
            End If

            If outgoing IsNot Nothing Then
                For Each c As ConnectedItem In outgoing
                    If Not TypeOf c.Item Is Queue Then
                        AddError(String.Format("Activity '{0}' is outgoing
connected to an activity instead a queue", a.Name.Value), entity)
                    End If
                Next
            Else
                AddError(String.Format("Activity '{0}' does not contain any
outgoing connection", a.Name.Value), entity)
            End If

        Next
    Else
        AddError(String.Format("There is no activity defined in the entity
'{0}'", entity.Name), entity)
    End If
End Sub
```

Figura 3.9: Exemplo de um procedimento utilizado na validação dos diagramas

Geração do diagrama global

O diagrama global é um elemento chave de todo o sistema (Figura 3.10). A sua análise permite, não só gerar o programa de simulação correspondente, como também perceber quais as entidades que participam em cada uma das atividades

que compõem a simulação. Uma vez que os diagramas individuais apenas se referem às entidades que modelam, a sua interpretação não é, por si só, suficiente para perceber a forma como estas cooperam.

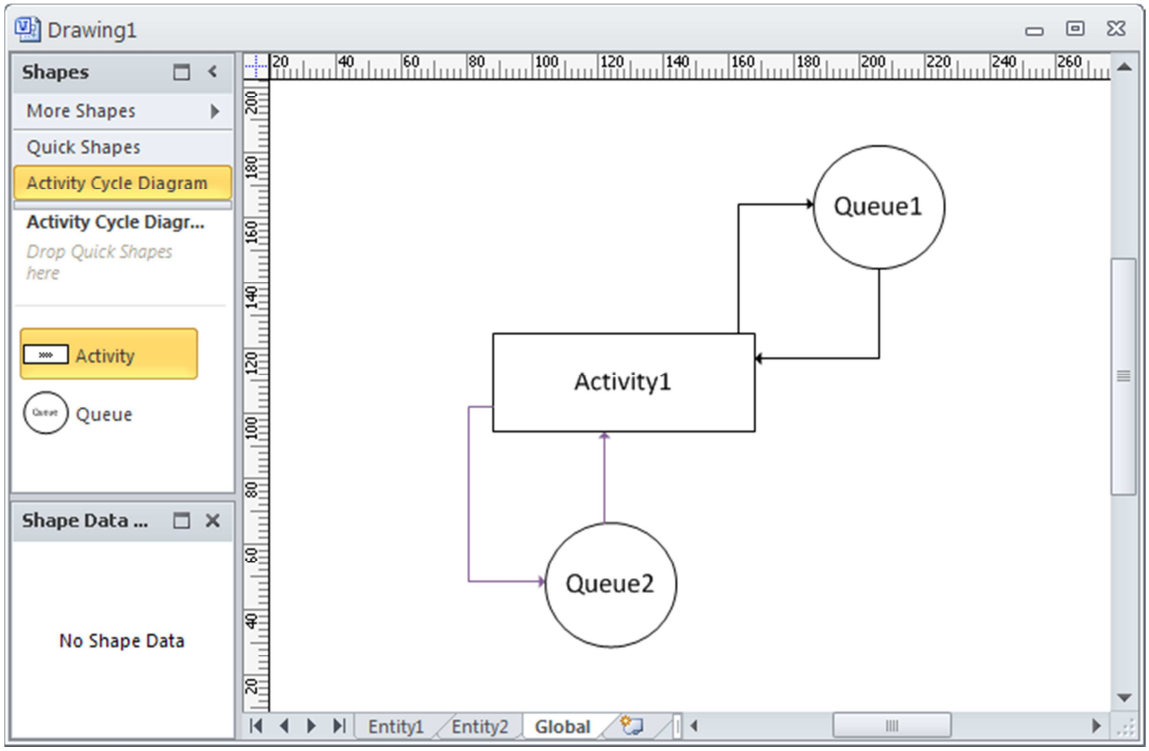


Figura 3.10: Exemplo de um diagrama global

Conforme já foi referido e face à importância do diagrama global para a simulação, a sua construção é antecedida de uma análise, onde se procura a existência de erros nos diagramas das entidades. No caso de existirem erros, a construção do diagrama global é cancelada, sendo os erros existentes apresentados conforme já se apresentou na Figura 3.8.

O mecanismo implementado, para a geração do diagrama global (Figura 3.11), começa por verificar se já existe um diagrama global no sistema, se existir esse será eliminado. Depois, são processados os diagramas das entidades, pela ordem que ocupam no documento. Verifica-se se as atividades da entidade em processamento já se encontram no diagrama global, caso não se encontrem estas

são adicionadas ao diagrama. De seguida, são adicionadas as filas de espera que estão ligadas à atividade.

Na ligação das filas às atividades, é utilizado um ligador de cor diferenciada, que permitirá identificar a entidade que intervém na atividade — a cor utilizada é determinada aleatoriamente.

As atividades e as filas de espera irão ocupar, no diagrama global, as posições que ocupavam nos diagramas que as originaram pelo que, se o diagrama gerado for confuso, deverá reposicionar as *shapes* nos diagramas das entidades e não no diagrama global.

```
Private Sub Generate()
    Dim globalQueue As Visio.Shape = Nothing
    Dim globalActivity As Visio.Shape = Nothing
    Dim globalConnector As Visio.Shape = Nothing
    Dim randomColor As Drawing.Color = Nothing

    If _simulation.Entities.Count = 0 Then
        Exit Sub
    End If

    Dim diagramPage As Visio.Page = Simulation.GetGlobalDiagramPage()
    If diagramPage IsNot Nothing Then
        _simulation.Handler.VisioHandler.RemovePage(diagramPage.ID)
    End If

    Dim pageId As Integer =
_simulation.Handler.VisioHandler.AddPage(Constants.GLOBAL_DIAGRAM_NAME)
    _simulation.Handler.VisioHandler.AddPageRow(pageId, Constants.TYPE_FIELD,
VisioPageDocType.GlobalDiagram, "")
    _globalDiagramPage = _simulation.Document.Pages.ItemFromID(pageId)
    _activityDependencies = New Dictionary(Of String, List(Of
ConnectedItem))())(StringComparison.CurrentCultureIgnoreCase)

    For Each entity As Entity In _simulation.Entities
        randomColor = GenerateRandomColor()
        If entity.Activities.Count = 0 Then
            Continue For
        End If

        For Each activity As Activity In entity.Activities
            If Not _activityDependencies.ContainsKey(activity.Name.Value) Then
                _activityDependencies.Add(activity.Name.Value, {New List(Of
ConnectedItem), New List(Of ConnectedItem)})
            End If

            globalActivity = GetShapeFromGlobalDiagram(activity)
            If globalActivity Is Nothing Then
                Dim assignText As String = String.Format("=""{0}""",
GetActivityOnExecuteAssign(activity, False))
```

```
                globalActivity = CreateShapeInGlobalDiagram(activity.Shape)
                Simulation.Handler.VisioHandler.ChangePropertyValue(pageId,
globalActivity.ID, activity.ON_EXECUTE_PROPERTY, assignText, False)
            End If

        Dim dependencies As ConnectedItem() = activity.GetConnectedTo()
        If dependencies Is Nothing Then
            Continue For
        End If

        For Each c As ConnectedItem In dependencies
            globalQueue = GetShapeFromGlobalDiagram(c.Item)
            If globalQueue Is Nothing Then
                globalQueue = CreateShapeInGlobalDiagram(c.Item.Shape)
            End If

            If c.Direction = ConnectedItem.DirectionEnum.Incoming Then
                globalQueue.AutoConnect(globalActivity,
Visio.VisAutoConnectDir.visAutoConnectDirNone)
            Else
                globalActivity.AutoConnect(globalQueue,
Visio.VisAutoConnectDir.visAutoConnectDirNone)
            End If

           _activityDependencies(activity.Name.Value)(c.Direction).Add(c)
           globalConnector = VisioHelper.GetConnectorGluedTo(globalActivity,
c.Direction = ConnectedItem.DirectionEnum.Incoming, globalQueue)
           VisioHelper.CloneShapeProperties(c.Connector.Shape, globalConnector)
           VisioHelper.SetLineColor(globalConnector, randomColor)
       Next
     Next
  Next
End Sub
```

Figura 3.11: Procedimento utilizado para a geração do diagrama global

Geração do programa de simulação

O processo de geração do programa de simulação consiste na tradução das representações expressas no diagrama global, para uma linguagem de programação. Conforme já foi referido neste capítulo, os programas de simulação gerados pelo protótipo são codificados em VBA. A escolha desta linguagem de programação baseou-se nos requisitos 1.1 e 1.4, apresentados por Ståhl [2000] e que estão representados no 0.

Numa primeira fase são declarados, e inicializados, os objetos que intervêm na simulação; depois são definidas as condições iniciais; e por fim, é definida a

dependência de recursos de cada atividade (Figura 3.12). Se o diagrama global ainda não existir, o primeiro passo irá consistir na sua criação.

```
Private Sub DeclareAssignVariables(dec As StringBuilder, assign As
StringBuilder)
    Dim name As String = Nothing
    Dim variable As String = Nothing
    Dim type As String = Nothing
    Dim quantity As Integer = 0
    dec.AppendLine(VBCodeSyntaxe.DeclareVariable("simulation",
VBCodeSyntaxe.VariableVisibilityEnum.Public, "Simulation", "Nothing"))
    assign.AppendLine(VBCodeSyntaxe.Assign("simulation", "New Simulation()"))
    assign.AppendLine(VBCodeSyntaxe.Assign("simulation.Length",
GlobalDiagram.Simulation.Variables(Simulation.LENGTH_PROPERTY).Value))
    For index As Integer = 0 To GlobalDiagram.Simulation.Variables.Count - 1
        Dim var As Variable = GlobalDiagram.Simulation.Variables(index)
        If Not var.IsSystemVariable Then
            DeclareVariable("simulation", var.Name, var.Value)
        End If
    Next
    For Each e As Entity In GlobalDiagram.Simulation.Entities
        name = e.Name
        variable = "entity" & name
        DeclareEntity(variable)
        AssignEntity(variable, name)
        For index As Integer = 0 To e.Variables.Count - 1
            Dim var As Variable = e.Variables(index)
            If Not var.IsSystemVariable Then
                DeclareVariable(variable, var.Name, var.Value)
            End If
        Next
        For index As Integer = 0 To e.Queues.Count - 1
            Dim q As Queue = e.Queues(index)
            name = q.Name.Value
            variable = "queue" & name
            type = q.Type.Value
            DeclareQueue(variable, name)
            AssignQueue(variable, q)
            QueueInitialConditions(variable, q)
        Next
    Next
    Dim activities As Activity() = GlobalDiagram.Simulation.GetAllActivities()
    If activities IsNot Nothing Then
        For index As Integer = 0 To activities.Length - 1
            Dim act As Activity = activities(index)
            name = act.Name.Value
            variable = "activity" & name
            DeclareActivity(variable)
            AssignActivity(variable, act)
            CreateActivityEventOnSchedule(act, sbMethods)
            CreateActivityEventOnExecute(act, sbMethods)
        Next
    End If
End Sub
```

Figura 3.12: Procedimento para a declaração e inicialização dos objetos

Para simplificar e reduzir o código-fonte do programa gerado, o mecanismo que controla toda a simulação, vulgarmente designado por executivo de simulação, foi implementado numa biblioteca à parte "*ActivityBasedSimulation.dll*". É nesta biblioteca onde estão implementados: os métodos para a criação e manipulação das atividades, entidades, filas de espera e variáveis; o organizador de dados estatísticos; e o gerador de valores de variáveis aleatórias (Figura 3.13).

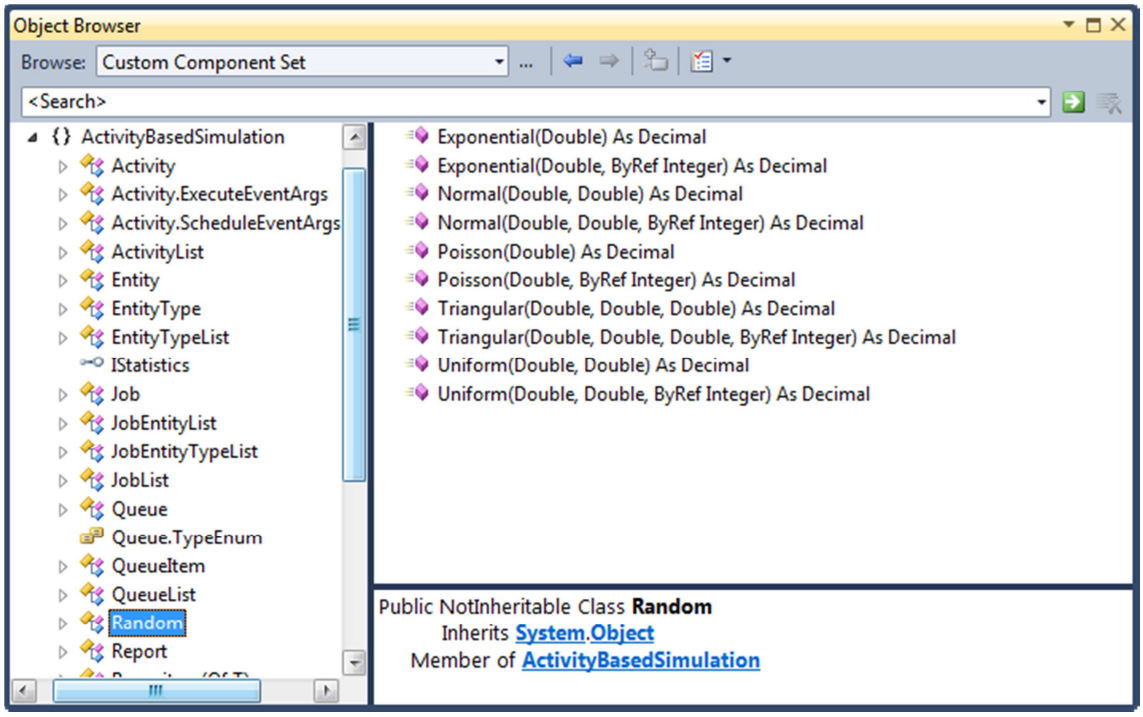


Figura 3.13: Métodos disponíveis para a geração de valores aleatórios

Apesar de a biblioteca ter sido construída para este projeto, funcionando como um componente do protótipo, os métodos existentes foram implementados de forma genérica e independente, para que possam ser utilizados na implementação de projetos de simulação que recorram a este paradigma. — Nas turmas compostas por alunos com conhecimentos consideráveis de programação, aqueles que preferirem poderão incorporar a biblioteca no Visual Studio e construir, manualmente, o programa de simulação.

3.4 Execução do Programa de Simulação

A execução do programa de simulação consiste na reprodução, linha a linha, do código fonte do programa de simulação, que é gerado automaticamente pelo protótipo. É através deste processo que são recolhidos os dados estatísticos relevantes para a análise da simulação. O facto de o programa ser gerado automaticamente não impede que o utilizador o possa alterar.

Uma vez que o programa que será executado é aquele que figurar na janela de interpretação de código, o utilizador poderá inicialmente gerar o programa da simulação, fazendo, posteriormente, as alterações que julgue necessárias. Se o programa introduzido possuir algum erro, a execução será interrompida, sendo o utilizador alertado para o erro cometido (Figura 3.14).

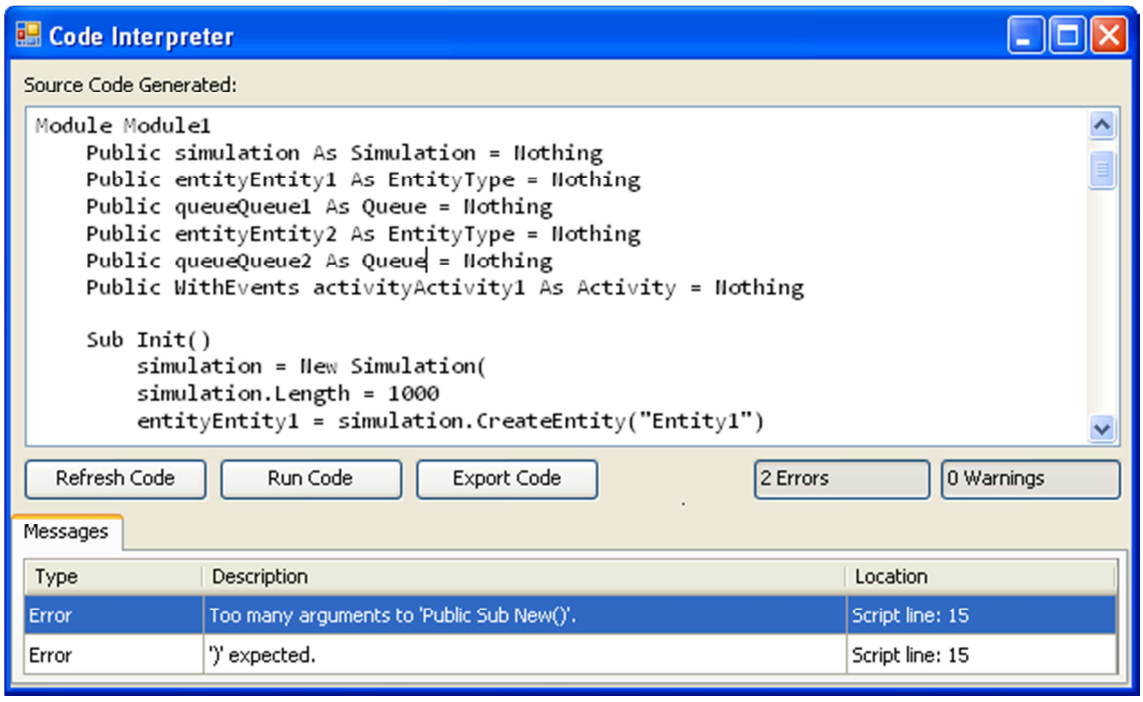


Figura 3.14: Janela de interpretação do programa de simulação

Refira-se também que o programa além de executado poderá ser exportado para o VB, ou para ficheiro de texto. Na sua essência, o mecanismo de execução recorre aos métodos disponibilizados para o efeito, que acompanham a plataforma .NET.

Estatísticas da simulação

Ao longo da execução da simulação são recolhidos dados para o tratamento estatístico das filas de espera, atividades e entidades. Esses dados são exibidos no final da execução da simulação (Figura 3.15).

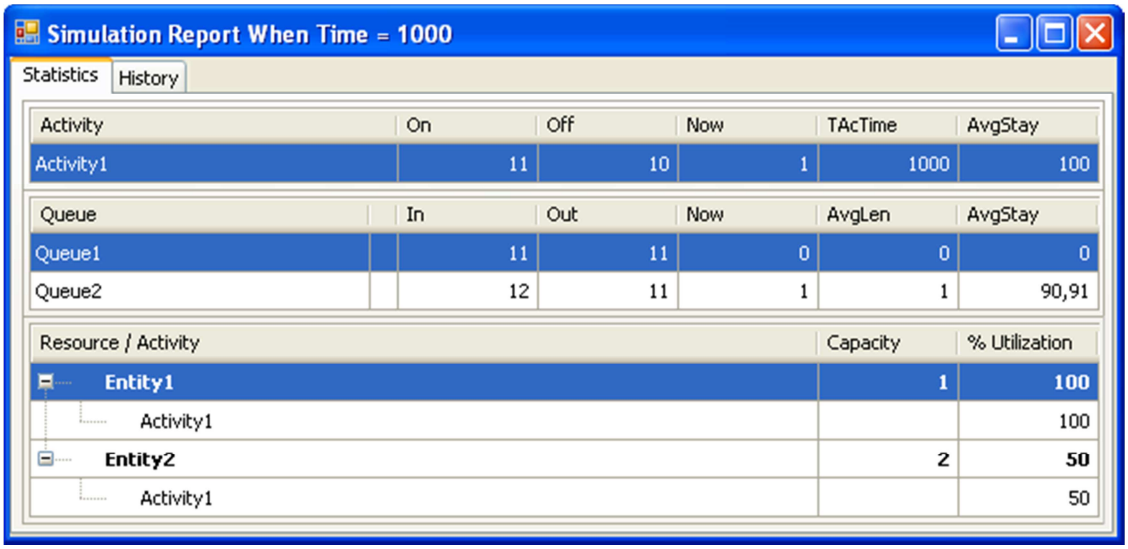
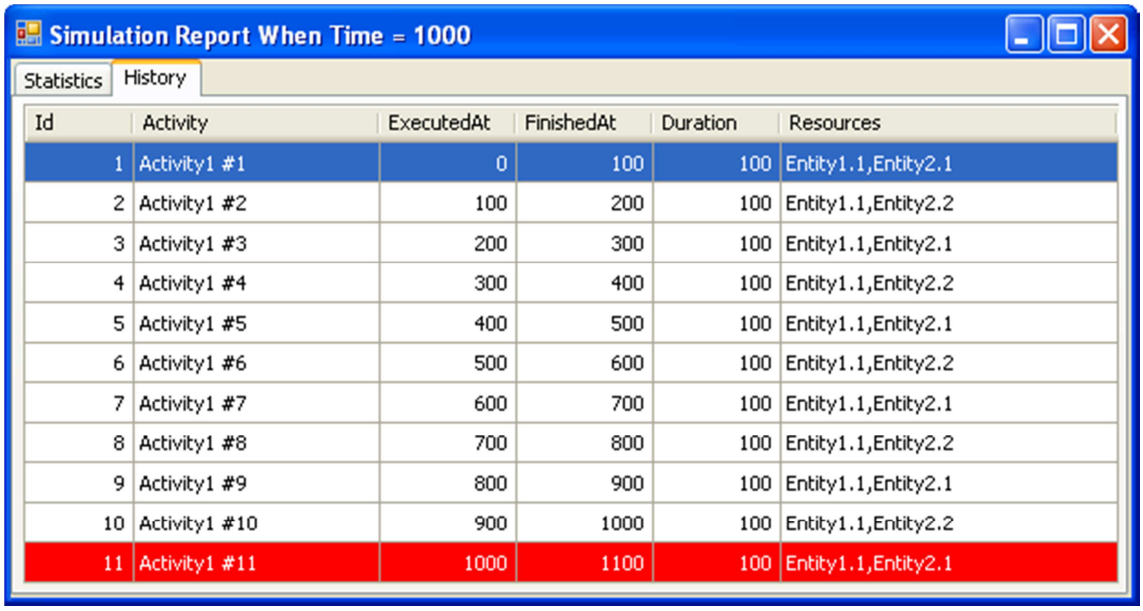


Figura 3.15: Janela de estatísticas da simulação

Histórico de agendamento e execução de atividades

Além das estatísticas da simulação, no fim da execução da simulação, é também exibido o histórico de agendamento e execução das atividades (Figura 3.16). Esse histórico permite ver quais as atividades que foram agendadas, a ordem de agendamento (*Id*), o instante do agendamento/execução (*StartAt*), o instante de conclusão (*EndAt*), a duração (*Duration*) e os recursos (*Resources*) envolvidos na simulação.



Id	Activity	ExecutedAt	FinishedAt	Duration	Resources
1	Activity1 #1	0	100	100	Entity1.1,Entity2.1
2	Activity1 #2	100	200	100	Entity1.1,Entity2.2
3	Activity1 #3	200	300	100	Entity1.1,Entity2.1
4	Activity1 #4	300	400	100	Entity1.1,Entity2.2
5	Activity1 #5	400	500	100	Entity1.1,Entity2.1
6	Activity1 #6	500	600	100	Entity1.1,Entity2.2
7	Activity1 #7	600	700	100	Entity1.1,Entity2.1
8	Activity1 #8	700	800	100	Entity1.1,Entity2.2
9	Activity1 #9	800	900	100	Entity1.1,Entity2.1
10	Activity1 #10	900	1000	100	Entity1.1,Entity2.2
11	Activity1 #11	1000	1100	100	Entity1.1,Entity2.1

Figura 3.16: Histórico da execução da simulação

3.5 Arquitetura e Lógica de Funcionamento

Da união dos esforços de desenvolvimento, referidos ao longo deste capítulo, resultou o protótipo, cuja estrutura se apresenta na Figura 3.17. Conforme se poderá observar, o protótipo apenas poderá ser utilizado através do Visio 2010, uma aplicação que apenas funciona nos sistemas operativos Windows.

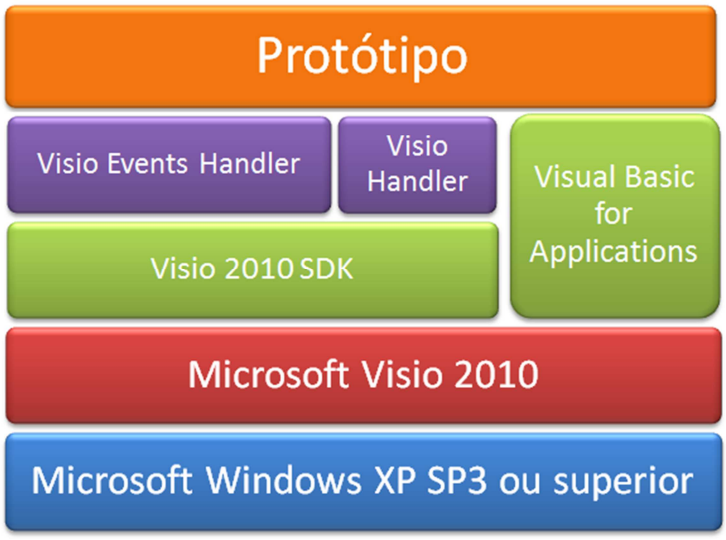


Figura 3.17: Estruturação interna do protótipo

Para poder interagir com o Visio, além do SDK disponibilizado, o protótipo recorre às bibliotecas de métodos auxiliares criadas para o efeito. Os programas de simulação, gerados pelo protótipo, são executados através do VBA.

De uma forma geral, a lógica de funcionamento do protótipo poderá ser descrita conforme apresentado na Figura 3.18.

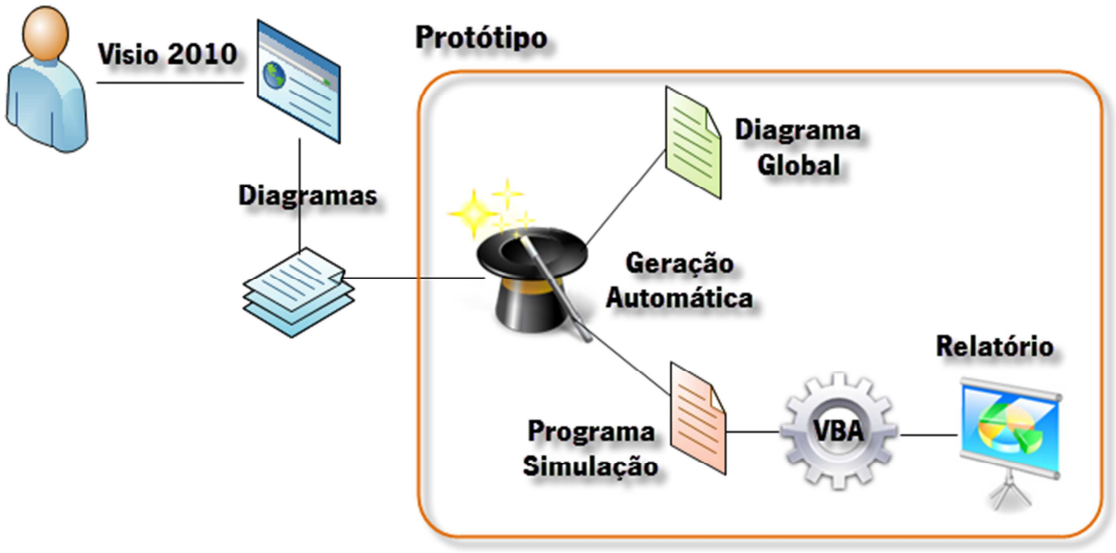


Figura 3.18: Lógica de funcionamento do protótipo

O utilizador representa, no Visio, os diagramas de todas as entidades que participam na simulação. Uma vez representados os diagramas, o utilizador poderá gerar o diagrama global da simulação, ou então, gerar o programa de simulação — tanto o diagrama global, como o programa de simulação correspondente, são gerados automaticamente pelo protótipo. Uma vez gerado o programa, que é executado recorrendo ao VBA, é apresentado o relatório de simulação, onde se podem analisar as estatísticas, e o histórico, da simulação.

4. Casos de Demonstração

4.1 Estratégia de Demonstração

De forma a demonstrar o funcionamento e a validade do protótipo, serão utilizados três casos de estudo normalmente utilizados no ensino de simulação. Os casos foram adaptados de Luís M S Dias [2005]; Rodrigues [1996], e serão apresentados por ordem crescente de complexidade.

Para cada um dos casos de estudo, será explicado e descrito o funcionamento do sistema, que será modelado. Após a modelação, serão apresentadas as estatísticas da simulação. Por uma questão de espaço, aspetos como os princípios gerais de funcionamento, o histórico de execução, a explicação do código gerado e a validação dos valores estatísticos, apenas serão apresentados no primeiro caso. Os programas de simulação dos casos 2 e 3 encontram-se disponíveis em anexo (Anexo 2 e Anexo 3).

4.2 Caso 1: Máquinas Semiautomáticas

O caso que passaremos a demonstrar foi já referido na explicação dos DCA (2.2), e tem com objetivo verificar se um trabalhador é suficiente para operar três máquinas semiautomáticas, iguais.

4.2.1 Descrição do sistema

As máquinas semiautomáticas requerem a atenção de um operador, que executa a sua preparação para que estas possam entrar em funcionamento. Completada a operação de produção as máquinas ficam novamente paradas, aguardando que um operador esteja livre para iniciar um novo ciclo. A preparação de cada máquina demora 3 unidades de tempo, e permite que cada máquina funcione durante 10 unidades de tempo.

4.2.2 Modelação do sistema

Uma vez compreendido o funcionamento, e efetuado o levantamento das entidades e atividades que compõem o sistema, podemos dar início à construção da simulação. Para construir uma nova simulação, no menu do Visio onde se encontra incorporado o protótipo (Figura 4.1), devemos invocar a opção *New Simulation* e depois no assistente, indicar as entidades que fazem parte da simulação, conforme se apresenta na Figura 4.2.

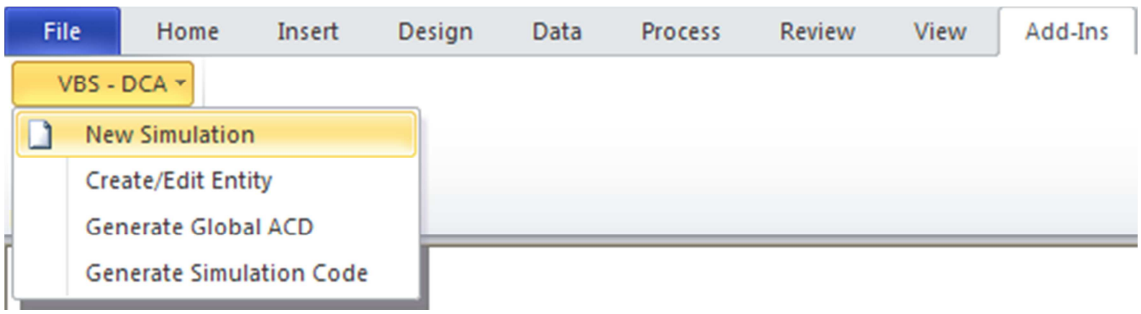


Figura 4.1: Menu de extensões do Visio

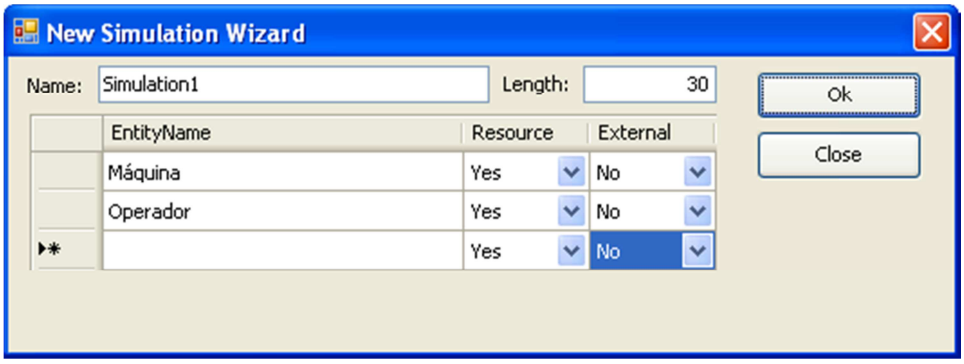


Figura 4.2: Assistente da criação da simulação das máquinas semiautomáticas

Depois de confirmadas as entidades que irão compor o sistema, é criado no Visio um documento com duas páginas em branco, uma para cada entidade. O passo seguinte consistirá em arrastar do *stencil*, as *shapes* necessárias, para representar o diagrama de cada uma das entidades (Figura 4.3). Depois de criado o documento, a inserção, ou edição, de entidades deverá ser efetuada através do comando *Create/Edit Entity*.

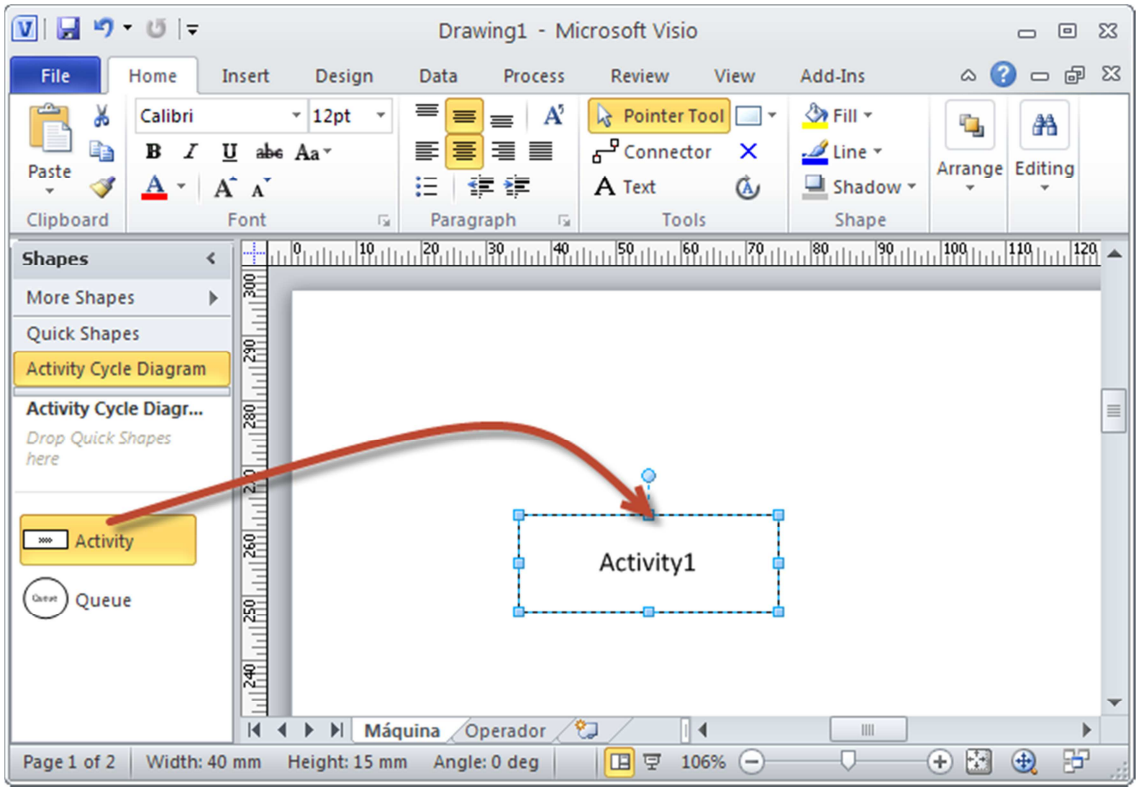


Figura 4.3: Inserção de *shapes* em diagramas

Sempre que uma nova *shape* for inserida é apresentado uma janela com os valores predefinidos dos atributos que compõem a *shape* (Figura 4.4).

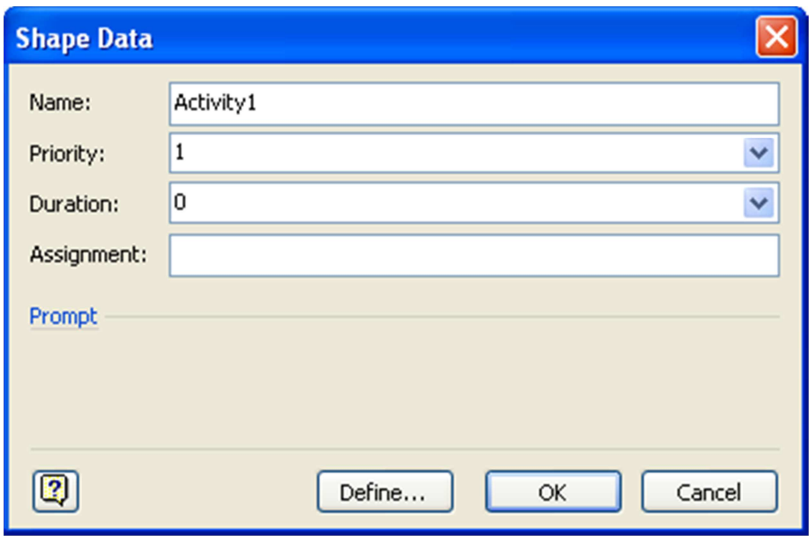


Figura 4.4: Atributos predefinidos da *shape Activity*

Os atributos das *shapes* poderão ser acedidos a qualquer momento, para isso deverá proceder conforme ilustrado na Figura 4.5.

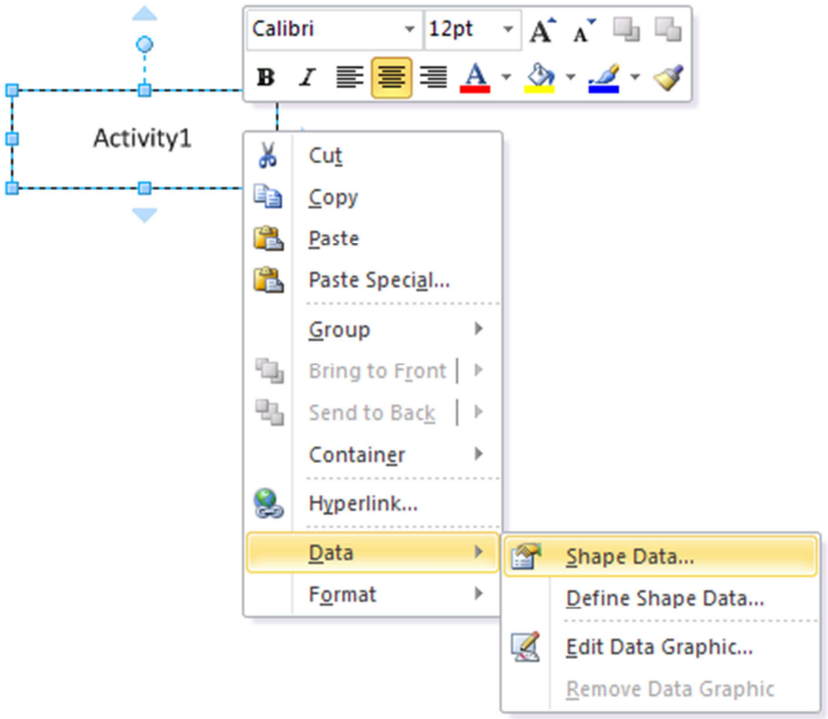


Figura 4.5: Procedimento para a invocação dos atributos das *shapes*

Uma vez invocado o comando, é apresentada a janela ilustrada na Figura 4.6. Através dessa janela o utilizador poderá consultar, ou alterar, os valores dos atributos.

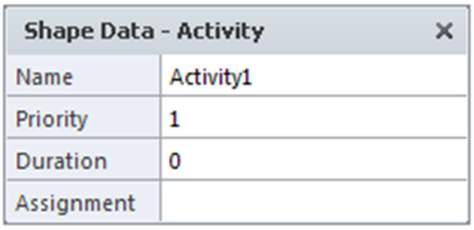


Figura 4.6: Janela de atributos das *shapes*

Se optar por não fechar a janela, sempre que selecionar uma outra *shape*, a janela é atualizada com os atributos da *shape* selecionada. Caso altere algum atributo e introduza um valor inválido, o protótipo reage conforme ilustrado na Figura 4.7.

Figura 4.7: Mensagem de erro de validação de atributos

Diagrama da entidade Máquina

Depois de construído o diagrama da entidade Máquina (Figura 4.8), podemos verificar que as máquinas estão inicialmente paradas, ficando, depois de preparadas, livres para entrar em funcionamento. Terminado o funcionamento, estas voltam ao estado inicial, ficando novamente paradas. Os atributos das filas de espera e das atividades estão representados, respetivamente, na Tabela 4.1 e na Tabela 4.2.

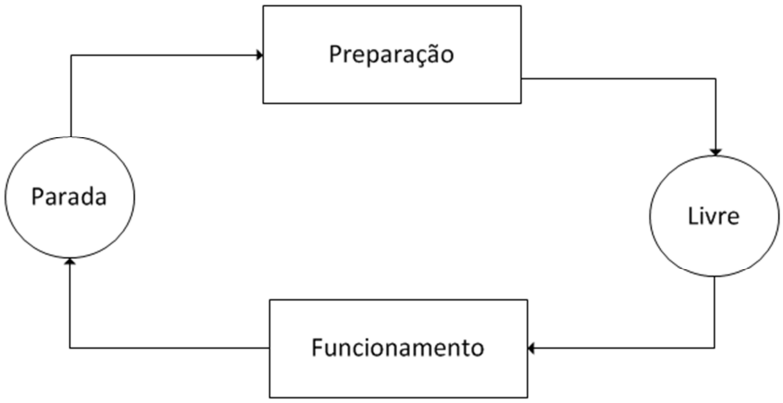


Figura 4.8: Diagrama da entidade Máquina

Tabela 4.1: Atributos das filas da entidade Máquina

Name	***Type***	***Quantity***	***Priority Field***
Parada	FIFO	3	
Livre	FIFO	0	

Uma vez que apenas são utilizadas filas do tipo *First-In-First-Out* (FIFO), não é necessário especificar o atributo que define a prioridade da entidade. O número de máquinas paradas, que corresponde ao número de máquinas existentes, é indicado no atributo *Quantity* da fila Parada.

Tabela 4.2: Atributos das atividades da entidade Máquina

Name	***Priority***	***Duration***	***Assignment***
Preparação	1	3	
Funcionamento	1	10	

Nesta simulação considerou-se que todas as atividades têm a mesma prioridade. Quanto menor o valor de *Priority*, maior será a prioridade.

Diagrama da entidade Operador

Depois da construção do diagrama da entidade Operador (Figura 4.9), verificamos também que, para que a atividade preparação possa ter início é necessário que existam operadores inativos. Após a preparação, os operadores volvam ao seu estado inicial, ou seja, ao estado inativo. Na Tabela 4.3 estão apresentados os atributos da única fila que existe na entidade. Uma vez que os atributos da

atividade preparação já foram definidos na entidade anterior, à exceção do nome da atividade, não é necessário definir novamente os atributos.

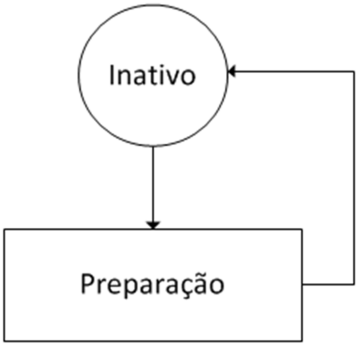


Figura 4.9: Diagrama da entidade Operador

Tabela 4.3: Atributos das filas da entidade Operador

Name	***Type***	***Quantity***	***Priority Field***
Inativo	FIFO	1	

Se em vez de um, se pretendesse verificar se dois operadores eram suficientes para preparar as máquinas, o atributo *Quantity* deveria possuir o valor 2.

Diagrama global da simulação

Porque os diagramas apenas ilustram o ciclo das entidades por si representadas, para percecionar o sistema como um todo, surge a necessidade de um diagrama global, que agrupe o ciclo de todas as entidades que intervenham no sistema. No protótipo que apresentamos, o diagrama global é construído de forma automática. O utilizador apenas precisa de invocar o comando *Generate Global ACD*, ou então proceder à geração do programa de simulação. Na Figura 4.10 está representado o diagrama global da simulação.

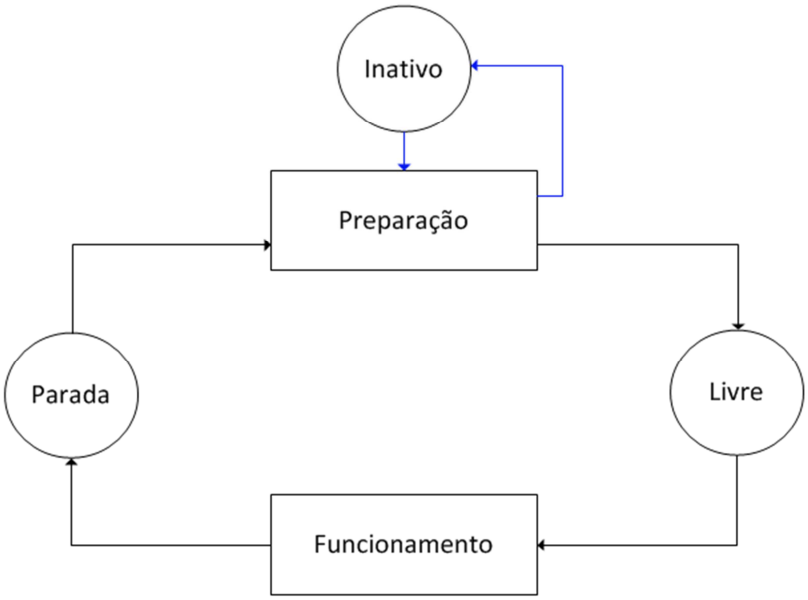


Figura 4.10: Diagrama global da simulação das máquinas semiautomáticas

No diagrama global, o ciclo (percurso) das entidades é representado recorrendo a diferentes cores. Conforme se poderá observar na Figura 4.10, o ciclo da máquina está representado a preto, e o ciclo do operador a azul. A utilização de cores diferentes permite delimitar o percurso de cada entidade. Na figura podemos facilmente verificar que o operador está confinado à atividade Preparação, enquanto as máquinas estão envolvidas nas atividades Preparação e Funcionamento.

4.2.3 Programa de simulação

Terminados os diagramas, podemos dar início à construção do programa de simulação. Tal como acontece com o diagrama global, o programa de simulação é construído automaticamente, e consiste num conjunto de instruções, que quando executadas simulam a execução do sistema modelado, durante o período de tempo especificado. Para construir o programa de simulação o utilizador apenas necessita de invocar o comando *Generate Simulation Code*.

De uma forma geral podemos decompor os programas de simulação gerados em quatro partes: (1) declaração das variáveis da simulação, (2) inicialização da simulação, (3) motor de simulação, e (4) descrição da movimentação dos recursos da simulação.

Declaração das variáveis da simulação

A declaração das variáveis é o processo no qual são definidos o nome e o tipo das variáveis que serão utilizadas na simulação. Uma vez que por cada entidade, fila e atividade será criada uma variável, o número de variáveis que irá compor a simulação irá depender do número de itens utilizados. Cada variável irá conter um prefixo que permitirá facilmente identificar o item a que diz respeito. Além dessas, existirá uma variável que designa a própria simulação. A Figura 4.11 apresenta o código VB referente à declaração das variáveis da simulação.

```
Public simulation As Simulation = Nothing
Public entityMáquina As EntityType = Nothing
Public queueParada As Queue = Nothing
Public queueLivre As Queue = Nothing
Public entityOperador As EntityType = Nothing
Public queueInativo As Queue = Nothing
Public WithEvents activityPreparação As Activity = Nothing
Public WithEvents activityFuncionamento As Activity = Nothing
```

Figura 4.11: Código para a declaração das variáveis da simulação

Inicialização da simulação

A inicialização da simulação é o processo onde são inicializados os valores das variáveis, onde são definidas as dependências de cada uma das atividades, e onde são definidas as condições iniciais da simulação. O código referente à inicialização da simulação aparecerá, sempre, contido dentro do método `Init` (Figura 4.12).

Motor de simulação

O motor de simulação corresponde ao conjunto de instruções necessárias para a execução da simulação. O código referente ao motor de simulação é sempre o

mesmo, independentemente da simulação, e aparecerá sempre contido dentro do método `Main` (Figura 4.13)

```
Sub Init()
    simulation = New Simulation()
    simulation.Length = 30
    entityMáquina = simulation.CreateEntity("Máquina")
    queueParada = simulation.CreateQueue("Parada",
Queue.TypeEnum.FirstInFirstOut, entityMáquina)
    queueLivre = simulation.CreateQueue("Livre", Queue.TypeEnum.FirstInFirstOut,
entityMáquina)
    entityOperador = simulation.CreateEntity("Operador")
    queueInativo = simulation.CreateQueue("Inativo",
Queue.TypeEnum.FirstInFirstOut, entityOperador)
    activityPreparação = simulation.CreateActivity("Preparação", 1)
    activityFuncionamento = simulation.CreateActivity("Funcionamento", 1)
    queueParada.Insert(entityMáquina.Generate(3))
    queueInativo.Insert(entityOperador.Generate(1))
    activityPreparação.NeedResource(entityMáquina, 1)
    activityPreparação.NeedResource(entityOperador, 1)
    activityFuncionamento.NeedResource(entityMáquina, 1)
End Sub
```

Figura 4.12: Código para a inicialização da simulação

```
Sub Main()
    Init()
    Dim simulationEnd As Boolean = False

    If Not simulation.ResourcesDefined() Then
        MsgBox("There aren't enough resources to start the simulation.",
MsgBoxStyle.Exclamation)
        Exit Sub
    End If

    Do Until simulationEnd
        For Each a As Activity In simulation.Activities
            Do While a.CanStart()
                simulation.ScheduleNewJob(a, simulation.CurrentTime)
            Loop
        Next
        Dim nextJob As Job = simulation.GetNextJob()
        simulationEnd = (nextJob Is Nothing OrElse nextJob.ExecuteAt >
simulation.Length)
        If Not simulationEnd Then
            simulation.UpdateCurrentTime(nextJob.ExecuteAt)
            For Each j As Job In
simulation.GetJobsByExecutionTime(simulation.CurrentTime)
                simulation.ExecuteJob(j)
            Next
        End If
    Loop

    simulation.Report()
End Sub
```

Figura 4.13: Código do motor de simulação

Descrição da movimentação de recursos

A descrição da movimentação de recursos consiste na definição da lógica de ocupação e libertação de recursos. Conforme se poderá observar na Figura 4.14, cada atividade será composta por dois métodos. Um deles será executado no agendamento da atividade (*OnSchedule*) e o outro no fim de execução da atividade (*OnExecute*).

Nos métodos referentes ao agendamento das atividades são definidas as filas de onde serão removidos os recursos que intervêm na atividade, e é especificada a duração de cada atividade. Nos métodos relativos à execução das atividades, apenas são especificadas as filas para onde serão deslocados os recursos após a conclusão da atividade.

```
Sub Preparação_OnSchedule(sender As Activity, e As Activity.ScheduleEventArgs)
Handles activityPreparação.OnSchedule
    e.Resources(entityMáquina).RemoveFrom = queueParada
    e.Resources(entityOperador).RemoveFrom = queueInativo
    e.Duration = 3
End Sub

Sub Preparação_OnExecute(sender As Job, e As Activity.ExecuteEventArgs) Handles
activityPreparação.OnExecute
    e.Resources(entityMáquina).InsertInto = queueLivre
    e.Resources(entityOperador).InsertInto = queueInativo
End Sub

Sub Funcionamento_OnSchedule(sender As Activity, e As
Activity.ScheduleEventArgs) Handles activityFuncionamento.OnSchedule
    e.Resources(entityMáquina).RemoveFrom = queueLivre
    e.Duration = 10
End Sub

Sub Funcionamento_OnExecute(sender As Job, e As Activity.ExecuteEventArgs)
Handles activityFuncionamento.OnExecute
    e.Resources(entityMáquina).InsertInto = queueParada
End Sub
```

Figura 4.14: Código para a movimentação de recursos

Depois de gerado o programa de simulação, para o executar, o utilizador apenas necessita de pressionar o botão *Execute Code*.

4.2.4 Resultados da simulação

Da execução do programa de simulação resultam as estatísticas e o histórico, apresentados, respetivamente, na Figura 4.15 e na Figura 4.16.

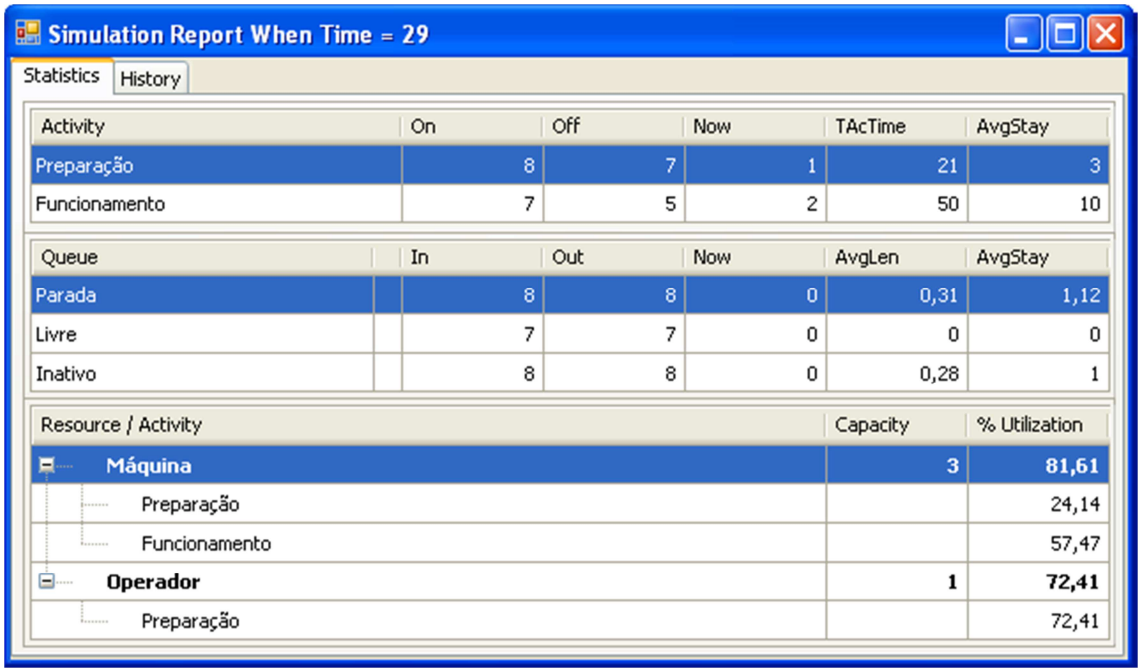


Figura 4.15: Estatísticas da simulação das máquinas semiautomáticas

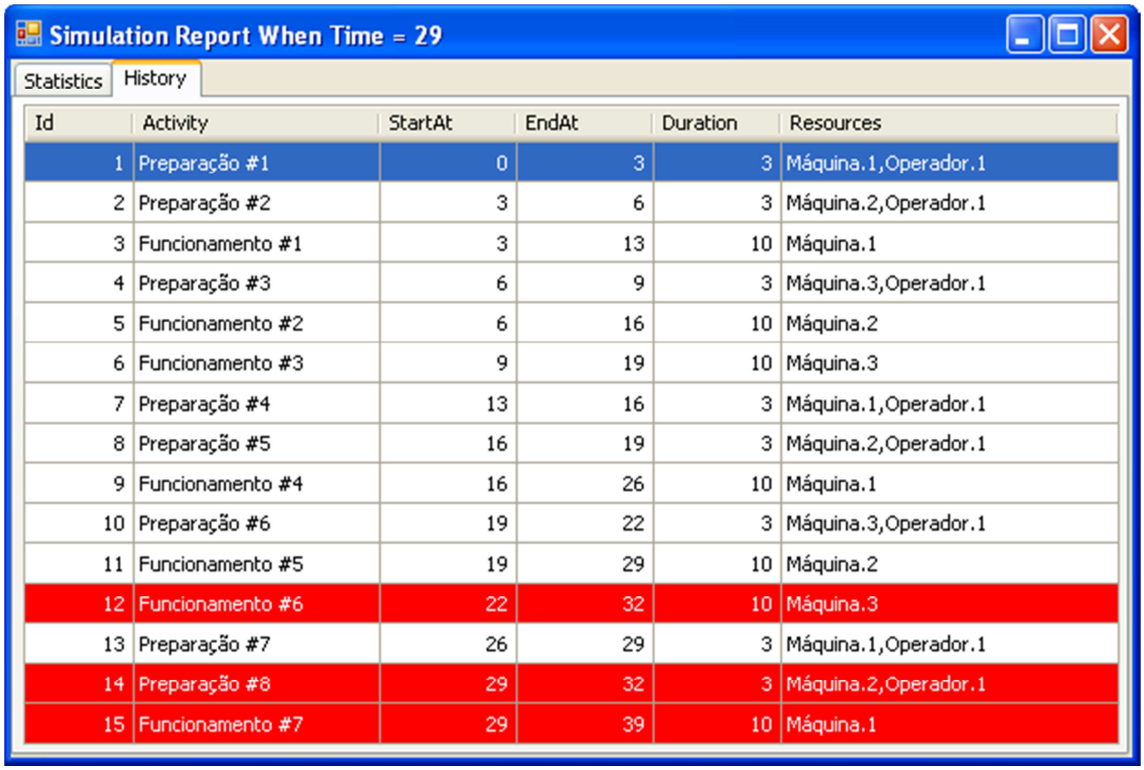


Figura 4.16: Histórico da simulação das máquinas semiautomáticas

Se confrontarmos o histórico apresentado na Figura 4.16 com o quadro resumo apresentado na Tabela 2.1, verificamos que o número de trabalhos agendados, a ordem de execução, e o número de trabalhos por concluir são coincidentes.

4.2.5 Validação de resultados da simulação

As estatísticas da simulação permitem-nos analisar a execução de atividades, a ocupação das filas de espera, e a ocupação dos recursos. Para aferirmos que os valores apresentados estão de acordo com as fórmulas utilizadas pela ferramenta, vamos efetuar os cálculos manualmente recorrendo ao histórico da simulação.

Estatísticas das atividades

Se repararmos no histórico da simulação (Figura 4.16), foram iniciados (*On*) 8 trabalhos de preparação, sendo que 7 já foram concluídos (*Off*), faltando apenas concluir 1 trabalho (*Now*). No que diz respeito ao funcionamento, foram já concluídos 5 dos 7 trabalhos iniciados. O tempo total de preparação é de 21 unidades de tempo e o tempo total de funcionamento, 50. — O tempo total da atividade Funcionamento é superior ao tempo total da simulação porque existem 3 máquinas a funcionar no sistema.

A duração média é de 3 unidades de tempo, no caso da preparação, e de 10 unidades no caso do funcionamento. Dado que não usamos valores estocásticos, a duração de cada atividade será constante e a duração média irá coincidir com a duração especificada.

Na Figura 4.17 e na Figura 4.18 estão apresentadas as fórmulas para o cálculo do tempo total de atividade e da duração média.

$$TAcTime = \sum_{i}^{\infty} (AcEnd_i - AcStart_i)$$

Figura 4.17: Fórmula para o cálculo da duração total de uma atividade

$$AvgStay = \frac{TActTime}{Off}$$

Figura 4.18: Fórmula para o cálculo da duração média de uma atividade

Na Tabela 4.4 estão apresentados os valores que resultam da aplicação das fórmulas. Se confrontarmos os valores apresentados pelo protótipo (Figura 4.15) com aqueles que resultaram da aplicação das fórmulas, verificamos que estes coincidem.

Tabela 4.4: Estatísticas das atividades

	On	***Off***	***Duration***	***TAcTime***	***AvgStay***
Preparação	8	7	3	**21**	**3**
Funcionamento	7	5	10	**50**	**10**

Estatísticas das filas de espera

Uma vez que a simulação se inicia com o operador no estado inativo, e porque foram concluídos 7 trabalhos de preparação, a fila Inativo deverá possuir um número de entradas (*In*) igual a 8 — recorde-se que sempre que a preparação termina, o operador fica inativo. Utilizando o mesmo raciocínio e uma vez que após a preparação as máquinas ficam livres, a fila de máquinas livres deverá registar 7 entradas. No que diz respeito à fila de máquinas paradas esta deverá registar 8 entradas, sendo que 3 são relativas ao estado inicial das máquinas e as restantes, à conclusão dos trabalhos de funcionamento.

Para que se possa verificar se as estatísticas das filas de espera estão corretas, temos que calcular o tempo total de permanência nas filas. Esse cálculo poderá efetuado através de duas formas. Nas situações em que um recurso apenas participa numa só atividade, como acontece com o operador, uma vez determinado o tempo total de atividade, apenas temos de retirar ao instante atual da simulação, o tempo total de atividade. Dado que a simulação termina no

instante 29 e porque o tempo total da Preparação é de 21 unidades de tempo, o tempo de permanência na fila será 8 (29 – 21 = 8).

Nas situações em que um mesmo recurso é utilizado em várias atividades, para apurarmos o tempo de permanência, temos de recorrer ao histórico da simulação e calcular a diferença entre o fim da atividade antecedente (*EndAt*) e o início da atividade consequente (*StartAt*). O tempo total de permanência é determinado pela soma das diferenças apuradas. Na Tabela 4.5 e na Tabela 4.6 está apresentado o tempo total de permanência nas filas Parada e Livre.

Tabela 4.5: Tempo de permanência na fila Parada

	Funcionamento		**Preparação**		**Permanência**
	Id	***EndAt***	***Id***	***StartAt***	
Máquina 1	–	0	1	0	0
	3	13	7	13	0
	9	26	13	26	0
	15	29			
Máquina 2	–	0	2	3	3
	5	16	8	16	0
	11	29	14	29	0
Máquina 3	–	0	4	6	6
	6	19	10	19	0
Total					**9**

Tabela 4.6: Tempo de permanência na fila Livre

	Preparação		**Funcionamento**		**Permanência**
	Id	***EndAt***	***Id***	***StartAt***	
Máquina 1	1	3	3	3	0
	7	16	9	16	0
	13	29	15	29	0
Máquina 2	2	6	5	6	0
	8	19	11	19	0
Máquina 3	4	9	6	9	0
	10	22			
Total					**0**

O cálculo do tempo médio de permanência é similar ao cálculo da duração média das atividades, consistindo na divisão do tempo total de permanência (*TotStay*) pelo número de entidades que já abandonaram a fila (*Out*).

$$AvgStay = \frac{TotStay}{Out}$$

Figura 4.19: Fórmula para o cálculo do tempo médio de permanência nas filas

Para calcular o comprimento médio de uma fila temos de verificar se existem entidades na fila (*Now*) e se existirem, apurar há quanto tempo lá se encontram. No caso de não existirem, o tempo de permanência (*NowStay*) é zero.

$$AvgLen = \frac{TotStay + NowStay}{SimTime}$$

Figura 4.20: Fórmula para o cálculo do comprimento médio das filas

Na Tabela 4.7 estão apresentados os valores que resultaram da aplicação das fórmulas apresentadas.

Tabela 4.7: Estatísticas das filas de espera

	TotStay	***In***	***Out***	***Now***	***AvgStay***	***AvgLen***
Parada	9	8	8	0	**1,12**	**0,31**
Livre	0	7	7	0	**0**	**0**
Inativo	8	8	8	0	**1**	**0,28**

Estatísticas dos recursos

Para que se possa validar os valores estatísticos da utilização de recursos é necessário determinar o tempo total da sua utilização, para isso iremos somar o tempo de atividade de todas as atividades onde cada um dos recursos esteve envolvido (Tabela 4.8).

Tabela 4.8: Tempo total de serviço por recurso

	Máquina	**Operador**
Preparação	21	21
Funcionamento	50	–
Total	71	21

A percentagem global de utilização de recursos resulta da aplicação da fórmula apresentada na Figura 4.21, em que *TResTime* designa o tempo total que um recurso foi utilizado, *SimTime*, o instante atual da simulação e *Capacity*, o número de recursos existentes, desse tipo.

$$\%Utilization = \frac{TResTime}{SimTime \times Capacity} \times 100$$

Figura 4.21: Fórmula para o cálculo da utilização global de recursos

Conforme podemos observar na Figura 4.22, o cálculo da percentagem de utilização de recursos por atividade é parecido com o anterior. Porque se pretende determinar a percentagem de utilização por atividade, ao invés de considerarmos o tempo total da utilização do recurso, iremos considerar o tempo total da atividade e multiplicar o resultado da primeira operação pelo número de recursos que participa na simulação (*Quantity*).

$$\%Utilization = \left(\frac{TActTime}{SimTime \times Capacity} \times 100\right) \times Quantity$$

Figura 4.22: Fórmula para o cálculo da utilização de recursos por atividade

Tabela 4.9: Estatísticas dos recursos

		Máquina	**Operador**
Nº de recursos		3	1
Tempo total de serviço		71	21
% Utilização	**Global**	81,61%	72,41%
	Preparação	24,14%	72,41%
	Funcionamento	57,47%	–

4.3 Caso 2: Agência Bancária

Este caso de estudo pretende ilustrar o funcionamento de uma agência bancária, onde são oferecidos dois serviços aos seus clientes. O objetivo da simulação é determinar o número de funcionários necessários para assegurar a qualidade de serviço estipulada pela agência. — Neste caso, a qualidade de serviço corresponde ao tempo de espera que a agência considera que os seus clientes estão dispostos a aguardar na fila de espera para os serviços.

4.3.1 Descrição do sistema

A agência bancária, em estudo, oferece dois serviços aos seus clientes: balcão e caixa. Sempre que um cliente chega à agência, este, primeiramente, deverá ser atendido no balcão, podendo ser posteriormente encaminhado para o caixa.

Para que os clientes possam aguardar a sua vez, existem duas filas de espera, uma para cada serviço. Os clientes que não necessitem de ir ao caixa abandonam a agência após o serviço de balcão, e os que necessitarem, após o serviço de caixa. Os funcionários da agência tanto poderão estar envolvidos no serviço de balcão como no serviço de caixa.

Com base em valores estatísticos recolhidos, ao longo o tempo, pela agência, apurou-se que 50% dos clientes que se dirigem ao balcão são posteriormente encaminhados para o caixa. Apurou-se também que a duração do atendimento é descrita por uma distribuição de *Poisson*, com média 4 no serviço de balcão; e com média 2 no serviço de caixa. A chegada de clientes é descrita por uma distribuição exponencial negativa de média 3.

Da análise ao funcionamento da agência podemos concluir que o sistema é composto por duas entidades (cliente e funcionário), e duas atividades (serviço de balcão e serviço de caixa). Uma vez que o cliente advém do exterior, dizemos que

a entidade cliente é uma entidade externa. Porque as atividades, na agência, são executadas pelos funcionários, consideramos que estes são recursos do sistema.

4.3.2 Modelação do sistema

Compreendido o funcionamento do sistema, podemos dar início à construção da simulação através do assistente criado para o efeito (Figura 4.23).

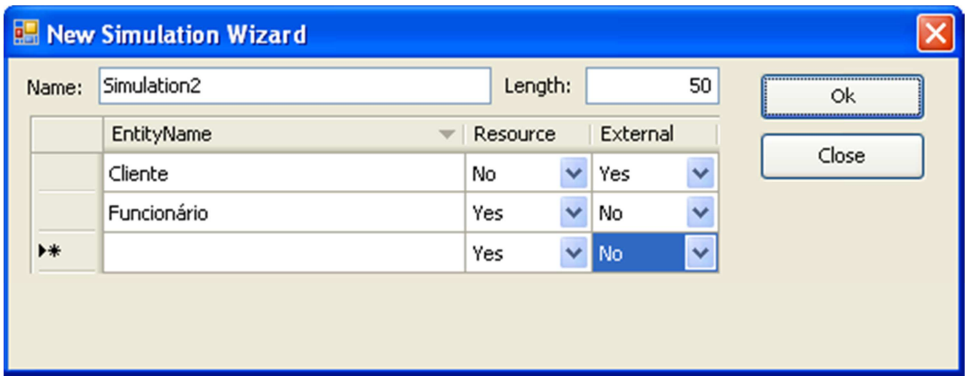


Figura 4.23: Assistente para a criação da simulação da agência bancária

Ao contrário do que aconteceu no caso anterior, o número de páginas criadas não irá coincidir com o número de entidades especificadas. Serão criadas três páginas, em vez de duas (Figura 4.24). Isto acontece porque, o mecanismo que gere as chegadas das entidades, cria, por cada entidade externa, uma entidade fictícia vulgarmente designada por porta.

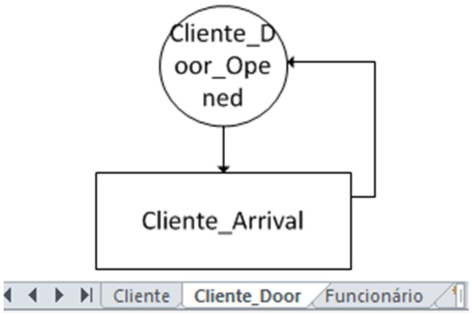


Figura 4.24: Porta da entidade Cliente

Diagrama da entidade Cliente

Em virtude do cliente ser uma entidade externa, o diagrama desta entidade vai estar já parcialmente construído com as *shapes* referentes ao mecanismo da porta (Figura 4.25).

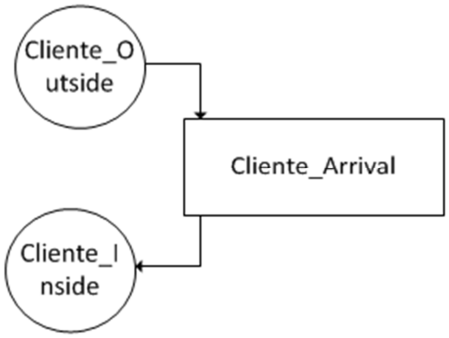


Figura 4.25: Diagrama da entidade cliente, parcialmente construído

Uma vez que, em média, apenas metade dos clientes recorrem ao serviço de caixa, sempre que chega um cliente, será necessário determinar se o mesmo irá, ou não deslocar-se ao caixa. Para isso, iremos adicionar, ao cliente, o atributo "VaiUtilizarCaixa". Conforme se poderá observar na Figura 4.26, a inserção é efetuada através da janela de edição de entidades, que está disponível através do comando *Create/Edit Entity*.

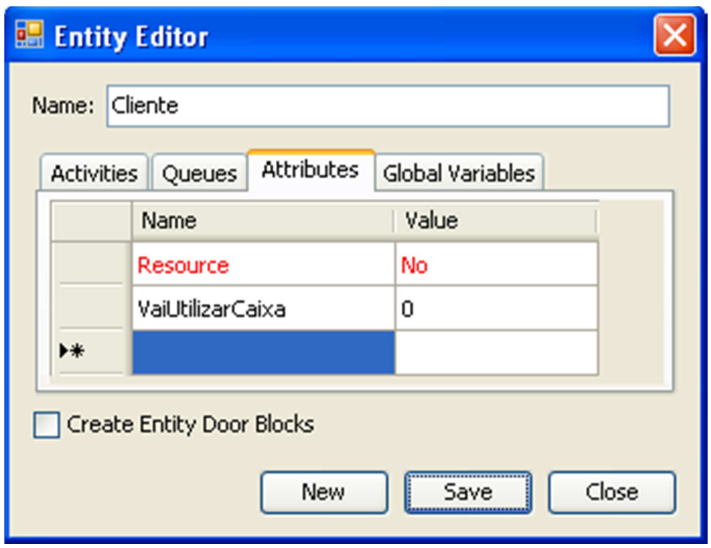


Figura 4.26: Inserção do atributo VaiUtilizarCaixa

O atributo criado irá depois receber valores de uma distribuição Uniforme. Os clientes cujo atributo seja maior ou igual a 0,50 irão deslocar-se ao caixa, os restantes irão apenas utilizar o serviço de balcão.

Para completar o diagrama da entidade cliente, será apenas necessário adicionar as atividades serviço de balcão (ServiçoBalcão) e serviço de caixa (ServiçoCaixa), e a fila de espera para o caixa (FilaEsperaCaixa). — Na Figura 4.27 está representado o diagrama completo da entidade. O conteúdo de cada uma das filas e atividades será apresentado na Tabela 4.10 e na Tabela 4.11.

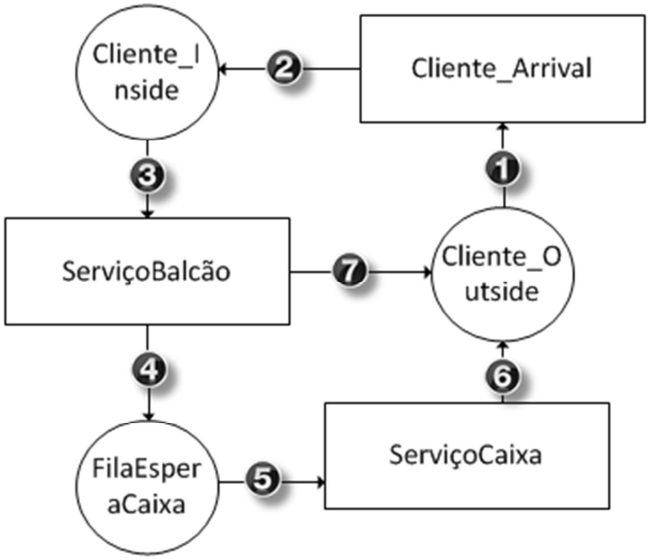


Figura 4.27: Diagrama completo da entidade Cliente

Conforme se poderá observar na Figura 4.27, os clientes chegam do exterior (1) à agência (2), onde aguardam até que chegue a vez de serem atendidos (3). Após serem atendidos no balcão, os clientes poderão ter que se deslocar ao caixa (4). Chegada a sua vez (5) e terminado o atendimento no caixa, os clientes regressam ao exterior (6). Os clientes que não necessitem de ir ao caixa, abandonam a agência após o serviço de balcão (7).

Tabela 4.10: Valores dos atributos das filas de espera

Name	*Type*	*Quantity*	*Priority Field*
Cliente_Inside	FIFO	0	
Cliente_Outside	FIFO	1000	
FilaEsperaCaixa	FIFO	0	

No início da simulação todos os clientes encontram-se no exterior da agência.

Tabela 4.11: Valores dos atributos das atividades

Name	*Priority*	*Duration*	*Assignment*
Cliente_Arrival	1	Exponential(3)	VaiUtilizarCaixa=Uniform(0,1)
ServiçoBalcão	1	Poisson(4)	
ServiçoCaixa	1	Poisson(2)	

Se numa mesma atividade for necessário efetuar várias atribuições, estas deverão ser separadas por ponto e vírgula. Exemplo: a=b; c=b/10; d=exponential(a*b).

Uma vez que, após o serviço de balcão, o cliente poderá seguir um de dois caminhos, é necessário indicar a condição que deverá ser satisfeita para se rumar em cada direção. Essa indicação apenas é necessária no caso de uma atividade possuir mais que uma fila consequente. Na Tabela 4.12 estão apresentadas as condições de seleção de destino da atividade ServiçoBalcão.

Tabela 4.12: Condições de seleção de destino

Origem	**Destino**	**Atributos**	
		Condition	***Quantity***
ServiçoBalcão	Cliente_Outside	VaiUtilizarCaixa<0,5	1
	FilaEsperaCaixa	VaiUtilizarCaixa>=0,5	1

O atributo *Quantity* indica o número de entidades que deverá seguir o caminho definido pelo conetor. Essa funcionalidade não será explorada neste trabalho pelo que deverá ser mantida a quantidade prefinida — valor 1.

Diagrama da entidade Funcionário

Porque os atributos das atividades já foram definidos na entidade cliente, o processo de construção do diagrama do funcionário será bastante mais celere. No que diz respeito às atividades, o utilizador apenas terá de indicar o nome das

atividades, sendo os atributos comuns preenchidos de forma automática. De todos os atributos apenas o atributo *Assignment* não será preenchido. Este atributo deverá ser utilizado para a atribuição de valores aos atributos da entidade. Na Figura 4.28 está apresentado o diagrama do funcionário.

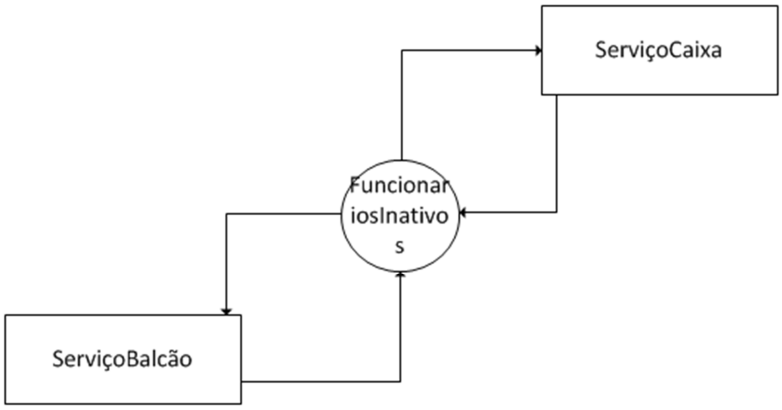


Figura 4.28: Diagrama da entidade Funcionário

Como podemos observar na Figura 4.28, os funcionários poderão estar ativos a prestar serviço no caixa, no balcão, ou então poderão estar inativos. Após a conclusão de um serviço, o funcionário volta ao seu estado inicial. Na Tabela 4.13 estão apresentados os valores dos atributos das filas de espera da entidade Funcionário.

Tabela 4.13: Valores dos atributos das filas da entidade Funcionário

Name	***Type***	***Quantity***	***Priority Field***
FuncionariosInativos	FIFO	2	

Uma vez que os atributos das duas atividades em que os funcionários estão envolvidos já foram definidos na entidade cliente, e porque nesta entidade não existem atividades com destinos múltiplos, não é necessário indicar o valor de mais nenhum atributo.

Diagrama global da simulação

Conforme se poderá observar no diagrama global apresentado na Figura 4.29, o ciclo do cliente está representado a preto, o ciclo do funcionário a azul, e o ciclo da porta, a verde. A utilização de cores diferentes permite delimitar o percurso de cada entidade.

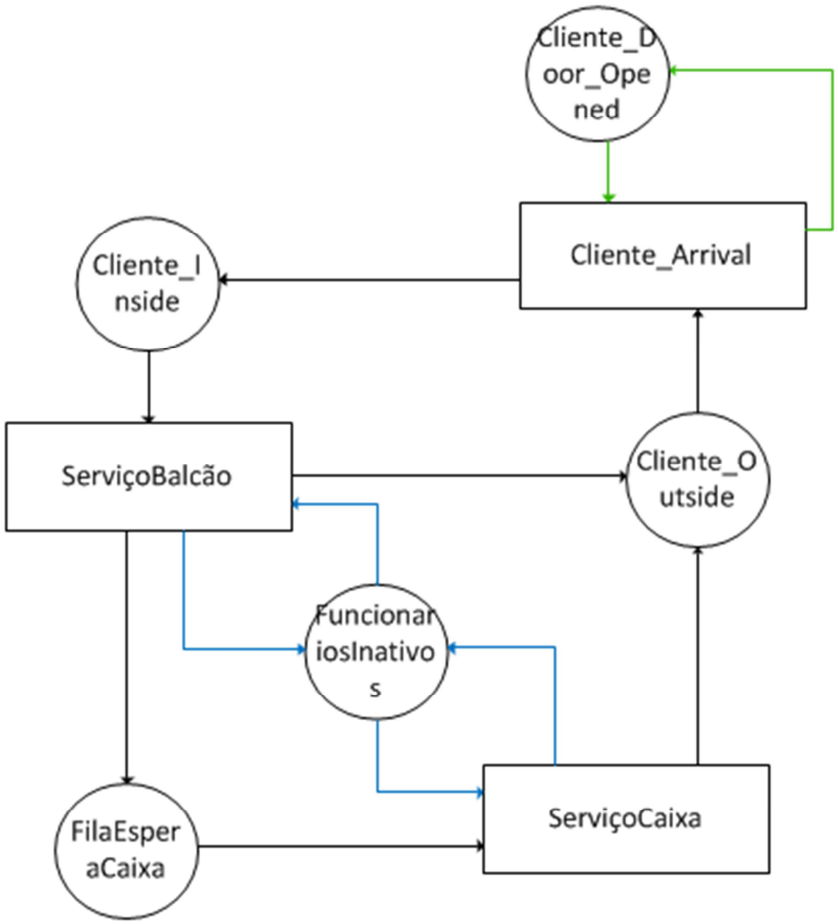


Figura 4.29: Diagrama global da agência bancária

4.3.3 Resultados da simulação

Na Figura 4.30 estão apresentadas as estatísticas da simulação, no instante de tempo 1000.

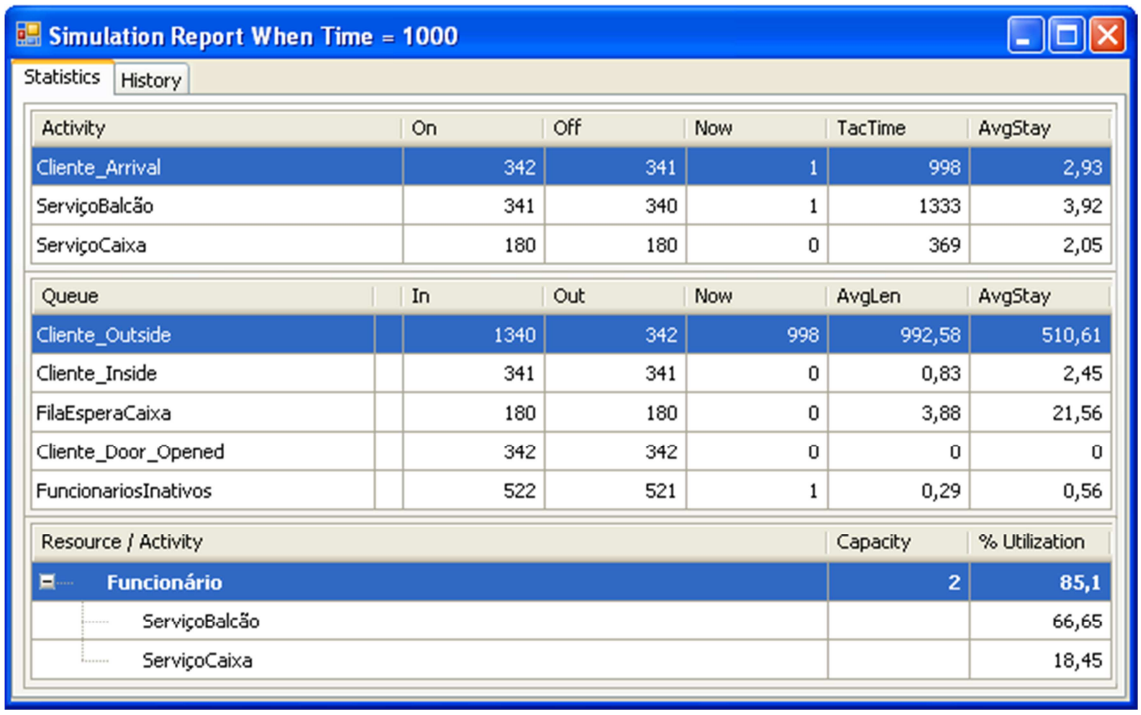


Figura 4.30: Estatísticas da simulação da agência bancária

4.4 Caso 3: Bar

O caso que a seguir passaremos a demonstrar pretende simular o funcionamento de um bar, onde os clientes se deslocam para beber. O objetivo da simulação é determinar se apenas um funcionário é suficiente para assegurar o serviço do bar.

4.4.1 Descrição do sistema

Os clientes, com vontade diferente de beber, chegam do exterior ao bar onde ficam à espera de serem atendidos. Sempre que um cliente é atendido, o *barman* pega num copo limpo e enche-o, ficando o cliente pronto a beber. Depois de beber e dependendo da sua sede, os clientes poderão aguardar que lhe encham novamente o copo, ou abandonar o bar e voltar ao exterior. Sempre que um cliente bebe, a sua vontade de beber decresce uma unidade. Os copos sujos serão, oportunamente, limpos em lotes de três unidades.

De acordo com o levantamento efetuado, os clientes chegam ao bar de acordo com uma distribuição exponencial negativa de média 20. A duração da atividade Encher é caracterizada de acordo com uma distribuição normal de média 6 e desvio padrão 1, e a atividade Beber, de acordo com uma distribuição uniforme no domínio [5, 10]. A atividade Lavar demora sempre 5 unidades de tempo. O bar entra em funcionamento com 12 copos, que se encontram inicialmente limpos.

4.4.2 Modelação do sistema

Uma vez que Cliente é uma entidade externa (Figura 4.31), a simulação será composta por quatro entidades: Cliente, Porta, Copo e Barman. — Por uma questão de normalização, os nomes dos objetos criados automaticamente pela ferramenta foram alterados para o seu equivalente em Português.

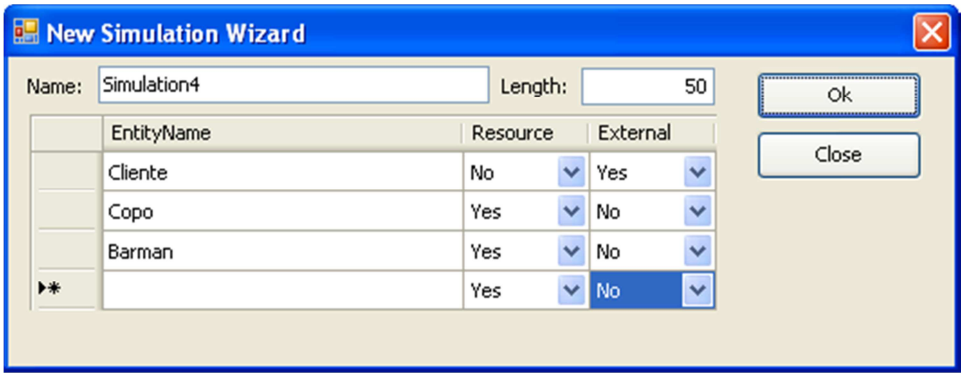


Figura 4.31: Assistente para a criação da simulação do bar

Diagrama da entidade Cliente

Conforme apresentado na Figura 4.32, os clientes chegam do exterior (1) e juntam-se à fila de espera (2) onde aguardam a vez de serem atendidos. Chegada a sua vez, o seu copo é cheio (3) e este encontra-se pronto para beber (4). Caso o cliente ainda tenha vontade de beber, este junta-se novamente à fila de espera, onde aguardará novamente a vez de ser atendido (5). Quando já não tiver sede, este abandona o bar e regressa ao exterior (6).

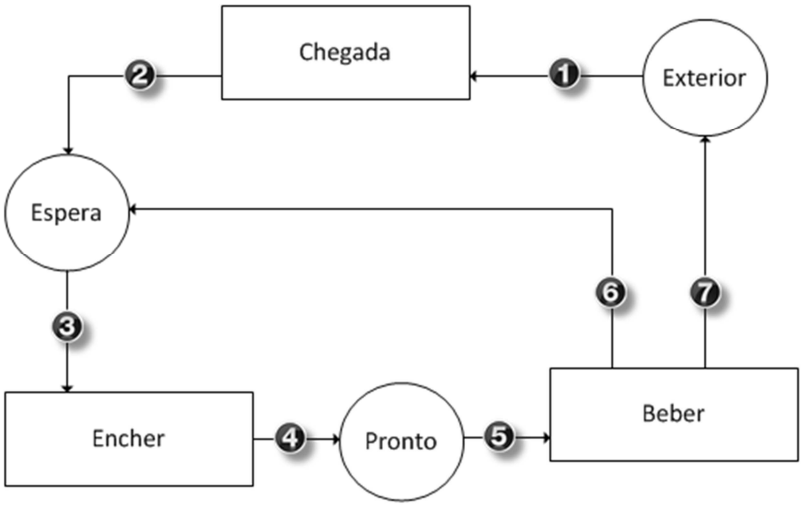


Figura 4.32: Diagrama da entidade Cliente

Os atributos das filas e das atividades da entidade Cliente estão representados, respetivamente, na Tabela 4.14 e Tabela 4.15.

Tabela 4.14: Atributos das filas da entidade Cliente

Name	*Type*	*Quantity*	*Priority Field*
Espera	FIFO	0	
Exterior	FIFO	1000	
Pronto	FIFO	0	

Para armazenar a vontade de beber, na entidade Cliente será criado o atributo Sede (Figura 4.33).

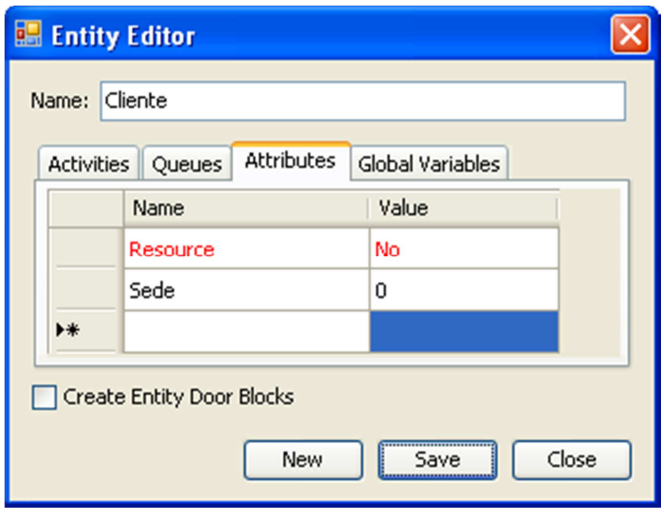


Figura 4.33: Inserção do atributo Sede na entidade Cliente

Como se poderá observar na Tabela 4.15 o valor do atributo será definido quando o cliente chega do exterior e será atualizado sempre que o cliente bebe. As condições de seleção de destino estão apresentadas na Tabela 4.16.

Tabela 4.15: Atributos das atividades da entidade Cliente

Name	*Priority*	*Duration*	*Assignment*
Chegada	1	Exponential(20)	Sede=Uniform(1,4)
Encher	1	Normal(6,1)	
Beber	1	Uniform(5,10)	Sede=Sede-1

Tabela 4.16: Condições de seleção de destino do cliente do bar

Origem	**Destino**	**Atributos**	
		Condition	***Quantity***
Beber	Espera	Sede>0	1
	Exterior	Sede<=0	1

Diagrama da entidade Copo

Os copos limpos (1) são cheios (2) ficando prontos a beber (3). Depois de bebidos (4) estes ficam sujos. Depois de lavados (5) os copos ficam novamente limpos (6) e prontos a usar.

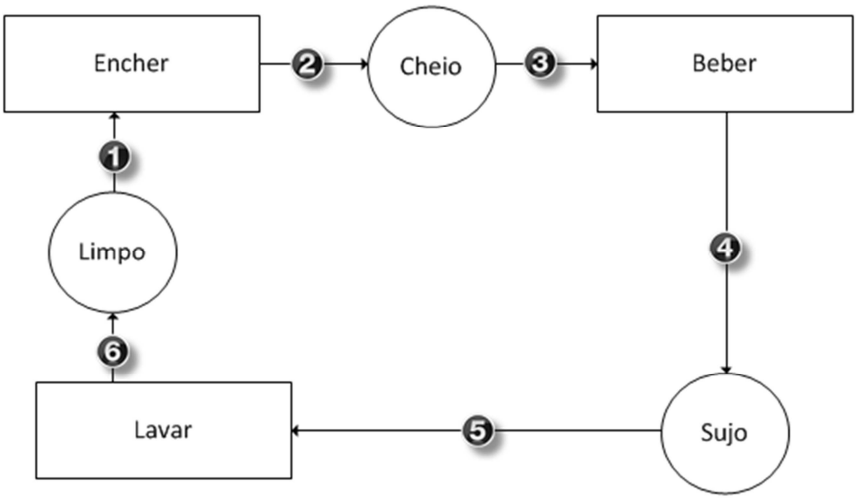


Figura 4.34: Diagrama da entidade Copo

Os atributos das filas e das atividades da entidade Copo estão apresentados, respetivamente na Tabela 4.17 e na Tabela 4.18. Uma vez que os copos são lavados em lotes de três unidades, é necessário indicar essa informação nos ligadores que se encontram ligados à atividade Lavar (Tabela 4.19).

Tabela 4.17: Atributos das filas da entidade Copo

Name	*Type*	*Quantity*	*Priority Field*
Limpo	FIFO	12	
Cheio	FIFO	0	
Sujo	FIFO	0	

Tabela 4.18: Atributos das atividades da entidade Copo

Name	*Priority*	*Duration*	*Assignment*
Lavar	1	5	

Estão omissos os atributos das atividades Encher e Beber, uma vez que estes já foram definidos na entidade Cliente (Tabela 4.15).

Tabela 4.19: Atributos dos ligadores da atividade Lavar

Origem	**Destino**	**Atributos**	
		Condition	***Quantity***
Sujo	Lavar		3
Lavar	Limpo		3

Diagrama da entidade Barman

O ciclo de atividade do Barman é bastante simples. Os barmans, quando não se encontram parados, encontram-se a encher, ou a lavar copos (Figura 4.35). Os atributos das filas de entidade Barman estão representados na Tabela 4.20. Uma vez que os atributos das atividades Encher e Lavar já foram definidos nas entidades Cliente (Tabela 4.15) e Copo (Tabela 4.17), estes não serão aqui apresentados.

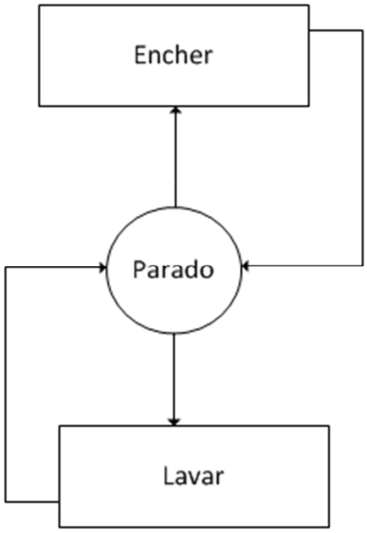


Figura 4.35: Diagrama da entidade Barman

Tabela 4.20: Atributos das filas da entidade Barman

Name	***Type***	***Quantity***	***Priority Field***
Parado	FIFO	1	

Diagrama global da simulação

No diagrama global da simulação do bar, apresentado na Figura 4.36, é utilizado o preto para representar o ciclo do cliente, o azul para representar o ciclo do copo, e o vermelho para representar o ciclo do *barman*. O ciclo da entidade fictícia, porta, está representado a verde.

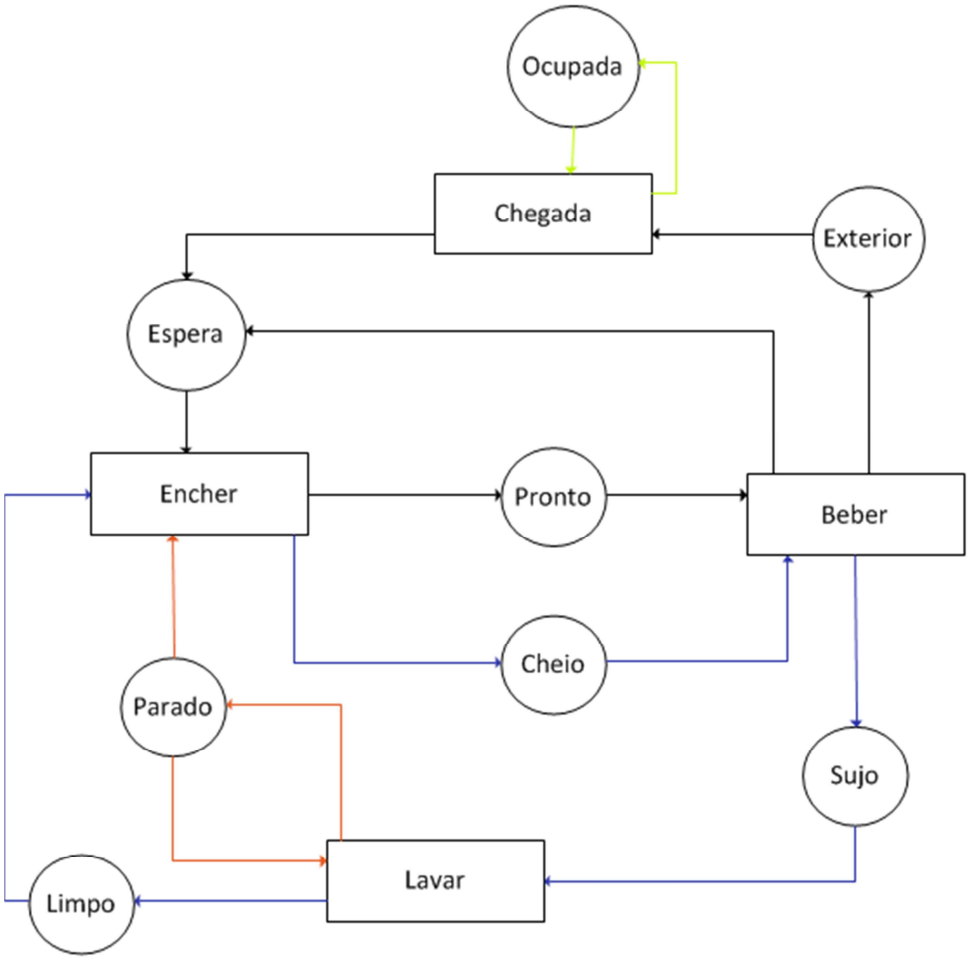
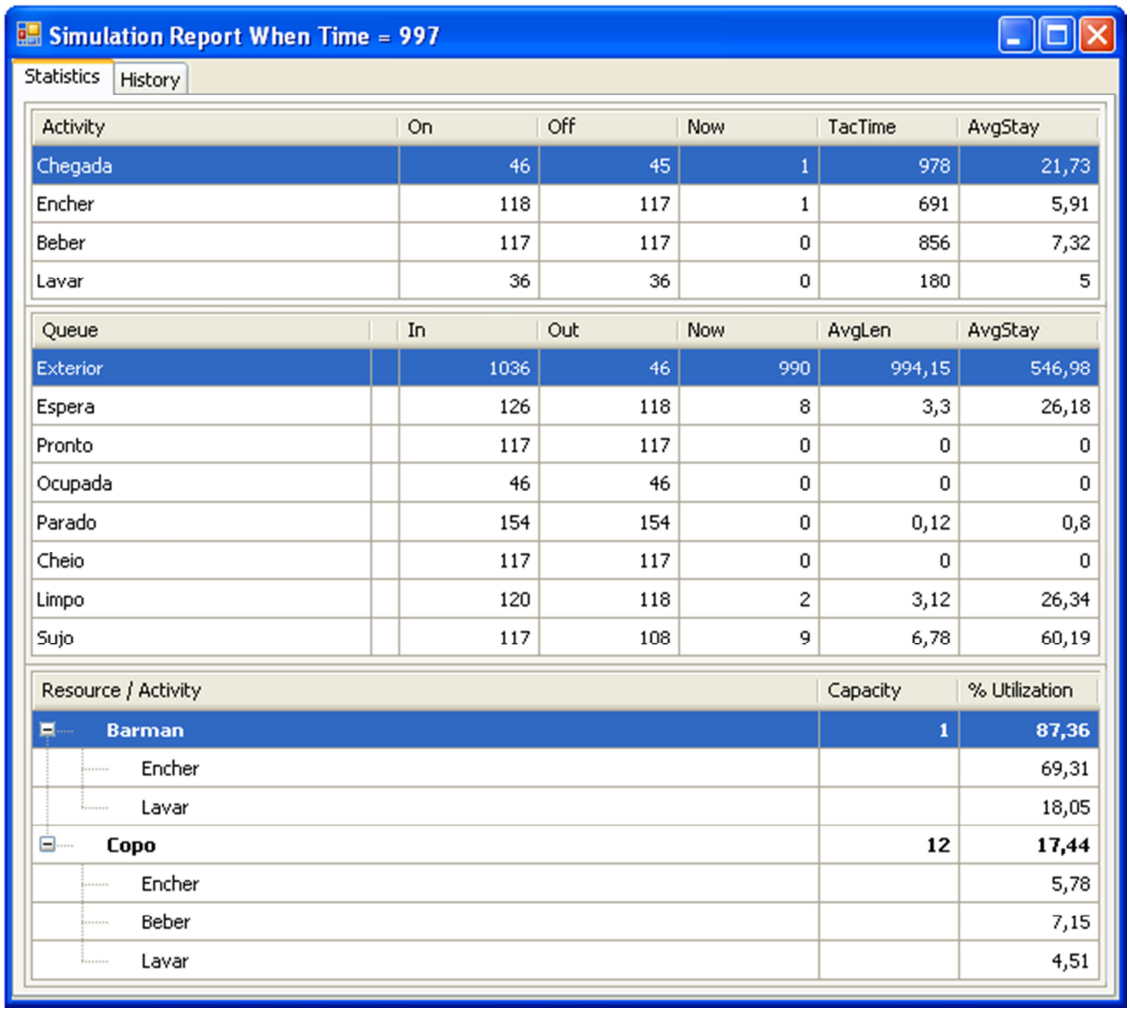


Figura 4.36: Diagrama global da simulação do bar

4.4.3 Resultados da simulação

Na Figura 4.37 estão apresentadas as estatísticas da simulação no instante de tempo 997.



Activity	On	Off	Now	TacTime	AvgStay
Chegada	46	45	1	978	21,73
Encher	118	117	1	691	5,91
Beber	117	117	0	856	7,32
Lavar	36	36	0	180	5

Queue	In	Out	Now	AvgLen	AvgStay
Exterior	1036	46	990	994,15	546,98
Espera	126	118	8	3,3	26,18
Pronto	117	117	0	0	0
Ocupada	46	46	0	0	0
Parado	154	154	0	0,12	0,8
Cheio	117	117	0	0	0
Limpo	120	118	2	3,12	26,34
Sujo	117	108	9	6,78	60,19

Resource / Activity	Capacity	% Utilization
Barman	**1**	**87,36**
Encher		69,31
Lavar		18,05
Copo	**12**	**17,44**
Encher		5,78
Beber		7,15
Lavar		4,51

Figura 4.37: Estatísticas da simulação do bar

5. Conclusões

5.1 Trabalho Realizado

Na busca da resposta à questão de investigação, o objetivo principal deste trabalho consistiu no desenvolvimento do protótipo de uma ferramenta de simulação, que permitisse, não só, a representação de sistemas no formalismo baseado no conceito de atividade; como também, a geração automática do programa de simulação correspondente, através da simples interpretação dos diagramas construídos. Sendo que o programa gerado deveria poder ser aumentado, modificado ou corrigido, e de cuja execução deveria resultar o relatório de simulação.

Das alternativas existentes, para a implementação do protótipo, optou-se pela integração do mesmo num editor gráfico já existente, o Visio 2010. Uma vez que deixávamos de ter que nos preocupar com a criação do editor gráfico, esta

alternativa fez com que focássemos apenas os aspetos mais essenciais do projeto e que passavam pela construção do protótipo de uma ferramenta para simulação.

O protótipo construído, em forma de *add-on* para o Visio, permite a qualquer utilizador, com poucos ou nenhuns conhecimentos de programação, a construção de programas de simulação. Estes apenas necessitam de estarem familiarizados com o formalismo de representação utilizado na representação de sistemas no paradigma de atividades.

O princípio de funcionamento do protótipo é bastante simples. Para cada simulação é criado um diagrama, com várias páginas. O ciclo de cada entidade é representado em páginas diferentes do diagrama. Representado o ciclo de todas as entidades, pela sobreposição dos ciclos de todas as entidades, é gerado o diagrama global da simulação. Uma vez gerado, o diagrama global é interpretado, *shape* a *shape*, sendo depois convertido em código fonte. A execução do código-fonte, também designado por programa de simulação, irá produzir o relatório da simulação.

Depois de instalado o protótipo, é adicionado ao Visio um novo menu, que conterá todas as funcionalidades disponibilizadas; e um novo *stencil*, que conterá as *shapes* que compõem os DCA. Através do menu disponibilizado, os utilizadores poderão construir novas simulações, adicionar ou editar entidades já existentes, gerar o diagrama global, e criar o programa de simulação. Para auxiliar o utilizador no processo de modelação, e de manipulação de código, foi também criado um depurador de erros, que alertará o utilizador para a existência de erros, sempre que estes forem detetados.

O protótipo desenvolvido foi submetido a três casos de estudo, de diferente complexidade, normalmente utilizados no ensino de simulação (máquinas semiautomáticas, agência bancária, e bar). Conforme aqui foi demonstrado, não

foram detetados problemas na modelação, nem na geração do código fonte, dos casos de estudo analisados. Verificou-se também que os resultados produzidos estavam de acordo com as fórmulas utilizadas.

5.2 Contribuições Técnicas e Científicas

Com o protótipo desenvolvido pretendeu-se, não só, satisfazer uma necessidade existente, como também recuperar um paradigma, tradicionalmente, pouco explorado — o paradigma de atividades. Sendo o número de ferramentas, que assenta neste paradigma, praticamente inexistente.

Através da geração automática dos programas pretendeu-se fazer chegar a simulação a outras plateias, principalmente aquelas que têm poucos, ou nenhuns, conhecimentos de programação. Uma vez que o código dos programas continua a existir na simulação, podendo ser alterado, os utilizadores mais experientes poderão acrescentar funcionalidades à simulação, seja para a enriquecer com novos métodos, ou para fazer frente a problemas mais complexos.

Dado que o protótipo desenvolvido utiliza o formalismo de representação no seu estado mais puro, não sendo acrescentados novos símbolos para lidar com questões que apenas têm a ver com a programação do modelo; e porque, ao não exigir conhecimentos prévios de programação, não esconde o código fonte que representa o sistema modelado, acreditamos que esta ferramenta possa ser utilizada no ensino de simulação, independentemente da abordagem de ensino adotada.

No desenvolvimento deste trabalho recorreu-se à utilização de tecnologia já existente, mas utilizada em outros contextos. A modelação recorre a um editor gráfico, genérico, conhecido; o código fonte gerado é produzido em VBA, sendo a sua execução, e depuração, efetuada através de bibliotecas disponibilizadas para o efeito, pela Microsoft.

5.3 Perspetivas de Trabalho Futuro

Sendo o protótipo, aqui apresentado, fruto de um trabalho de dissertação, existe ainda algum trabalho que, por restrições de tempo, não foi feito, e que necessita de ser efetuado. O trabalho que falta efetuar pretende, não só, enriquecer o protótipo com novas funcionalidades, mas sobretudo, demonstrar que este possui utilidade no ensino introdutório de simulação. De seguida irá perspetivar-se o trabalho que falta realizar.

Apesar do funcionamento do protótipo ter sido demonstrado, através da resolução de casos de estudo utilizados no ensino de simulação, falta avaliar o seu funcionamento em contexto de sala de aula — fase 5 da metodologia. Esta fase não foi considerada uma vez que de acordo com o plano de trabalhos, o protótipo apenas estaria concluído depois de a atividade letiva da disciplina de simulação, onde o protótipo poderia ser experimentado, já ter terminado.

Na execução do programa de simulação são recolhidos os valores estatísticos referentes à execução de apenas uma instância do problema. Apesar de esta limitação não ter impacto nas simulações que não recorram a fenómenos aleatórios, a execução de várias instâncias do problema é vital para determinar o nível de confiança dos fenómenos aleatórios nas simulações estocásticas. O que se pretende é que o protótipo possibilite a execução de várias replicações, para que a análise estatística da performance do sistema incida sobre as replicações e não se baseie apenas numa replicação.

Outro dos aspetos a melhorar no protótipo é a informação apresentada nos diagramas. Atualmente, apenas é apresentado o nome dos objetos que intervêm na simulação, o que obriga, sempre que o utilizador deseja saber o valor de algum atributo, a abrir a janela de propriedades da *shape* correspondente. Nos conetores

que ligam as atividades, que possuem múltiplas filas de espera de destino, falta indicar a condição de seleção de destino.

Uma das funcionalidades a implementar é a exportação do relatório de simulação para o formato *Comma-Separated Values* (CSV). Esta funcionalidade irá permitir que os resultados possam ser analisados através do Microsoft Excel, ou através de outras ferramentas que importem ficheiros nesse formato.

O facto de o protótipo ser disponibilizado em forma de *add-on* para o Visio, obriga a que o cliente possua, não só, essa aplicação como também o sistema operativo Windows — visto que esse é o único sistema operativo onde o Visio funciona. Dependendo do futuro que se pretenda dar à ferramenta, poderá ser necessário rever a forma como essa será disponibilizada, o que poderá obrigar à construção de um editor gráfico próprio.

Pretende-se também que o protótipo aqui apresentado possa vir a ser integrado numa ferramenta de ensino de simulação já existente, o Visio Basic for Simulations (VBS). Uma vez que o VBS é baseado no paradigma de acontecimentos, a integração do protótipo com esta ferramenta irá permitir obter uma ferramenta global de simulação, baseada em qualquer um dos dois paradigmas — acontecimentos ou atividades.

Referências

Altiok, T., Kelton, W. D., L'Ecuyer, P., Nelson, B. L., Schmeiser, B. W., Schriber, T. J., . . . Wilson, J. R. (2001). *Various ways academics teach simulation: are they all appropriate?* Paper presented at 33nd Conference on Winter Simulation, Arlington, Virginia.

April, J., Better, M., Glover, F., Kelly, J., & Laguna, M. (2006). *Enhancing Business Process Management with Simulation Optimization.* Paper presented at 38th Conference on Winter Simulation, Monterey, California.

Banks, J., John S. Carson, I., Nelson, B. L., & Nicol, D. M. (2010). *Discrete-Event System Simulation* (5th ed.): Prentice Hall.

Born, R. G. (2003). *Teaching Discrete Event Simulation to Business Students: The Alpha and Omega.* Paper presented at 35th Winter Simulation Conference, New Orleans, LA.

Carson, J. S. (1993). *Modeling and simulation worldviews.* Paper presented at 25th Conference on Winter Simulation, Los Angeles, California.

Carson, J. S. (2004). *Introduction to modeling and simulation.* Paper presented at 36th Conference on Winter simulation, Washington, D.C.

Chase, R., Jacobs, F. R., & Aquilano, N. (2005). *Operations Management for Competitive Advantage* (11th ed.): McGraw-Hill.

Clementson, A. T. (1986). *Simulation with activities using C.A.P.S/E.C.S.L (the British approach to discrete-event simulation).* Paper presented at 18th Conference on Winter Simulation, Washington, D.C., United States.

Dias, L. M. S. (2005). *Modelação Automática Interactiva de Simulação.* Tese de Doutoramento, Universidade do Minho.

Dias, L. M. S., Pereira, G. A. B., & Rodrigues, A. J. M. G. (2002). *Towards simplicity in modelling for simulation.* Paper presented at Operational Research Society - Simulation Study Group. First biennial Simulation Study Group Workshop, Birmingham, UK.

Dias, L. M. S., Pereira, G. A. B., & Rodrigues, A. J. M. G. (2006). *Activity Based Modelling With Automatic Prototype Generation Of Process Based Arena Models.* Paper presented at 2nd European Modeling and Simulation Symposium, Barcelona, Spain.

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

Dias, L. M. S., Rodrigues, A. J. M. G., & Pereira, G. A. B. (2005). *An Activity Oriented Visual Modelling Language With Automatic Translation to*

Different Paradigms. Paper presented at 19th European Conference on Modelling and Simulation, Riga, Latvia.

Goldsman, D. (2007). *Introduction to simulation.* Paper presented at 39th Conference on Winter Simulation, Washington D.C.

Herper, H., & Ståhl, I. (1999). *Micro-GPSS on the Web and for Windows: a tool for introduction to simulation in high schools.* Paper presented at Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future - Volume 1, Phoenix, Arizona, United States.

Hlupic, V. (2000). *Simulation software: an Operational Research Society survey of academic and industrial users.* Paper presented at 32nd Winter Simulation Conference, Orlando, FL.

Hutchinson, G. K. (1975). Introduction to the use of activity cycles as a basis for system's decomposition and simulation. *SIGSIM Simul. Dig., 7*(1), 15-20. doi: 10.1145/1102722.1102725

Ingalls, R. G. (2008). *Introduction to simulation.* Paper presented at 40th Conference on Winter Simulation, Miami, Florida.

K. Preston White, J., & Ingalls, R. G. (2009). *Introduction to simulation.* Paper presented at Winter Simulation Conference, Austin, Texas.

Kang, D., & Choi, B. K. (2011). The extended activity cycle diagram and its generality. *Simulation Modelling Practice and Theory, 19*(2), 785-800. doi: 10.1016/j.simpat.2010.11.004

Kelton, W. D., Sadowski, R. P., & Sadowski, D. A. (2001). *Simulation with Arena* (2nd ed.): Mcgraw-Hill

Khoshnevis, B. (1994). *Discrete Systems Simulation*: McGraw-Hill.

Lackner, M. R. (1962). *Toward a general simulation capability.* Paper presented at Proceedings of the May 1-3, 1962, Spring Joint Computer Conference, San Francisco, California.

Lorenz, P., & Schriber, T. J. (1996). *Teaching introductory simulation in 1996: from the first assignment to the final presentation.* Paper presented at 28th Conference on Winter Simulation, Coronado, California.

Nance, R. E. (1995). *Simulation programming languages: an abridged history.* Paper presented at 27th Conference on Winter Simulation, Arlington, Virginia, United States.

Nance, R. E. (2000). *Simulation education: past reflections and future directions.* Paper presented at 32nd Conference on Winter Simulation, Orlando, Florida.

Paul, R. J. (1993). *Activity Cycle Diagrams and the Three Phase Method.* Paper presented at 25th Winter Simulation Conference, Los Angeles, CA.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research.

Journal of Management Information Systems, 24(3), 45-78. doi: 10.2753/MIS0742-1222240302

Pereira, G. A. B., Dias, L. M. S., & Ferreira, B. L. S. (2009). *Flowchart Simulation – A Tool For The Automatic Generation Of Simulation Programs.* Paper presented at IX Congreso Galego de Estatística e Investigación de Operacións, Ourense, Spain.

Pereira, G. A. B., Dias, L. M. S., & Rocha, H. T. C. (2009). *Teaching Simulation Basics Through Flowchart Simulation the Event Scheduling World View.* Paper presented at The 6th International Mediterranean Modeling Multiconference, Tenerife, Spain.

Pidd, M. (1992). *Computer Simulation in Management Science* (3rd ed.): Wiley.

Pidd, M. (2004). *Simulation worldviews: so what?* Paper presented at 36th Conference on Winter Simulation, Washington, D.C.

Rodrigues, A. J. M. G. (1996). *Simulação*: Universidade do Minho.

Schriber, T. J., & Brunner, D. T. (2008). *Inside Discrete-Event Simulation Software: How It Works And Why It Matters.* Paper presented at Winter Simulation Conference, Miami, FL.

Shannon, R. E. (1998). *Introduction to the art and science of simulation.* Paper presented at 30th Conference on Winter Simulation, Washington, D.C.

Ståhl, I. (2000). *How Should We Teach Simulation?* Paper presented at 32nd Conference on Winter Simulation, Orlando, Florida.

Ståhl, I. (2007). *Teaching simulation to business students summary of 30 years' experience.* Paper presented at 39th Conference on Winter Simulation, Washington D.C.

Ståhl, I., Henriksen, J. O., Born, R. G., & Herper, H. (2011). *GPSS 50 Years Old, But Still Young.* Paper presented at 43rd Winter Simulation Conference, Phoenix, AZ.

Ståhl, I., Herper, H., Hill, R. R., Harmonosky, C. M., Donohue, J. M., & Kelton, W. D. (2003). *Teaching The Classics Of Simulation To Beginners (Panel).* Paper presented at 35th Winter Simulation Conference, New Orleans, LA.

Sulistio, A., Yeo, C. S., & Buyya, R. (2004). A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Softw. Pract. Exper., 34*(7), 653-673. doi: 10.1002/spe.585

Turban, E., Sharda, R., & Delen, D. (2010). *Decision Support and Business Intelligence Systems* (9th ed.): Prentice Hall.

Zee, D.-J. V. d., & Vorst, J. G. A. J. V. d. (2007). *Guiding principles for conceptual model creation in manufacturing simulation.* Paper presented at 39th Conference on Winter Simulation, Washington, D.C.

Zhou, M., Son, Y. J., & Chen, Z. (2004). *Knowledge representation for conceptual simulation modeling.* Paper presented at 36th Conference on Winter Simulation, Washington, D.C.

Bibliografia

Abran, A., & Moore, J. W. (2004). Guide to the Software Engineering Body of Knowledge (SWEBOK).

Aurum, A., & Wohlin, C. (2005). Engineering and Managing Software Requirements: Springer.

Berndtsson, M., Olsson, B., Hansson, J., & Lundell, B. (2008). *Thesis Projects A Guide for Students in Computer Science and Information Systems* (2nd ed.): Springer.

Brocke, J. v., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., & Cleven, A. (2009). *Reconstructing The Giant: On The Importance Of Rigour In Documenting The Literature Search Process*. Paper presented at 17th European Conference on Information Systems, Verona, Italy.

Hevner, A., & Chatterjee, S. (2010). Design Research in Information Systems Theory and Practice: Springer.

Knuth, D. E. (1981). The Art Of Computer Programming, Volume 2: Seminumerical Algorithms, 3/E (2nd ed. Vol. 2): Addison-Wesley.

Microsoft. (2010). *Visio 2010 SDK* Acedido através de http://msdn.microsoft.com/en-us/library/ff758690.aspx

Parker, D. J. (2010). Microsoft Visio 2010 Business Process Diagramming and Validation: Packt Publishing.

Troelsen, A., & Agarwal, V. V. (2010). *Pro VB 2010 and the .NET 4.0 Platform*: Apress.

Anexos

Anexo 1. Requisitos desejados para as ferramentas de simulação

#ID	Descrição do Requisito
1 – FACILIDADE DE APRENDIZAGEM	
1.1	A ferramenta não deverá exigir conhecimentos prévios de programação, excetuando alguns possíveis conhecimentos, elementares, de Visual Basic.
1.2	Uma vez que se pretende que os alunos se foquem na modelação e não nos detalhes de sintaxe, a ferramenta e a sua linguagem deverão ser fáceis de aprender, para que estes não tenham que aprender um novo conceito sempre que tenha que fazer alguma coisa.
1.3	A ferramenta deverá ser divertida de aprender, permitindo fazer coisas interessantes depois de períodos curtos de aprendizagem.
1.4	A ferramenta deverá restringir-se aos detalhes essenciais de programação, negligenciando aspetos como *case sensitive*.
1.5	Para evitar que a ferramenta se torne difícil de compreender (devido á existência de funções diferentes para se efetuar a mesma coisa), não deverão ser tomadas precauções para lidar com a compatibilidade com diferentes versões da ferramenta. Essa compatibilidade é apenas do interesse dos utilizadores mais velhos, que já aprenderam os conceitos elementares de simulação.
1.6	A ferramenta deverá disponibilizar o resultado da simulação de forma automática, visto que os alunos menos experientes poderão não saber quais as estatísticas relevantes para a análise do resultado da simulação.
1.7	A ferramenta deverá ser simples de forma a permitir que os seus conteúdos possam ser cobertos de forma pedagógica, e deverá estar dotada de vários exemplos e de um manual de utilização, que não deverá ser muito extenso e exigir o recurso a manuais não pedagógicos.
1.8	A ferramenta deverá facilitar o ensino nos laboratórios de simulação e estimular a autoaprendizagem.
1.9	Para facilitar a autoaprendizagem a ferramenta deverá estar dotada de exemplos práticos, tutoriais e de um sistema de ajuda.
1.10	Todos os elementos que compõem a ferramenta deverão estar também visíveis mesmo quando se utilize um videoprojector.
1.11	Por razões pedagógicas, deverá existir uma forma mais simples para a invocação de funções com aspetos mais difíceis de compreender (ex.: as sementes da geração de números aleatórios).
1.12	A ferramenta não deverá exigir conhecimentos profundos de modelação, permitindo que os alunos iniciados, ou mais experientes, possam

efetuar modelos de complexidade crescente. Deverá ser também possível parar a simulação, depois de um período de tempo, após um determinado número de clientes, ou em ambas as situações.

2 – FACILIDADE DE MODELAÇÃO

2.1 O processo de modelação deverá ser bastante simples e basear-se em interfaces gráficos, para que o aluno, com a ajuda do rato, possa selecionar os blocos (símbolos) que pretende usar no modelo.

2.2 A seleção dos blocos deverá ser efetuada através de técnicas "*drag-and-drop*", ou "*point-and-click*". Na primeira técnica, o utilizador seleciona primeiro o bloco, arrastando-o depois para o local pretendido. Na segunda, o utilizador clica no bloco pretendido, sendo este automaticamente colocado no modelo.

2.3 Os blocos disponíveis deverão ser os estritamente essenciais.

2.4 A especificação dos parâmetros de um determinado bloco deverá ser efetuada através de uma janela própria, depois de ter sido identificado o bloco.

2.5 Deverá ser também possível criar manualmente o programa, sem recorrer à modelação. A partir do código introduzido a ferramenta deverá criar o respetivo modelo, e vice-versa.

2.6 É importante que código gerado não seja demasiado longo.

3 – FACILIDADE DE LEITURA DOS RESULTADOS

3.1 As estatísticas da simulação deverão ser fáceis de compreender. Para que estas possam ser compreendidas pelos alunos menos experientes, a ferramenta deverá apenas apresentar as estatísticas mais elementares da simulação.

3.2 O código fonte, gerado pela ferramenta, deverá ser também de fácil compreensão.

3.3 Deverá ser possível complementar o código fonte gerado pela ferramenta.

3.4 Para que as estatísticas sejam mais fáceis de compreender, deverão ser utilizados gráficos e histogramas.

3.5 Para facilitar a validação do programa e perceber como o este funciona é essencial a existência de um mecanismo de animação. A animação poderá resumir-se à movimentação das entidades por entre os blocos que compõem os diagramas.

4 – FACILIDADE DA CONDUÇÃO DE EXPERIÊNCIAS

4.1 Uma vez que é importante que os alunos percebam que os programas de simulação têm ser executados várias vezes, devido à ocorrência dos fenómenos aleatórios, é importante que a ferramenta permita, de uma forma simples, a replicação das experiências.

4.2 Os resultados estatísticos deverão ter em consideração o número de replicações efetuadas, apresentado também o intervalo de confiança.

5 – PROGRAMAÇÃO EFICIENTE

5.1 Para minimizar o risco de erros lógicos, assim como para evitar que o aluno perca demasiado tempo na depuração de erros, a ferramenta deverá possuir um depurador de erros, onde sejam facilmente visíveis os erros lógicos cometidos.

5.2 Para evitar situações difíceis de depurar, a ferramenta não deverá possuir palavras reservadas. Um aluno que desconheça que está a usar uma palavra reservada no nome de uma variável, como por exemplo *random*, poderá não se aperceber porque é que o programa não se está a comportar como seria esperado.

5.3 As mensagens de erro do depurador deverão ser claras, e não ambíguas.

Deverá existir um mecanismo para validar e verificar a simulação, passo-a-passo.

5.4 É importante que o código gerado seja totalmente depurável e que o mecanismo interno possa ser objeto de escrutínio científico.

6 – EFICIÊNCIA DE EXECUÇÃO

6.1 Apesar de a eficiência não ser um aspeto vital, a execução da simulação não deverá demorar demasiado tempo para não desencorajar os alunos a replicarem a experiência. A replicação é um aspeto importante para determinação do intervalo de confiança.

7 – DISPONIBILIDADE

7.1 A ferramenta de simulação deverá funcionar da mesma forma que as aplicações tradicionais podendo ser instalada em qualquer computador, para que o código gerado num computador possa ser testado noutro.

7.2 A ferramenta deverá estar disponível a um preço bastante acessível, ou de forma gratuita, para que os alunos possam adquirir a sua própria cópia da ferramenta.

7.3 É desejável que a ferramenta esteja disponível na Internet, para garantir que se está a usar a versão mais recente, para que o aluno, após deixar a universidade, possa continuar a usar a ferramenta que estava acostumado, ou para simplesmente experimentar a ferramenta, antes

de a descarregar.

7.4 É recomendável que existam bastantes livros sobre a ferramenta, assim como exemplos, para que se possa explorar a ferramenta.

8 – LIGAÇÃO A OUTRAS FERRAMENTAS

8.1 Por vezes é também desejável que os alunos, após terem experimentado uma ferramenta didática, consigam rapidamente passar para uma ferramenta mais amplamente utilizada, tipicamente uma ferramenta comercial de simulação.

Anexo 2. Programa de simulação do Caso 2

```
Public simulation As Simulation = Nothing
Public entityCliente As EntityType = Nothing
Public queueCliente_Outside As Queue = Nothing
Public queueCliente_Inside As Queue = Nothing
Public queueFilaEsperaCaixa As Queue = Nothing
Public entityCliente_Door As EntityType = Nothing
Public queueCliente_Door_Opened As Queue = Nothing
Public entityFuncionário As EntityType = Nothing
Public queueFuncionariosInativos As Queue = Nothing
Public WithEvents activityCliente_Arrival As Activity = Nothing
Public WithEvents activityServiçoBalcão As Activity = Nothing
Public WithEvents activityServiçoCaixa As Activity = Nothing

Sub Init()
    simulation = New Simulation()
    simulation.Length = 50
    entityCliente = simulation.CreateEntity("Cliente")
    entityCliente.IsResource = False
    entityCliente.Variables.Create("VaiUtilizarBalcao", 0)
    queueCliente_Outside = simulation.CreateQueue("Cliente_Outside",
Queue.TypeEnum.FirstInFirstOut, entityCliente)
    queueCliente_Inside = simulation.CreateQueue("Cliente_Inside",
Queue.TypeEnum.FirstInFirstOut, entityCliente)
    queueFilaEsperaCaixa = simulation.CreateQueue("FilaEsperaCaixa",
Queue.TypeEnum.FirstInFirstOut, entityCliente)
    entityCliente_Door = simulation.CreateEntity("Cliente_Door")
    entityCliente_Door.IsResource = False
    queueCliente_Door_Opened = simulation.CreateQueue("Cliente_Door_Opened",
Queue.TypeEnum.FirstInFirstOut, entityCliente_Door)
    entityFuncionário = simulation.CreateEntity("Funcionário")
    queueFuncionariosInativos = simulation.CreateQueue("FuncionariosInativos",
Queue.TypeEnum.FirstInFirstOut, entityFuncionário)
    activityCliente_Arrival = simulation.CreateActivity("Cliente_Arrival", 1)
    activityServiçoBalcão = simulation.CreateActivity("ServiçoBalcão", 1)
    activityServiçoCaixa = simulation.CreateActivity("ServiçoCaixa", 1)
    queueCliente_Outside.Insert(entityCliente.Generate(1000))
    queueCliente_Door_Opened.Insert(entityCliente_Door.Generate(1))
    queueFuncionariosInativos.Insert(entityFuncionário.Generate(2))
    activityCliente_Arrival.NeedResource(entityCliente, 1)
    activityCliente_Arrival.NeedResource(entityCliente_Door, 1)
    activityServiçoBalcão.NeedResource(entityCliente, 1)
    activityServiçoBalcão.NeedResource(entityFuncionário, 1)
    activityServiçoCaixa.NeedResource(entityCliente, 1)
    activityServiçoCaixa.NeedResource(entityFuncionário, 1)
End Sub

Sub Main()
    Init()
    Dim simulationEnd As Boolean = False

    If Not simulation.ResourcesDefined() Then
        MsgBox("There aren't enough resources to start the simulation.",
MsgBoxStyle.Exclamation)
        Exit Sub
    End If

    Do Until simulationEnd
        For Each a As Activity In simulation.Activities
            Do While a.CanStart()
```

```
                simulation.ScheduleNewJob(a, simulation.CurrentTime)
            Loop
        Next

        Dim nextJob As Job = simulation.GetNextJob()
        simulationEnd = (nextJob Is Nothing OrElse nextJob.ExecuteAt >
simulation.Length)

        If Not simulationEnd Then
            simulation.UpdateCurrentTime(nextJob.ExecuteAt)
            For Each j As Job In
simulation.GetJobsByExecutionTime(simulation.CurrentTime)
                simulation.ExecuteJob(j)
            Next
        End If
    Loop

    simulation.Report()
End Sub

Sub Cliente_Arrival_OnSchedule(sender As Activity, e As
Activity.ScheduleEventArgs) Handles activityCliente_Arrival.OnSchedule
    e.Resources(entityCliente).RemoveFrom = queueCliente_Outside
    e.Resources(entityCliente_Door).RemoveFrom = queueCliente_Door_Opened
    e.Duration = Random.Exponential(3)
End Sub

Sub Cliente_Arrival_OnExecute(sender As Job, e As Activity.ExecuteEventArgs)
Handles activityCliente_Arrival.OnExecute

    For Each item As Entity In sender.Entities(entityCliente)
        item.Variables("VaiUtilizarBalcao") = Random.Uniform(0, 1)
    Next

    e.Resources(entityCliente).InsertInto = queueCliente_Inside
    e.Resources(entityCliente_Door).InsertInto = queueCliente_Door_Opened
End Sub

Sub ServiçoBalcão_OnSchedule(sender As Activity, e As
Activity.ScheduleEventArgs) Handles activityServiçoBalcão.OnSchedule
    e.Resources(entityCliente).RemoveFrom = queueCliente_Inside
    e.Resources(entityFuncionário).RemoveFrom = queueFuncionariosInativos
    e.Duration = Random.Poisson(4)
End Sub

Sub ServiçoBalcão_OnExecute(sender As Job, e As Activity.ExecuteEventArgs)
Handles activityServiçoBalcão.OnExecute

    For Each item As Entity In sender.Entities(entityCliente)
        If item.Variables("VaiUtilizarBalcao") < 0.5 Then
            e.Resources(entityCliente).InsertInto = queueCliente_Outside
        Else
            e.Resources(entityCliente).InsertInto = queueFilaEsperaCaixa
        End If
    Next

    e.Resources(entityFuncionário).InsertInto = queueFuncionariosInativos
End Sub

Sub ServiçoCaixa_OnSchedule(sender As Activity, e As Activity.ScheduleEventArgs)
Handles activityServiçoCaixa.OnSchedule
    e.Resources(entityCliente).RemoveFrom = queueFilaEsperaCaixa
```

```
    e.Resources(entityFuncionário).RemoveFrom = queueFuncionariosInativos
    e.Duration = Random.Poisson(2)
End Sub

Sub ServiçoCaixa_OnExecute(sender As Job, e As Activity.ExecuteEventArgs)
Handles activityServiçoCaixa.OnExecute
    e.Resources(entityCliente).InsertInto = queueCliente_Outside
    e.Resources(entityFuncionário).InsertInto = queueFuncionariosInativos
End Sub
```

Anexo 3. Programa da simulação do Caso 3

```
Public simulation As Simulation = Nothing
Public entityCliente As EntityType = Nothing
Public queueExterior As Queue = Nothing
Public queueEspera As Queue = Nothing
Public queuePronto As Queue = Nothing
Public entityPorta As EntityType = Nothing
Public queueOcupada As Queue = Nothing
Public entityBarman As EntityType = Nothing
Public queueParado As Queue = Nothing
Public entityCopo As EntityType = Nothing
Public queueCheio As Queue = Nothing
Public queueLimpo As Queue = Nothing
Public queueSujo As Queue = Nothing
Public WithEvents activityChegada As Activity = Nothing
Public WithEvents activityEncher As Activity = Nothing
Public WithEvents activityBeber As Activity = Nothing
Public WithEvents activityLavar As Activity = Nothing

Sub Init()
    simulation = New Simulation()
    simulation.Length = 50
    entityCliente = simulation.CreateEntity("Cliente")
    entityCliente.IsResource = False
    entityCliente.Variables.Create("Sede", 0)
    queueExterior = simulation.CreateQueue("Exterior",
Queue.TypeEnum.FirstInFirstOut, entityCliente)
    queueEspera = simulation.CreateQueue("Espera",
Queue.TypeEnum.FirstInFirstOut, entityCliente)
    queuePronto = simulation.CreateQueue("Pronto",
Queue.TypeEnum.FirstInFirstOut, entityCliente)
    entityPorta = simulation.CreateEntity("Porta")
    entityPorta.IsResource = False
    queueOcupada = simulation.CreateQueue("Ocupada",
Queue.TypeEnum.FirstInFirstOut, entityPorta)
    entityBarman = simulation.CreateEntity("Barman")
    queueParado = simulation.CreateQueue("Parado",
Queue.TypeEnum.FirstInFirstOut, entityBarman)
    entityCopo = simulation.CreateEntity("Copo")
    queueCheio = simulation.CreateQueue("Cheio", Queue.TypeEnum.FirstInFirstOut,
entityCopo)
    queueLimpo = simulation.CreateQueue("Limpo", Queue.TypeEnum.FirstInFirstOut,
entityCopo)
    queueSujo = simulation.CreateQueue("Sujo", Queue.TypeEnum.FirstInFirstOut,
entityCopo)
    activityChegada = simulation.CreateActivity("Chegada", 1)
    activityEncher = simulation.CreateActivity("Encher", 1)
    activityBeber = simulation.CreateActivity("Beber", 1)
    activityLavar = simulation.CreateActivity("Lavar", 1)
    queueExterior.Insert(entityCliente.Generate(1000))
    queueOcupada.Insert(entityPorta.Generate(1))
    queueParado.Insert(entityBarman.Generate(1))
    queueLimpo.Insert(entityCopo.Generate(12))
    activityChegada.NeedResource(entityCliente, 1)
    activityChegada.NeedResource(entityPorta, 1)
    activityEncher.NeedResource(entityCliente, 1)
    activityEncher.NeedResource(entityBarman, 1)
    activityEncher.NeedResource(entityCopo, 1)
    activityBeber.NeedResource(entityCliente, 1)
    activityBeber.NeedResource(entityCopo, 1)
```

```
    activityLavar.NeedResource(entityBarman, 1)
    activityLavar.NeedResource(entityCopo, 3)
End Sub

Sub Main()
    Init()
    Dim simulationEnd As Boolean = False

    If Not simulation.ResourcesDefined() Then
        MsgBox("There aren't enough resources to start the simulation.",
MsgBoxStyle.Exclamation)
        Exit Sub
    End If

    Do Until simulationEnd
        For Each a As Activity In simulation.Activities
            Do While a.CanStart()
                simulation.ScheduleNewJob(a, simulation.CurrentTime)
            Loop
        Next

        Dim nextJob As Job = simulation.GetNextJob()
        simulationEnd = (nextJob Is Nothing OrElse nextJob.ExecuteAt >
simulation.Length)

        If Not simulationEnd Then
            simulation.UpdateCurrentTime(nextJob.ExecuteAt)
            For Each j As Job In
simulation.GetJobsByExecutionTime(simulation.CurrentTime)
                simulation.ExecuteJob(j)
            Next
        End If
    Loop

    simulation.Report()
End Sub

Sub Chegada_OnSchedule(sender As Activity, e As Activity.ScheduleEventArgs)
Handles activityChegada.OnSchedule
    e.Resources(entityCliente).RemoveFrom = queueExterior
    e.Resources(entityPorta).RemoveFrom = queueOcupada
    e.Duration = Random.Exponential(20)
End Sub

Sub Chegada_OnExecute(sender As Job, e As Activity.ExecuteEventArgs) Handles
activityChegada.OnExecute

    For Each item As Entity In sender.Entities(entityCliente)
        item.Variables("Sede") = Random.Uniform(1, 4)
    Next

    e.Resources(entityCliente).InsertInto = queueEspera
    e.Resources(entityPorta).InsertInto = queueOcupada
End Sub

Sub Encher_OnSchedule(sender As Activity, e As Activity.ScheduleEventArgs)
Handles activityEncher.OnSchedule
    e.Resources(entityCliente).RemoveFrom = queueEspera
    e.Resources(entityBarman).RemoveFrom = queueParado
    e.Resources(entityCopo).RemoveFrom = queueLimpo
    e.Duration = Random.Normal(6, 1)
End Sub
```

```
Sub Encher_OnExecute(sender As Job, e As Activity.ExecuteEventArgs) Handles
activityEncher.OnExecute
    e.Resources(entityCliente).InsertInto = queuePronto
    e.Resources(entityBarman).InsertInto = queueParado
    e.Resources(entityCopo).InsertInto = queueCheio
End Sub

Sub Beber_OnSchedule(sender As Activity, e As Activity.ScheduleEventArgs)
Handles activityBeber.OnSchedule
    e.Resources(entityCliente).RemoveFrom = queuePronto
    e.Resources(entityCopo).RemoveFrom = queueCheio
    e.Duration = Random.Uniform(5, 10)
End Sub

Sub Beber_OnExecute(sender As Job, e As Activity.ExecuteEventArgs) Handles
activityBeber.OnExecute

    For Each item As Entity In sender.Entities(entityCliente)
        item.Variables("Sede") = item.Variables("Sede") - 1
        If item.Variables("Sede") <= 0 Then
            e.Resources(entityCliente).InsertInto = queueExterior
        ElseIf item.Variables("Sede") > 0 Then
            e.Resources(entityCliente).InsertInto = queueEspera
        End If
    Next

    e.Resources(entityCopo).InsertInto = queueSujo
End Sub

Sub Lavar_OnSchedule(sender As Activity, e As Activity.ScheduleEventArgs)
Handles activityLavar.OnSchedule
    e.Resources(entityBarman).RemoveFrom = queueParado
    e.Resources(entityCopo).RemoveFrom = queueSujo
    e.Duration = 5
End Sub

Sub Lavar_OnExecute(sender As Job, e As Activity.ExecuteEventArgs) Handles
activityLavar.OnExecute
    e.Resources(entityBarman).InsertInto = queueParado
    e.Resources(entityCopo).InsertInto = queueLimpo
End Sub
```