limite texto

Resumo

A crise, a dificuldade das democracias europeias em fazer face aos desafios causados pela mesma e a concorrência dos países emergentes criam um clima de forte competitividade entre as empresas conducentes a uma margem estreita por onde orientar os seus negócios.

A utilização de metodologias que auxiliem a eliminar incerteza nos processos de decisão empresarial são de enorme importância actualmente. O advento da Modelação e Simulação proporciona-se como uma ferramenta de suporte inovadora e útil na melhoria da gestão dos processos e negócios.

Neste trabalho, apresenta-se uma breve introdução teórica aos conceitos da Modelação e Simulação, uma descrição do funcionamento do programa Anylogic e um exemplo de aplicação.

Foi analisado o caso da empresa de serviços de assistência e desenvolveu-se um modelo conceptual que emule o seu processo. De seguida criou-se um modelo de programação baseado na filosofia *Agent Based Modeling* e implementou-se o mesmo no software específico de simulação Anylogic. Por último verificou-se e validou-se o modelo, fez-se análise de dados e a optimização de parâmetros importantes para o a eficiência do negócio.

Abstract

The crisis, the difficulty European democracies demonstrated facing up to the challenges it poses and competition from emerging countries have created a climate of strong competition between companies leading to a very narrow margin to conduct their businesses.

The use of methods that help eliminate uncertainty from decision making processes is of the utmost relevance currently. The advent of Modeling & Simulation provides an innovating and useful tool in the improvement of the management of business processes.

In this essay, a brief introduction to the theoretical concepts of M&S is presented, a description of the operation of Anylogic software is done as well as an application example.

A review of the case of a services company is completed, and a conceptual model that emulates its process is developed. Then, a model is created using *Agent Based Modeling* philosophy and is implemented in Anylogic. Lastly, verification and validation is performed on the model, resulting data is analysed and parameters optimized in order to improve business efficiency.

Agradecimentos

À família

Aos amigos

Ao orientador, o Professor Daniel Gaspar

Conteúdo

Índice de Figuras

Índice de Tabelas

1. Introdução

Há muitas razões que guiam a procura de conhecimento e a necessidade de perceber o que nos rodeia. Resolver problemas está na natureza humana. O primeiro método de aprendizagem acerca do meio em que nos inserimos foi a experimentação, e durante milhares de anos foi o único disponível, continuando a ser uma das principais formas de resolução de problemas. Na experimentação há duas entidades sob avaliação, o objecto estudado e o quadro de condições que define como decorre a experiência.

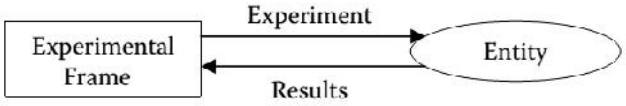


Figura 1- Resolução de problemas através de experimentação (Wainer, 2009)

No entanto, por vezes, os problemas a solucionar são de tal ordem complexos que a experimentação se pode revelar uma metodologia desadequada devido a questões éticas, de risco ou custo. Noutros casos a resolução é simplesmente impossível de colocar em prática. A ciência soube encontrar formas de abordar estas questões, uma das quais passa por abstrairmo-nos do problema e pensar nele usando um modelo do mesmo. Embora diferentes técnicas de modelação tenham sido propostas ao longo dos últimos séculos, o formalismo das equações diferenciais de Newton e Leibniz tem sido uma das ferramentas de eleição para modelação e resolução de problemas. Estas equações proporcionam uma forma matemática (analítica) de estudar a entidade em causa.

As técnicas usadas neste tipo de resolução de problemas são analíticas no sentido em que são simbólicas e baseadas em raciocínio, fornecendo soluções genéricas aos problemas enfrentados. É feita uma abstracção da entidade e o modelo resultante é então usado para avaliar o sistema. Esta abstracção é feita com perda de informação (simplificações), mas representa o comportamento das entidades, permite análise e é suficiente para demonstrar as propriedades do

modelo proposto. É fundamental que os resultados do modelo estejam de acordo com os observados na entidade, sendo nesse caso válidos.

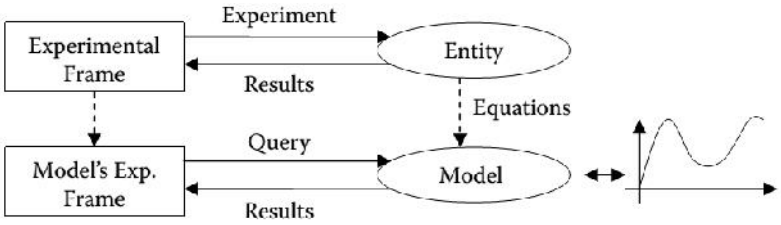


Figura 2 - Resolução de problemas através de modelação analítica (Wainer, 2009)

No caso de modelos com um maior grau de complexidade, é necessário recorrer a equações diferenciais e outros métodos numéricos, aproximando a equação através da discretização do seu comportamento (calculado em diversos pontos no tempo). Os resultados serão menos exactos do que os modelos analíticos, mas fornecem valores com precisão suficiente para estudar o problema em análise. O processo inicia-se obtendo dados experimentais da entidade em estudo, modelando depois matematicamente o comportamento observado. No entanto, devido à complexidade envolvida, as equações têm de ser resolvidas através de aproximação numérica. Com a massificação da utilização de computadores, isto é conseguido de forma recursiva, calculando os valores das variáveis de estado em momentos específicos, e usando estes valores como base para a iteração seguinte. O modelo deve fornecer uma solução para o problema, no entanto esta é uma aproximação e implica uma perca de precisão quando comparado com modelos analíticos. Na figura 3 observa-se que é necessário introduzir um passo para verificar a precisão dos resultados obtidos através de aproximação numérica. Os resultados têm de ser comparados com dados experimentais, para que o modelo emule o melhor possível a realidade. Durante séculos estas abordagens permitiram avanços muito importantes na área da ciência e tecnologia.

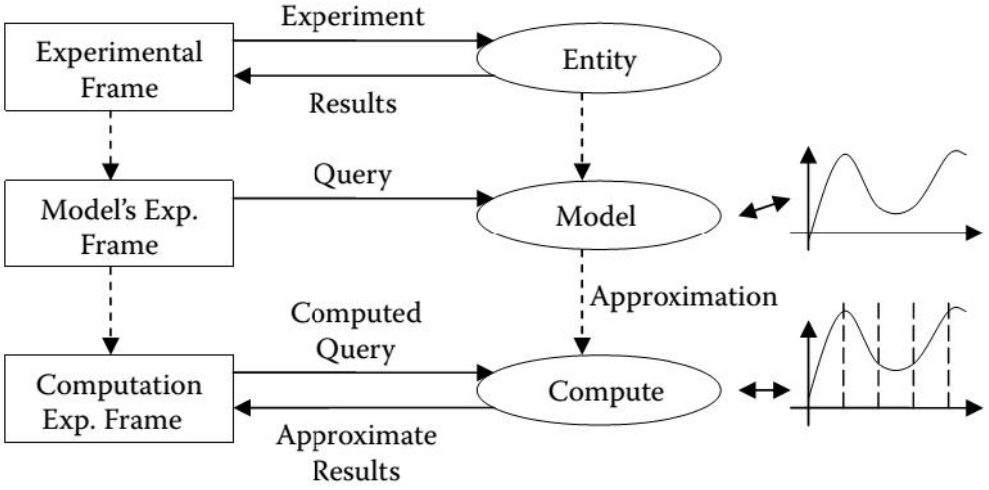


Figura 3 – Resolução de problemas com recurso a computação (Wainer, 2009)

A evolução do nosso conhecimento e a invenção de dispositivos avançados (p.ex. sistemas de controlo, fabricação inteligente, gestão e controlo de tráfego) tornou impossível a

continuação da utilização exclusiva de métodos analíticos para a resolução de problemas. Estes são analiticamente impossíveis e numericamente difíceis de avaliar sem simplificar excessivamente os modelos, o que pode resultar em soluções incoerentes com a realidade.

A partir dos anos 40, os computadores possibilitaram formas alternativas de lidar com estas questões, uma vez que estão bem equipados para aplicar técnicas de aproximação, eliminam erros de computação humanos e resolvem problemas a maior velocidade. Assim, desde o advento da computação os modelos numéricos foram convertidos em soluções computacionais (também chamadas simulações computacionais). Um ciclo de simulação computacional também pode ser caracterizado pela figura 3, com a diferença que o modelo é executado por dispositivos especializados. A verificação continua a ser necessária, já que limitações de precisão podem causar resultados erróneos e divergir das soluções expectáveis.

A simulação possibilita experiências com ambientes virtuais, avançando o nível de análise de aplicações naturais e artificiais a níveis sem precedentes na história científica, e permitindo a concepção e análise de aplicações complexas. A simulação também proporcionou soluções de treino com boa relação custo/benefício e sem risco quando contraposto à experimentação.

Em suma, a simulação é uma abordagem madura e económica de resolução de problemas, que evolui continuamente com o aumento do poder computacional disponível.

2. Modelação e Simulação (M&S)

De acordo com Banks, no âmago da disciplina de M&S está a noção de que "os modelos são aproximações do mundo real". O primeiro passo em M&S é a criação de um modelo que represente um evento ou sistema, que possa depois ser modificado e a simulação permita observação do seu comportamento. Depois de efectuar simulações sobre o modelo, ocorre a análise para tirar conclusões, verificar e validar a pesquisa relevante e fazer recomendações. A visualização fornece uma forma de interagir com o modelo, e é uma forma de representação de dados. As fundações da M&S assentam sobre estes quatro conceitos:

1. Modelação
2. Simulação
3. Visualização
4. Análise

"Um sistema é uma colecção, ou construído a partir, de elementos diferentes que juntos produzem resultados não atingíveis pelos elementos em separado" - International Council of Systems Engineering – INCOSE (traduzido). Em M&S um sistema é referido como o sujeito ou coisa a ser investigado, e como tal é o objecto do desenvolvimento do modelo. Durante este estudo, uma avaliação quantitativa do sistema é importante para o modelador, observando como o sistema reage a várias entradas em ambientes diferentes usando critérios de avaliação objectivos. Existem bastantes tipos de sistemas, um tipo de classificação abrangente foi proposto por Cassandras e Lafortune (em 2008):

-*sistemas estáticos e dinâmicos* – nos sistemas estáticos a saída é independente dos valores que a entrada tomou anteriormente. No caso dos dinâmicos, a entrada influencia a saída. Equações de diferenças ou diferenciais são necessárias para caracterizar o comportamento de sistemas dinâmicos.

-*sistemas variantes e invariantes no tempo* – sistemas invariantes não alteram o seu estado com o passar do tempo. Esta propriedade implica que podemos aplicar uma entrada a um sistema e esperar que ele responda sempre da mesma forma.

-*sistemas lineares e não lineares* – de uma forma simplificada, sistemas lineares podem ser combinados entre si e o sistema resultante pode ser descrito como a combinação das propriedades matemáticas dos sistemas iniciais. Em sistemas não lineares isto não se verifica e o seu comportamento é menos previsível.

-*sistemas de estado contínuo e estado discreto* – em sistemas de estado contínuo as variáveis podem tomar qualquer valor (real ou complexo). Em sistemas de estado discreto as variáveis são elementos de um conjunto discreto (p.ex. inteiros não negativos).

-*sistemas accionados por tempo (time driven) e sistemas accionados por eventos (event driven)* – no primeiro caso, o estado varia continuamente ao longo do tempo, enquanto no segundo a ocorrência de eventos discretos força a alteração instantânea de estado. No período entre eventos o sistema mantém-se inalterado.

-*sistemas determinísticos e estocásticos* – um sistema é estocástico quando uma ou mais das suas variáveis de saída é uma variável aleatória. Neste caso o estado do sistema é determinado por um processo estocástico cujo comportamento apenas pode ser descrito com recurso a probabilidades. Quando isto não se verifica o sistema é determinístico.

-*sistemas discretos e sistemas contínuos (tempo)* – nos sistemas discretos as variáveis de estado variam instantaneamente em pontos espaçados no tempo. No caso dos sistemas contínuos as suas variáveis de estado variam de forma contínua ao longo do tempo e podem tomar qualquer valor.

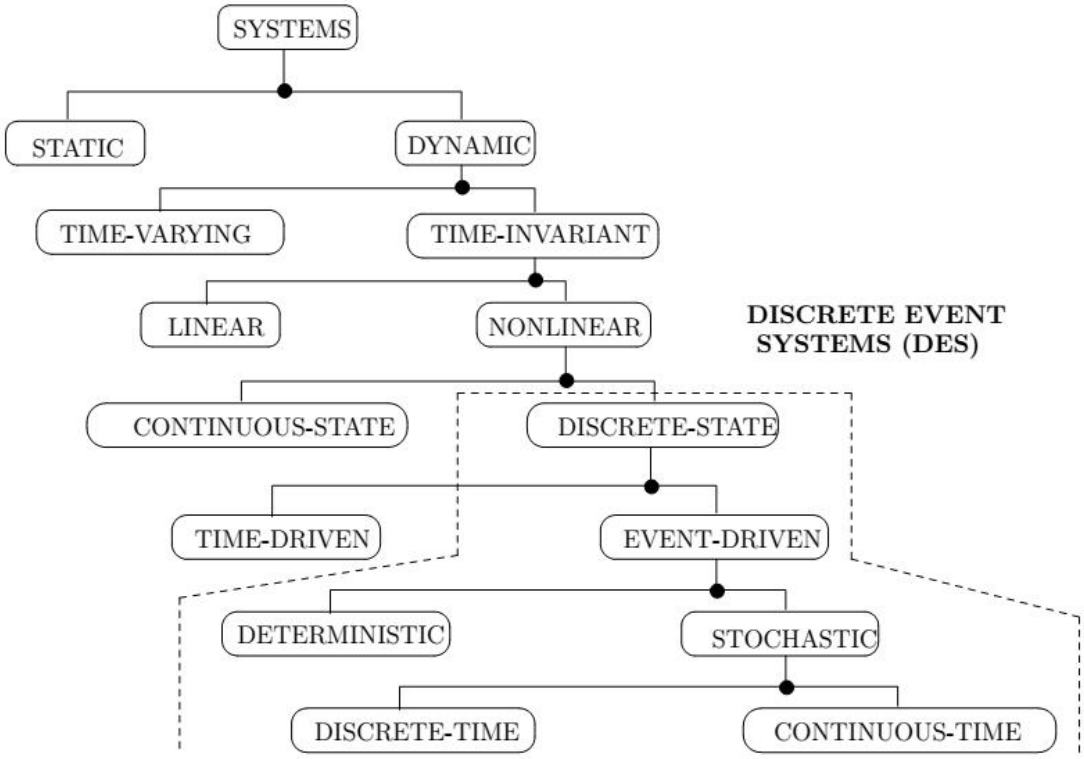


Figura 4 – Principais classificações de sistemas (Cassandras e Lafortune – 2008)

Existem várias formas de estudar um sistema (Sokolowski & Banks, 2010):

1. O sistema real vs. um modelo
2. Uma representação física vs. uma representação matemática

3. Solução analítica vs. simulação

Quando estuda um sistema, o modelador deve focar-se em três questões fundamentais:

1. A análise quantitativa dos sistemas
2. As técnicas para o desenvolvimento, controlo e uso
3. A avaliação e mensuração da performance do sistema (p.ex. custo, lucro, alocação de recursos, tempos)

Como foi referido anteriormente, um modelo é uma representação física, matemática ou lógica de um sistema, entidade, fenómeno ou processo. A sua função é servir como representação de eventos reais e/ou coisas. Pode ser uma representação de um sistema real, principalmente porque estes podem ser difíceis, senão impossíveis, de investigar. Sistemas essenciais que não podem ser retirados de funcionamento, sistemas teóricos sem partes físicas onde conduzir experiências, em ambas as situações devem ser desenvolvidos modelos, e o estudo realiza-se sobre estes. O desenvolvimento de um modelo ocorre abstraindo da realidade uma descrição do sistema. Apenas as características do sistema que afectam a sua performance devem ser representadas no modelo, a inclusão exaustiva de todas as características do sistema real implicaria elevados custos, consumo de tempo, e complexidade excessiva sendo, talvez impossível. O modelo deve ser o mais simples possível, mas representar o sistema de forma fiável. É um desafio para o modelador decidir quais as características do sistema que devem ser incluídas no modelo.

Um modelo pode ser físico ou um conjunto de equações matemáticas ou afirmações lógicas que descrevem o comportamento de um sistema (sistemas teóricos/nocionais). Modelos matemáticos podem resultar em soluções analíticas (prova matemática) se a complexidade for baixa, ou soluções numéricas noutros casos (com grau de incerteza associado). O processo de resolver numericamente um problema neste contexto é referido como simulação computacional, a definição de simulação é mais complexa que a de um modelo, e segundo Sokolowski & Banks (traduzido), estas são cinco possibilidades:

- Uma forma de implementação de um modelo ao longo do tempo
- Uma técnica para teste, análise ou treino em que são usados sistemas reais ou onde sistemas reais e sistemas conceptuais são reproduzidos por um modelo
- Um método científico de investigação não intrusivo envolvendo experiências com um modelo, em vez da fracção da realidade que o modelo representa
- Uma metodologia para extrair informação de um modelo através da observação do seu comportamento quando executado
- Um termo não técnico que significa não real, imitação

Portanto, uma simulação pode ser entendida como um método que descreve o comportamento de um sistema usando um modelo matemático ou simbólico. A sua necessidade deve-se ao facto de que a interacção com um sistema real pode não ser possível devido a inacessibilidade, risco, ser inaceitável a interacção com o mesmo e ainda a possibilidade de o sistema nem sequer existir. Uma execução única da simulação é referida como ensaio, uma série de ensaios da simulação é um exercício, podendo ocorrer como parte de um processo de aprendizagem.

Outro conceito importante é a M&S em si mesmo, refere-se ao processo de desenvolvimento de um modelo de um sistema e depois à execução de simulações sobre esse modelo de forma a recolher dados relativos à performance do mesmo. Os dados resultantes podem servir como base a decisões de gestão, formação e técnicas e, quando sistemas de grande dimensão e complexidade estão em causa, a simulação pode ser mesmo a única ferramenta viável para a sua

análise. Em suma, M&S começa com o desenvolvimento de um modelo de simulação computacional baseado num sistema real ou teórico, depois corre-se o referido modelo num computador e procede-se à análise dos dados resultantes.

Segundo Banks, o ciclo de desenvolvimento M&S passa ciclicamente por 4 fases, cada uma delas necessitando um conjunto diferente de tecnologias associadas, tal como é demonstrada na figura 5.

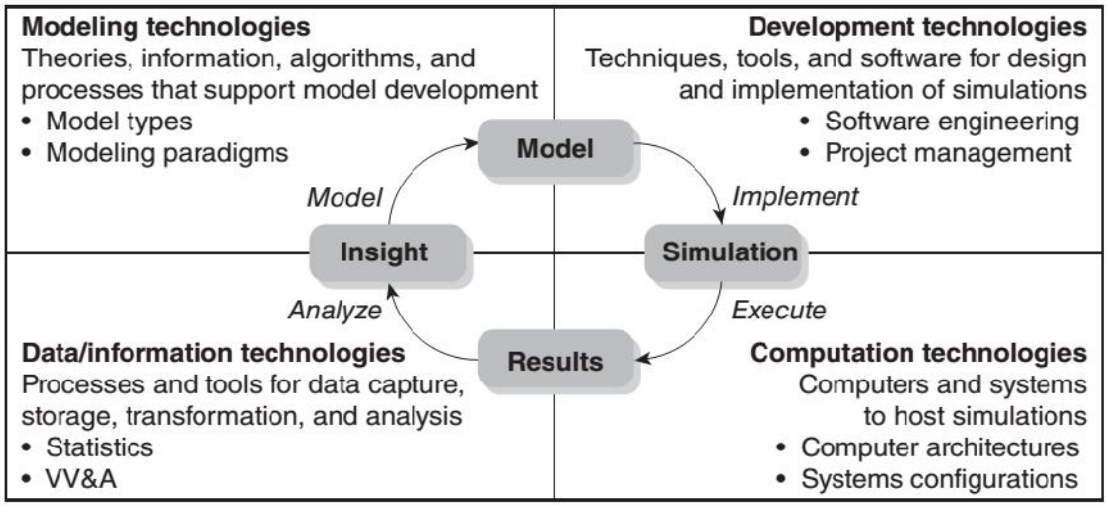


Figura 5 – Ciclo de desenvolvimento do processo M&S (Sokolowski & Banks – 2010)

Só depois de um certo número de iterações é que o processo atingirá a sua eficiência ideal. A primeira iteração é útil para fornecer informação e dados com vista à optimização do modelo. As boas práticas recomendam a repetição do processo até que os membros das equipas intervenientes estejam satisfeitos com os resultados obtidos que produzam uma representação fiável do sistema estudado.

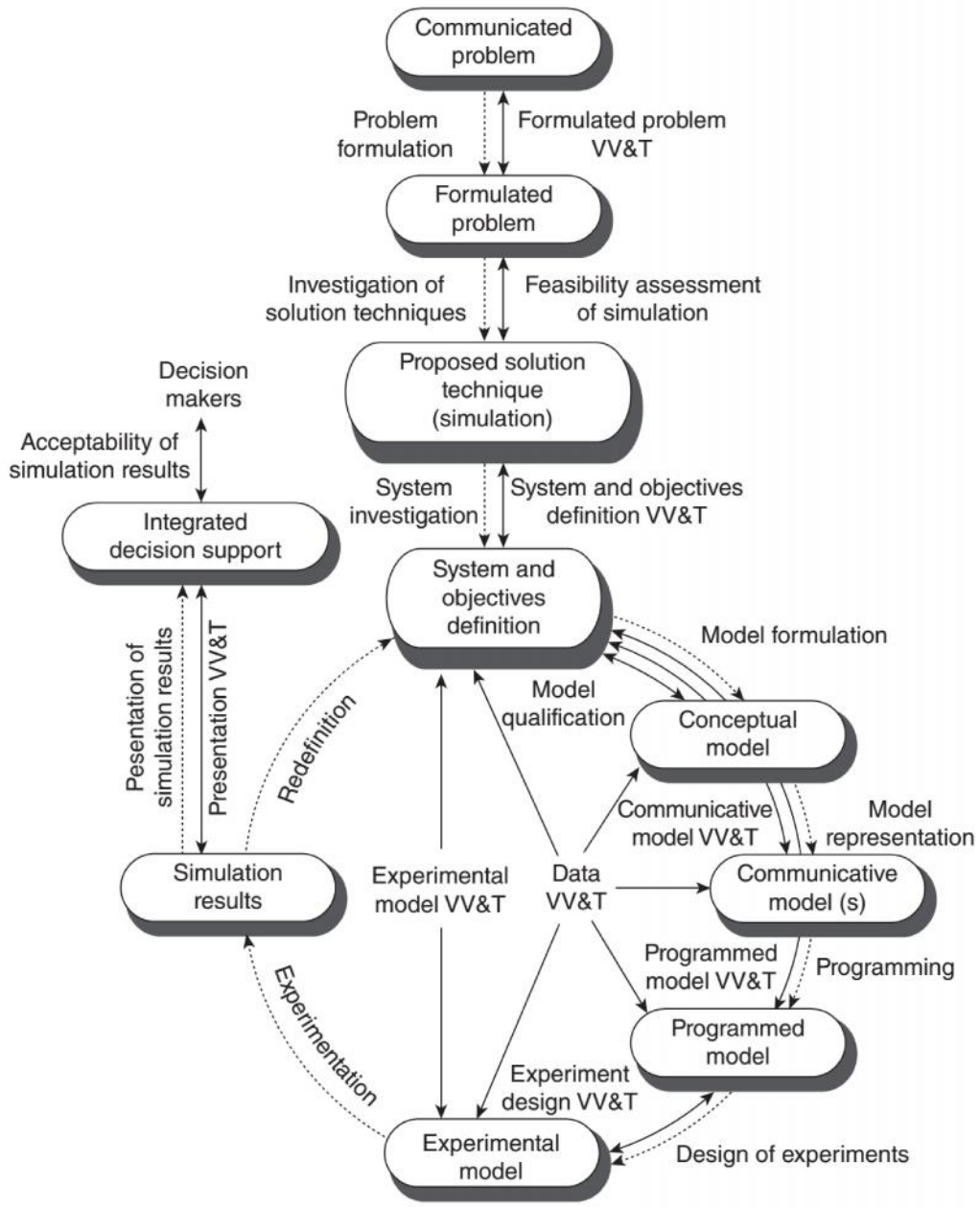


Figura 6 – Ciclo de vida M&S (Sokolowski & Banks, 2010)

Existem cinco grandes disciplinas necessárias à condução correcta de um processo de M&S:

-*Probabilidade e estatística* – a maioria dos sistemas implica graus diferenciados de incerteza, na modelação proceder simplesmente ao cálculo do valor médio da resposta não é suficiente porque então faltaria ao modelo validade. Tem de se ter em conta o grau de incerteza e variabilidade e, consequentemente, as variáveis do sistema devem ser representadas como variáveis ou processos aleatórios. O tratamento de variáveis ou processos aleatórios implica o domínio de conceitos e teorias de probabilidade e estatística. Estas são usadas para gerar variações aleatórias nas variáveis de entrada que simulam a incerteza e variabilidade do sistema, e analisam as saídas resultantes dos modelos ou sistemas estocásticos. Modelos estocásticos são definidos por parâmetros de entrada que contêm variáveis afectadas com um grau de incerteza e, assim, de simulações com esses modelos resultam saídas que são também variáveis aleatórias.

-*Análise e pesquisa operacional* – de um estudo de simulação computacional resultam grandes quantidades de dados de performance do sistema. *Análise* é o processo de transformação destes dados em informação útil e facilmente interpretável que descreva o comportamento do sistema. *Pesquisa operacional* consiste numa série de técnicas e abordagens matemáticas e de engenharia de sistemas para condução de análise. A M&S fornece como resultados da simulação dados que representam a resposta dinâmica do modelo quando sujeito a certas condições e entradas, sobre estes resultados efectua-se análise com o propósito de encontrar resposta às questões que motivaram o estudo inicial. Este estudo pode procurar cumprir várias funções (Sokolowski & Banks – 2010):

1 – Concepção de experiências – o desenvolvimento de um conjunto de simulações experimentais por forma a dar resposta a questões de performance do sistema.

2 – Avaliação de desempenho – medição da performance do sistema, avaliando o quanto se aproxima de um determinado nível de desempenho.

3 – Análise de sensibilidade – sensibilidade do sistema a um conjunto de parâmetros de entrada.

4 – Comparação de sistemas – comparação entre dois ou mais sistemas alternativos para encontrar a solução com melhor desempenho para determinadas condições.

5 – Optimização condicionada – determinação dos parâmetros ideais para atingir o objectivo de desempenho.

A análise ocorre para tirar conclusões, verificar e validar a pesquisa e fazer recomendações baseada em simulações do modelo

-*Visualização computacional* – visualização é uma forma de representar os dados e fornecer um interface com o modelo. Grandes quantidades de dados são gerados correndo simulações sobre o modelo, a visualização oferece alternativas a grandes folhas de cálculo que não permitem uma fácil avaliação do comportamento do sistema. Usando gráficos gerados por computador, criam-se modelos em duas e três dimensões da resposta do sistema permitindo um entendimento mais fácil do seu comportamento.

-*Factores humanos* – normalmente as simulações incluem pessoas como componentes do modelo. Isto é designado como modelação de comportamento humano e o modelador tem de possuir um entendimento básico da percepção e conhecimento humano. Com esta informação o modelador pode criar o interface humano-computador tendo em conta as forças e fraquezas do utilizador humano. O principal foco desta abordagem recai sobre o processo computacional do processo de decisão humano.

-*Gestão de projectos* – a utilização de simulação M&S para resolver problemas reais é um desafio exigente e se não for gerido competentemente pode gerar mais problemas do que os que se propõe resolver. Quando é simulado um processo de elevada complexidade, M&S torna-se um projecto técnico de grande escala que requer supervisão e gestão experiente e conhecedora. Assim, profissionais de M&S devem possuir sólida formação em gestão de projectos.

Segundo Banks e Sokolowski, os paradigmas de simulação que predominantes em M&S são:

- a *simulação Monte Carlo* – onde valores amostrados são aleatoriamente adquiridos de cada distribuição de variáveis de entrada e são usados para o cálculo dos valores de saída do modelo. Este processo é repetido ciclicamente até que se perceba a forma como valores de entrada variáveis afectam a saída do sistema. A simulação Monte Carlo usa probabilidades para modelar o comportamento do sistema;

- *a simulação contínua* – onde as variáveis são funções contínuas do tempo e se recorre a equações diferenciais para a concepção do modelo.

- *a Simulação de Eventos Discretos* – onde as variáveis são funções discretas do tempo e apenas variam em momentos específicos (eventos do sistema). Este tipo de simulação usa a teoria das filas de espera.

Há três atributos principais que definem as propriedades ou características de um modelo:

-*Fidelidade* – descreve o grau de precisão com que um modelo retracta a realidade. Atingir um grau de fidelidade elevado não é fácil, os modelos são concebidos para emular apenas as características dos sistemas que são relevantes para a avaliação da sua performance. Diferentes níveis de fidelidade são aceites dependendo da relevância dos componentes do sistema. Exige-se maior fidelidade a simulações de pesquisa académica do que a simulações usadas em formação e treino.

-*Resolução* – é o nível de detalhe com que a simulação representa o sistema real. Quanto mais detalhado for um modelo mais resolução tem.

-*Escala* – é a dimensão do cenário simulado. Quanto maior for um cenário, logicamente maior será a escala. Maior escala implica mais componentes e maior nível de complexidade do modelo.

As relações entre fidelidade, resolução e escala não são tão claras quanto podem parecer ao primeiro olhar. Um maior grau de fidelidade não implica necessariamente maior resolução. Ao mesmo tempo é possível aumentar a escala sem a que isso corresponda uma diminuição na fidelidade. No entanto, não é prático aumentar a escala mantendo um elevado nível de resolução, já que os recursos necessários seriam proibitivos e a complexidade da simulação excessiva.

Simulações podem ser de três tipo, "*live" (ao vivo)*, *virtual* e *construtiva:*

-*Live* – este tipo de simulação envolve pessoas reais a operar num sistema real, frequentemente em ambiente de treino (p.ex. exercícios militares).

-*Virtual* – uma simulação virtual envolve pessoas reais a operar em sistemas simulados. Os participantes são inseridos num ambiente simulado realístico. Exemplos deste tipo de simulações envolvem simuladores de vôo, simulações militares de cenários de guerra, entre outras.

-*Construtiva* – neste caso os participantes e os sistemas são simulados. Pessoas simuladas operam em ambientes simulados. Estes sistemas são operados por não-participantes na simulação.

Os domínios de aplicação de M&S incluem formação e treino, análise, experimentação, engenharia e aquisição de dados e ocorrem em vários domínios, incluindo, entre outros, o ramo militar, de transportes, suporte a decisões, educação (aprendizagem com teoria de jogos), simulação médica, ciências sociais e ambientes virtuais.

Modelo Conceptual

Conforme preconiza Dale (2010), o modelo conceptual de uma simulação responde ao contexto da mesma, como satisfará os seus requisitos e como as suas entidades e processos devem ser representados. Embora não exista nenhuma abordagem universalmente aceite para decompor a representação do objecto da simulação nas entidades e processos que o representam, para abstrair essa representação a partir de informação acerca do objecto da simulação, ou para

descrever ou documentar o modelo conceptual, este é essencial na elaboração da simulação e deve possuir quatro atributos fundamentais:

1 – Ser completo – o modelo conceptual identifica todas as entidades e processos do domínio do problema a resolver, e todas as características de controlo e operação da simulação necessárias para garantir que a simulação responde de forma satisfatória a todos os seus requisitos.

2 – Consistência – os processos e entidades representativas dentro do modelo conceptual são abordados de perspectivas compatíveis em relação a características tais como sistemas de coordenadas e unidades, níveis de agregação e desagregação, precisão e paradigmas descritivos.

3 – Coerência – o modelo conceptual deve ser organizado de forma a que todos os elementos tenham função (não existam elementos alheios) e potencial (não existam partes do modelo impossíveis de activar).

4 – Exactidão – o modelo deve ser apropriado para o fim pretendido e tem potencial para desempenhar a sua função de forma a satisfazer todos os requerimentos da simulação.

2.1. Simulação de Eventos Discretos (DES)

Evento

No âmbito de DES, um evento ocorre instantaneamente e implica a transição de um estado para outro. Segundo Cassandras e Lafortune, pode ser identificado por uma acção específica tomada (p.ex. ligar a luz), uma ocorrência espontânea ditada pela natureza (um computador avaria por uma qualquer razão demasiado complicada para avaliar) ou o resultado de um conjunto de condições que subitamente são todas concretizadas (p.ex. o nível de líquido num tanque excedeu determinado valor).

Sistemas *time-driven* vs. sistemas *event-driven*

Em sistemas de estado contínuos, o estado altera-se ao longo do tempo. Isto torna-se particularmente evidente em modelos tempo-discreto. O tempo é o que guia o caminho do sistema, com cada intervalo de tempo é expectável que o estado do sistema varie, já que variáveis de estado contínuas variam com o tempo. Devido a esta propriedade estes sistemas denominam-se de sistemas accionados por tempo (*time-driven*). A variável tempo é uma variável naturalmente independente que surge como argumento de todas as funções de estado, entrada e saída do sistema.

Em sistemas de estado discretos, as variações ocorrem instantaneamente em certos pontos no tempo. A cada mudança de estado associamos um evento. O mecanismo de temporização dos eventos pode assumir uma de duas formas (Cassandras & Lafortune – 2008):

1 – A cada intervalo de tempo pré-definido um evento "*e*" é seleccionado do conjunto de eventos "*E*". Se nenhum evento ocorrer escolhe-se um evento "nulo" de "*E*", cuja propriedade definidora é não causar alterações no sistema.

2 – Em vários instantes no tempo (não necessariamente conhecidos antecipadamente, nem coincidindo obrigatoriamente com os intervalos de tempo pré definidos no relógio), um evento *e* ocorre.

No primeiro caso, as transições de estado são sincronizadas pelo relógio. Nos intervalos definidos ou ocorre um evento (ou um não evento), altera-se o estado e o processo repete-se.

Já na segunda possibilidade, todos os eventos *"e"* pertencentes a *"E"* definem um processo distinto através de quais são determinados os instantes em que ocorrem. As transições de estado são um resultado da combinação destes eventos assíncronos e concorrentes. Não é necessário que estes processos sejam independentes uns dos outros.

A distinção entre 1 e 2 leva à distinção entre sistemas accionados por tempo e sistemas accionados por eventos (*event-driven*). Estes últimos são mais complexos de modelar e analisar devido à natureza assíncrona de transição entre eventos.

Propriedades características de sistemas DES (Cassandras e Lafortune – 2008)

Sistemas de eventos discretos satisfazem duas propriedades:

1 – O espaço de estado é um conjunto discreto.
2 – O mecanismo de transição de estado é accionado por eventos.

Definição de sistema DES:

"É um sistema de estado discreto, accionado por eventos, ou seja, a evolução do seu estado depende inteiramente da ocorrência de eventos discretos assíncronos ao longo do tempo."

Níveis de abstracção

Sistemas de eventos discretos podem usar como representação do caminho que percorrem sequências de pares ordenados listando os eventos, e o tempo em que ocorrem. Esta representação pode ser mais conveniente do que a representação gráfica do caminho seguido pelo sistema (fig. 7).

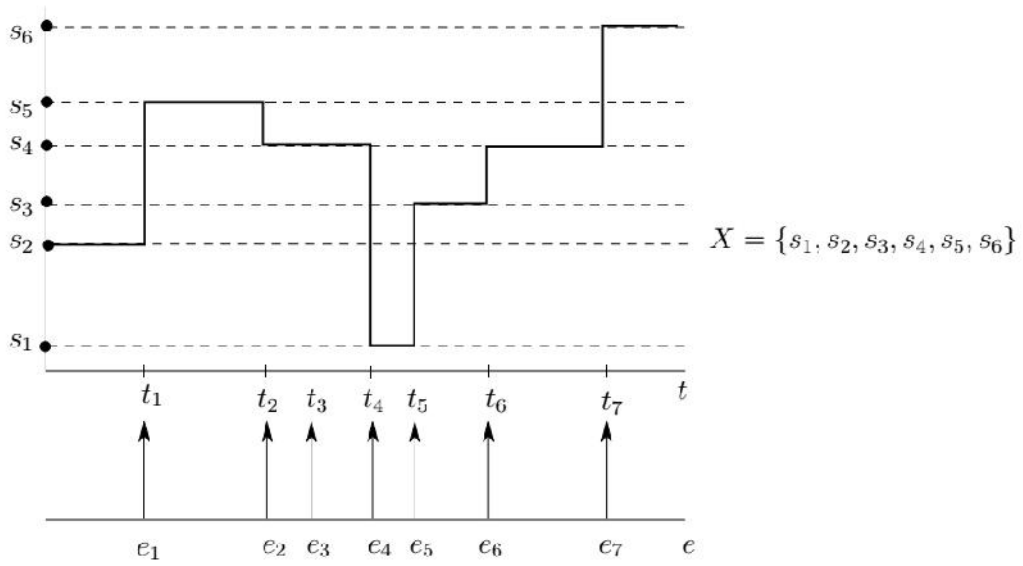


Figura 7 – Caminho seguido por um sistema DES (Cassandras e Lafortune – 2008)

O sistema da figura pode ser descrito da seguinte forma:

- *(e1,t1), (e2,t2), (e3,t3), (e4,t4), (e5,t5), (e6,t6), (e7,t7).*

Esta representação contém a mesma informação que a figura anterior. O evento *e1* ocorre no instante *t1*, o estado inicial *s2* é conhecido e o sistema é determinístico no sentido em que cada estado após a ocorrência de um evento é único. Pode-se designar o conjunto de todas as sequências temporais de eventos que o sistema pode executar como modelo em *linguagem*

temporal do sistema. Se estiver disponível uma função de distribuição de probabilidade acerca da "vida útil" de cada evento, isto é, o tempo decorrido entre ocorrências sucessivos de cada evento, pode-se adicionar esta informação ao modelo de linguagem, obtendo uma *linguagem temporal estocástica* do modelo do sistema. Este modelo elenca todos os possíveis caminhos que o sistema pode percorrer, juntamente com informação estatística acerca destes. Este é o tipo de linguagem mais detalhada acerca de um sistema discreto. Se ao modelo for retirada a informação estatística e a temporal, ficamos apenas com a ordem dos eventos, e neste caso ao modelo resultante chamamos a *linguagem* do sistema.

A linguagem, linguagem temporal, e *linguagem temporal estocástica* são os três níveis de abstracção usados para o estudo de sistemas de eventos discretos. Se apenas nos interessar o comportamento lógico do sistema, uma ordenação precisa dos eventos que ocorrem perante um conjunto de especificações, o modelo de *linguagem* do sistema é suficiente. Para uma aferição mais precisa da performance do sistema (tempo de resposta, rendimento), é necessário usar a *linguagem temporal* do modelo do sistema. Frequentemente, os sistemas de eventos discretos trabalham com cenários estocásticos e necessariamente têm de introduzir informações estatísticas nos seus modelos. Estas três abordagens são complementares, uma vez que abordam questões diversas acerca do comportamento de sistemas DES. Embora a abordagem das "*linguagens*" seja conveniente para apresentar questões de modelação e propriedades teóricas dos sistemas, não é de todo conveniente para verificação, controlo e avaliação de performance de sistemas. Existe como possibilidade a utilização de formalismos de modelação de eventos discretos que permitem a representar as linguagens realçando informação estrutural acerca do comportamento do sistema sendo passíveis de manipulação quando necessário para avaliação de questões de análise e controlo. Estes formalismos podem ser não-temporizados, temporizados ou estocásticos conforme o nível de abstracção requerido.

2.2. Simulação Agent Based Modeling (ABM)

A simulação DES envolve entidades genéricas que seguem regras definidas por agentes ou processos que alteram o seu estado e levam as entidades a esperar. Isto é, as entidades não tomam decisões nem se adaptam a circunstâncias diferentes, o que não deixa de ser útil para analisar/avaliar muitos tipos de sistemas. No entanto, por vezes, é interessante representar entidades que se adaptam e fazem escolhas. Para suprir esta necessidade, surgiu a simulação ABM que apresenta uma nova abordagem no domínio da simulação tornada possível pelo poder de computação dos equipamentos mais modernos. Estes sistemas são compostos por agentes autónomos, que interagem entre si, fornecendo outra forma de modelar as dinâmicas de sistemas complexos. Foca-se nos indivíduos e interacções entre eles, e embora se encontrem raízes nos anos 1940 a primeira modelação computorizada ocorreu com a introdução da linguagem SWARM em meados dos anos 1990 pelo *Santa Fe Institute* (Allen – 2011).

Há três ideias centrais na modelação com ABM (Allen):

1- Emergência (a formação de padrões complexos a partir de uma multiplicidade de interacções simples);
2- Agentes sociais como objectos;
3- Complexidade (frequentemente, o objecto modelado pelos agentes é o mesmo de modelos físicos).

Portanto, as entidades participantes no sistema não são participantes passivas, mas agentes capazes de aprender e interagir entre eles. Comportamentos pouco expectáveis para o modelador resultam de simulações sucessivas, e as interacções e fenómenos daí resultantes não são facilmente previsíveis por análise prévia devido à complexidade envolvida. Assim, modelos

ABM são particularmente indicados para aferir quando o estado de equilíbrio de sistemas poderá estar prestes a deixar de existir, que comportamento pode então ser esperado e que eventos poderão causar estabilidade ou instabilidade e quão robusto é expectável que o sistema seja.

Estrutura

Segundo Macal e North (2010), a estrutura de um modelo baseado em agentes tem três elementos:

1- Um conjunto de *agentes*, os seus atributos e comportamentos;
2- Um conjunto de *relações* e métodos de interacção entre agentes: define como e com quem interagem os agentes;
3- O *ambiente* dos agentes: os agentes interagem com o ambiente que os rodeia e uns com os outros.

Um modelador precisa identificar, modelar e programar estes elementos para poder criar um modelo ABM. É necessário um *motor* computacional para correr as simulações do modelo, ou seja observar o comportamento dos agentes e suas interacções.

Agentes autónomos

A característica mais importante num agente é a autonomia, ou seja, a capacidade de agir sem orientação externa perante as situações que encontra. Os agentes possuem comportamentos que lhes permitem tomar decisões independentes, agindo para atingir os seus objectivos internos. Casti (1997) advoga que devem possuir regras de comportamento de *baixo nível*, que permitam um conjunto de respostas passivas ao ambiente, e regras de *alto nível* que implicam a alteração das primeiras, e que permitem o processo de adaptação e aprendizagem.

Segundo Macal e North (2010) os agentes possuem estas características essenciais:

1- Um agente é um indivíduo independente, modular e único. O requisito de modularidade implica que o agente tem uma fronteira, é facilmente identificável o que pertence ou não ao agente. Os agentes têm atributos que lhes permitem ser distinguidos de outros agentes.
2- Um agente é autónomo e auto motivado. Pode agir independentemente no seu ambiente e interacções com outros agentes, pelo menos no âmbito das situações do interesse do modelo. Um agente tem comportamentos que fornecem informações por ele apreendidas acerca das suas decisões e acções (interacções com ambiente e outros agentes). O seu comportamento pode ser definido de várias formas, desde regras simples a modelos abstractos.
3- Um agente possui um *estado* que varia ao longo do tempo, este estado representa as principais variáveis associadas à sua situação no momento considerado. O estado de um sistema ABM é composto pelo estado de todos os agentes que o compõem em conjunto com o ambiente do sistema.
4- Um agente é *social*, tendo interacções dinâmicas com outros agentes que influenciam o seu comportamento. Possuem protocolos para interacção, comunicação, movimento e disputa de espaço com outros agentes, para resposta ao ambiente, entre outros. Têm a capacidade de reconhecer os atributos de outros agentes.

Podem ainda possuir outras características úteis, por exemplo ser adaptativos (capacidade de aprendizagem), possuir e ser dirigidos por objectivos (o que permite comparar o resultado das suas acções com o pretendido e adaptar-se).

Um tipo de agente representativo pode ser observado na figura:

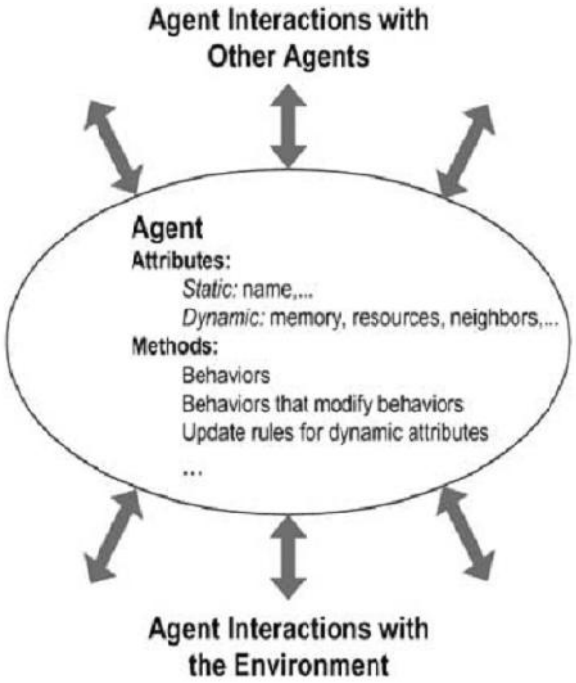


Figura 8 – Agente típico (Macal e North)

Interacção entre agentes

A modelação ABM dá o mesmo enfâse à concepção das relações entre agentes quanto oferece à modelação dos seus comportamentos. Os dois principais desafios na modelação das interacções entre agentes são quem "está, ou pode estar ligado a quem?" e o mecanismos das dinâmicas das interacções (Macal e North - 2010). Um dos dogmas dos sistemas ABM é que apenas está disponível a um qualquer agente informação localizada. Estes sistemas são descentralizados, nenhuma autoridade central divulga informação com o objectivo de melhorar o comportamento individual dos agentes. Agentes interagem uns com os outros, no entanto não o fazem com todos os agentes o tempo todo. As interacções de um agente num dado intervalo de tempo dão-se com um subconjunto do universo de agentes (os seus *vizinhos*) e do ambiente localizado. Estes conjuntos de vizinhos alteram-se com o decorrer da simulação e com a deslocação destes pelo ambiente. A forma como os agentes estão ligados é designada como a *topologia* do modelo ABM. Macal e North preconizam que topologias típicas podem incluir grelhas espaciais ou redes de ligações (relações) e nós (agentes).

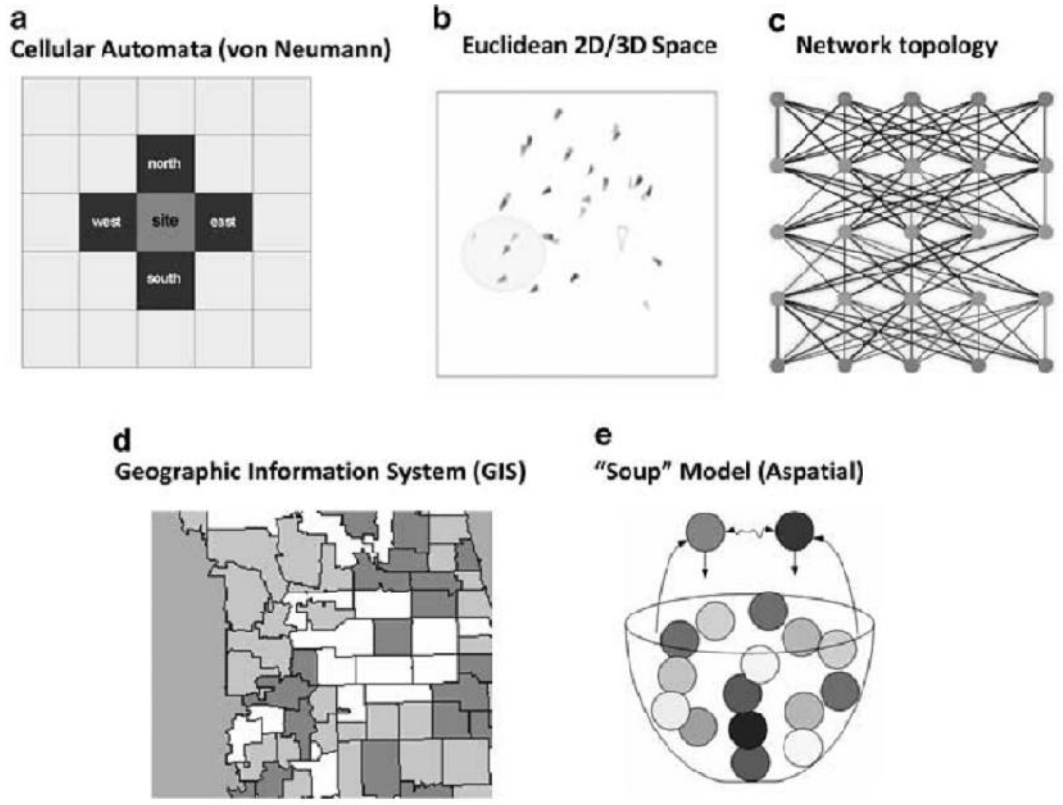


Figura 9 - Relações entre agentes e interacções sociais (Macal e North – 2010)

Ambiente

Os agentes estão inseridos num ambiente com o qual interagem em conjunto com outros agentes. O ambiente pode servir para fornecer apenas informação acerca da localização espacial de um agente relativo a outros ou fornecer informação geográfica detalhada. A localização de um agente, definida como atributo dinâmico, pode ser necessária para seguir agentes enquanto se movem ao longo do ambiente, competem por espaço ou adquirem recursos (Macal e North).

O ambiente influencia e constrange as acções que estão acessíveis aos agentes, tanto pela disponibilidade de recursos localizadas, como por constrangimentos de ordem geográfica, entre outros.

Aplicações

As aplicações de modelos ABM podem abranger áreas muito diversas, das ciências sociais à biologia, física à gestão, entre outras. Os modelos ABM podem modelar explicitamente a complexidade resultante das acções e interacções individuais que existem no mundo real.

Os modelos vão desde a elegância dos modelos minimalistas académicos, a modelos de grande escala de suporte a decisões. Estes últimos normalmente incluem dados reais, são sujeitos a testes de validação e propõem-se responder a questões (de gestão, política, entre outras) reais.

2.3. Verificação e Validação

"A credibilidade dos resultados da simulação depende não só da correcção do modelo, mas também do quão precisa seja a formulação do problema." - Pace, D. K. 1993. A *verificação* e

validação são uma parte essencial de qualquer projecto de simulação, e assim, um conhecimento mínimo destes conceitos é pré-requisito necessário.

As simulações são usadas com frequência em situações que envolvem riscos financeiros ou humanos elevados, no pressuposto que o modelo utilizado produz resultados fiáveis. Esta assumpção não pode ser apenas baseada na crença e na boa vontade dos modeladores, num projecto bem elaborado o rigor do modelo é avaliado num processo de *validação* e *verificação* (V&V), e em certos casos de *acreditação* (VV&A).

Verificação

Segundo Sokolowski & Banks, na M&S a *verificação* é definido como o processo de determinação se um modelo é implementado de forma consistente com as suas especificações. Preocupa-se também em avaliar se a concepção do modelo satisfará os requisitos de aplicação do mesmo, e deve produzir respostas às seguintes questões:

1 - O código do programa do modelo executável implementa correctamente o modelo conceptual?
2 - O modelo conceptual consegue satisfazer a utilização pretendida para o modelo?
3 - O modelo executável produz resultados quando necessário, e no formato pretendido?

Wainer (2009) refere ainda que uma simulação correcta reproduz fielmente o comportamento do modelo em todas as execuções do mesmo.

Validação

Ainda segundo Sokolowski & Banks, no âmbito de M&S a *validação* corresponde ao processo de determinação do grau de precisão com que o modelo representa o que é simulado. A validação avalia a precisão da representação do que é simulado no modelo conceptual e nos resultados fornecidos pelo modelo executável. A precisão requerida pode ter diferentes graus dependendo do uso final pretendido, e as perguntas típicas a responder durante este processo podem incluir:

1 - O modelo conceptual é uma representação correcta do que é simulado?
2 - Quão próximos são os resultados produzidos pelo modelo executável do comportamento do que é simulado?
3 - Quais os valores de entrada para os quais os resultados do modelo são credíveis e úteis?

Wainer observa que no ciclo de vida representado na figura seguinte, um projecto de simulação necessita de incluir as seguintes actividades:

1 – Verificação do modelo conceptual e do problema analisado. Garantir que o problema que se está a resolver corresponde ao sistema real

2 – Verificação do *design.* Garantir que as especificações reflectem fielmente o modelo conceptual.

3 – Validação do modelo do sistema – verificar que o modelo corresponde ao sistema que interessa dentro do quadro experimental. Verificar que o modelo é representado com um suficiente grau de precisão e garantir a qualidade dos dados usados nas diferentes fases do modelo.

4 – Verificação do simulador – assegurar que a implementação informática do modelo obedece às especificações do mesmo. Testar de forma a cobrir o máximo de casos possíveis.

5 – Validação dos resultados experimentais – a credibilidade do modelo é resultado de ter completado as actividades anteriores e garantir que os resultados obtidos no simulador são compatíveis com o sistema real.

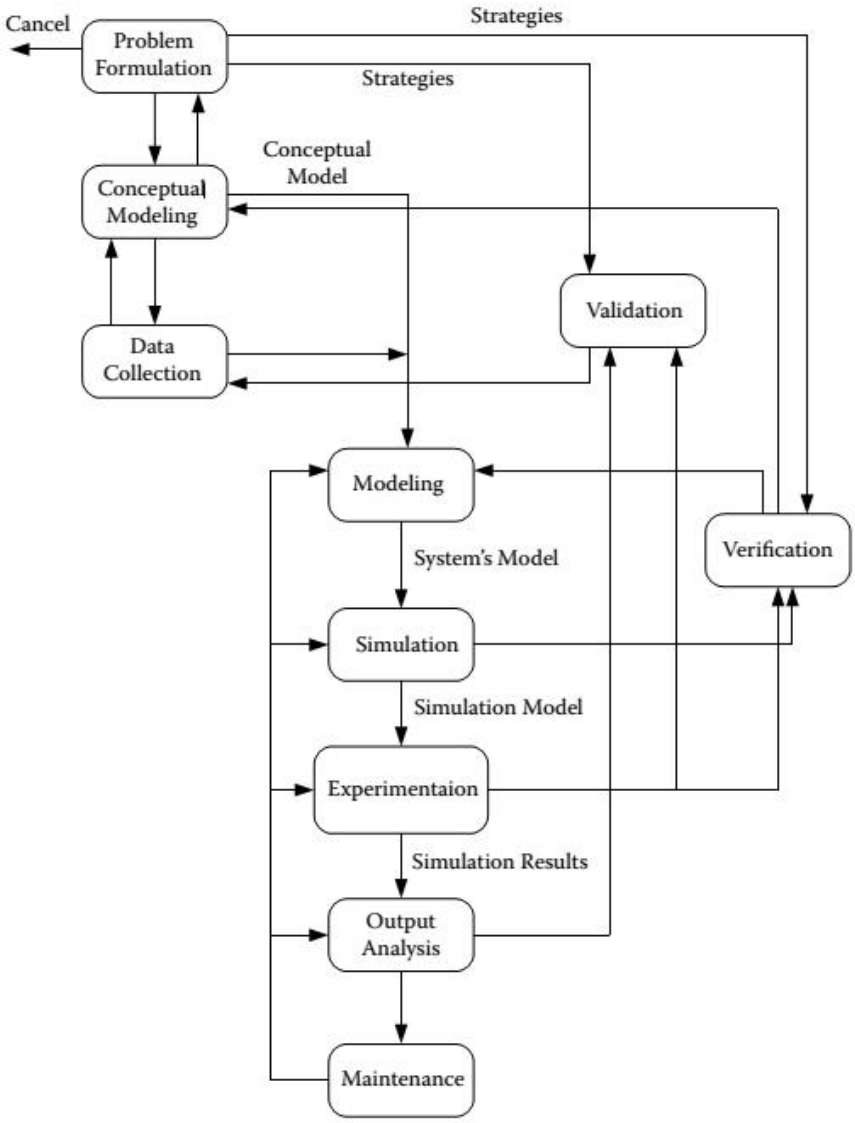


Figura 10 – V&V em estudos de M&S (Wainer – 2009)

3. *Software* utilizado – Anylogic

Para a elaboração do modelo computacional o *software* escolhido foi o Anylogic v7.0. Este programa permite o desenvolvimento de modelos de simulação recorrendo às três principais metodologias em uso: Dinâmica de Sistemas, Simulação de Eventos Discretos e *Agent Based Modeling.* É possível combinar várias destas abordagens de acordo com as necessidades do sistema a modelar. A sua linguagem básica é o JAVA e como tal tem grande grau de compatibilidade e flexibilidade, existem versões para os principais sistemas operativos (Linux, OSX, Windows). Os modelos desenvolvem-se graficamente, com recurso às vastas bibliotecas de objectos que permitem configurar rapidamente elementos prontos a usar. O grande número de exemplos de modelos ajuda na aprendizagem e facilitam a implementação de funcionalidades recorrendo ao que foi feito previamente por especialistas. Permite a utilização de tempo/espaço contínuo ou discreto.

As suas aplicações estendem-se a vários domínios, nomeadamente:

- Cadeias de abastecimento e logística;
- Fluxos de peões (aeroportos, estações, centros comerciais);
- Transporte e armazenamento;
- Gestão de projectos e de activos;
- Processos de negócios e serviços;
- Ferrovia;
- Saúde e defesa;
- Tecnologias da informação e telecomunicações;
- Recursos humanos;
- Gestão e planeamento estratégico.

3.1. Ambiente Gráfico

Como se pode ver na seguinte figura, o ambiente gráfico consiste em cinco elementos principais:

- o separador *Projects* onde se acede aos agentes, simulações e optimizações;
- atrás deste está o separador *Palette*, a que regressaremos em detalhe mais à frente;

- a zona central onde está representado o agente *Main*, é aqui que é feita toda a programação gráfica com as ligações entre objectos;

- o separador *Properties* onde se definem as características e comportamento dos objectos;

- a parte inferior onde é dada informação acerca de erros nos separadores *Problems* e *Console*.

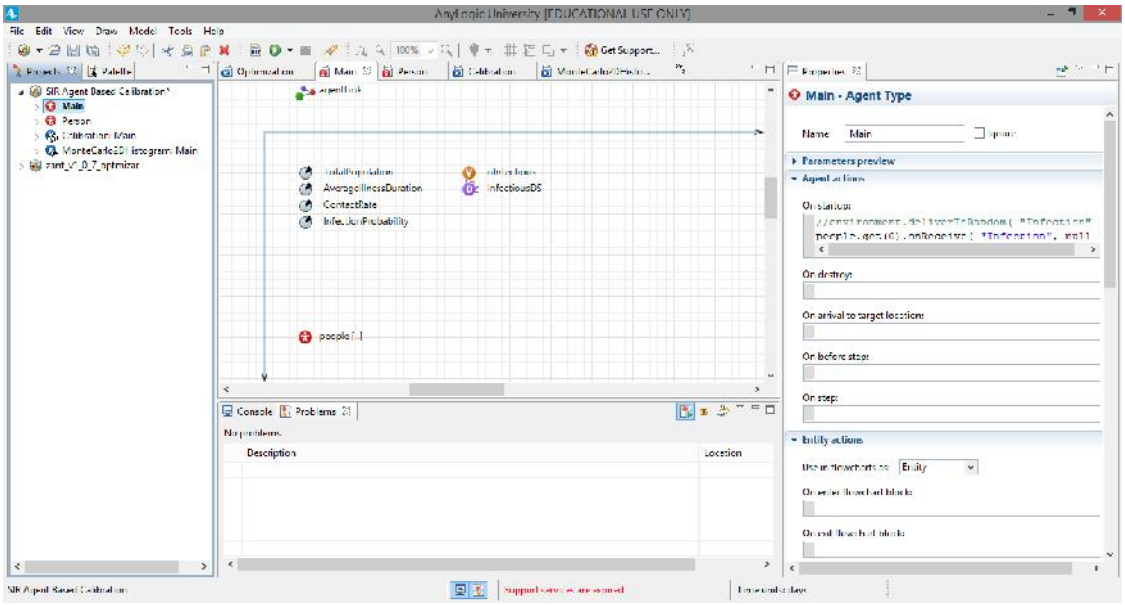


Figura 11 - Ambiente gráfico Anylogic 7.0

No separador *Palette* (fig. 12 e 13) a quantidade de objectos pré configurados dá resposta a praticamente todas as necessidades para a elaboraçao do modelo de forma gráfica com ligações simbólicas entre os elementos, eliminando na sua grande maioria a necessidade de recurso a linguagens de programação convencionais. No primeiro separador procede-se à criação de agentes, parâmetros, variáveis e funções entre outros. Nos dois separadores seguintes temos acesso a elementos gráficos para a apresentação ou delimitação de espaços. O separador Analysis permite a colecção e apresentação de dados em Datasets, agregador de estatísticas que permite elaborar gráficos de vários tipos. Com os elementos constituintes de *Controls* podemos introduzir elementos de controlo úteis para variar parâmetros no decorrer da execução da simulação por forma a variar os resultados. Por último temos os elementos necessários à criação dos diagramas de transição de estado. Nestes podem ser introduzidas acções tanto nas entradas como nas saídas de estado, as transições podem ser automáticas, sujeitas a temporização, mediante condições específicas (como receber uma mensagem de outro agente) entre outras possibilidades. Os losangos indicam possibilidades de várias saídas para uma entrada, em que uma das saídas acontece por omissão e as outras mediante certas condiçoes parametrizáveis.

Existem ainda outras bibliotecas mais específicas destinadas a aplicações em fluxos de peões, de comboios, possibilidade de exportação e ligação a ficheiros externos, imagens 2D/3D.

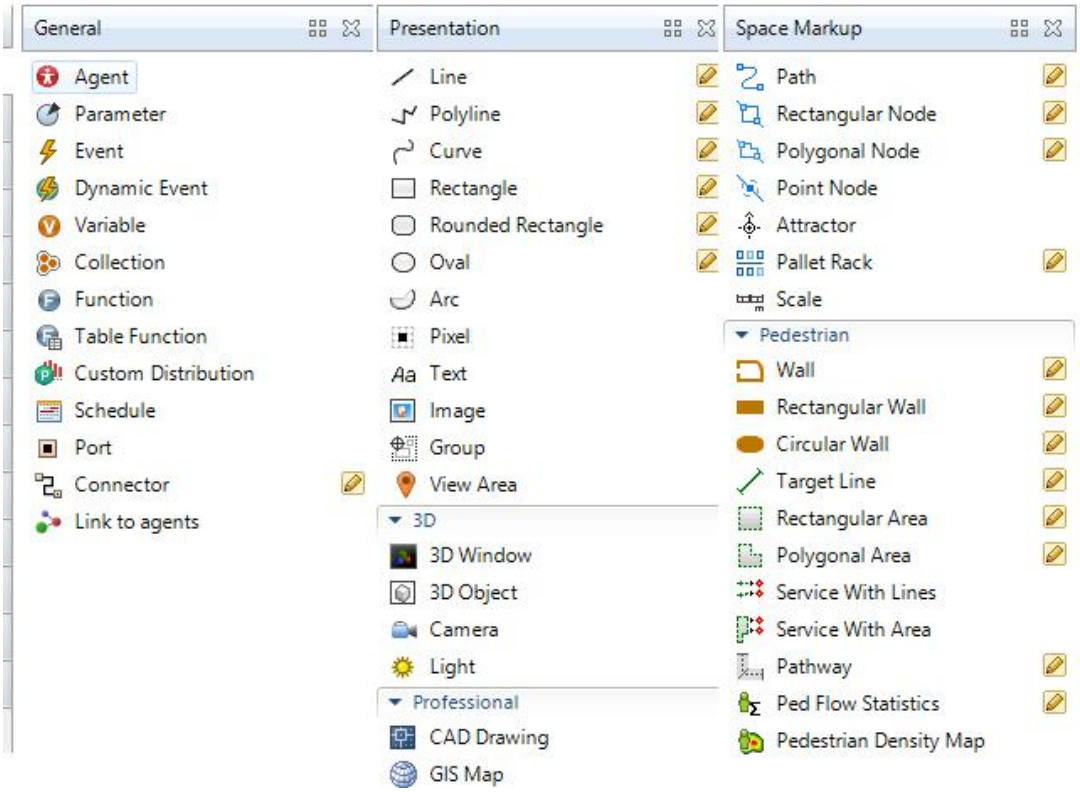


Figura 12- Bibliotecas de objectos disponíveis – parte 1

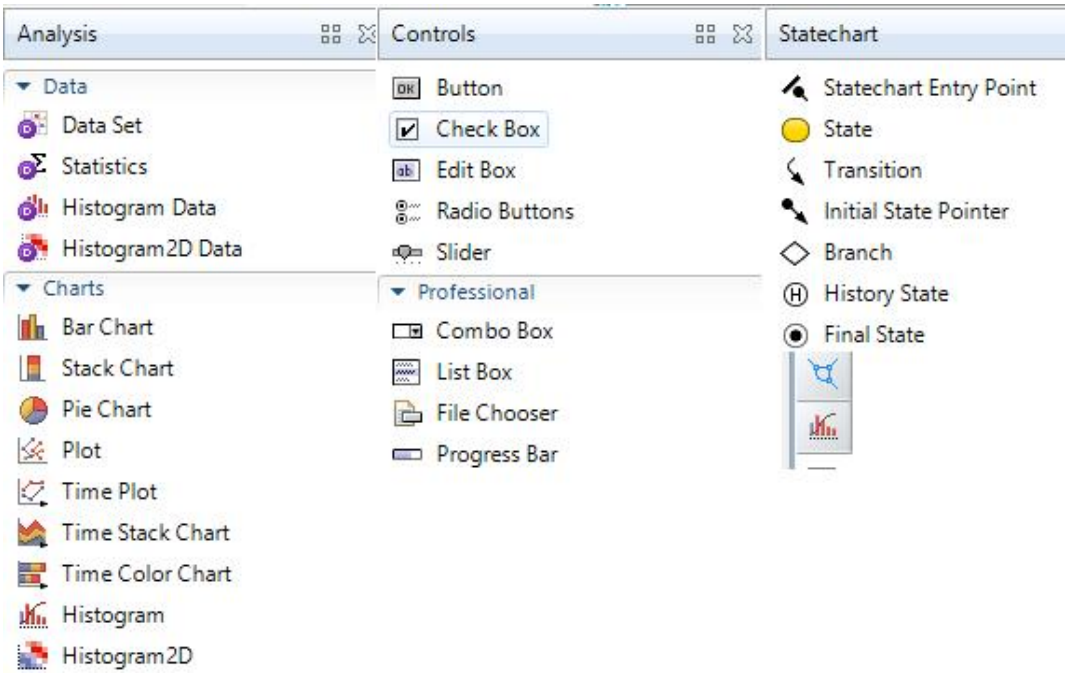


Figura 13 - Bibliotecas de objectos disponíveis – parte 2

3.2. Exemplo de modelo de contágio de doença infecciosa – "*SIR Agent Based Calibration*"

Este modelo está disponível nos exemplos fornecidos com o *software* e serve para ilustrar de forma simples as potencialidades do mesmo. É composto por quatro partes:

– o agente *Main*;

– o agente *people* (que está ligado ao *Person*);

– o Optimizador;

– a Simulação.

No agente *Main* é corrida uma instrução quando é inicializado o modelo que inicia o contágio. Tem quatro parâmetros, a população (que indica o número de agentes *Person* existentes), duração média da doença, taxa de contacto entre as pessoas e a probabilidade de infecção. A variável *nInfectious* regista as pessoas infectadas e tem ainda o *DATASET InfectiousDS*.

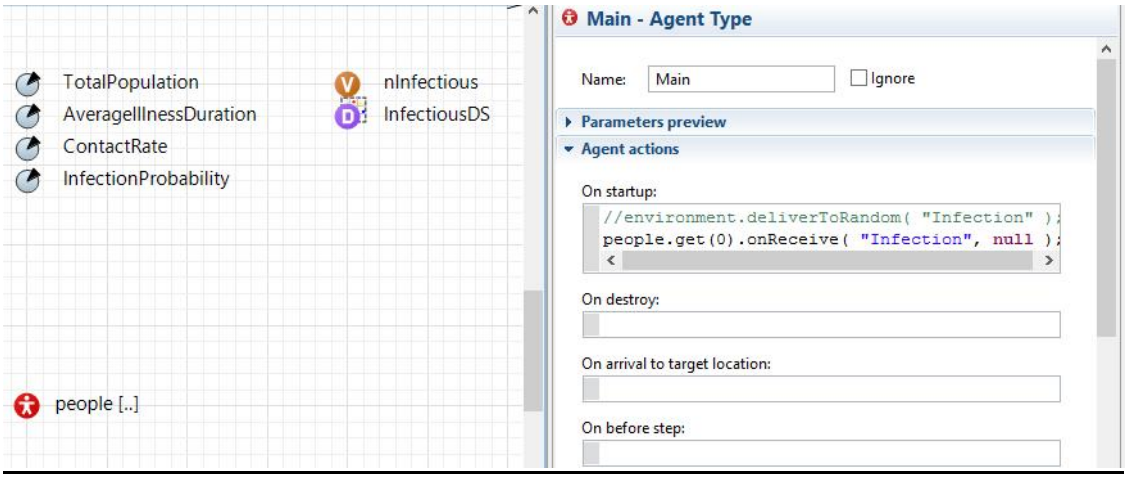


Figura 14 – Modelação ABM de uma epidemia de Tuberculose

O fluxograma do agente *Person* possui estados sucessivos ligados por transições, inicialmente está no estado *Susceptible*, que indica poder vir a ser infectado pela doença. Passa ao estado *Infectious* quando recebe uma mensagem, e nesta situação pode vir a infectar com quem se encontre. Ao entrar neste estado é corrido um código que regista na variável *nInfectious* do *Main* que este agente está infeccioso, ele deixa este estado sujeito a uma taxa dada pelo parâmetro *AverageIllnessDuration* do objecto de topo *Main*. Ao sair deste estado corre o código que retira este agente da variável *nInfectious* e passa ao estado *Recovered*.

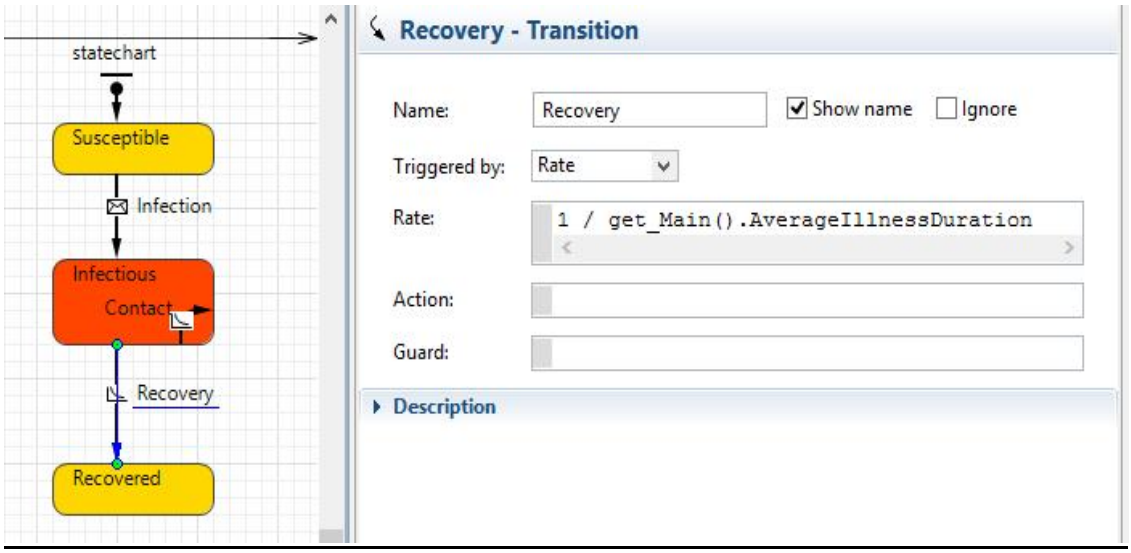


Figura 15 – Agente "Person" do modelo de contágio de uma doença infecciosa

Isto acontece sucessivamente para o número de agentes que vierem a ser infectados.

Este modelo recorre ao Optimizador do *Anylogic*, uma ferramenta poderosa para procura dos parâmetros do modelo que melhores resultados proporcionam. Define-se um objectivo que pode ser por exemplo a maximização de lucro, minimização do tempo de espera de um dado agente, o que se pretender analisar. Estabelecem-se que parâmetros podem ser variados e entre que valores, que pré-requisitos e constrangimentos devem ser aplicados, qual o número de replicações a utilizar ou se se prefere antes que este número varie por forma a atingir objectivos de erro e níveis de confiança específicos. Depois de definidos os objectivos, corre-se o optimizador e usam-se os parâmetros obtidos na simulação do modelo.

No caso apresentado na figura abaixo, para a população potencialmente infectável pela doença corre-se o optimizador permitindo a variação dos valores de *ContactRate* e *InfectionProbability* por forma a minimizar a diferença entre os valores obtidos e os valores historicamente registados.

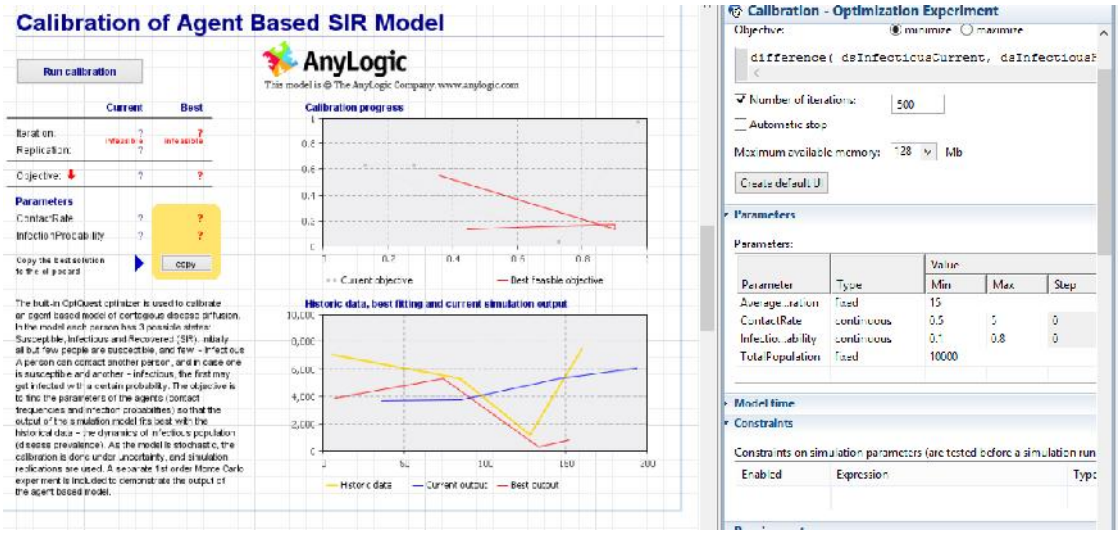


Figura 16 – Calibração dos entre os parâmetros do modelo e dados históricos conhecidos

Com 500 iterações obteve-se um valor de 4.535 para a *ContactRate* e 0.136 para a *InfectionProbability*.

Após isto, corre-se a simulação do modelo com os valores e os parâmetros obtidos e tem-se o modelo da evolução da doença nesta população, com o pico de infecções a ser atingido cerca de 20 dias após o paciente zero, e a erradicação da mesma a acontecer cerca de 80 dias depois do começo do surto.

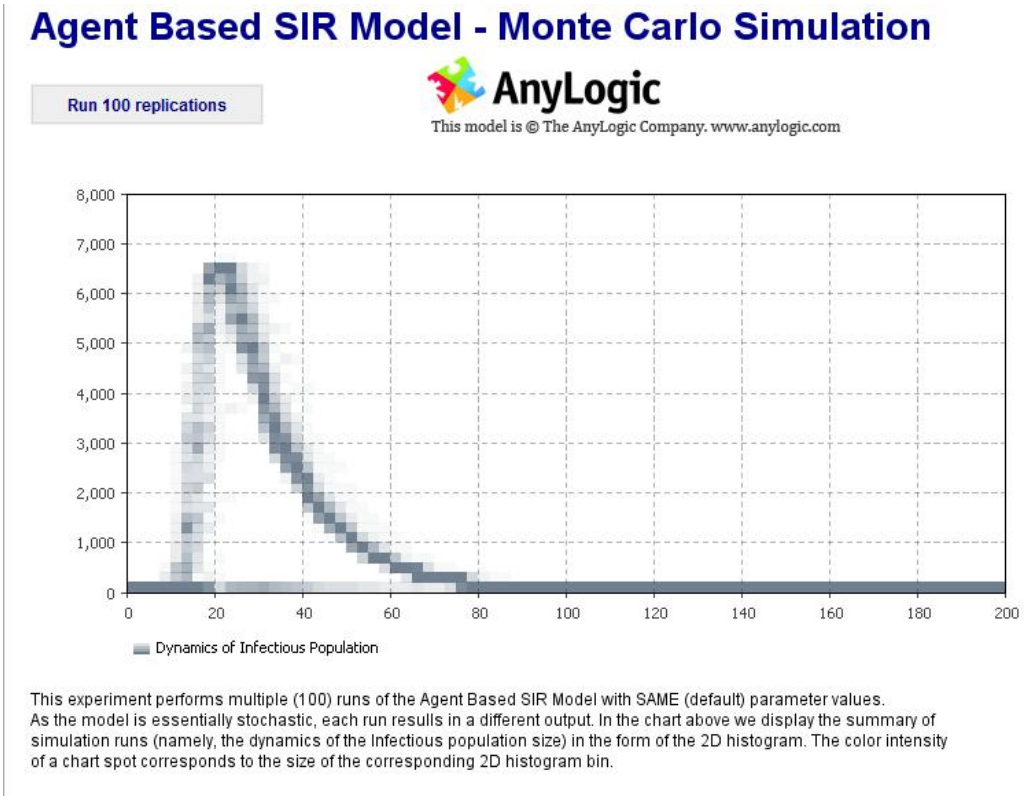


Figura 17 – Resultados da simulação

3.3. Vantagens/desvantagens

Segundo Osgood (*Anylogic Lectures – 2012*), as principais vantagens do Anylogic quando comparadas com outros programas de modelação ABM são:

- Mecanismos essencialmente declarativos na construção dos modelos. Isto permite ao modelador focar-se mais no que está a modelar e não na forma como implementar, o programa trata de encontrar maneira de implementar.

- Menos código;

- Grande flexibilidade;

- Acesso a bibliotecas *JAVA*;

- Suporte a vários tipos de modelação;

- Suporte a combinações de tipos de modelação.

Desvantagens:

- Exportação de resultados dos modelos ainda com pouco suporte para análise da forma como se chegou a eles;

- Necessidade de conhecimento do código JAVA;
- Modelo com muitas partes/secções;
- Preço da versão com acesso a *debugger* (cerca do dobro da normal).

4. Empresa

A empresa em estudo presta serviços de assistência técnica, manutenção e reparação de sistemas de aquecimento, climatização e produção de água quente. O seu leque de clientes abrange tanto a área industrial, como clientes domésticos. O cerne da sua actividade foca-se na região centro do país, mas para a elaboração deste estudo apenas consideraremos a zona que abrange os distritos de Aveiro, Viseu e Guarda, parte de Coimbra e Castelo Branco.

A empresa é sediada em Viseu.

4.1. Objectivo

Pretende-se simular a actividade da empresa 4 Climas ao nível da prestação de serviços de assistência pós-venda e manutenção periódica à carteira de clientes, tendo em conta as taxas de avaria previsíveis, necessidades de substituição de equipamentos, procurando dimensionar o número de equipas de assistência necessário para garantir o nível de serviço previsto. De uma forma simples, a empresa recebe um pedido de assistência ou manutenção, este é colocado numa lista de espera após o que é entregue a uma equipa que presta a assistência. Pode ser dito que o caso desta empresa é semelhante ao de qualquer outra prestadora de serviços pós-venda e, como tal, o modelo a criar será aplicável à maioria das empresas deste tipo.

4.2. Dados

Para o efeito de simulação do serviço de assistência da empresa e dimensionamento da quantidade de equipas necessárias para cumprimento do mesmo foram considerados os seguintes dados fornecidos pela própria e ainda alguns outros recolhidos por Pedro Campos para o seu trabalho de dissertação. O foco do trabalho foi na modelação do sistema e menos no

tratamento estatísticos dos dados de origem, trabalho esse feito de forma completa pelo colega referido.

O número de clientes é de cerca de 800, entre clientes domésticos e industriais, considerou-se uma distribuição triangular para os tempos das intervenções, o MTBF segue uma distribuição normal com o desvio padrão referido na tabela. O período de manutenção recomendado é o indicado.

Reparação (horas)			Manutenção (horas)			Substituição (horas)		
Optimista	Normal	Pessimista	Optimista	Normal	Pessimista	Optimista	Normal	Pessimista
0.5	1	2.5	1	1.5	2	6	8	10
MTBF (dias)			Desvio Padrão (dias)			Período Manutenção (dias)		
333			60			365		

Tabela 1 – Dados relativos às intervenções

5. Modelo

Para modelar o sistema de prestação de serviços de assistência e manutenção da empresa, e tendo em conta o objectivo de implementação posterior deste modelo como um modelo *ABM* cada um dos intervenientes no sistema é modelado enquanto agente individual.

De forma simples, pode-se dizer que a empresa funciona da seguinte forma:

Figura 18 – Modelo de funcionamento do serviço pós venda da empresa

O pedido de assistência ou manutenção é recebido pelos serviços da empresa, é colocado numa fila de espera e quando chega a sua vez uma equipa é destacada para prestar assistência ao cliente.

Podemos neste modelo simples identificar dois agentes: a equipa de assistência que presta o serviço e o equipamento do cliente em que este é prestado. Assim, é necessário conceptualizar o comportamento destes dois intervenientes no sistema, sendo que eles interagem num espaço comum, o da área onde a empresa exerce as suas funções.

5.1. Espaço

Uma vez que o cerne da actividade da empresa se centra nos distritos de Aveiro, Viseu e Guarda, consideramos um espaço contínuo rectangular de 170 quilómetros de largura por 100 de altura com a cidade de Viseu em posição central. No modelo, as equipas de assistência viajarão linearmente entre a sede da empresa e os locais onde prestam assistência.

Embora a versão 7.1 do programa apresente a possibilidade de recurso a mapeamento GIS e a movimentação segundo percursos rodoviários, na versão disponibilizada (7.0) isto ainda não é

possível. As coordenadas das localidades foram retiradas com recurso ao sistema de coordenadas do programa Autocad.

As distâncias são calculadas linearmente, afectadas de um factor de correcção teórico encontrado com recurso às distâncias fornecidas pelo sítio Via Michelin para principais localizações. Para obter maior rigor dever-se-ia considerar a frequência de intervenções em cada local, mas não se considerou necessário para o nível de precisão necessário no cálculo de tempos de deslocação e é implementada no modelo reduzindo a velocidade de deslocação do agente na percentagem indicada na tabela seguinte.

Local	Coord.(km)		Dist. Calc. (km)	Via Michelin	
	x	y		Km	%
Viseu	0	0	0,0	0	
Águeda	-44,4	-8,6	45,2	68	50,4%
Aguiar da Beira	32,5	15,6	36,1	42	16,5%
Aveiro	-54,8	3,4	54,9	86	56,6%
Castro Daire	1	28,8	28,8	35	21,5%
Celorico da Beira	43,9	-2,8	44,0	50	13,7%
Figueira da Foz	-75	-50	90,1	126	39,8%
Guarda	56,3	-16,8	58,8	76	29,4%
Lamego	11,24	49,2	50,5	62	22,9%
Mangualde	12	-7	13,9	17	22,4%
Mealhada	-44	-28	52,2	72	38,1%
Moimenta da Beira	24,23	34,5	42,2	52	23,3%
Nelas	5	-14,1	15,0	22	47,1%
Oliveira do Hospital	2,6	-33,8	33,9	52	53,4%
Penalva do Castelo	19	3,8	19,4	25	29,0%
Santa Maria da Feira	-49,3	28,9	57,1	99	73,2%
Tondela	-12	-18	21,6	27	24,8%
Vila Nova de Foz Côa	66,6	41,4	78,4	113	44,1%
Vouzela	-13,4	9	16,1	28	73,5%
				Desvio	**38,0%**

Tabela 2 – Distâncias das principais localidades da área de operação

5.2. Pressupostos

Para a elaboração do modelo foram assumidas algumas simplificações. Isto acontece por várias razões, desde inexistência de dados para poder emular com fiabilidade as opções pretendidas, por entender que o aumento de complexidade de programação não seria compensado com correspondente ganho de rigor nos resultados e/ou por ter sido atenção de trabalhos anteriores.

Assim, em cada simulação os seguintes pressupostos foram considerados:

- carteira de clientes estável;

- ausência de sazonalidade;

- clientes aceitam sempre substituição de equipamentos se necessária;

- localização de clientes aleatória no espaço (embora a concentração nos aglomerados urbanos seja inegável, os ganhos em rigor no modelo seriam negligenciáveis face à necessidade de introdução quase individual de registos);

- o modelo e seus intervenientes funcionam em modo contínuo, nas equipas de assistência isto implica que se considera como velocidade a distância semanal possível por equipa, e a duração das intervenções é multiplicada por 4.2 (horas de uma semana a dividir por horário de trabalho normal);

- usou-se como unidade de tempo o dia e para distância o quilómetro.

5.3. Modelação Anylogic

As unidades consideradas neste modelo são de dias para o tempo, e km para a distância e explicitamente definidas no programa. Para modelar no Anylogic, as equipas de assistência e equipamentos dos clientes serão duas classes de agentes, o espaço, quantidades de cada um destes agentes, gestão de pedidos de assistência/manutenção, gestão de listas de espera e todos os parâmetros globais serão definidos no elemento base do modelo *Main.*

5.3.1. Objecto de topo - *Main*

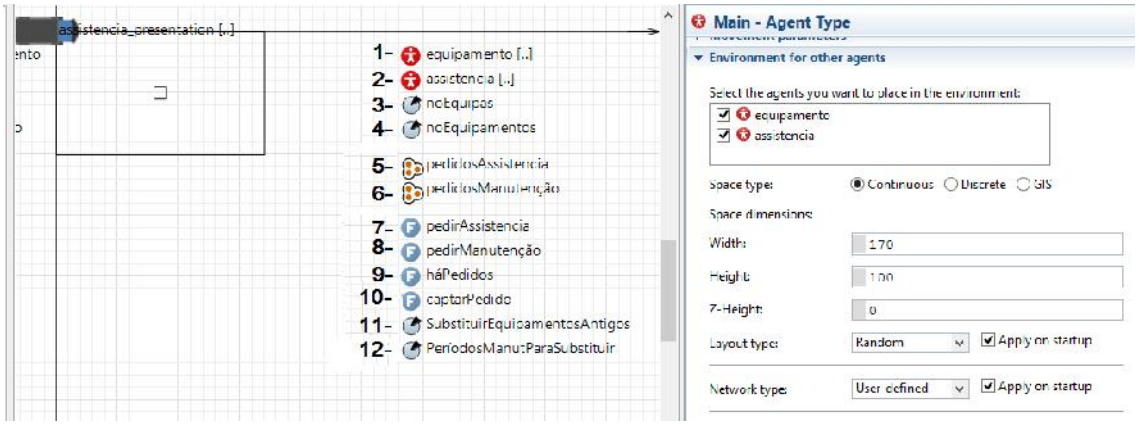


Figura 19 – Objecto *Main*

Aqui, para além do referido anteriormente, são também colocados elementos ligados à apresentação gráfica da simulação, desde gráficos a animações, assunto a que se voltará aquando da descrição do correr da simulação. Existem ainda outras funções deste objecto *Main* que apenas serão descritas quando forem ligadas com a descrição dos agentes aqui intervenientes, como por exemplo a colecção de estatísticas relativas aos agentes e todo o sistema de gestão de filas de espera e de geração de pedidos.

No separador *Environment* é definido o espaço contínuo onde os agentes são colocados. As localizações dos equipamentos dos clientes são geradas de forma aleatória ao inicializar a simulação, as equipas de assistência são colocadas na sede da empresa, no ponto *(x,y)=(85,50)*.

Os agentes *equipamento* e *assistência* aqui representados respectivamente pelo número 1 e 2 da figura 19 representam a ligação aos agentes *EquipaAsssistência* e *EquipamentoCliente* cuja descrição em pormenor será efectuada mais à frente. Neste objecto são definidas as quantidades de cada agente que interagem no espaço físico. Para o caso do *EquipamentoCliente*, este agente foi definido como uma "População de agentes" e o seu número é dado pelo parâmetro *noEquipamentos* (número 4 da figura 19). Uma vez que é um número inteiro este valor deve ser definido com o tipo de dados *int*, números decimais serão definidos como *double*.

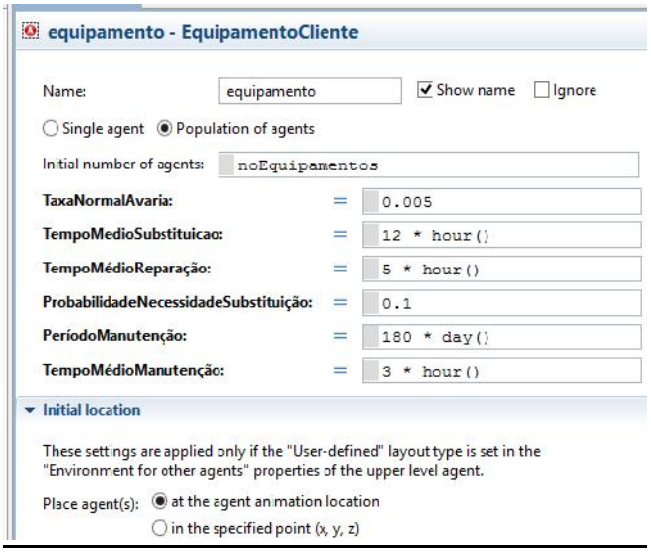


Figura 20 - Propriedades do agente *equipamento* em *Main*

Os parâmetros aqui apresentados representam uma ligação aos definidos no fluxograma do agente, e podem ser alterados em qualquer dos locais, embora por questões de coerência isto apenas seja feito no objecto de topo *Main*.

O agente *assistência* em *Main* apenas tem como parâmetro *noEquipas* (nº 3 na figura), definindo a quantidade de equipas que actuam no modelo. O parâmetro 11 é do tipo booleano e corresponde à política de substituição de equipamentos.

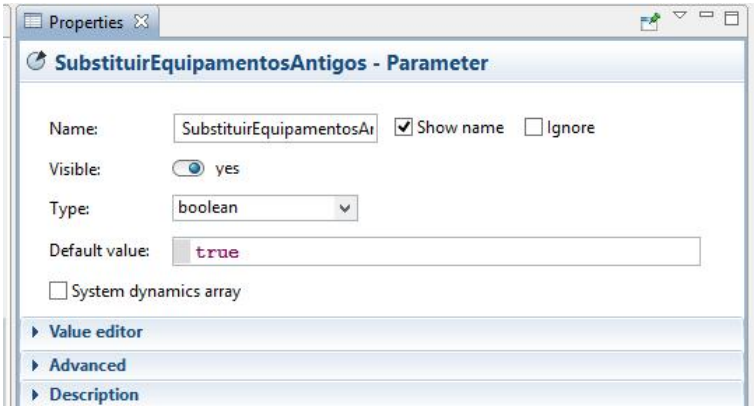


Figura 21 - Parâmetro que define política de substituição de equipamentos

Por último, o parâmetro 12 da figura 19 indica (caso a política de substituição esteja activa) qual o número de períodos de manutenção que devem corresponder ao prazo de substituição dos equipamentos.

5.3.2. Sistema de gestão de filas de espera e geração de pedidos de assistência e manutenção

A necessidade de assistência e/ou manutenção é gerado nos equipamentos dos clientes por *timeouts* que serão explicados no diagrama de estado do agente *EquipamentoCliente.* No entanto, toda a gestão das filas de espera desses pedidos é feita no objecto de topo *Main.*

Consiste em dois elementos de colecta de dados do tipo *LinkedList* e classe de elemento *EquipamentoCliente* (nº 5 e 6 da figura 19) para registo da fila de espera dos pedidos:

Figura 22 – *Collection* dos pedidos de assistência e manutenção

Quando o *timeout* de avaria de um *EquipamentoCliente* é activado, é enviado um pedido de assistência para esse equipamento, recorrendo à função número 7 – *pedirAssistência.*

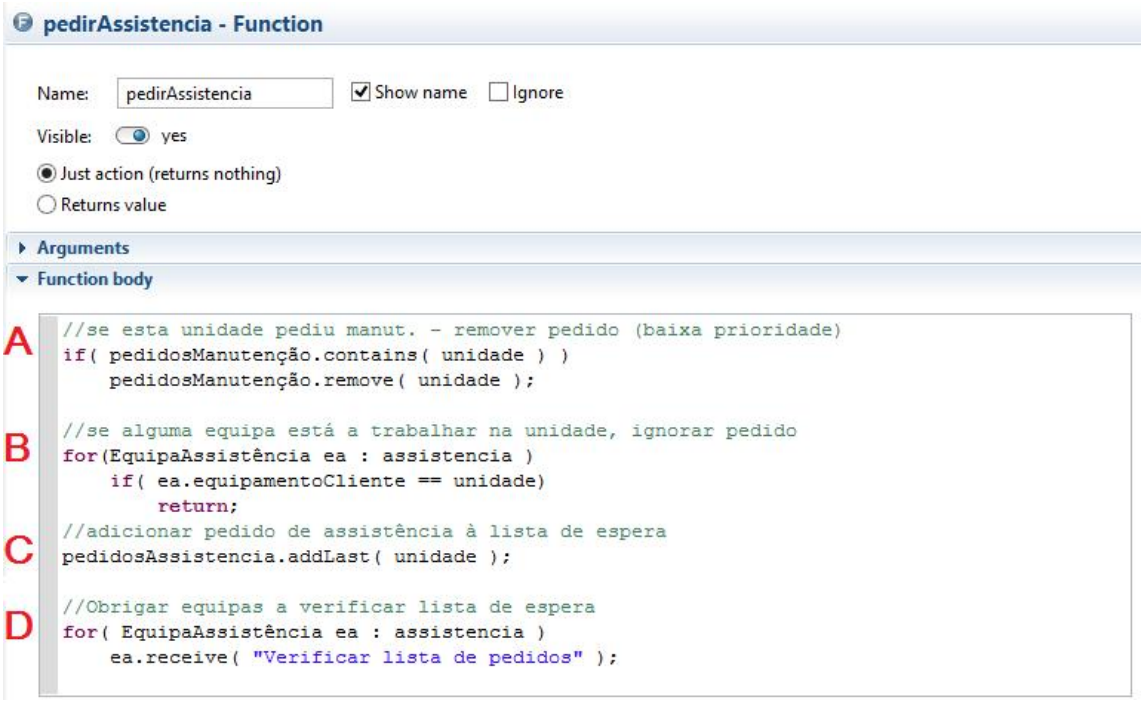


Figura 23 - Função pedirAssistência

Esta função apenas toma as seguintes acções:

A: Se a unidade pediu manutenção antes de efectuar o pedido de assistência, remove esse pedido da lista de espera uma vez que tem menor prioridade do que o pedido de reparação uma vez que o equipamento está indisponível.

B: Verifica se alguma equipa de assistência está a trabalhar no equipamento, corre todas as equipas e se nalguma esta situação for verdadeira: "*ea.equipamentoCliente = = unidade*" executa o comando *return;* e sai da função *pedirAssistência*.

C: Tendo passado por A e B, acrescenta aos pedidos de assistência o equipamento do cliente em causa.

D: Força todas as equipas de assistência a verificar a lista de pedidos.

Para a função 8 – *pedirManutenção*, o funcionamento é análogo.

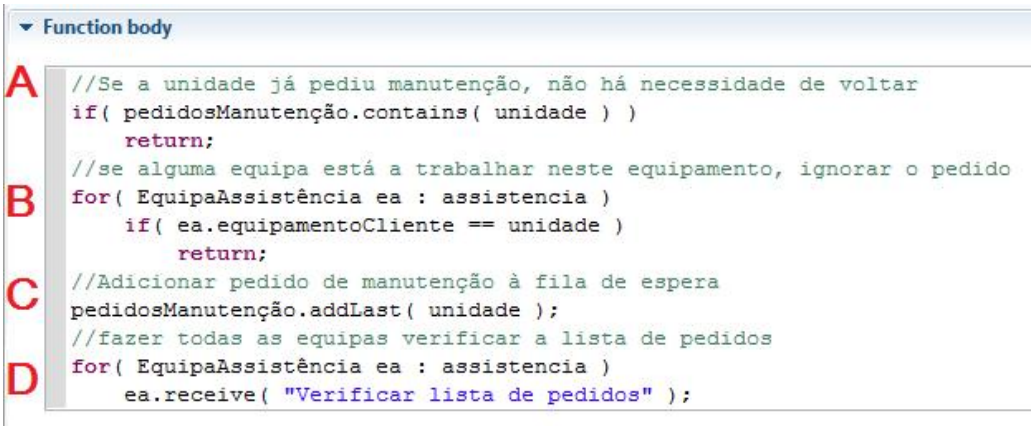


Figura 24 - Função *pedirManutenção*

A: Se ao verificar os pedidos de assistência esta unidade já tiver pedido assistência ou manutenção não há necessidade de pedir de novo e a instrucção *"return;"* sai da função .

B: Verifica se alguma equipa de assistência está a trabalhar no equipamento, corre todas as equipas e se nalguma esta situação for verdadeira: "*ea.equipamentoCliente = = unidade*" executa o comando *return;* e sai da função *pedirManutenção*.

C: Se não tiver saído da função em A ou B, acrescenta aos pedidos de manutenção o equipamento do cliente.

D: Força todas as equipas de assistência a verificar a lista de pedidos.

A função número 9 - *háPedidos* é uma condição booleana, e é activada a partir do agente *EquipaAssistência* quando este pretende informar-se se existem pedidos.

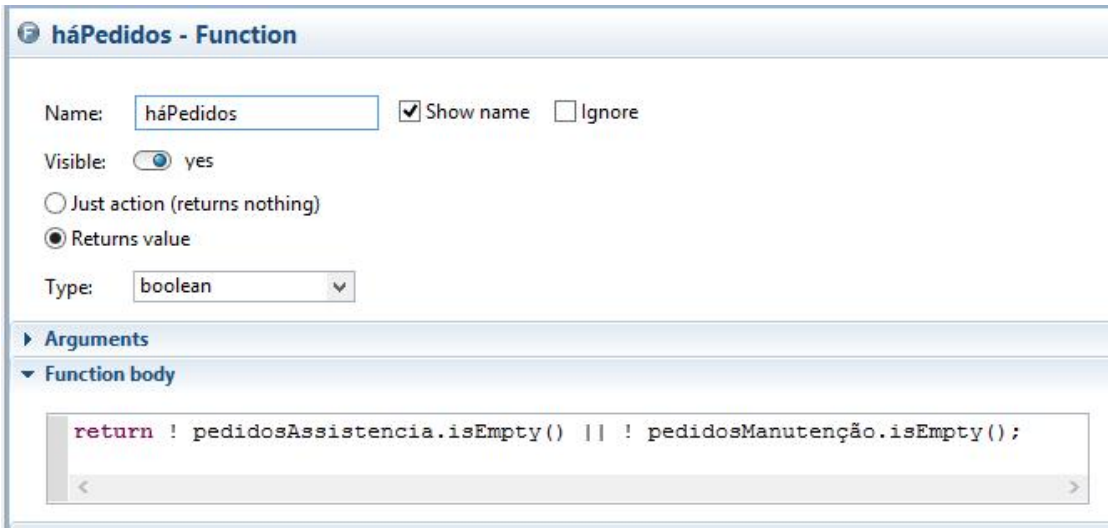


Figura 25 - função *háPedidos*

Esta função ao ser activada responde verdadeiro se o complementar (o *"!"* em linguagem Java) de *pedidosAssistência.isEmpty( )* ou ("||") *pedidos.Manutenção.isEmpty( )* ocorrer. Ou seja, se existirem pedidos de manutenção ou assistência o valor da função será verdadeiro.

No caso da condição imposta pela função ser verdadeira, o agente *EquipaAssistência* invocará a função 10 *captarPedido* cujo resultado será um equipamento de cliente:

Figura 26 – função *captarPedido*

Começa por verificar a lista de pedidos de assistência e se não estiver vazia apaga o primeiro Equipamento de cliente da lista e dá o como resultado (onde será prestada assistência pela equipa que captou o pedido). Se não existirem passa aos pedidos de manutenção onde procede da mesma forma.

5.3.3. Agente - *EquipaAssistência*

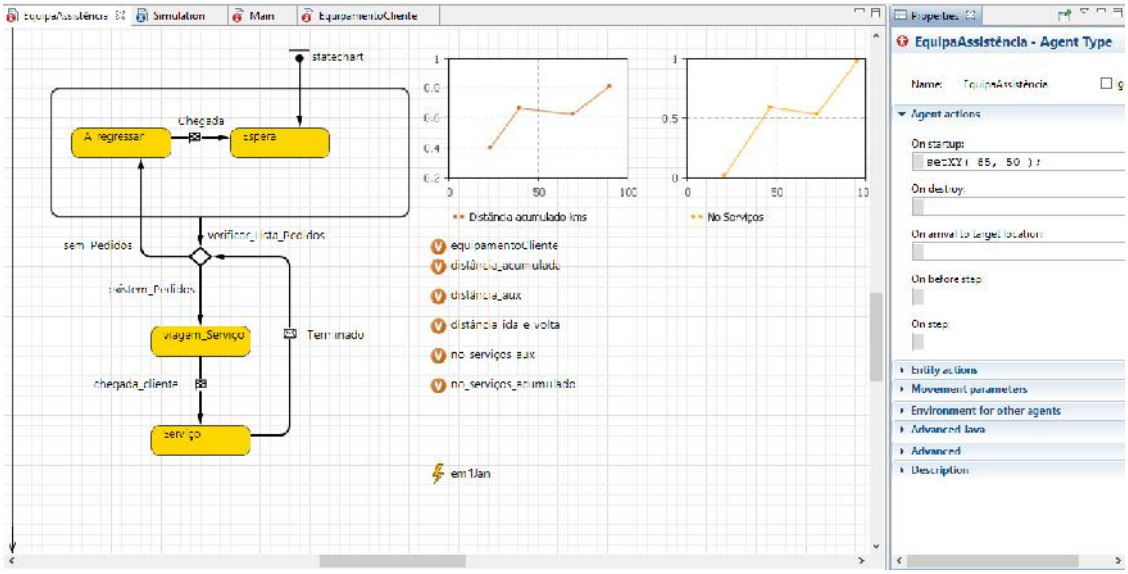


Figura 27 - Diagrama de estado do agente *EquipaAssistência*

Ao correr a simulação, o número de agentes definidos pelo parâmetro *noEquipas* no objecto *Main* são colocados no ponto correspondente à sede da empresa. Os agentes passam automaticamente para o estado *Espera*, já que é a única função do ponto de entrada. O estado *Espera*, *A_regressar*, a transição de estado 6 e 7 correspondem a um estado composto. Existem uma série de variáveis para cálculos de distâncias e número de serviços e uma que regista o equipamento em que é prestada assistência.

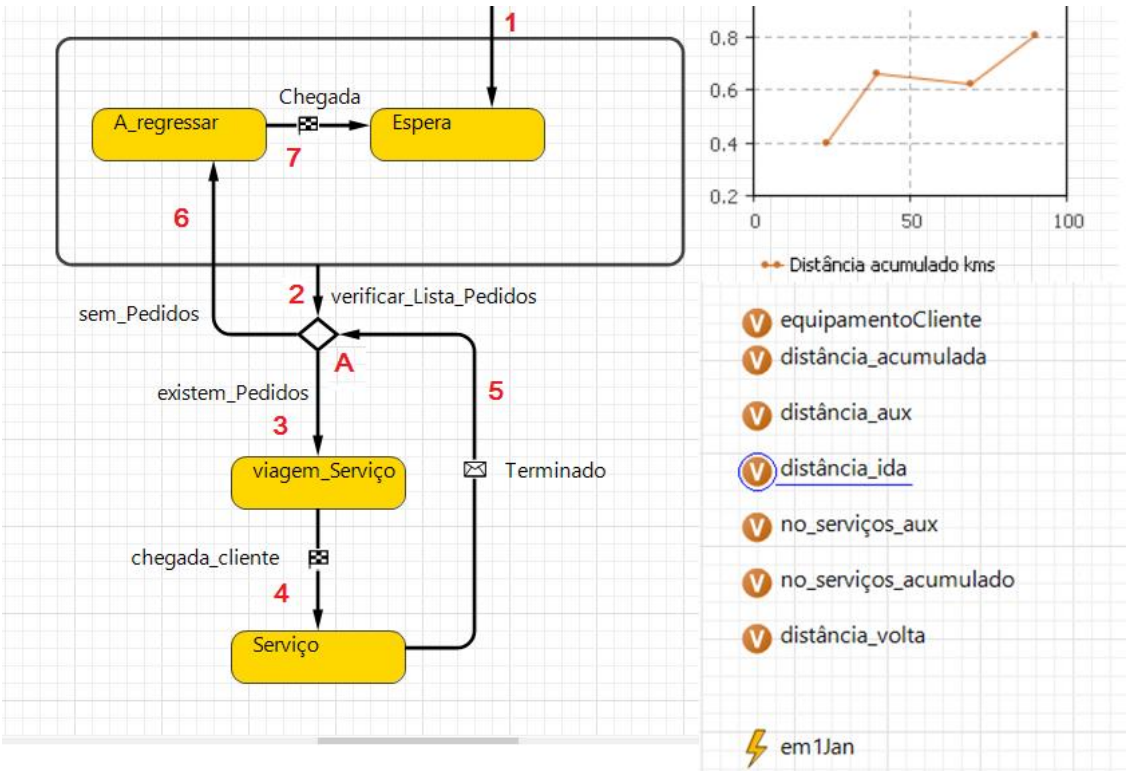


Figura 28 - Transições de estado do agente

A equipa de assistência aguarda até receber ordem (gerada pelo sistema de gestão de pedidos do objecto *Main*) na forma de uma mensagem do tipo *String* com o formato e conteúdo indicado na figura seguinte para verificar a lista de pedidos (transição 2), esta ordem é recebida em qualquer ponto do estado composto, transições 6 e 7, estado *A_regressar* ou *espera*:

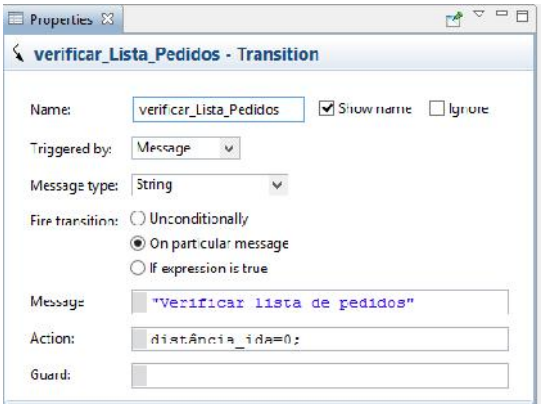


Figura 29 – Mensagem que activa a transição 2

Aqui, o diagrama de estado apresenta o *Branch* A, representado por um losango que permite duas entradas e duas saídas. A transição 6 *sem_Pedidos* é o caminho *default* (no Anylogic estes ramos são seguidos se todas as outras opções forem falsas) que é tomado se não existirem pedidos. Aqui regista-se a variável *distância_ida* como tendo o valor nulo por questões que se relacionam com o cálculo de distâncias acumulado e se tornarão transparentes posteriormente.

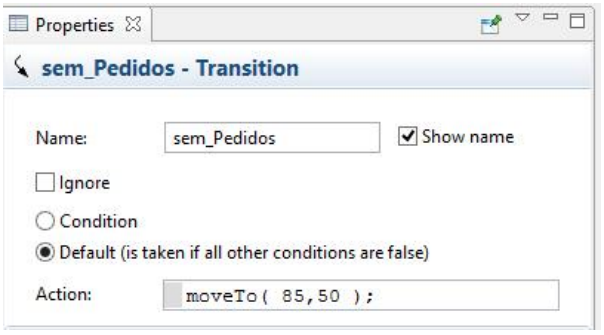


Figura 30 - Transição 6 - *sem_Pedidos*

Esta transição é comum à que é tomada quando a equipa regressa após efectuar assistência num qualquer equipamento do cliente e como tal as instruções são genéricas e aplicáveis nos dois casos, sendo que se a equipa estiver em espera na sede da empresa a instrução *moveTo* não terá qualquer efeito enquanto para uma equipa situada num qualquer equipamento de cliente no espaço terá como resultado uma deslocação linear à velocidade definida nos parâmetros do agente.

No caso de existirem pedidos, ocorrerá a transição 3 – *existem_Pedidos*, transição que só acontecerá se as condições definidas forem verdadeiras.

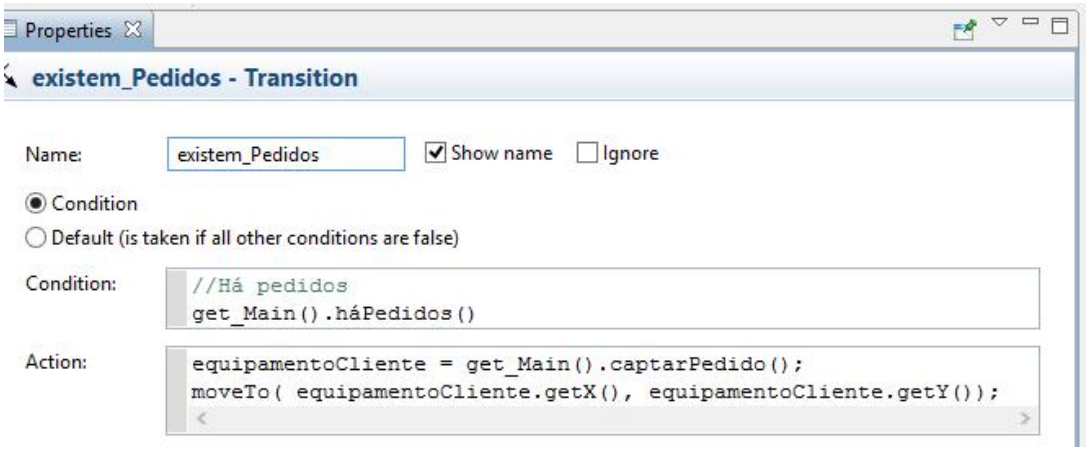


Figura 31 – Transição 3 – *existem_Pedidos*

Aqui o agente terá que verificar se existem pedidos com a condição de *get_Main.háPedidos()* ser verdadeira. Em caso afirmativo é tomada a acção de registar na variável *equipamentoCliente* qual o equipamento que necessita de assistência e é efectuada a movimentação linear para o mesmo com a instrução *moveTo* coordenada *x* e *y* do equipamento após o que entra no estado *viagem_Serviço.*

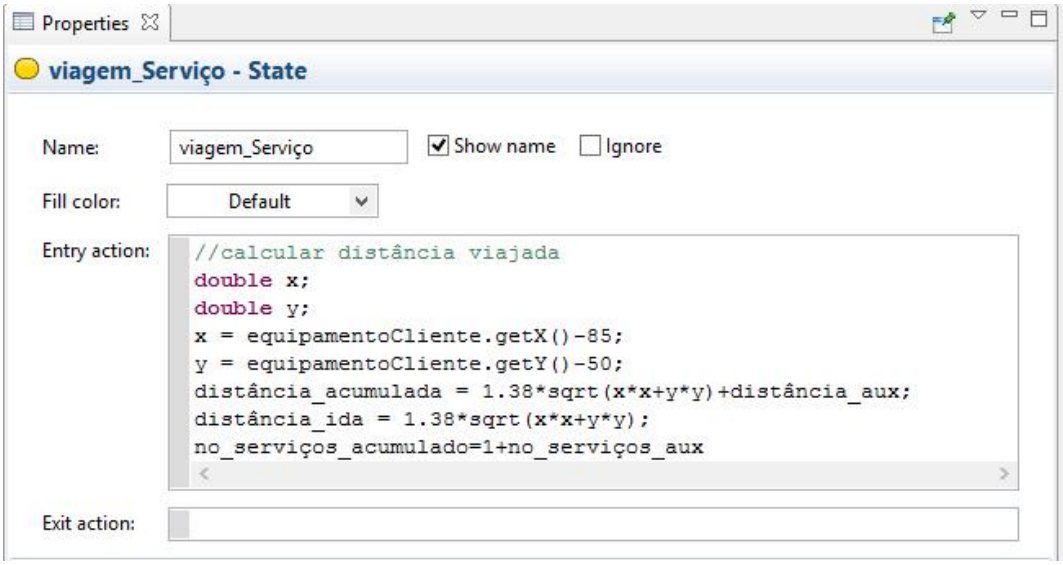


Figura 32 – Estado *viagemServiço*

Este estado é utilizado para alguns cálculos auxiliares não fundamentais para o desenvolvimento da simulação, mas que podem revelar-se úteis para aferir se a distribuição da carga de trabalho pelas equipas está a ser feita de forma equilibrada. Assim, ao entrar no estado definimos *x* e *y* como sendo números do tipo *double*, após o que definimos *x* como o valor em quilómetros da diferença segundo as abcissas da posição do equipamento e a sede da empresa. Procede-se de igual modo para o *y*. Calculamos a distância da viagem de ida com recurso ao teorema de Pitágoras acrescido dos 38% de excesso anteriormente definidos como valor necessário para correcção de distância e com recurso a variáveis auxiliares registada num ponto posterior do diagrama de estada calculamos a distância acumulada. Procede-se de igual forma para o número de serviços efectuado por esta equipa de assistência.

A transição 4 é despoletada pela chegada do agente equipa_Assistência à localização do equipamento do cliente. Aqui o agente toma a acção *send(this, equipamentoCliente)*, que envia a informação que *este* agente está a trabalhar neste *equipamentoCliente* específico.

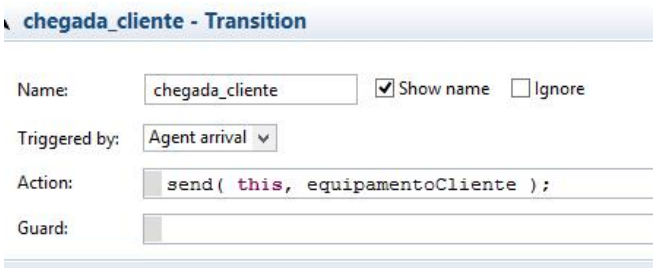


Figura 33 – Transição 4 – *chegada_cliente*

A equipa de assistência passa então para o estado *Serviço*, que é aproveitado para fazer o registo das variáveis auxiliares necessárias para cálculo de distâncias e número de serviços acumulado.

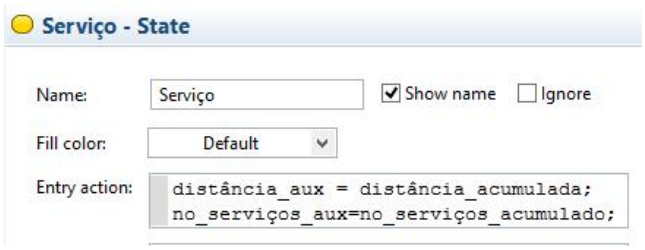


Figura 34 - Estado *Serviço*

O estado *Serviço* é deixado quando é recebida a mensagem informando que o serviço de assistência e ou manutenção está terminado. Esta gestão é feita no diagrama de estado do agente EquipamentoCliente segundo os *timeouts* lá definidos para o tipo de serviço a efectuar, que quando terminado procede ao envio da mensagem que liberta a equipa de assistência.

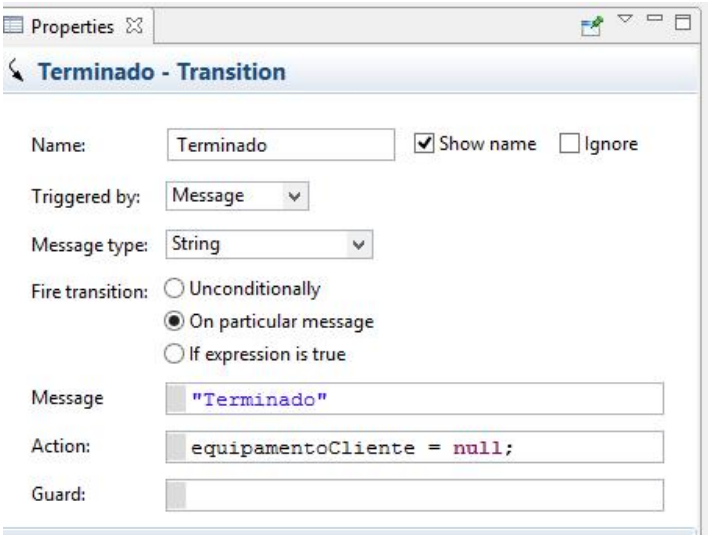


Figura 35 - Transição 5 – *Terminado*

Uma vez que a equipa deixa o equipamento do cliente, é registado o valor desta variável como sendo nulo.

Após a transição 5, a equipa de assistência retorna ao *branch* A, e assumindo que não há pedidos toma a transição 6 e move-se em direcção à sede. O estado A_regressar não tem nenhuma acção, e a transição *chegada* é despoletada pela chegada do agente.

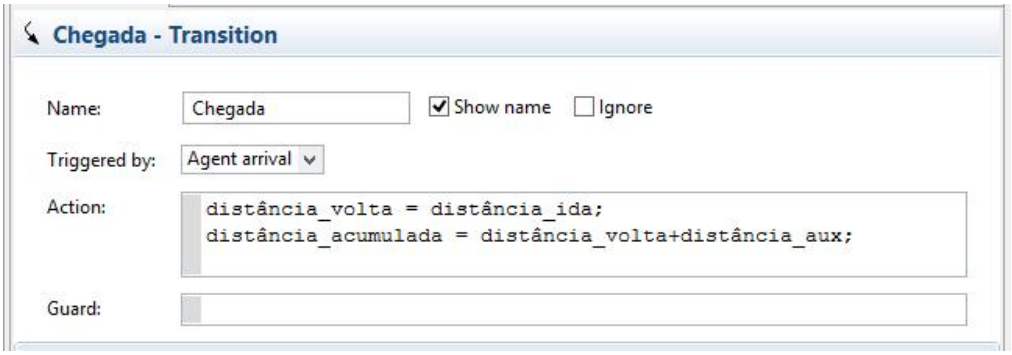


Figura 36 – Transição *chegada*

Aqui, uma vez que a equipa de assistência não seguiu para outro equipamento sabemos que efectuou a viagem de regresso. Desta forma, igualamos a distância da viagem de volta à da viagem de ida, e calculamos a nova distância acumulada.

Finalmente, o agente volta ao estado inicial enquanto aguarda pela chegada de pedidos de assistência, onde é registado de novo o valor da variável auxiliar para cálculo de distâncias.

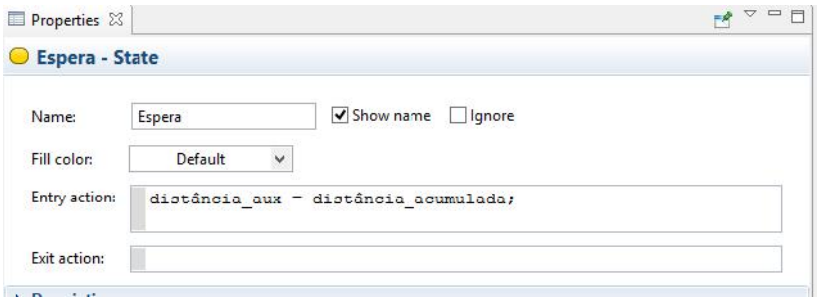


Figura 37 – Estado *Espera*

Este diagrama de estado do agente possui ainda dois gráficos que quando corremos a simulação nos dão o número de serviços que a equipa de assistência prestou e o número de quilómetros que viajou.

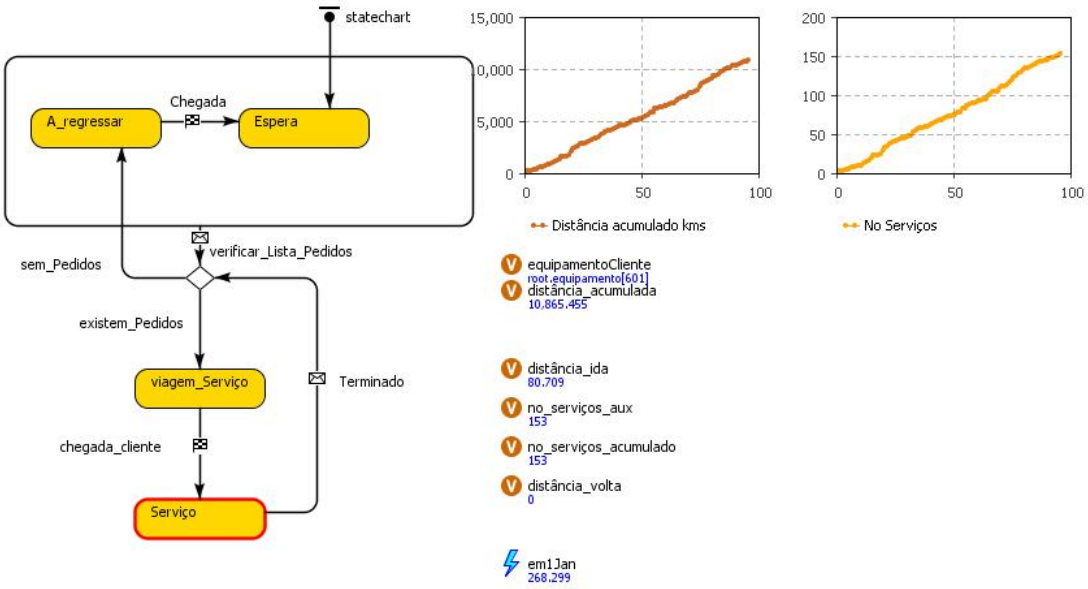


Figura 38 – aspecto da simulação de um dos agentes *equipaAssistência*

Finalmente tem-se um evento que corre todos os dias 1 de Janeiro, que pode repor os valores de distância e número de serviços a zero, para comparar valores anuais ou estar inactivo permitindo nos gráficos a visualização dos totais acumulados.

5.3.4. Agente – *EquipamentoCliente*

No arranque da simulação define-se desde logo o temporizador de manutenção para cada equipamento. Uma questão importante é ter em conta que se for inicializado no instante zero e com o período fixo previsto, teremos números de pedidos igual ao número de equipamentos com prazo igual ao período de manutenção. Isto não é nem viável, nem emula minimamente a realidade. Para evitar isto o arranque do temporizador de manutenção é feito da seguinte forma:

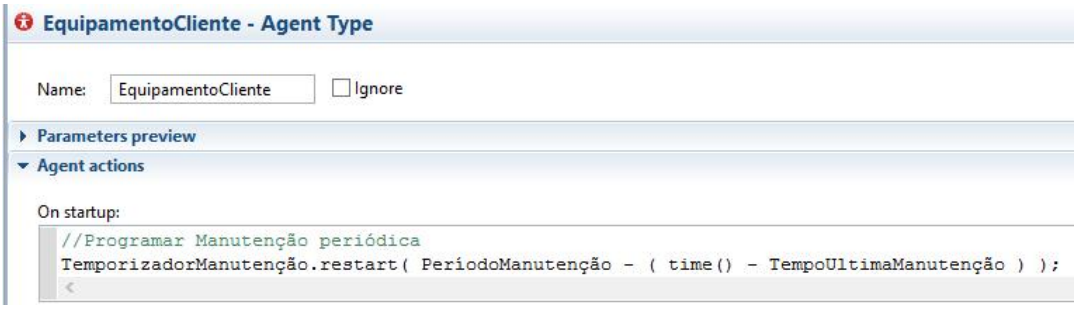


Figura 39 - Inicialização do temporizador de manutenção no arranque da simulação

O temporizador de manutenção é inicializado com o prazo definido no parâmetro Período de manutenção, a que é retirada a diferença entre o tempo actual (que no arranque será zero) e o valor registado na variável *TempoÚltimaManutenção*. Este valor inicialmente é definido de

forma aleatória (instrução *Java* "*uniform*") estando compreendido num intervalo entre – PeríodoManutenção e zero. Evitando-se assim picos incomportáveis de pedidos.

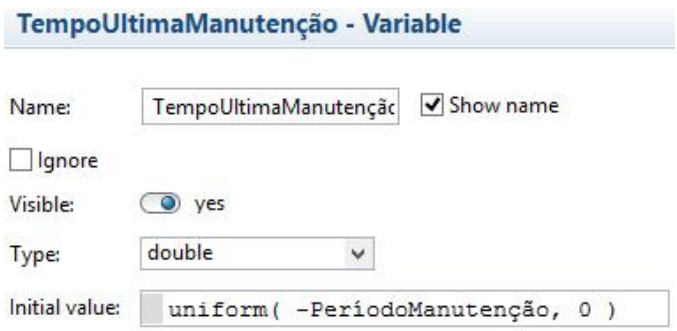


Figura 40 – valor inicial *TempoUltimaManutenção*

O mesmo tem de ser garantido para a variável *TempoÚltimaSubstituição*, no entanto uma vez que o número de períodos de manutenção recomendados para substituir o equipamento estão definidos no objecto de topo *Main* tem de ser recorrer à função *get_Main( )* para este cálculo.

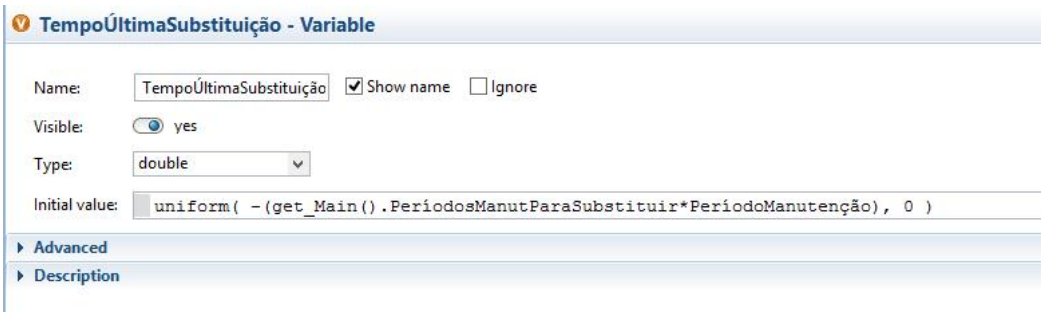


Figura 41 – valor inicial *TempoÚltimaSubstituição*

Repete-se ainda este procedimento para os valores iniciais das variáveis *TempoÚltimaAssistência* e *TempoÚltimaManutenção*, sendo a distribuição uniforme entre o valor negativo do período de manutenção (ou prazo médio entre avarias) e zero.

Como já foi referido anteriormente, os parâmetros do agente *EquipamentoCliente* são definidos no objecto de topo *Main*. As variáveis de registo de tempo são registadas em vários pontos do diagrama de estado para efeitos de verificação de bom funcionamento do modelo e cálculo de funções relativas a idade e necessidades de manutenção, assistência e/ou substituição. A variável *equipaAssistência* regista qual a equipa que está a trabalhar no equipamento.

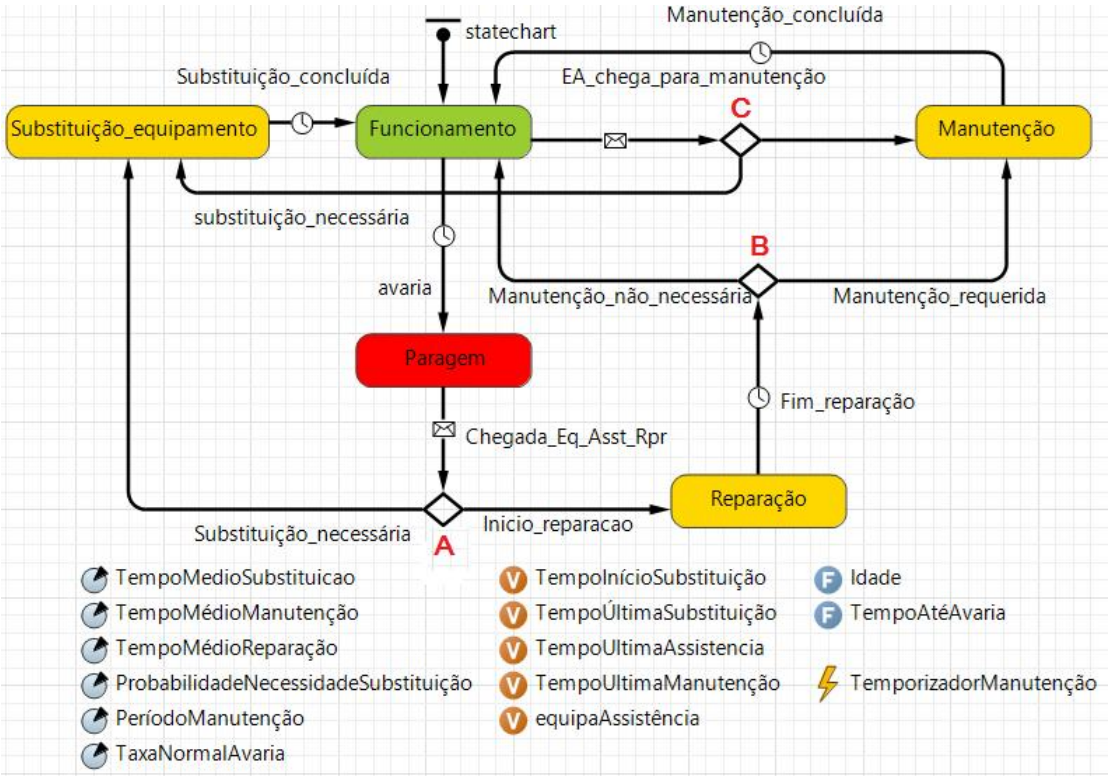


Figura 42 - Diagrama de estado *EquipamentoCliente*

A simulação inicia-se com o agente a entrar no estado *Funcionamento*, com a inicialização do *TemporizadorManutenção* e com a inicialização do *timeout* da transição *avaria*:

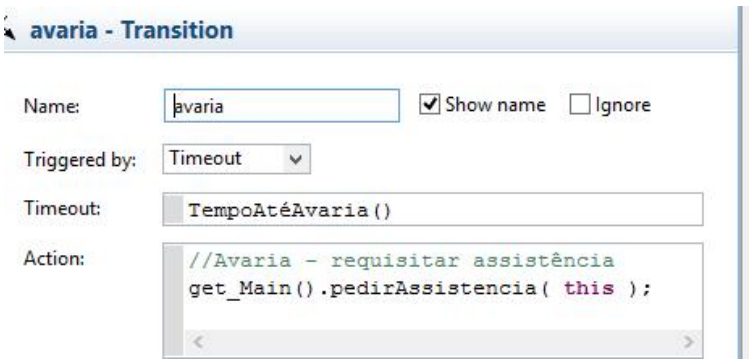


Figura 43 - timeout *avaria*

Este é dado pelo valor resultante da função *TempoAtéAvaria*. Por outro lado, esta função depende da função *Idade*, que devolve como valor a idade do equipamento:

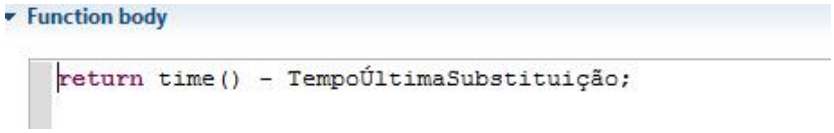


Figura 44 - Função *Idade*

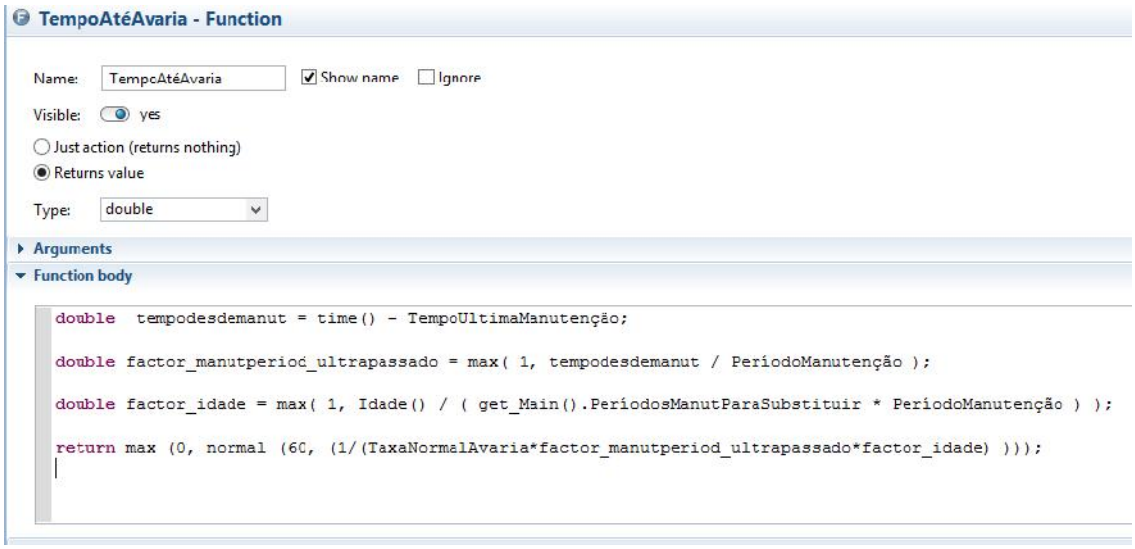


Figura 45 – Função *TempoAtéAvaria*

Optou-se por implementar dois factores que diminuem um pouco o prazo entre avarias se para o dado equipamento forem ultrapassados os prazos de manutenção ou substituição previstos. Uma vez que a manutenção depende de um temporizador que gera pedidos quando necessário, e é feita uma verificação acerca da idade do equipamento sempre que é feita uma intervenção no mesmo, e uma vez que não existem dados que permitam o cálculo real dos mesmos, a implementação destes factores é feita mais como prova de conceito e terão sempre uma influência residual no prazo entre falha.

O *factor_manutperiod_ultrapassado* dá o valor maior entre 1 e o quociente entre o tempo passado desde que o equipamento foi objecto de manutenção e o período de manutenção recomendado. O mesmo acontece para o *factor_idade* em relação à última substituição do equipamento e o período de substituição previsto.

Esta função dá como resultado o maior valor entre zero, e o resultado da geração aleatória normal com desvio padrão de 60 dias e valor médio dado pelo MTBF (correspondente à inversa da taxa de avaria), afectado pelos factores de idade e de período de manutenção ultrapassado.

Com o recurso a esta função, o timeout *TempoAtéAvaria* é activado, o equipamento entra no estado *Avaria*, tomando como acção activar a função *PedirAssistência* do objecto de topo *Main* para este agente.

A transição *Chegada_Eq_Ass_Rpr* é activada pelo recebimento de uma mensagem do tipo *EquipaAssistência*, de forma incondicional, e toma como acção registar qual a equipa de assistência que está a prestar serviço na variável *equipaAssistência.*

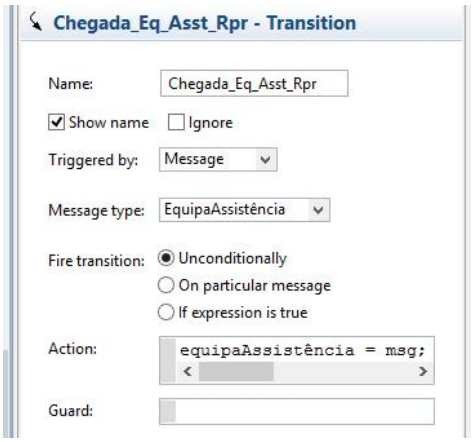


Figura 46 - Transição *Chegada_Eq_Asst_Rpr*

No *branch* A é verificado se o equipamento deve ser substituído. O ramo que leva à substituição é seguido se se verificarem as seguintes condições:

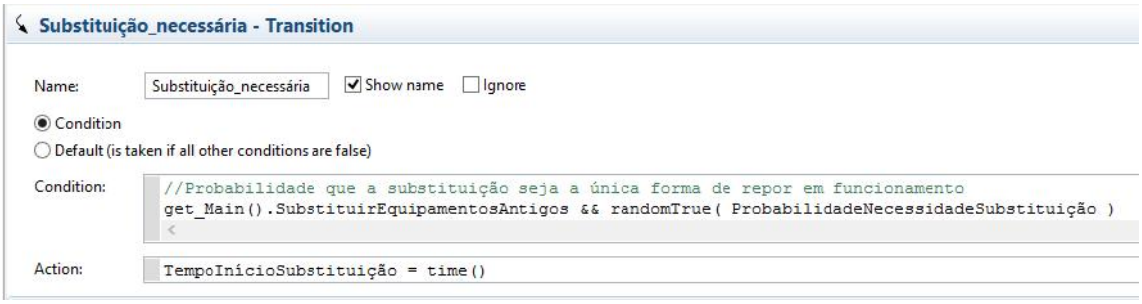


Figura 47 - Transição *Substituição_necessária*

É verificado se a política *SubstituirEquipamentosAntigos* no objecto *Main* está activada e, em caso afirmativo, qual a probabilidade que seja necessária substituição. Por exemplo para um valor de 0.1 um em cada 10 equipamentos serão substituídos. Se a condição for verdadeira é registado o tempo do início de substituição.

O equipamento entra no estado *Equipamento_substituição*, que deixa com a transição *Substituição_concluída* activada pelo *timeout* de valor igual ao definido no parâmetro *TempoMédioSubstituição.*

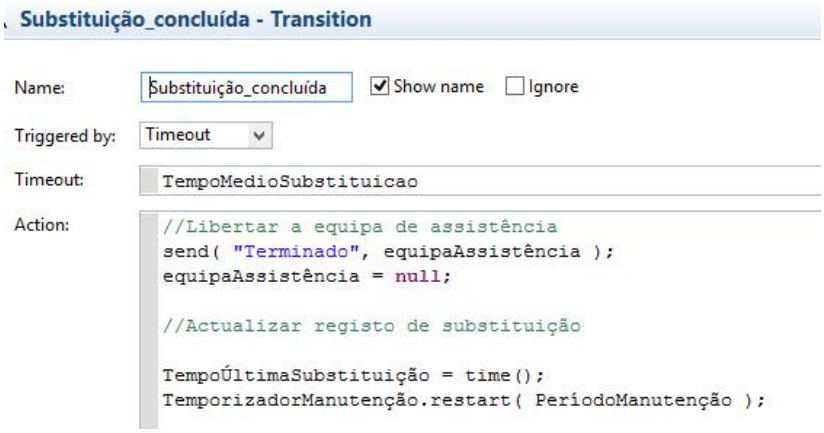


Figura 48 - Transição *Substituição_concluida*

Com o final da substituição, liberta-se a equipa de assistência enviando-lhe a mensagem "Terminado", o valor da variável *equipaAssistência* passa a nulo, reinicia-se o temporizado de manutenção e regista-se o tempo da substituição. Com isto o equipamento volta ao estado *Funcionamento.* Voltando ao *branch* A, se não for necessária a substituição dá-se início à reparação, sendo a transição *Início_reparação* atravessada sem acção e o estado *Reparação* é deixado através da transição *fim_reparação* depois do *timeout*:

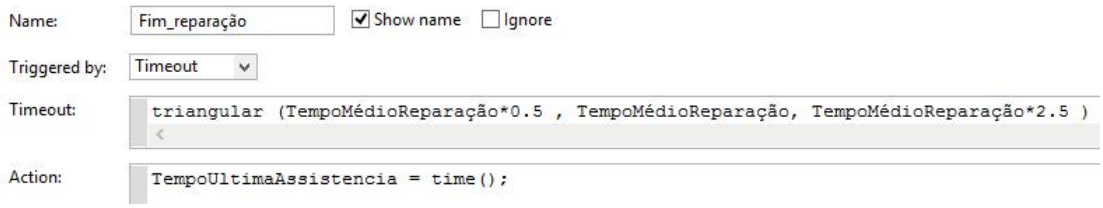


Figura 49 - *Transição Fim_reparação*

Este *timeout* obedece a uma distribuição triangular, com o valor optimista a ser metade do *TempoMédioReparação*, o valor normal a corresponder ao *TempoMédioReparação*, e ao valor pessismista a ser duas vezes e meia esta duração.

Aqui, chega-se ao *branch* B onde é verificado se é necessária manutenção:

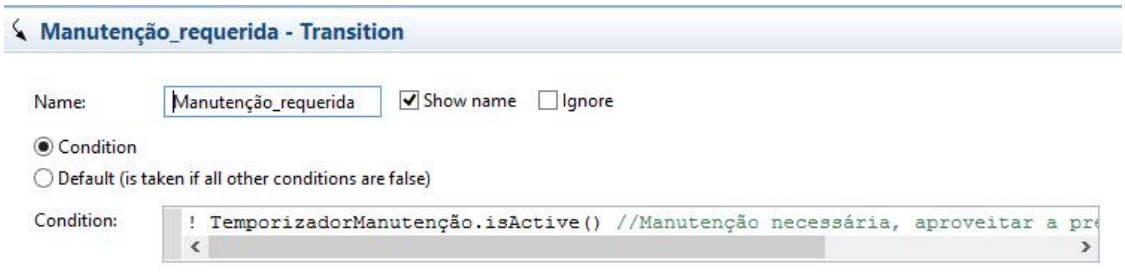


Figura 50 - Transição *Manutenção_Requerida*

A condição necessária para que esta transição seja activada é que o temporizador de manutenção não esteja activo (negação *Java "!"*).

Caso o temporizador esteja activo, é seguida a transição *Manutenção_não_necessária*, sendo tomada a acção de libertar de novo a equipa de assistência de forma análoga ao que foi feito no final da substituição com a mensagem:

send("Terminado", equipaAssistência);

equipaAssistência = null;

Se o temporizador estiver inactivo, passa-se para o estado *Manutenção*, que se deixa após o *timeout* da transição *Manutenção_concluída*:

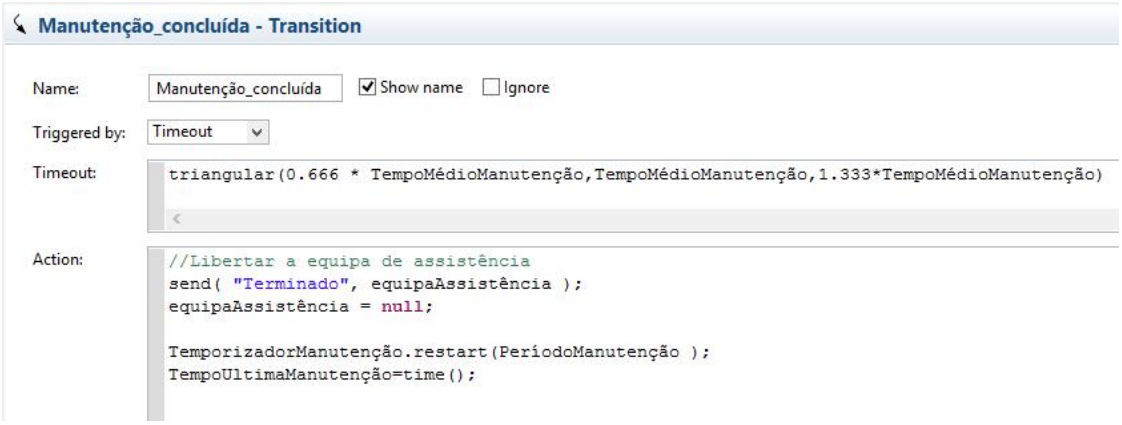


Figura 51 - Transição *Manutenção_concluída*

Este *timeout* também obedece a uma distribuição triangular, após terminar liberta a equipa de assistência e regista como nula a variável *equipaAssistência.* Reinicia o temporizador de manutenção e regista o tempo em que esta foi feita e retorna o equipamento ao estado de *Funcionamento*.

Existe uma última possibilidade, se tiver sido feito um pedido de manutenção e o equipamento não tiver tido falha antes de este ser satisfeito chegamos ao *branch* C. Aqui é de novo verificada a necessidade de substituição exactamente da mesma forma que no *branch* A (mesmos parâmetros). Se esta for necessária procede-se à mesma da forma que foi feita anteriormente. Em caso negativo passa para o estado *Manutenção* que abandona após o *timeout* que acabou de ser descrito na figura 51 e reentra em *Funcionamento*.

5.3.5 Sistema de recolha de estatísticas

O modelo embora possa ser funcional, necessita de que os seus resultados sejam quantificados e avaliáveis de forma objectiva. É necessário implementar um sistema que recolha e apresente dados estatísticos que permita avaliar o desempenho do sistema. Para cada um dos agentes no objecto de topo *Main* acrescentamos no separador *Statistics* os dados que pretendemos guardar. Para o equipamento do cliente pretendemos registar quantidade de equipamentos que estão em cada um dos estados: *Funcionamento, Paragem, Manutenção, Reparação* e *Substituição*.

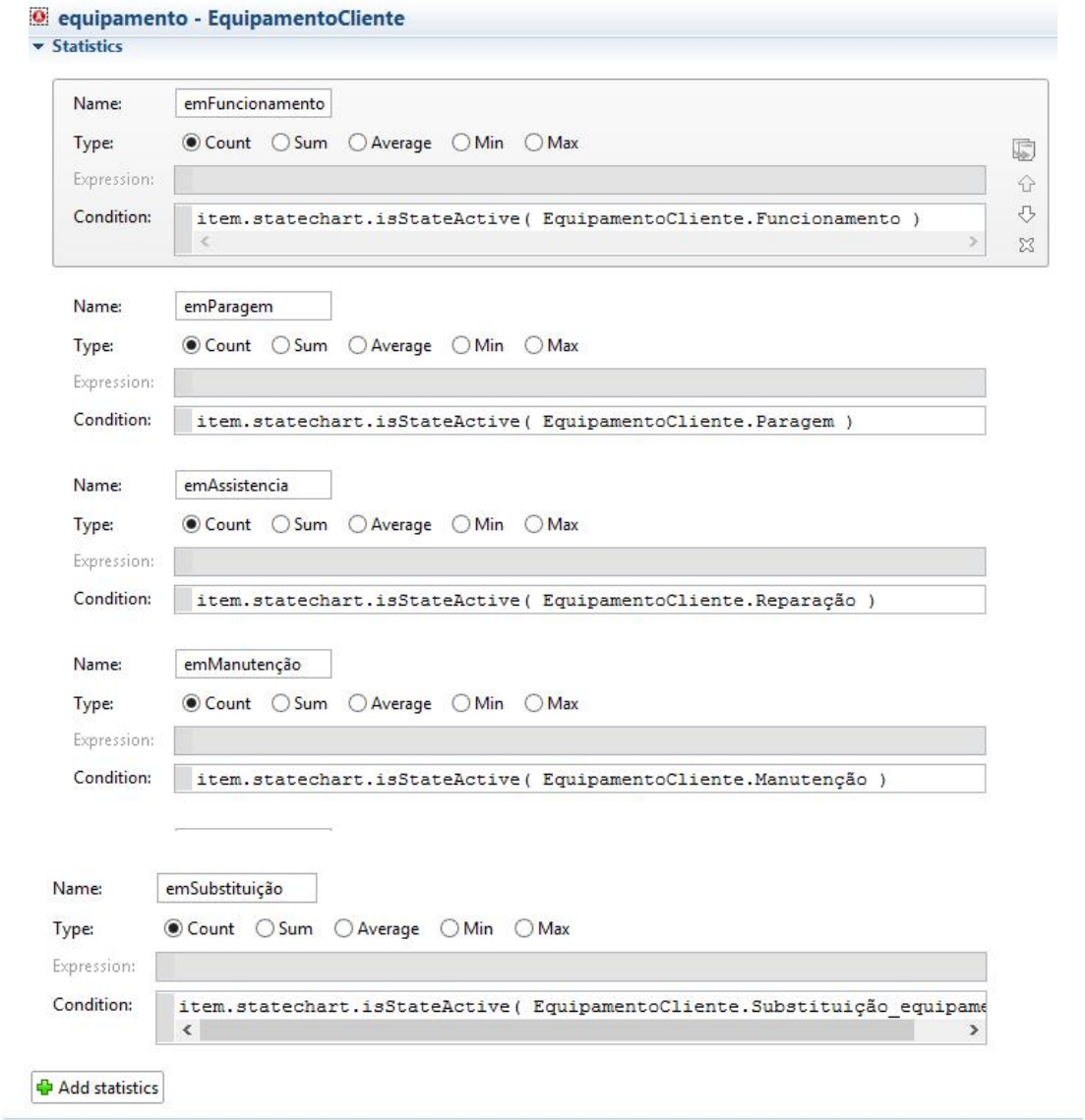


Figura 52 – Colecção de estatísticas do agente equipamento de cliente

Para a equipa de assistência o procedimento é rigorosamente igual, registando neste caso quantas equipas estão em *Espera*, *Viagem* ou a prestar *Assistência*.

Registamos estes dados no colector de estatísticas do Anylogic com actualização horária:

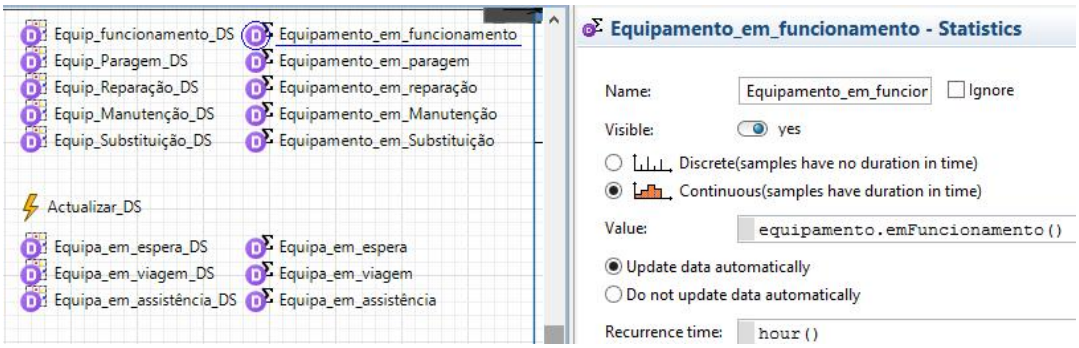


Figura 53 – Agregador de estatísticas para o equipamento no estado em *Funcionamento*

Cronologicamente, o agregador vai buscar os dados recolhidos estatisticamente no agente *equipamento* do objecto *Main* identificados como *emFuncionamento*. Procede-se da mesma forma para todos os outros valores relativos aos equipamentos dos clientes e equipas de assistência.

Para que estes valores sejam utilizáveis criou-se o evento *Actualizar_DS*, que adiciona mensalmente os valores médios destes estados aos *Datasets* criados para cada um dos casos. Aproveita-se ainda este evento para fazer mensalmente um cálculo de Disponibilidade dos equipamentos (número de equipamentos em funcionamento dividido pelo total) e de ocupação das equipas (equipas em trabalho ou viagem, a dividir pelo número de equipas) igualando estes valores a duas variáveis com o respectivo nome.

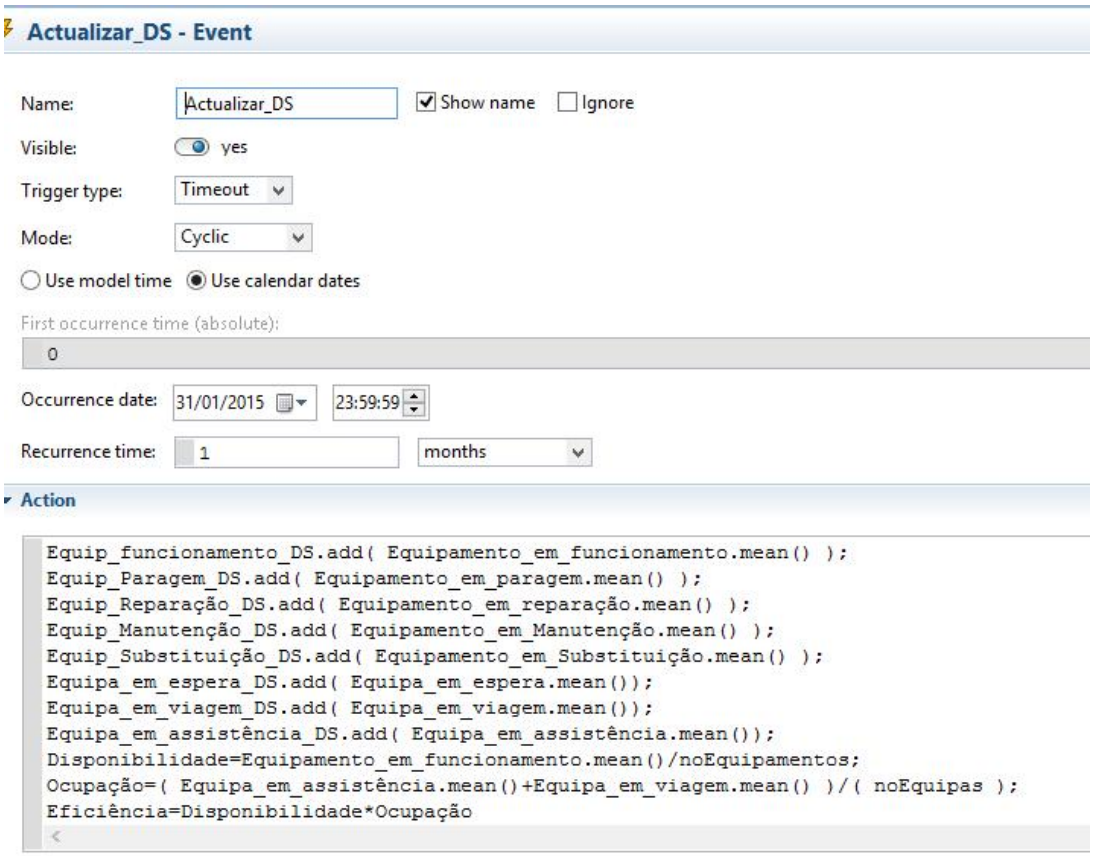


Figura 54 - Evento *Actualizar_DS*

Com estes dados criamos dois gráficos a representar no objecto *Main*, que em conjunto com o rectângulo representativo do espaço físico (170x100) e representações simbólicas dos agentes nos permitem uma visualização do desempenho do sistema enquanto decorrer a simulação. A animação 2D seleccionada, embora simples e objectivamente não necessária para o desenrolar da simulação, ajuda a identificar rapidamente e de forma visual problemas que possam existir no modelo.

Os gráficos apresentam os valores recolhidos pelos *Datasets* recolhidos no evento mensal.

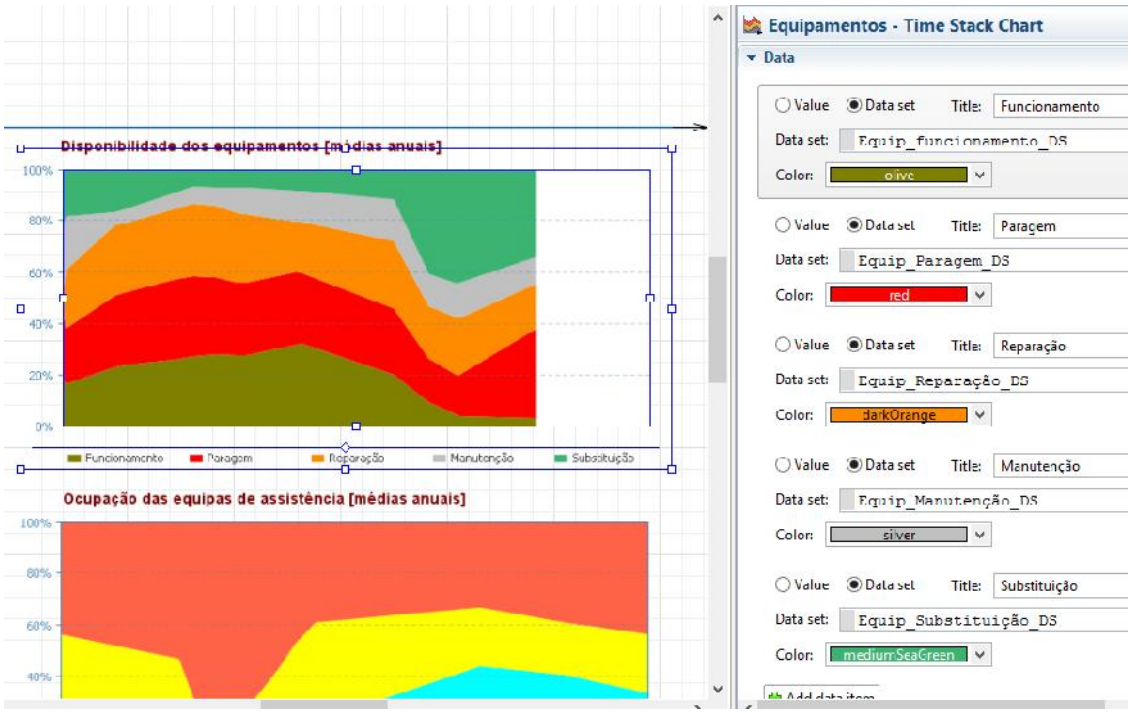


Figura 55 – Gráfico estado Equipamentos

5.3.6 Desempenho do sistema

Em conjunto com as variáveis criadas para registar a *Disponibilidade* (dos equipamentos), e a *Ocupação* (das equipas de assistência), para avaliar o desempenho do sistema criou-se a função *Eficiência_Sistema* que multiplica estes dois valores e que será objectivo maximizar.

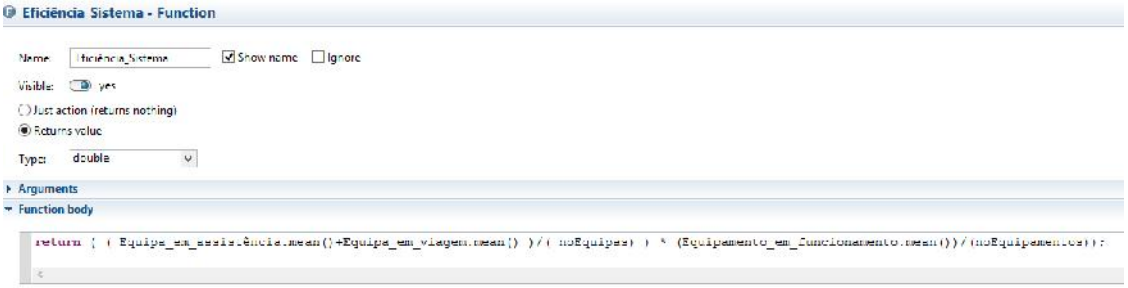


Figura 56 – Função *Eficiência_Sistema*

5.3.6 Simulação

As simulações foram definidas para correr ao longo de 10 anos com início a 1 de Janeiro de 2015. A aleatoriedade é garantida impondo *seeds* aleatórias para cada execução da simulação do modelo.

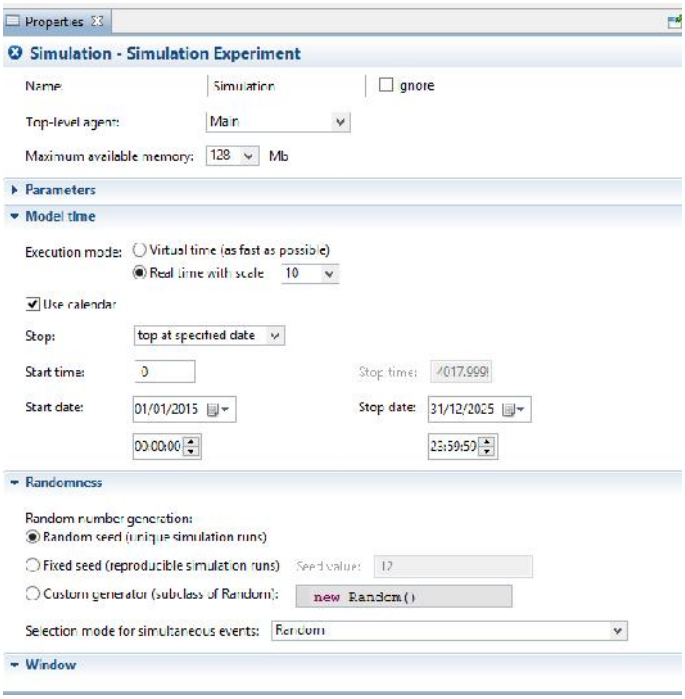


Figura 57 – Parâmetros da simulação

Ao correr o simulador temos controlo sobre a velocidade a que esta acontece e podemos visualizar tanto os dados fornecidos no objecto de topo graficamente e/ou nas variáveis, bem como a movimentação dos elementos na animação 2D. Temos sempre visíveis os parâmetros em uso para a simulação em curso.

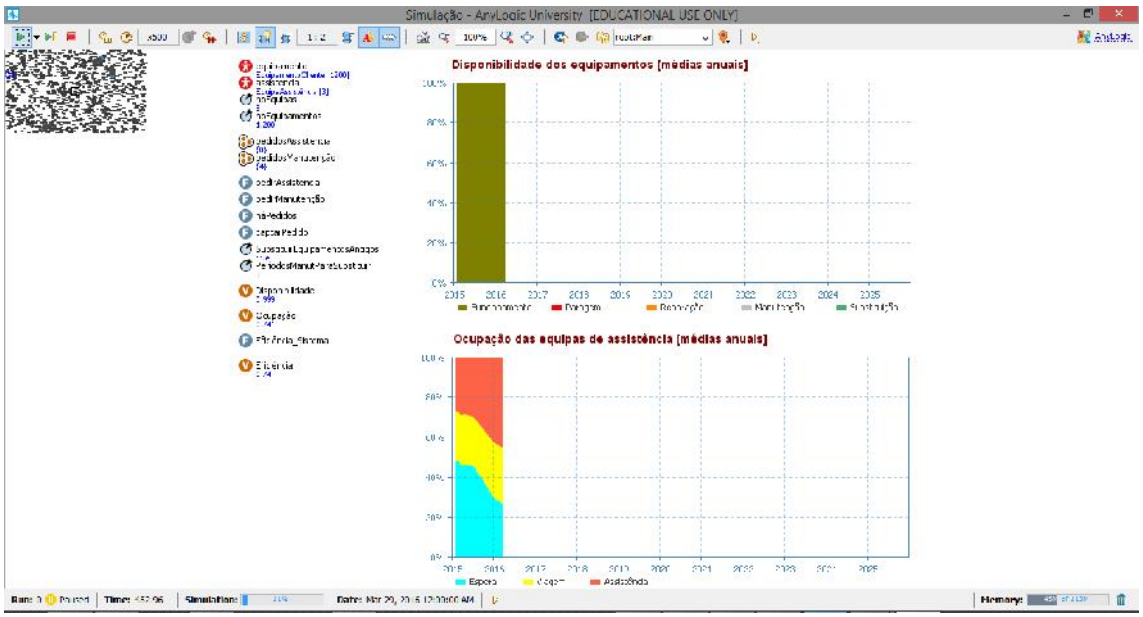


Figura 58 - Aspecto da simulação no objecto *Main*

Na equipa de assistência podemos para cada uma das equipas verificar em que estados se encontram e dados acumulados acerca de distâncias e serviços efectuados, no caso da figura seguinte temos a equipa 1:

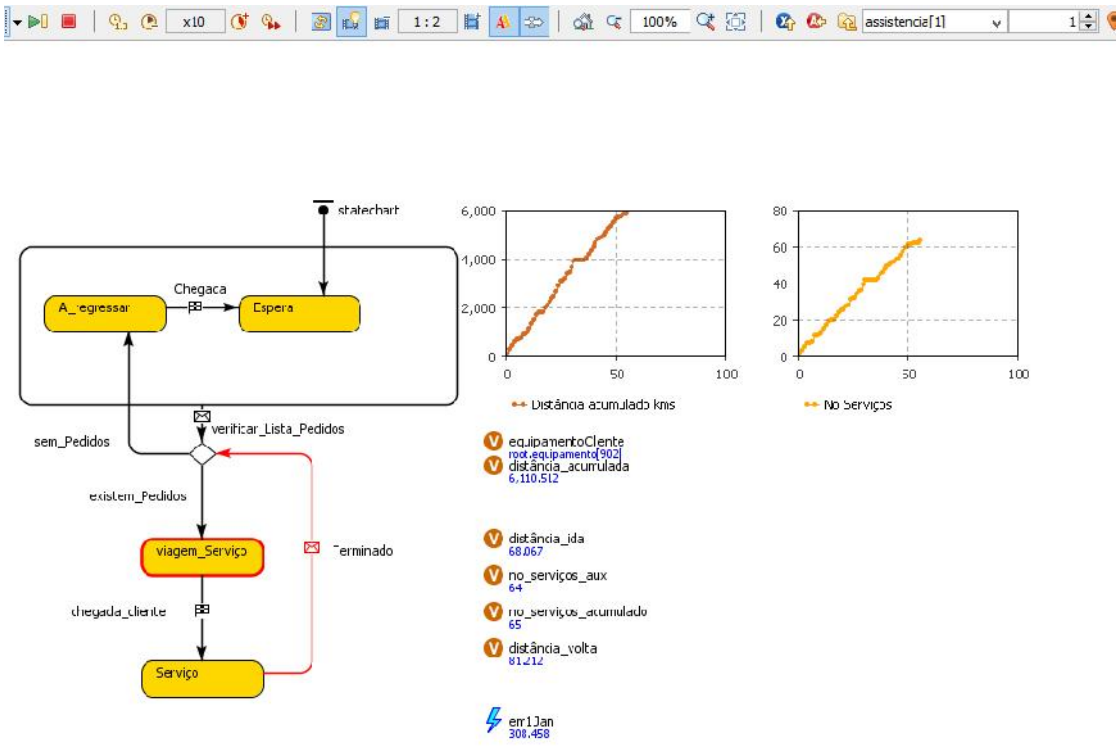


Figura 59 – Agente *EquipaAssistência*

Para os equipamentos, temos, além dos estados, informação acerca do tempo em que ocorreram os últimos eventos bem como os temporizadores activos a informar quando vai ocorrer a próxima avaria e quando será necessária manutenção. Neste caso temos o equipamento 345.

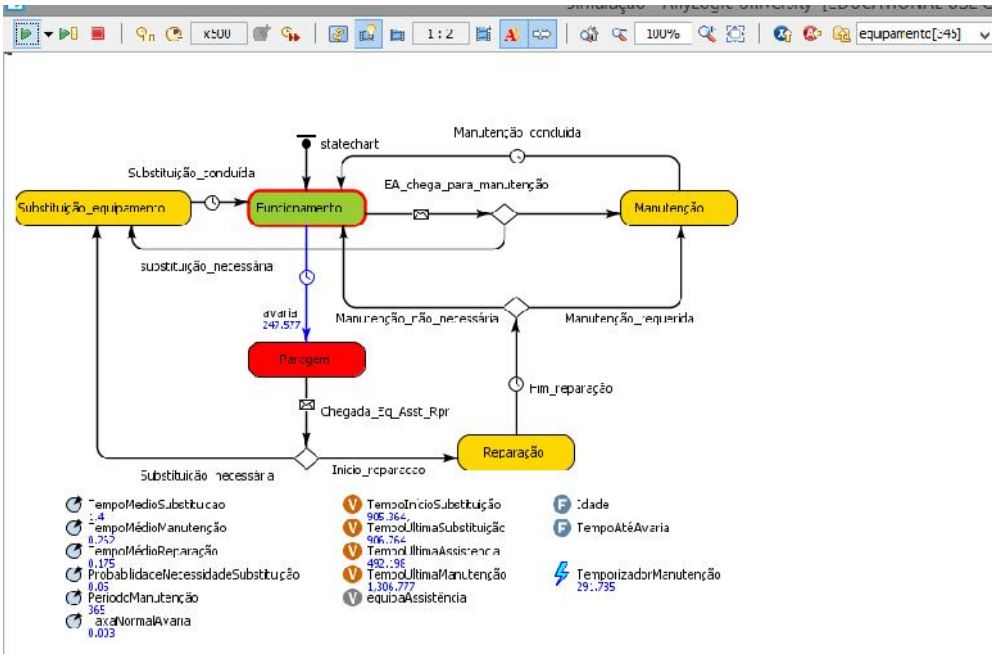


Figura 60 – Agente Equipamento do cliente 345

5.3.7 Optimizador

No Anylogic existe um optimizador de resultados de simulação. O objectivo deste passa por, definindo os parâmetros de partida procurar maximizar (ou minimizar) uma função que nos possa oferecer o melhor resultado para o desempenho do sistema. No caso em apreço optou-se por criar uma função "Eficiência" que relaciona a disponibilidade dos equipamentos com o nível de ocupação das equipas de assistência. Idealmente os dois níveis seriam totais, mas tal não é possível e procura-se o melhor compromisso. Ao executar as simulações, temos de variar os parâmetros manualmente e observar como se altera o comportamento do sistema. Com o recurso ao optimizador podemos definir quais os parâmetros que têm liberdade para ser variados em busca da solução.

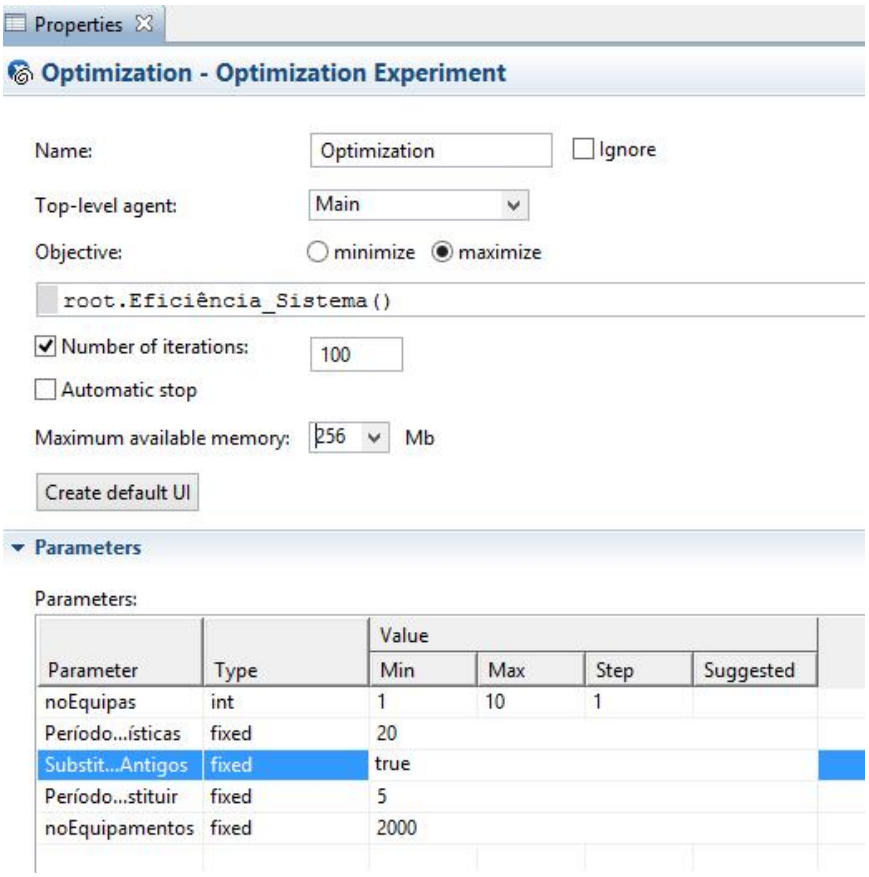


Figura 61 – Definições da experiência de optimização

Definiu-se como objectivo maximizar a função *Eficiência_Sistema* definida anteriormente e descrita atrás. Deu-se como liberdade a variação do número de equipas entre um valor mínimo de uma equipa e o máximo de 10. A optimização foi limitada a 100 iterações, valor que não será necessário atingir.

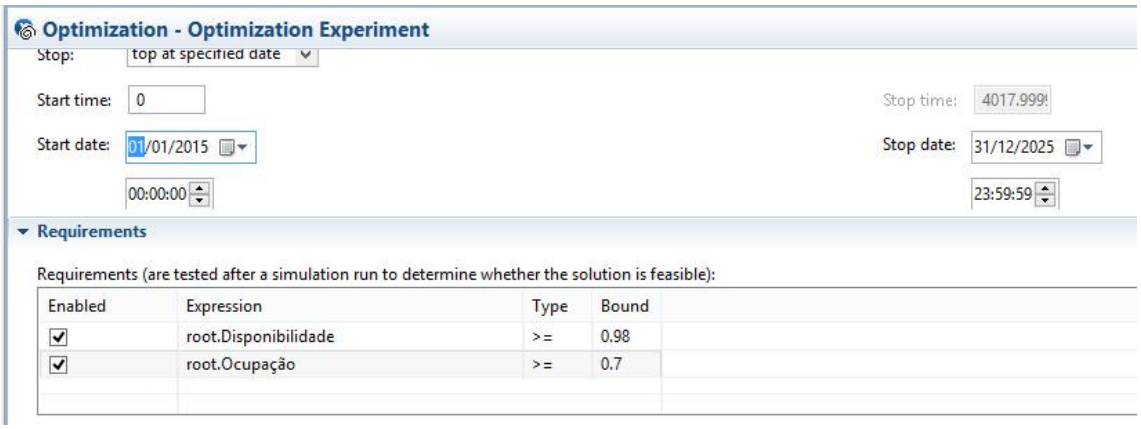


Figura 62 - Definições de tempo e requerimentos para a viabilidade da solução

Foram copiados da simulação os parâmetros relativos ao período de tempo de execução do modelo. No final de cada execução do mesmo verifica-se que os pré-requisitos são cumpridos. Definiu-se como objectivo uma disponibilidade de equipamentos de 98% e ocupação de equipas de 70%. No final de cada execução o optimizador verifica o valor destas variáveis e regista a solução como viável ou inviável.

As definições de aleatoriedade mantêm-se. Uma vez que cada simulação é única e há pequenas variações nos resultados, definiu-se um número de repetições da simulação com os mesmos parâmetros (replicações) para reduzir a dispersão de valores.

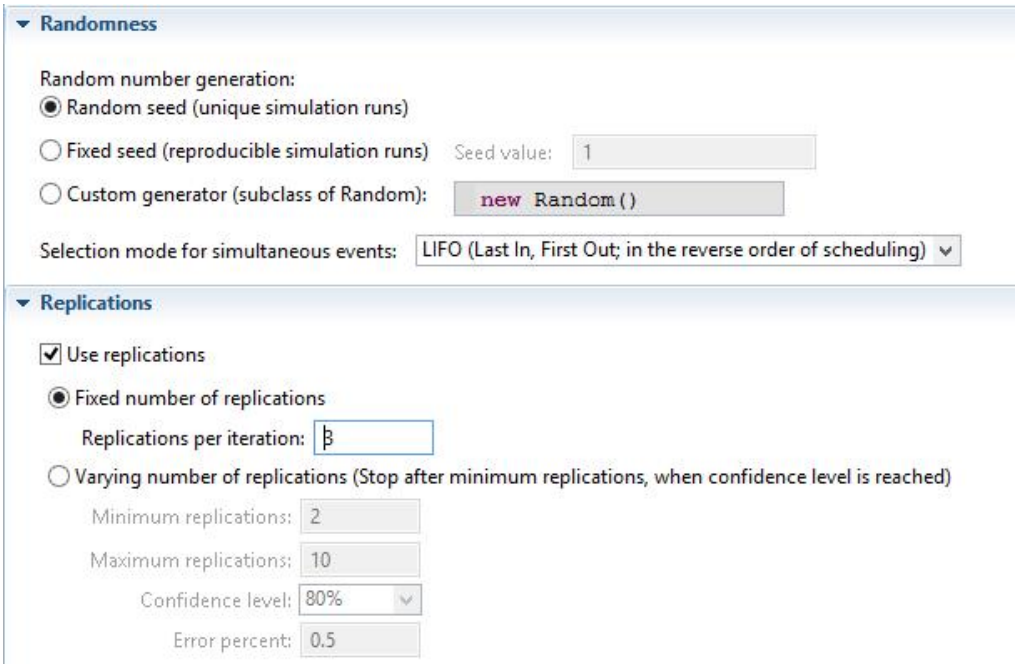


Figura 63 - Definições de aleatoriedade e replicações do optimizador

O optimizador correrá 30 iterações da simulação. Esta quantidade de iterações levam tempo a calcular e para minimizar este tempo é necessário tornar o modelo mais leve computacionalmente. Assim todos os gráficos são eliminados e apenas os *Datasets* necessários ao cálculo da função *Eficiência* são mantidos.

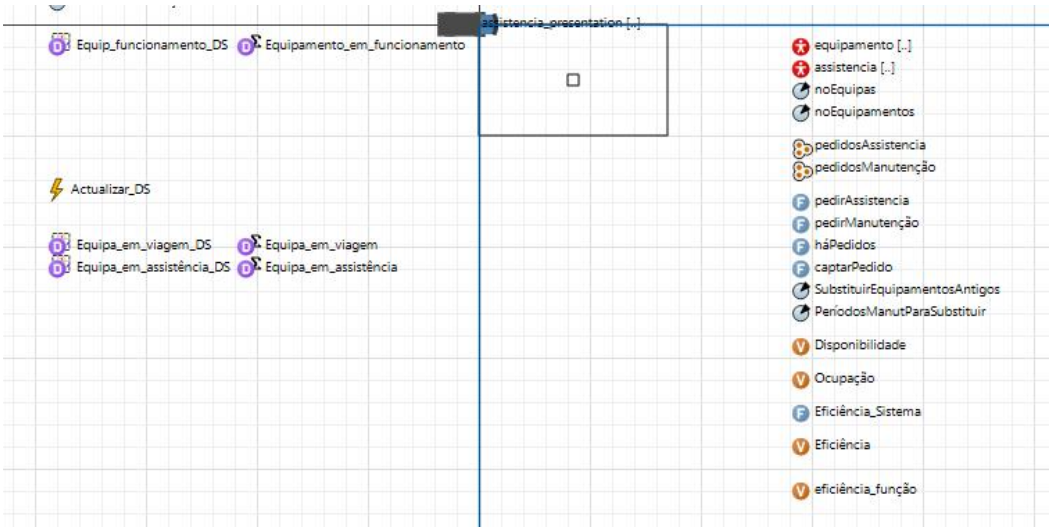


Figura 64 - *Main* após eliminação dos elementos desnecessários para optimização

No objecto *EquipaAssistência* eliminaram-se os gráficos com registo das distâncias e serviços realizados. Após estes passos apenas é necessário criar o interface de utilizador, que é gerado de forma automaticamente e fornece todas as informações que se pretendem.

A título de exemplo, correu-se uma experiência de optimização para 800 equipamentos de cliente.

Figura 65 - Resultados da experiência de optimização

Com os valores obtidos para o número de equipas, corremos a simulação e observa-se o comportamento do sistema para o melhor caso.

5.4 Resultados

O trabalho consistiu em simular vários cenários, procurando identificar qual o número mínimo de equipas que garante bom funcionamento do sistema. Assim, começámos por definir como objectivo identificar o número de equipas mínimo que permita um nível de disponibilidade dos equipamentos de 98% para várias quantidades de equipamentos, pretendemos ainda como objectivo secundário ter uma ocupação das equipas no mínimo de 70% do seu tempo.

No primeiro cenário temos 1200 equipamentos, os parâmetros de MTBF e tempos de intervenções são os constantes nos dados referidos anteriormente. Assim, para duas equipas temos o seguinte resultado:

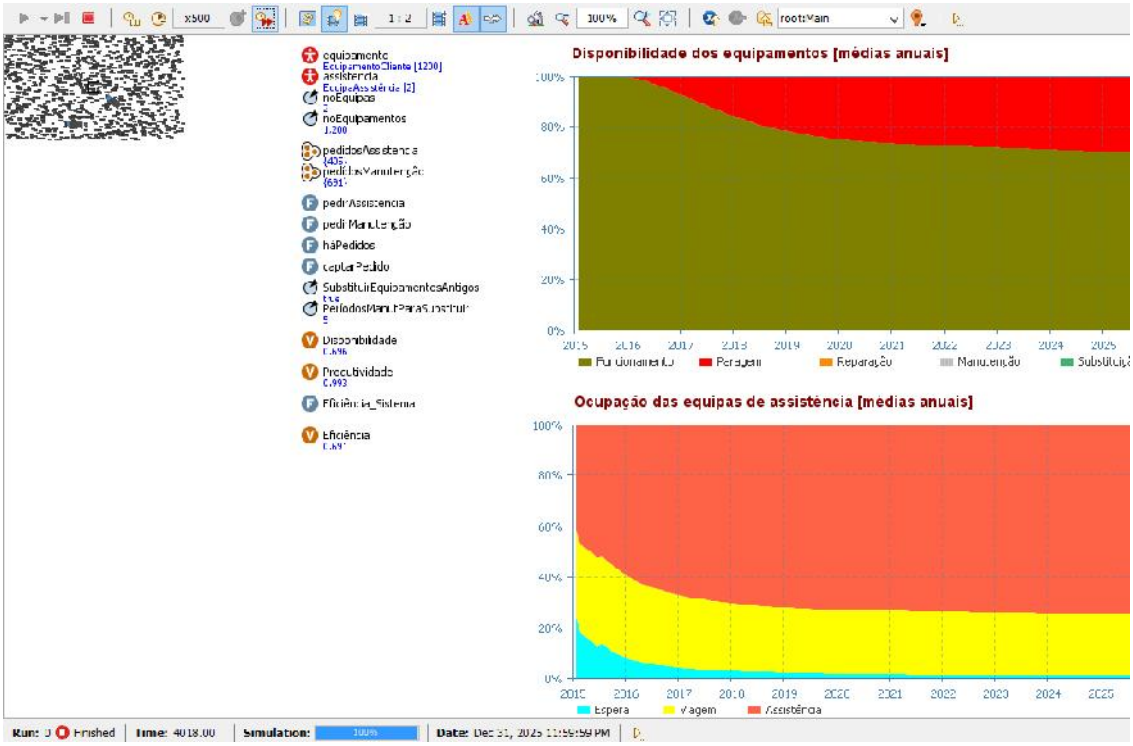


Figura 66 - Resultado da simulação para 1200 equipamentos e 2 equipas

O resultado da disponibilidade é de 0.696 o que é claramente insuficiente. Ao executar a simulação pela segunda vez obteve-se uma disponibilidade de 0.689. A produtividade ronda os 100% uma vez que as equipas estão ocupadas em pleno e nem assim conseguem responder.

Estabelecem-se diferentes cenários para níveis de procura dos pedidos de intervenção mantendo como condição o nível de serviço. Uma vez que cada simulação é única e produz resultados ligeiramente diferentes correu-se o modelo duas vezes para cada uma das situações descritas abaixo. Os resultados podem ser consultados na seguinte tabela.

	Equipamentos	Nº Equipas	Disponibilidade Equipamentos	Ocupação Equipas	Viável
Cenário 1	600	1	0.701	0.989	Não
		2	**0.998**	**0.726**	**Sim**
Cenário 2	800	**2**	**0.998**	**0.921**	**Sim**
		3	0.998	0.661	Não
Cenário 3	1200	**3**	**0.998**	**0.925**	**Sim**
		4	**0.998**	**0.738**	**Sim**
Cenário 4	2000	4	0.885	0.986	Não
		5	**0.998**	**0.929**	**Sim**

Tabela 3 - Resultados dos diferentes cenários

Pode ser observado que para o nível de actividade actual o número de equipas ideal para responder à procura é de 2 pois consegue garantir o nível de serviço com uma ocupação superior a 90%. Se houver uma quebra de procura de 25% para cerca de 600 clientes a ocupação das equipas cai mas ainda se mantém acima do nível mínimo estabelecido, no entanto era impossível manter o nível de serviço com apenas uma equipa. Com uma mais forte implantação no mercado atingindo 1200 clientes a empresa poderia responder com 3 equipas com uma elevada eficiência, se o número de clientes atingisse os 2000 já seriam necessárias 5 equipas.

6. Conclusões e trabalhos futuros

6.1. Conclusão

Para a criação de um modelo que se aproxime o mais possível de um sistema real é fundamental a modelação conceptual. Sem uma definição correcta de todo o sistema, desde a representação das entidades e processos intervenientes e definição das relações entre os mesmos, não é possível sequer considerar a possibilidade de conceber um modelo fiável.

Existem uma multiplicidade de metodologias de M&S, no entanto a opção por ABM confirmou integralmente todas a expectativas acerca das possibilidades que apresenta. A modelação ABM é mais natural, descrever e criar fluxogramas acerca do comportamento de um determinado interveniente é muito mais intuitivo do que procurar equações que o descrevam, o que se revelou uma enorme vantagem em relação a outras abordagens.

O programa *Anylogic* demonstrou ser uma excelente ferramenta por todas as possibilidades de modelação que apresenta: vastas bibliotecas de elementos pré configurados; opção de visualização de resultados que permite rapidamente identificar problemas; a enorme quantidade de modelos exemplo incluídos oferecem possibilidade de aprendizagem por observação da forma como foram construídos, implementados e as soluções para as questões propostas.

Ao mesmo tempo, apesar de todo o trabalho feito para tornar o programa de modelação acessível a não informáticos, um modelo mais complexo ainda exige bastantes conhecimentos de programação *Java*, o que não se torna simples de implementar por quem não é munido de formação em programação.

Em suma, pode ser dito que o programa pode ser uma mais-valia para a procura de soluções em várias áreas eliminando a dispendiosa necessidade de tentativa/erro em sistemas reais. Para o caso em análise, mediante cuidada aferição da qualidade dos dados de partida, o programa

responde sem dificuldade aos objectivos de dimensionar o número de equipas necessário para o desempenho da actividade de empresa com o nível requerido.

6.2. Trabalhos futuros

As possibilidades de trabalho futuro nesta área são enormes, procurando restringir as propostas apenas ao essencial:

-uma vez que o cerne de actividade da empresa é a prestação de serviços em equipamentos que não possui, o registo de dados de frequência e tipo de avaria não é feito por equipamento individual, mas por cliente;

-obter dados que permitam melhor identificar as características individuais de ocorrências, para melhor calibrar estatisticamente as funções de geração de avarias na simulação e que poderão melhorar o comportamento e fiabilidade do sistema;

-implementar um sistema de cálculo de custos/lucros/rentabilidade baseados no tempo despendido, distâncias percorridas, material e tempo facturado por tipo de intervenção;

-melhorar o cálculo de distâncias com recurso ao mapeamento de rotas GIS, eliminando a necessidade de factores de correcção necessariamente induzidores de erro;

-mediante recolha de dados, implementar sazonalidade na procura dos serviços procurando avaliar a resposta do sistema perante aumentos de procura localizadas no tempo;

-introduzir variação na procura, perca e ganho de clientes, variações causadas por ciclos económicos positivos ou negativos;

-embora seja ultrapassável da forma que aqui se demonstrou, introduzir horários de trabalho, turnos diferenciados tanto para equipas como para clientes.

Algumas destas propostas poderiam eventualmente violar um dos fundamentos da modelação, que é o de não introduzir no modelo complexidade demasiada que pode levar a problemas no desempenho do mesmo sem oferecer correspondente aumento de rigor nos resultados, no entanto podem ser áreas que importa abordar no futuro.

Referências

Christos G. Cassandras; Stéphane Lafortune – *"Introduction to Discrete Event Systems – 2nd Ed."* – Springer – 2008

C.M. Macal and M.J. North - *"Tutorial on agent-based modelling and simulation"* –Center for Complex Adaptive Agent Systems Simulation, Decision & Information Sciences Division, Argonne National Laboratory, Argonne, Il, USA; and Computation Institute, The University of Chicago, Chicago, Il, USA – Journal of Simulation (2010) 4, p. 151-162

Dale K. Pace – *"Ideas About Simulation Conceptual Model Development"* – Johns Hopkins APL Technical Digest, Volume 21, Number 3 (2000)"

Dale K. Pace - *"Naval modeling and simulation verification, validation, and accreditation. Proceedings of WSC '93: Proceedings of the 25th Winter Simulation Conference,Los Angeles"*, p. 1077–1080. – 1993

Department of Defence USA Instruction (DoDI) 5000.61: DoD Modelling and Simulation (M&S) Verification, Validation, Accreditation (VV&A), April 1996

Gabriel A. Wainer - *"Discrete-Event Modeling and Simulation - A Practitioner's Approach"*– CRC Press – 2009

J. Casti – Wiley – "*Would-Be Worlds: How Simulation is Changing the World of Science*" – 1997

John A. Sokolowski; Catherine M. Banks - "*MODELING AND SIMULATION FUNDAMENTALS -Theoretical Underpinnings and Practical Domains"* – Wiley – 2010

Nathaniel Osgood - Associate Professor, University of Saskatchewan - "*MIT Lectures – Spring 2012*", , http://www.cs.usask.ca/faculty/ndo885/Classes/MIT15879/Lectures.html

Pedro António Marques Campos – *"A inovação de uma empresa de serviços usando a simulação"* - ESTGV – IPV – 2012

Theodore T. Allen -*"Introduction to Discrete Event Simulation and Agent-based Modeling "* – Springer – 2011

AnyLogic – http://www.anylogic.com

Matlab Simulink - http://www.mathworks.com/products/simulink/

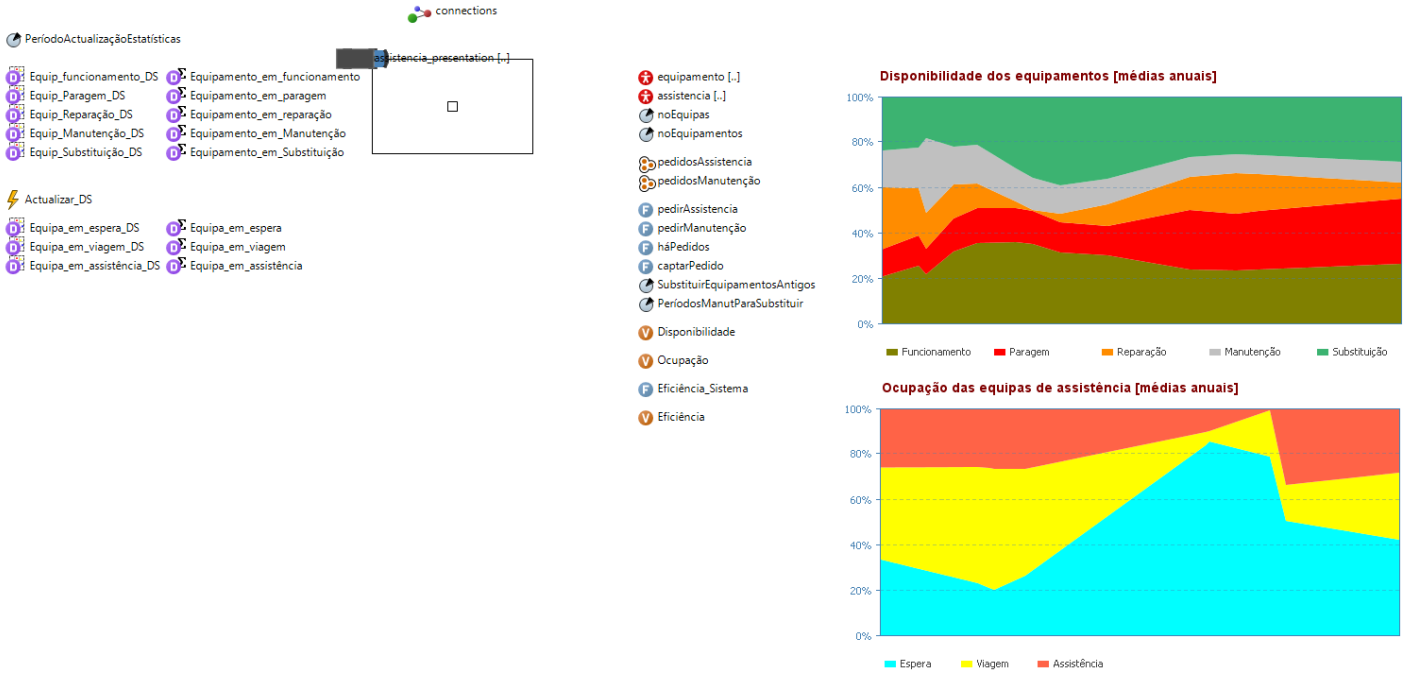
Anexos – Documentação gerada pelo *Anylogic*

Model: simulacao

Name	Value
General	
Model Time Units	Day
Numerical methods	
Differention Equations Method	EULER
Algebraic Equations Method	MODIFIED_NEWTON
Mixed Equations Method	RK45_NEWTON
Absolute Accuracy	1.0E-5
Time Accuracy	1.0E-5
Relative Accuracy	1.0E-5
Fixed Time Step	0.0010
Advanced	
Java Package Name	simulacao
File Name	C:\Users\Filipe\Google Drive\Mestrado\Tese\modelo tese - zant\Simulação Equipas Assistência\simulacao\simulacao.alp

Agent Type: Main

Name	Value
Entity actions	
Flowcharts Usage	ENTITY
Movement parameters	
Velocity	10
Rotate Animation Towards Movement	false
Environment for other agents	
Space Type	CONTINUOUS
Dynamic: Width	170
Dynamic: Height	100
Dynamic: z Height	0
Layout Type	RANDOM
Layout Type Apply On Startup	true
Network Type	USER_DEF
Network Type Apply On Startup	true
Enable Steps	false
Advanced Java	
Generic	false
Advanced	
Make Default View Area	true
Auto-create Datasets	true
Recurrence	1
Limit the number of array elements	false



Parameter: noEquipas

Name	Value
General	
Array	false
Default Value	1
Type	int
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Parameter: PeríodoActualizaçãoEstatísticas

Name	Value
General	
Array	false
Default Value	20
Type	double
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Parameter: SubstituirEquipamentosAntigos

Name	Value
General	
Array	false
Default Value	true
Type	boolean
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	CHECK_BOX

Parameter: PeríodosManutParaSubstituir

Name	Value
General	
Array	false
Default Value	5
Type	int
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Parameter: noEquipamentos

Name	Value
General	
Array	false
Default Value	600
Type	int
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Function: pedirAssistencia

Name	Value
General	
ReturnModificator	VOID
Show At Runtime	true
Show name	true
Arguments	
Parameters	(EquipamentoCliente unidade)
Function body	

Body	//se esta unidade pediu manut. – remover pedido (baixa prioridade)
	if(pedidosManutenção.contains(unidade))
	pedidosManutenção.remove(unidade);
	
	//se alguma equipa está a trabalhar na unidade, ignorar pedido
	for(EquipaAssistência ea : assistencia)
	if(ea.equipamentoCliente == unidade)
	return;
	//adicionar pedido de assistência à lista de espera
	pedidosAssistencia.addLast(unidade);
	
	//Obrigar equipas a verificar lista de espera
	for(EquipaAssistência ea : assistencia)
	ea.receive("Verificar lista de pedidos");

Advanced	
Static	false
Access Type	default
Use Units	false

Arguments:

Name	Type
unidade	EquipamentoCliente

Function: háPedidos

Name	Value
General	
Return Type	boolean
ReturnModificator	RETURNS_VALUE
Show At Runtime	true
Show name	true
Arguments	
Parameters	()
Function body	
Body	return ! pedidosAssistencia.isEmpty() || ! pedidosManutenção.isEmpty();
Advanced	
Static	false
Access Type	default
Use Units	false

Function: captarPedido

Name	Value
General	
Return Type	EquipamentoCliente
ReturnModificator	RETURNS_VALUE

Name	Value
Show At Runtime	true
Show name	true
Arguments	
Parameters	()
Function body	

Body	//Verificar primeiro Avarias
	if(! pedidosAssistencia.isEmpty())
	return pedidosAssistencia.removeFirst();
	else //depois verificar pedidos de manutenção periódica
	return pedidosManutenção.removeFirst();

Advanced	
Static	false
Access Type	default
Use Units	false

Function: pedirManutenção

Name	Value
General	
ReturnModificator	VOID
Show At Runtime	true
Show name	true
Arguments	
Parameters	(EquipamentoCliente unidade)
Function body	

Body	//Se a unidade já pediu manutenção, não há necessidade de voltar
	if(pedidosManutenção.contains(unidade))
	return;
	//se alguma equipa está a trabalhar neste equipamento, ignorar o pedido
	for(EquipaAssistência ea : assistencia)
	if(ea.equipamentoCliente == unidade)
	return;
	//Adicionar pedido de manutenção à fila de espera
	pedidosManutenção.addLast(unidade);
	//fazer todas as equipas verificar a lista de pedidos
	for(EquipaAssistência ea : assistencia)
	ea.receive("Verificar lista de pedidos");

Advanced	
Static	false
Access Type	default
Use Units	false

Arguments:

Name	Type
unidade	EquipamentoCliente

Function: Eficiência_Sistema

Name	Value
General	
Return Type	double
ReturnModificator	RETURNS_VALUE
Show At Runtime	true

Name	Value
Show name	true
Arguments	
Parameters	()
Function body	
Body	return ((Equipa_em_assistência.mean()+Equipa_em_viagem.mean())/(Equipa_em_assistência.mean()+Equipa_em_viagem.mean()+Equipa_em_espera.mean()) * (Equipamento_em_funcionamento.mean())/(Equipamento_em_funcionamento.mean()+Equipamento_em_paragem.mean()+Equipamento_em_reparação.mean()+Equipamento_em_Manutenção.mean()));
Advanced	
Static	false
Access Type	default
Use Units	false

Event: Actualizar_DS

Name	Value
General	
Event Recurrence Time Unit	MONTH
Recurrence	1
Event Occurrence Date	Sat Jan 31 23:59:59 GMT 2015
Occurs At Time	false
Mode	cyclic
Trigger Type	timeout
Show At Runtime	true
Show name	true
Action	
Action	Equip_funcionamento_DS.add(Equipamento_em_funcionamento.mean()); Equip_Paragem_DS.add(Equipamento_em_paragem.mean()); Equip_Reparação_DS.add(Equipamento_em_reparação.mean()); Equip_Manutenção_DS.add(Equipamento_em_Manutenção.mean()); Equip_Substituição_DS.add(Equipamento_em_Substituição.mean()); Equipa_em_espera_DS.add(Equipa_em_espera.mean()); Equipa_em_viagem_DS.add(Equipa_em_viagem.mean()); Equipa_em_assistência_DS.add(Equipa_em_assistência.mean()); Disponibilidade=Equipamento_em_funcionamento.mean()/noEquipamentos; Ocupação=(Equipa_em_assistência.mean()+Equipa_em_viagem.mean())/(noEquipas);

Variable: Disponibilidade

Name	Value
General	
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false

Name	Value
Constant	false
Save In Snapshot	true
Use Units	false

Variable: Ocupação

Name	Value
General	
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: Eficiência

Name	Value
General	
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Collection: pedidosAssistencia

Name	Value
General	
Element Class	EquipamentoCliente
Collection Class	LinkedList
Show At Runtime	true
Show name	true
Advanced	
Colleciton Initializer Code	{}
Access Type	public
Save In Snapshot	true
Static	false

Collection: pedidosManutenção

Name	Value

Name	Value
General	
Element Class	EquipamentoCliente
Collection Class	LinkedList
Show At Runtime	true
Show name	true
Advanced	
Colleciton Initializer Code	{}
Access Type	public
Save In Snapshot	true
Static	false

Time Stack Chart: Equipamentos

Name	Value
General	
Public	true
Data update	
Analysis Auto Update	false
Dataset Samples To Keep	50000
Scale	
Time Window	11 * 365 * day()
Time Window Units	MODEL_TIME_UNIT
Vertical Scale	HUNDRED_PERCENTS
Chart Vertical Scale: To	1
Appearance	
Labels Horizontal Position	DEFAULT
Labels Vertical Position	DEFAULT
Label Format	MODEL_DATE_CUSTOM
Custom Label Format	yyyy
Labels Text Color	steelBlue
Chart Area Grid Color	steelBlue
Position and size	
x	500.0
Width	610.0
y	20.0
Height	300.0
Legend	
Show Legend	true
Legend Size	20.0
Legend Text Color	darkGray
Legend Place	SOUTH
Chart area	
Chart Area: X Offset	40.0
Chart Area: Width	550.0
Chart Area: Y Offset	20.0
Chart Area: Height	240.0
Chart Area: Background Color	white
Chart Area Border Color	steelBlue

Name	Value
Advanced	
Time Windows Movement Type	MOVEMENT_WITH_DATA
Show name	false

Plot Items:

Title	Type	Dataset / Value	Color
Funcionamento	dataset	Equip_funcionamento_DS	olive
Paragem	dataset	Equip_Paragem_DS	red
Reparação	dataset	Equip_Reparação_DS	darkOrange
Manutenção	dataset	Equip_Manutenção_DS	silver
Substituição	dataset	Equip_Substituição_DS	mediumSeaGreen

Time Stack Chart: chart1

Name	Value
General	
Public	true
Data update	
Analysis Auto Update	false
Dataset Samples To Keep	50000
Scale	
Time Window	11 * 365 * day()
Time Window Units	MODEL_TIME_UNIT
Vertical Scale	HUNDRED_PERCENTS
Chart Vertical Scale: To	1
Appearance	
Labels Horizontal Position	DEFAULT
Labels Vertical Position	DEFAULT
Label Format	MODEL_DATE_CUSTOM
Custom Label Format	yyyy
Labels Text Color	steelBlue
Chart Area Grid Color	steelBlue
Position and size	
x	498.0
Width	610.0
y	350.0
Height	300.0
Legend	
Show Legend	true
Legend Size	20.0
Legend Text Color	darkGray
Legend Place	SOUTH
Chart area	
Chart Area: X Offset	40.0
Chart Area: Width	550.0
Chart Area: Y Offset	20.0
Chart Area: Height	240.0
Chart Area: Background Color	white
Chart Area Border Color	steelBlue

Name	Value
Advanced	
Time Windows Movement Type	MOVEMENT_WITH_DATA
Show name	false

Plot Items:

Title	Type	Dataset / Value	Color
Espera	dataset	Equipa_em_espera_DS	cyan
Viagem	dataset	Equipa_em_viagem_DS	yellow
Assistência	dataset	Equipa_em_assistência_DS	tomato

Statistics: Equipamento_em_funcionamento

Name	Value
General	
Statistics Value	equipamento.emFuncionamento()
Discrete	false
Show At Runtime	true
Data update	
Analysis Auto Update	true
Recurrence	hour()

Data Set: Equip_funcionamento_DS

Name	Value
General	
Dataset Samples To Keep	100
Axis Data Freeze X Axis	true
Show At Runtime	true
Show name	true
Data update	
Analysis Auto Update	false

Statistics: Equipamento_em_paragem

Name	Value
General	
Statistics Value	equipamento.emParagem()
Discrete	false
Show At Runtime	true
Data update	
Analysis Auto Update	true
Recurrence	hour()

Data Set: Equip_Paragem_DS

Name	Value
General	
Dataset Samples To Keep	150
Axis Data Freeze X Axis	true

Name	Value
Show At Runtime	true
Show name	true
Data update	
Analysis Auto Update	false

Statistics: Equipamento_em_reparação

Name	Value
General	
Statistics Value	equipamento.emAssistencia()
Discrete	false
Show At Runtime	true
Data update	
Analysis Auto Update	true
Recurrence	hour()

Data Set: Equip_Reparação_DS

Name	Value
General	
Dataset Samples To Keep	150
Axis Data Freeze X Axis	true
Show At Runtime	true
Show name	true
Data update	
Analysis Auto Update	false

Statistics: Equipamento_em_Manutenção

Name	Value
General	
Statistics Value	equipamento.emManutenção()
Discrete	false
Show At Runtime	true
Data update	
Analysis Auto Update	true
Recurrence	hour()

Data Set: Equip_Manutenção_DS

Name	Value
General	
Dataset Samples To Keep	150
Axis Data Freeze X Axis	true
Show At Runtime	true
Show name	true
Data update	
Analysis Auto Update	false

Statistics: Equipa_em_espera

Name	Value
General	
Statistics Value	assistencia.Eq_em_Espera()
Discrete	false
Show At Runtime	true
Data update	
Analysis Auto Update	true
Recurrence	hour()

Data Set: Equipa_em_espera_DS

Name	Value
General	
Dataset Samples To Keep	150
Axis Data Freeze X Axis	true
Show At Runtime	true
Show name	true
Data update	
Analysis Auto Update	false

Statistics: Equipa_em_viagem

Name	Value
General	
Statistics Value	assistencia.Eq_em_viagem()
Discrete	false
Show At Runtime	true
Data update	
Analysis Auto Update	true
Recurrence	hour()

Data Set: Equipa_em_viagem_DS

Name	Value
General	
Dataset Samples To Keep	150
Axis Data Freeze X Axis	true
Show At Runtime	true
Show name	true
Data update	
Analysis Auto Update	false

Statistics: Equipa_em_assistência

Name	Value
General	
Statistics Value	assistencia.Eq_em_serviço()
Discrete	false

Name	Value
Show At Runtime	true
Data update	
Analysis Auto Update	true
Recurrence	hour()

Data Set: Equipa_em_assistência_DS

Name	Value
General	
Dataset Samples To Keep	150
Axis Data Freeze X Axis	true
Show At Runtime	true
Show name	true
Data update	
Analysis Auto Update	false

Statistics: Equipamento_em_Substituição

Name	Value
General	
Statistics Value	equipamento.emSubstituição()
Discrete	false
Show At Runtime	true
Data update	
Analysis Auto Update	true
Recurrence	hour()

Data Set: Equip_Substituição_DS

Name	Value
General	
Dataset Samples To Keep	150
Axis Data Freeze X Axis	true
Show At Runtime	true
Show name	true
Data update	
Analysis Auto Update	false

Rectangle: rectangle

Name	Value
General	
Show At Runtime	true
Lock	false
Embedded Icon	false
Public	true
Appearance	
Line Color	black
Line Width	1.0

Name	Value
Line Style	SOLID
Position and size	
x	0.0
Width	170.0
y	0.0
Height	100.0
z	0.0
z Height	0.0
Rotation	0.0
Dynamic: X Scale	20
Dynamic: Y Scale	20
Dynamic: Z Scale	0
Advanced	
Shape Draw Mode	SHAPE_DRAW_2D3D
Show name	false

Agent Presentation: equipamento_presentation

Name	Value
General	
Show At Runtime	false
Public	true
Position and size	
x	0.0
y	0.0
z	0.0
Rotation	0.0
Embedded Object Presentation Scale Type	DEFINED_EXPLICITLY
Dynamic: X Scale	0.3
Dynamic: Y Scale	0.3
Advanced	
Shape Draw Mode	SHAPE_DRAW_2D3D
Embedded Object Presentation Drawing Mode	false
Show name	false

Agent Presentation: assistencia_presentation

Name	Value
General	
Show At Runtime	true
Public	true
Position and size	
x	-10.0
y	0.0
z	0.0
Rotation	0.0
Embedded Object Presentation Scale Type	DEFINED_EXPLICITLY

Name	Value
Dynamic: X Scale	0.3
Dynamic: Y Scale	0.3
Dynamic: Z Scale	1
Advanced	
Shape Draw Mode	SHAPE_DRAW_2D
Embedded Object Presentation Drawing Mode	false
Show name	true

Polyline: Sede

Name	Value
General	
Polyline closed	false
Show At Runtime	true
Lock	false
Embedded Icon	false
Public	true
Appearance	
Line Color	black
Line Width	1.0
Line Style	SOLID
Position and size	
x	80.0
y	45.0
z	0.0
z Height	0.0
Advanced	
Shape Draw Mode	SHAPE_DRAW_2D3D
Show name	false

Text: text4

Name	Value
General	
Show At Runtime	true
Lock	false
Embedded Icon	false
Public	true
Text	
Text	Disponibilidade dos equipamentos [médias anuais]
Appearance	
Bold Font Style	true
Font Size	14
Font Name	Arial
Color	maroon
Alignment	LEFT
Position and size	
x	538.0

Name	Value
y	10.0
z	0.0
Rotation	0.0
Advanced	
Shape Draw Mode	SHAPE_DRAW_2D
Show name	false

Text: text5

Name	Value
General	
Show At Runtime	true
Lock	false
Embedded Icon	false
Public	true
Text	
Text	Ocupação das equipas de assistência [médias anuais]
Appearance	
Bold Font Style	true
Font Size	14
Font Name	Arial
Color	maroon
Alignment	LEFT
Position and size	
x	540.0
y	340.0
z	0.0
Rotation	0.0
Advanced	
Shape Draw Mode	SHAPE_DRAW_2D
Show name	false

Agent Type: EquipamentoCliente

Name	Value
Agent actions	

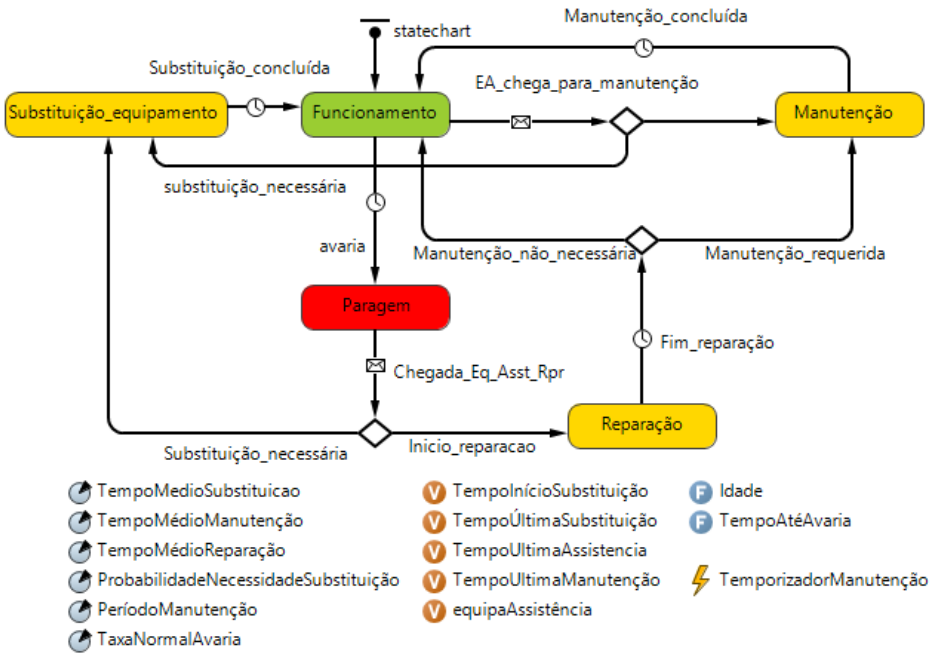
Startup Code	//Programar Manutenção periódica
	TemporizadorManutenção.restart(PeríodoManutenção - (time() - TempoUltimaManutenção));

Entity actions	
Flowcharts Usage	ENTITY
Movement parameters	
Velocity	6000/ day()
Rotate Animation Towards Movement	true
Rotate Animation Vertically	true
Advanced Java	
Generic	false
Advanced	
Make Default View Area	true
Auto-create Datasets	true

Name	Value
Recurrence	1
Limit the number of array elements	false

main

connections



Parameter: TaxaNormalAvaria

Name	Value
General	
Array	false
Type	double
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Parameter: TempoMedioSubstituicao

Name	Value
General	
Array	false

Name	Value
Type	double
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Parameter: TempoMédioReparação

Name	Value
General	
Array	false
Type	double
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Parameter: ProbabilidadeNecessidadeSubstituição

Name	Value
General	
Array	false
Type	double
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Parameter: PeríodoManutenção

Name	Value
General	
Array	false
Type	double
Show At Runtime	true
Show name	true
Advanced	

Name	Value
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Parameter: TempoMédioManutenção

Name	Value
General	
Array	false
Type	double
Show At Runtime	true
Show name	true
Advanced	
Modificator	STATIC
Use Units	false
Save In Snapshot	true
Editor	
Editor Control	TEXT_BOX

Function: Idade

Name	Value
General	
Return Type	double
ReturnModificator	RETURNS_VALUE
Show At Runtime	true
Show name	true
Arguments	
Parameters	()
Function body	
Body	return time() - TempoÚltimaSubstituição;
Advanced	
Static	false
Access Type	default
Use Units	false

Function: TempoAtéAvaria

Name	Value
General	
Return Type	double
ReturnModificator	RETURNS_VALUE
Show At Runtime	true
Show name	true
Arguments	
Parameters	()
Function body	

Name	Value
Body	double tempodesdemanut = time() - TempoUltimaManutenção; double factor_manutperiod_ultrapassado = max(1, tempodesdemanut / PeríodoManutenção); double factor_idade = max(1, Idade() / (get_Main().PeríodosManutParaSubstituir * PeríodoManutenção)); return max (0, normal (60, (1/(TaxaNormalAvaria*factor_manutperiod_ultrapassado*factor_idade))));
Advanced	
Static	false
Access Type	default
Use Units	false

Event: TemporizadorManutenção

Name	Value
General	
Occurence Time	0
Occurs At Time	true
Mode	occuresOnce
Trigger Type	timeout
Show At Runtime	true
Show name	true
Action	

Action	//É necessária manutenção
	get_Main().pedirManutenção(this);

Variable: TempoÚltimaSubstituição

Name	Value
General	
Initial Value	uniform(-get_Main().PeríodosManutParaSubstituir*PeríodoManutenção, 0)
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: equipaAssistência

Name	Value
General	
Type	EquipaAssistência

Name	Value
Show At Runtime	true
Show name	true
Advanced	
Access Type	private
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: TempoUltimaAssistencia

Name	Value
General	
Initial Value	uniform(-1/TaxaNormalAvaria, 0)
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: TempoUltimaManutenção

Name	Value
General	
Initial Value	uniform(-PeríodoManutenção, 0)
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

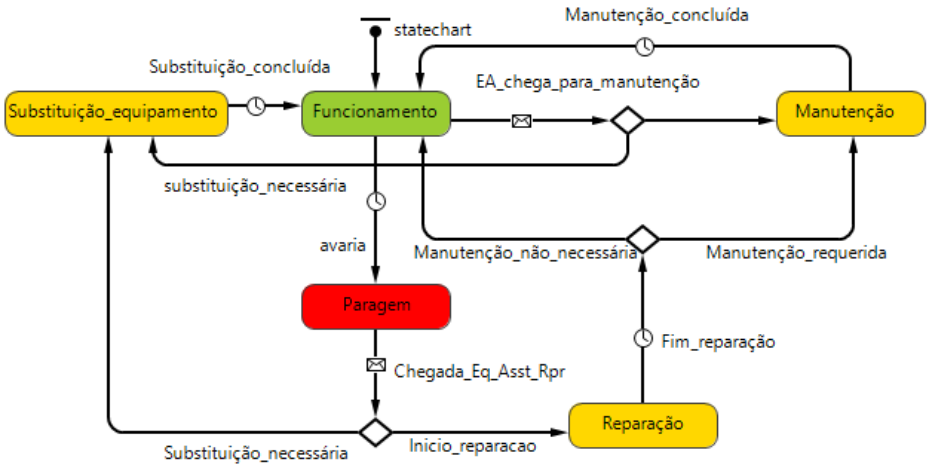
Variable: TempoInícioSubstituição

Name	Value
General	
Initial Value	0
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public

Name	Value
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Statechart Entry Point: statechart

Name	Value
General	
Show At Runtime	true
Show name	true



Transition: avaria

Name	Value
General	

Action	//Avaria - requisitar assistência
	get_Main().pedirAssistencia(this);

Timeout	TempoAtéAvaria()
Trigger Type	timeout
Show name	true

Transition: Chegada_Eq_Asst_Rpr

Name	Value
General	
Action	equipaAssistência = msg; //registar equipa que foi prestar assistência
Filter Type	unconditionally
Message Type	EquipaAssistência
Trigger Type	message
Show name	true

Transition: Inicio_reparacao

Name	Value
General	
Default Transition	true
Show name	true

Transition: Fim_reparação

Name	Value
General	
Action	TempoUltimaAssistencia = time();
Timeout	triangular (TempoMédioReparação*0.5 , TempoMédioReparação, TempoMédioReparação*2.5)
Trigger Type	timeout
Show name	true

Transition: Manutenção_requerida

Name	Value
General	
Condition	! TemporizadorManutenção.isActive() //Manutenção necessária, aproveitar a presença da equipa de assistência
Default Transition	false
Show name	true

Transition: Manutenção_concluída

Name	Value
General	

Action	//Libertar a equipa de assistência
	send("Terminado", equipaAssistência);
	equipaAssistência = null;
	
	TemporizadorManutenção.restart(PeríodoManutenção);
	TempoUltimaManutenção=time();

Timeout	triangular(0.666 * TempoMédioManutenção,TempoMédioManutenção,1.333*Tempo MédioManutenção)
Trigger Type	timeout
Show name	true

Transition: Manutenção_não_necessária

Name	Value
General	

Action	//Libertar a equipa de assistência
	send("Terminado", equipaAssistência);
	equipaAssistência = null;

Default Transition	true
Show name	true

Transition: Substituição_necessária

Name	Value
General	
Action	TempoInícioSubstituição = time()

Condition	//Probabilidade que a substituição seja a única forma de repor em funcionamento
	get_Main().SubstituirEquipamentosAntigos && Idade() > get_Main().PeríodosManutParaSubstituir * PeríodoManutenção || get_Main().SubstituirEquipamentosAntigos && randomTrue(ProbabilidadeNecessidadeSubstituição)

Default Transition	false
Show name	true

Transition: Substituição_concluída

Name	Value
General	

Action	//Libertar a equipa de assistência
	send("Terminado", equipaAssistência);
	equipaAssistência = null;
	
	//Actualizar registo de substituição
	
	TempoÚltimaSubstituição = time();
	TemporizadorManutenção.restart(PeríodoManutenção);

Timeout	triangular (6/8*TempoMedioSubstituicao,TempoMedioSubstituicao, 10/8*TempoMedioSubstituicao)
Trigger Type	timeout
Show name	true

Transition: EA_chega_para_manutenção

Name	Value
General	
Action	equipaAssistência = msg; //registar a equipa que chegou
Filter Type	unconditionally
Message Type	EquipaAssistência
Trigger Type	message
Show name	true

Transition: transition

Name	Value
General	
Default Transition	true
Show name	false

Transition: substituição_necessária

Name	Value
General	
Action	TempoInícioSubstituição=time()

Name	Value
Condition	//policy is active and equipment is older than 3 mtce periods get_Main().SubstituirEquipamentosAntigos && Idade() > get_Main().PeríodosManutParaSubstituir * PeríodoManutenção
Default Transition	false
Show name	true

State: Funcionamento

Name	Value
General	
Fill Color	yellowGreen
Show name	true

State: Paragem

Name	Value
General	
Fill Color	red
Show name	true

State: Reparação

Name	Value
General	
Show name	true

State: Manutenção

Name	Value
General	
Show name	true

State: Substituição_equipamento

Name	Value
General	
Show name	true

Branch: branch1

Name	Value
General	
Show name	false

Branch: branch

Name	Value
General	
Show name	false

Branch: branch3

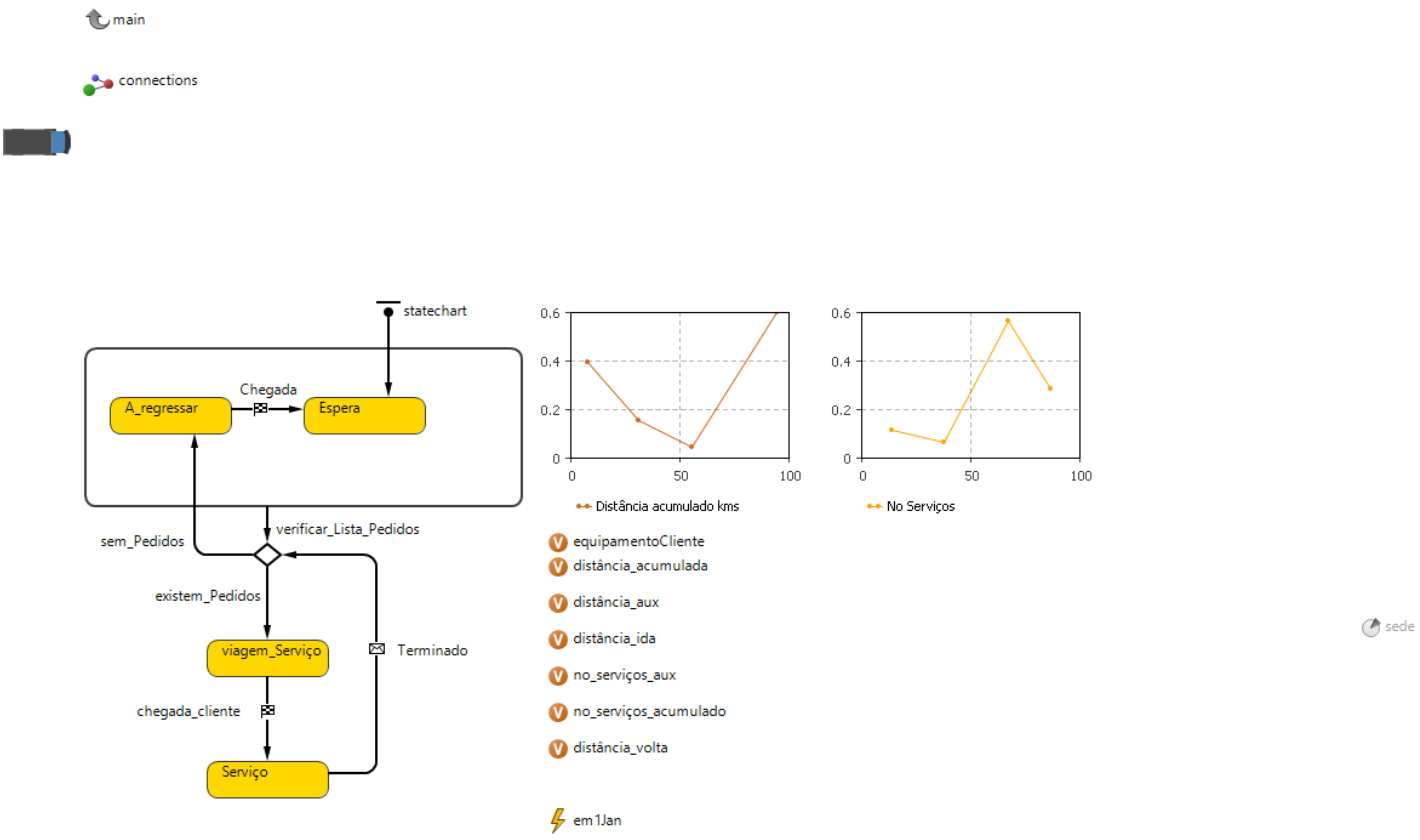
Name	Value
General	
Show name	false

3D Object: factory_2

Name	Value
General	
Axis Order	XYZ_AXIS_ORDER
Scale	0.1
File Name	x3d/factory_2.x3d
Show At Runtime	true
Lock	false
Public	true
Position	
x	0.0
y	0.0
z	0.0
Rotation	0.0
Advanced	
Shape Draw Mode	SHAPE_DRAW_2D3D
Apply Shading	true
Show name	false

Agent Type: EquipaAssistência

Name	Value
Agent actions	
Startup Code	setXY(85, 50);
Entity actions	
Flowcharts Usage	ENTITY
Movement parameters	
Velocity	(560/day())/1.38
Rotate Animation Towards Movement	true
Rotate Animation Vertically	false
Advanced Java	
Generic	false
Advanced	
Make Default View Area	true
Auto-create Datasets	true
Recurrence	1
Limit the number of array elements	false



Event: em1Jan

Name	Value
General	
Event Recurrence Time Unit	YEAR
Recurrence	1
Event Occurrence Date	Fri Jan 01 00:00:01 GMT 2016
Occurs At Time	false
Mode	cyclic
Trigger Type	timeout
Show At Runtime	true
Show name	true
Action	

Action	//distância_acumuladaDS.reset();
	distância_acumulada = 0;
	distância_aux = 0

Variable: equipamentoCliente

Name	Value
General	
Type	EquipamentoCliente
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false

Name	Value
Constant	false
Save In Snapshot	true
Use Units	false

Variable: distância_acumulada

Name	Value
General	
Initial Value	0
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: distância_aux

Name	Value
General	
Initial Value	0
Type	double
Show At Runtime	false
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: distância_ida

Name	Value
General	
Initial Value	0
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: no_serviços_aux

Name	Value
General	
Initial Value	0
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: no_serviços_acumulado

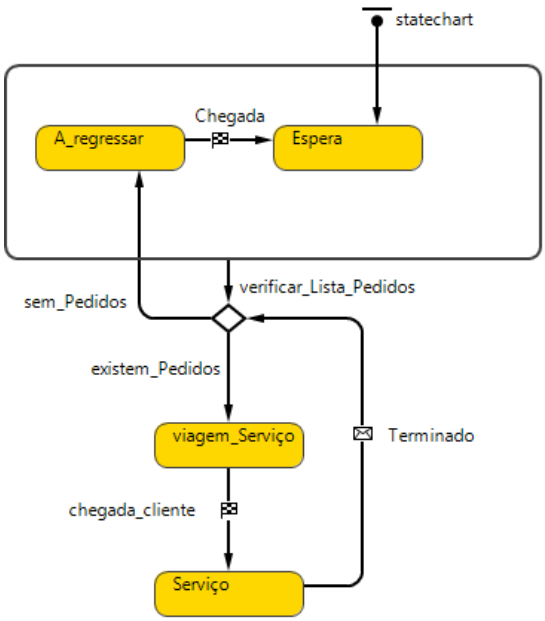
Name	Value
General	
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Variable: distância_volta

Name	Value
General	
Type	double
Show At Runtime	true
Show name	true
Advanced	
Access Type	public
Static	false
Constant	false
Save In Snapshot	true
Use Units	false

Statechart Entry Point: statechart

Name	Value
General	
Show At Runtime	true
Show name	true



Transition: verificar_Lista_Pedidos

Name	Value
General	
Action	distância_ida=0;
Equals	"Verificar lista de pedidos"
Filter Type	equalsTo
Message Type	String
Trigger Type	message
Show name	true

Transition: existem_Pedidos

Name	Value
General	
Action	equipamentoCliente = get_Main().captarPedido(); moveTo(equipamentoCliente.getX(), equipamentoCliente.getY());

Condition	//Há pedidos
	get_Main().háPedidos()

Default Transition	false
Show name	true

Transition: chegada_cliente

Name	Value
General	
Action	send(this, equipamentoCliente);
Trigger Type	arrival
Show name	true

Transition: Terminado

Name	Value

Name	Value
General	
Action	equipamentoCliente = null;
Equals	"Terminado"
Filter Type	equalsTo
Message Type	String
Trigger Type	message
Show name	true

Transition: sem_Pedidos

Name	Value
General	
Action	moveTo(85,50);
Default Transition	true
Show name	true

Transition: Chegada

Name	Value
General	
Action	distância_volta = distância_ida; distância_acumulada = distância_volta+distância_aux;
Trigger Type	arrival
Show name	true

State: Espera

Name	Value
General	
Entry Action	distância_aux = distância_acumulada;
Show name	true

State: state

Name	Value
General	
Show name	false

State: viagem_Serviço

Name	Value
General	

Entry Action	//calcular distância viajada
	double x;
	double y;
	x = equipamentoCliente.getX()-85;
	y = equipamentoCliente.getY()-50;
	distância_acumulada = 1.38*sqrt(x*x+y*y)+distância_aux;
	distância_ida = 1.38*sqrt(x*x+y*y);
	no_serviços_acumulado=1+no_serviços_aux

Show name	true

State: Serviço

Name	Value
General	
Entry Action	distância_aux = distância_acumulada; no_serviços_aux=no_serviços_acumulado;
Show name	true

State: A_regressar

Name	Value
General	
Entry Action	
Show name	true

Branch: branch

Name	Value
General	
Show name	false

Time Plot: plot

Name	Value
General	
Public	false
Data update	
Analysis Auto Update	true
Recurrence	1
Dataset Samples To Keep	100
Scale	
Time Window	100
Time Window Units	MODEL_TIME_UNIT
Vertical Scale	AUTO
Appearance	
Labels Horizontal Position	DEFAULT
Labels Vertical Position	DEFAULT
Label Format	MODEL_TIME_UNITS
Labels Text Color	darkGray
Chart Area Grid Color	darkGray
Draw Line	true
Interpolation	LINEAR
Position and size	
x	390.0
Width	260.0
y	110.0
Height	210.0
Legend	
Show Legend	true
Legend Size	30.0

Name	Value
Legend Text Color	black
Legend Place	SOUTH
Chart area	
Chart Area: X Offset	50.0
Chart Area: Width	180.0
Chart Area: Y Offset	30.0
Chart Area: Height	120.0
Chart Area: Background Color	white
Chart Area Border Color	black
Advanced	
Time Windows Movement Type	MOVEMENT_WITH_TIME
Show name	false

Plot Items:

Title	Type	Dataset / Value	Point Style	Color	Line	Width	Interpolation
Distância acumulado kms	value	distância_acumulada	CIRCLE	chocolate	true	1	LINEAR

Time Plot: no_serviços

Name	Value
General	
Public	false
Data update	
Analysis Auto Update	true
Recurrence	1
Dataset Samples To Keep	100
Scale	
Time Window	100
Time Window Units	MODEL_TIME_UNIT
Vertical Scale	AUTO
Appearance	
Labels Horizontal Position	DEFAULT
Labels Vertical Position	DEFAULT
Label Format	MODEL_TIME_UNITS
Labels Text Color	darkGray
Chart Area Grid Color	darkGray
Draw Line	true
Interpolation	LINEAR
Position and size	
x	630.0
Width	260.0
y	110.0
Height	210.0
Legend	
Show Legend	true
Legend Size	30.0
Legend Text Color	black
Legend Place	SOUTH

Name	Value
Chart area	
Chart Area: X Offset	50.0
Chart Area: Width	180.0
Chart Area: Y Offset	30.0
Chart Area: Height	120.0
Chart Area: Background Color	white
Chart Area Border Color	black
Advanced	
Time Windows Movement Type	MOVEMENT_WITH_TIME
Show name	false

Plot Items:

Title	Type	Dataset / Value	Point Style	Color	Line	Width	Interpolation
No Serviços	value	no_serviços_acumulado	CIRCLE	orange	true	1	LINEAR

3D Object: lorry

Name	Value
General	
Axis Order	YZX_AXIS_ORDER
Scale	1.0
File Name	x3d/lorry.x3d
Show At Runtime	true
Lock	false
Public	true
Position	
x	0.0
y	0.0
z	0.0
Rotation	0.0
Advanced	
Shape Draw Mode	SHAPE_DRAW_2D3D
Apply Shading	true
Show name	false

Simulation Experiment: Simulation

Name	Value
General	
Maximum Available Memory	128
Active Object Class	Main
Model time	
Execution Mode	realTimeScaled
Real Time Scale	10.0
Use Calendar	true
Stop Option	Stop at specified date
Initial Time	0.0
Initial Date	Thu Jan 01 00:00:00 GMT 2015
Final Date	Wed Dec 31 23:59:59 GMT 2025

Name	Value
Randomness	
Random Number Generation Type	randomSeed
Selection Mode For Simultaneous Events	random
Window	
Real Time Of Simulation	false
Model Date	true
View	true
Animation setup	true
Title	Simulação
Enable Panning	true
Enable Zoom	true
Maximized	true
Close Confirmation	false
Advanced	
Enable Antialiasing	true
Enable Enhanced Model Elements Animation	true
Adaptive Frame Management	true
CPU Time Balance	ratio_1_2
Load Root From Snapshot	false

Simulação

Run

Text: text

Name	Value
General	
Show At Runtime	true
Lock	false
Text	
Text	Simulação
Appearance	
Font Size	24
Font Name	SansSerif
Color	royalBlue
Alignment	LEFT
Position and size	
x	40.0
y	30.0
z	0.0
Rotation	0.0
Advanced	
Show name	false

Button: button

Name	Value
General	
Dynamic: Label	getState() == IDLE ? "Run" : "Top level agent"
Label Text	Run
Action	

Action	if (getState() == IDLE)
	run();
	getPresentation().setPresentable(getEngine().getRoot());

Position and size	
x	40.0
Width	100.0
y	80.0
Height	30.0
Advanced	
Font Size	11
Font Name	Dialog
Show name	false