Proceedings of the 2013 Workshop on
Complex Systems Modelling and Simulation

CoSMoS 2013

Susan Stepney, Paul S. Andrews
Editors

CoSMoS 2013

Luniver Press
2013

Published by Luniver Press
Frome BA11 6TT United Kingdom

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

CoSMoS 2013

ISBN-10: 1-905986-39-4
ISBN-13: 978-1-905986-39-2

Preface

The CoSMoS workshops series has been organised to disseminate best practice in complex systems modelling and simulation, with its genesis in the similarly-named CoSMoS research project, a four year EPSRC funded research project at the Universities of York and Kent in the UK. Funding for the CoSMoS project has now completed, but we have continued to run the workshop series as a forum for research examining all aspects of the modelling and simulation of complex systems. To allow authors the space to describe their systems in depth we put no stringent page limit on the submissions.

We are pleased to be running the sixth CoSMoS workshop as a satellite event at the 12th International Conference on Unconventional Computation and Natural Computation (UCNC 2013) at the Università degli Studi di Milano-Bicocca, Milan, Italy. UCNC explores all aspects of unconventional and natural computation, an area rich in the inherent complexity within systems, providing a natural complement to the issues addressed by the CoSMoS workshop.

The main session of the workshop is based on seven accepted full paper submissions:

Barr et al. describe an algorithm for efficient simulation of a kind of unconventional computation: quantum random walks on graphs.

Evora et al. describe an analysis technique applied to the output from a complex Smart Grid system simulation, used to aid the decision making process.

Fehér et al. describe a more abstract approach to transforming Simulink models that should aid model reuse.

Garnett examines "tipping points", the rapid flipping of a complex system from one quasi-stable state to another, in the context of the banking sector acquisitions and mergers.

Li et al. use ideas from the CoSMoS approach to repurpose a simulation to apply to a different domain.

Stepney describes how the ODD protocol can be used to help present CoSMoS simulation experiments in a format that aids their reproducibility.

Tao & Liu examine self-organisation in complex healthcare systems, using an Autonomy-Oriented Computing approach.

Our thanks go to all the contributors for their hard work in getting these papers prepared and revised. All submissions received multiple reviews, and we thank the programme committee for their prompt, extensive and in-depth reviews. We would also like to extend a special thanks

to the organising committee of UCNC 2013 for enabling our workshop to be co-located with this conference. We hope that readers will enjoy this set of papers, and come away with insight on the state of the art, and some understanding of current progress in complex systems modelling and simulation.

Programme Committee

Table of Contents

CoSMoS 2013

x

Simulation methods for quantum walks on graphs applied to perfect state transfer and formal language recognition

Katie Barr, Toby Fleming, Viv Kendon

School of Physics and Astronomy, E C Stoner Building, University of Leeds, Leeds, LS2 9JT, UK

Abstract. We describe an algorithm which automates the generation of appropriate shift and coin operators for a discrete time quantum walk, given the adjacency matrix of the graph over which the walk is run. This gives researchers the freedom to numerically investigate any discrete time quantum walk over graphs of a computationally tractable size by greatly reducing the time required to initialise a given walk. We then describe two situations in which the swift initialisation of walks has enabled systematic investigations of walks over a large number of structures. The results of these simulations and their reliability, as well as the general suitability of numerical analysis as a tool for investigating discrete time quantum walks, are briefly discussed. We also mention specific Python packages which facilitate our simulations and analysis, motivating the use of high level programming languages in this context.

1 Introduction

Recently, there has been much interest in the discrete time quantum walk, due to its applicability in creating efficient quantum algorithms [1, 27, 32]. The inspiration for the development of these walks came from the fact that classical random walks have been used to develop new algorithms which outperform their predecessors. For example, the best known algorithms to solve the constraint satisfiability problem, where one determines whether a collection of objects have a certain set of properties, use the classical random walk [24, 26]. It was therefore natural, given that quantum algorithms have been shown to outperform known classical algorithms, particularly at searching [12] and factorization [28], to develop a quantum version of the random walk as an algorithmic tool.

Whilst this strongly motivates the development of algorithms based on quantum principles, in order to attain their efficiency in practice these algorithms would have to be run on a quantum computer.

Discrete time quantum walks are an example of a composite quantum system. In this case it is composed of two systems which correspond to the structure the walker traverses and the degrees of freedom introduced for performing a 'quantum coin flip.' Composite quantum systems swiftly increase in size as degrees of freedom are added to their sub-systems, so accurate simulations can be difficult. Quantum walks are theoretical models which can be exactly simulated in some cases. In general they can be simulated with a high degree of precision. Exact analytical results concerning these walks are very difficult to obtain, so they are particularly suited to numerical simulation. While quantum walks on simple or regular structures are easy to simulate up to computationally tractable sizes [16], simulations over arbitrary graphs require customised operators at each node of different degree.

In this paper we describe a simple algorithm to generate time evolution operators for the discrete time quantum walk over arbitrary structures. This has greatly facilitated the authors' own investigations into the discrete time quantum walk in a variety of contexts, in one case extending their applicability, and hence constitutes a significant contribution to the theory of discrete time quantum walks. We start in Section 2 by defining discrete time quantum walks over a given graph structure. In Section 3 we describe the algorithm. We then outline two specific applications of the discrete time quantum walk and the simulations we performed. The first, in Section 4.1 is a brute force search for perfect state transfer over structures having certain properties. The second, described in Section 4.2 applies the quantum walk to language acceptance problems, interpreting acceptance as the walker being at a specific node after a set number of timesteps.

2 Quantum walks

All purely quantum states are represented by a complex state vector, the components of which are called amplitudes. The time evolution is represented by unitary operators, which fulfil the criteria $\mathcal{U}\mathcal{U}^\dagger = \mathcal{U}^\dagger\mathcal{U} = \mathbb{I}$ where $\mathcal{U}^\dagger$ is the conjugate transpose, otherwise known as the adjoint, of $\mathcal{U}$. A discrete time quantum walk evolves over an arbitrary graph structure $\mathcal{G} = \{E, V\}$ where V is a set of nodes and E is a set of edges of the form (i, j) where $i, j \in V$. The adjacency matrix $A_\mathcal{G}$ of $\mathcal{G}$ has ones in the entries (i, j) if node i is connected to node j, and zeroes elsewhere. The number of nodes is denoted $|V|$. The number of edges

incident on a given node is called the degree of that node and is denoted $|v|$. The quantum walk model used in this paper assumes that the graph is undirected, so for every $(i, j) \in E$ we have that $(j, i) \in E$, but there are models of quantum walks which used directed graphs [21, 33].

The time evolution of the walk is determined by a unitary operator $\mathcal{U} = \mathcal{SC}$ with $\mathcal{S}$ being a shift operation, and $\mathcal{C}$ being a 'coin' operation. The coin operation is required in order to guarantee unitary, that is to say validly quantum, evolution. To see the role of the coin operator, consider a walk on a line. In contrast with the classical coin, which definitely sends the walker either left or right, the quantum coin sends the walker both left and right in proportions depending on the precise choice of coin. The coin operator acts on the nodes of the graph, each node has a set of coin states, each coin state indicating the edge along which amplitude arrived at the node. There can be a different coin operator at each node, and to get the operator over the entire graph, the direct sum of the individual operators is required. The coin operator controls which node amplitude is directed to in the next step of the walk, as the coin state also specifies which edge amplitude should leave by.

The shift operator acts such that $\mathcal{S}|v, c\rangle = |w, d\rangle$, so moves amplitude from the c^{th} coin state of v to the d^{th} coin state of w [17], where c and d label the ends of the edge between nodes v and w. There is a one to one correspondence between edges (v, w) and coin states (c, d). This bijection guarantees the existence of a consistent labelling scheme for the coin states at each node.

The state of the walker after T steps is

$$\psi(T) = \sum_{v,c} \alpha_{v,c}(T)|v, c\rangle \tag{1}$$

where $\alpha_{v,c} \in \mathbb{C}$ is called the amplitude of the walker at position v in coin state c and $|v, c\rangle$ denotes a basis state on node v with coin state c. The probability of the walker being found at node v after T steps is the summation over coin states at v, $p(v, t) = \sum_i |v, c_i|^2$. The state of the quantum walker is simply a complex vector, and for numerical analysis the time evolution operators are represented by complex unitary matrices which are not generally sparse.

An example coin operator, which was used heavily in the applications described in Section 4 below, is the Grover operator:

$$G_{|v|} = \begin{pmatrix} \frac{2-|v|}{|v|} & \frac{2}{|v|} & \cdots & \frac{2}{|v|} \\ \frac{2}{|v|} & \frac{2-|v|}{|v|} & \cdots & \frac{2}{|v|} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{|v|} & \frac{2}{|v|} & \cdots & \frac{2-|v|}{|v|} \end{pmatrix} \tag{2}$$

Where $|v|$ is the dimension of the node we are operating at (the degree of that node in the graph), and the operator is a $|v| \times |v|$ matrix.

As mentioned in the introduction, the discrete time quantum walk is a composite system, its state describes both the coin state and the position of the walker. This state must have basis states which describe every possible position and coin state, so the number of required states increases quickly with the number of nodes of the graph and connections between them. It is the size of the state vector describing the walker, coupled with the matrix multiplication required to evolve it, which makes the simulation of quantum walks, or indeed any large quantum system, a nontrivial computational task. Due to the degrees of freedom introduced by the coin space, the discrete time quantum walk is very difficult to investigate analytically. Some cases on regular lattices have been solved exactly using path counting and Fourier space techniques. The case of the walk on the line was solved by Ambainis *et al* [2], the hypercube was treated by Moore and Russell [22] and then Kempe [15], and higher dimensional lattices were treated by Gottlieb *et al* in [11]. General solutions for walks over arbitrary structures have not yet been attempted. In cases where the initial states and entries in the coin operator are represented by numbers which can be handled exactly by the simulation, such as those in the Grover operator with the walker initially localised in a specific coin state of a given node, they can be exactly simulated. In general, exact simulations are rare, only being possible for a small number of timesteps or in cases where the evolution is periodic. Our own simulations coupled with subsequent analytic work for a few exactly solvable, highly symmetric, cases indicate that the walks can be simulated to a very high degree of precision for at least 100 timesteps. Much longer walks can be simulated [16] but eventually numerical accuracy may become an issue. Their difficulty in being treated analytically coupled with their suitability for numerical investigation is why, thus far, the vast majority of results concerning the discrete time quantum walk have been obtained numerically [18, 20, 29, 31]. This contrasts with the case of the continuous time walk where numerical methods are less suitable, as

numerical integration is required to simulate the time evolution. These walks are, however more amenable to analytic techniques [5, 6, 19, 25]. As they obey the Schrödinger equation their evolution can be written in terms of the eigensystems of their Hamiltonian, which is either the adjacency matrix or Laplacian of their graph structure. In cases where the eigenvalues and vectors have simple expressions, it is easy to write out the full time evolution of the walk for a given initial state.

3 Algorithm to generate a quantum walk from the adjacency matrix of a graph

The problem solved by the algorithm outlined in this section can be stated thus: Given the adjacency matrix of a graph, generate appropriate time evolution operators to simulate a discrete time quantum walk over that structure. Mathematically, this corresponds to performing the appropriate matrix tensor product operations. As should be clear from the definition of a quantum walk, two such operators are required, the coin operator $\mathcal{C}$, and the shift operator $\mathcal{S}$. The coin operator is simpler to generate, so we discuss this first.

3.1 Generating the coin operation

The coin operator acts locally on each node of the graph. Due to the structure and ordering of the direct sum, we know that the operator will be represented by a block diagonal matrix, with each block acting on a different node of the graph, and the block's dimension will be equal to the degree of that node. In order to generate the operator, the relevant coin acting at each node is generated, and then we must ensure that it is placed in the appropriate position in the large coin matrix. To do this, the degree of each node is required, and this can be found easily by taking the sum of the row representing that node in the adjacency matrix. Then the operations at each node must be specified, and there is a large amount of freedom here. The simplest case is to have coins of the same type operate at each node. For example, the Grover or DFT operators of appropriate dimensions can be specified easily. In some walks discussed below, different types of coin are used at different nodes. Exceptions for nodes of a specific degree, or even for specific nodes, can easily be added. A very simple example of generating the coin operation can be given if we consider a cycle of two nodes, which in standard notation is referred to as C_2 and is depicted in Figure 1. Using the two dimensional DFT operator, also known as the Hadamard, at both nodes gives rise to the following coin operator:

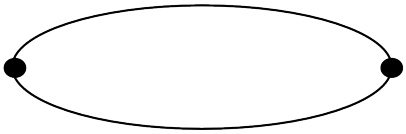


Fig. 1. The cycle of two nodes

$$\mathcal{C} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \tag{3}$$

3.2 Generating the shift operation

In order to ensure that the consistent labelling scheme required to guarantee unitarity of the walk is taken into account, care must be taken in generating the shift operator. For each node, we need not only the degree, but the indices of the other nodes it is joined to. In Python it is easy to generate a list of these by looping over the relevant row in the adjacency matrix $A_{\mathcal{G}}$, and this list can then be used to specify the ordering of the coin states at that node. This list has the form:

$$\text{list\_1} = ([i, j, k], [l, m] ... [v, w])$$

where the index of each entry specifies which node the nodes i, j, k etc. are attached to. It is also useful to create another list:

$$\text{list\_2} = [0, 3, 5 ...]$$

specifying the index of the first coin state for each node. So the nth entry of list_1 tells us which nodes node n is joined to, and the corresponding entry of list_2 tells us which coin state joins n to the first entry list_1 $[n]$. The dimension of the shift operator is simply the sum of the number of coin states at each node. At a given node v, which corresponds to the row/column numbered v in $A_{\mathcal{G}}$, the correct permutation between coin states is the most difficult part to find. The algorithm can be described thus:

```
Initialise list_1: an empty list of length |V|
Initialise list_2: an empty list of length |V|
Set coin_state_counter to zero
For v < |V|
      For x < |v|
         If A_G[v][x] == 1
            Append x to list_1[v]
      list_2 [v] = coin_state_counter
      Add |v| to coin_state_counter
shift dimension = coin_state_counter
Initialise shift operator, a square array of size shift
dimension
For v < |V|
     // The first coin state, a_v, at the node v is list_2[v]
     a_v = list_2[v]
     For x < |v|
          // The node a_v + x joins to, j, is given by
          // list_1[v][x]
           j = list_1[v][x]
          a_j = list_2[j]
          For y < |j|
             // Find the coin state of j which joins to
             // node v
             When list_1[j][y] == v
                b = y
          shift[a_v + x][a_j + b] = 1
```

In Figure 2 the relations between the values used by this algorithm are shown schematically. Performing this routine for each node of the graph gives half of the shift operator. For each nonzero entry (i, j) in the array simply populate the corresponding entry (j, i) and we have the required permutation matrix.

This routine can be used to generate an appropriate shift operator for any standard adjacency matrix, that is, any whose entries are only zeroes and ones, including those which contain self loops. More care must

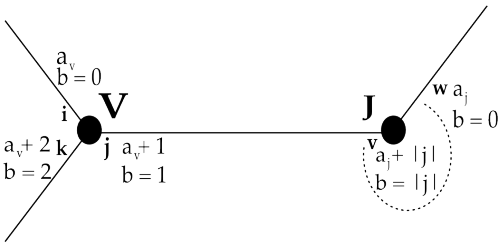


Fig. 2. Relation between nodes (bold capitals), edge labels (bold lower case) and book keeping quantities for a pair of nodes on an arbitrary graph structure. The dashed line at node j indicates that there are possibly other edges

be taken if we wish to generate shift operators for graphs with multiple connections between edges, but the principles remain the same. Clearly, the size of the structure which can be simulated depends on the available memory. The authors have been able to use the functions created using the above routine to run walks over graphs with more than 7500 nodes for approximately 100 timesteps in less than five minutes using a standard desktop computer. Currently simulations of quantum walks on regular graphs, which can be simulated in a much more compact way, cannot have more than 10^{12} sites. The limits imposed by memory considerations for the size of quantum walk we can classically simulate are discussed in more detail and for a variety of situations in [16].

Once the appropriate shift and coin operators have been created, the only things left to be specified are the number of timesteps the walk should be run for, the initial state of the walker and the information we would like to gain about the walk.

3.3 Example

Before moving onto the applications of this algorithm, we illustrate it for the simple case of the walk on the line. Clearly it is not possible to simulate a walk on an infinite line, but analytic proofs of the asymptotic behaviour can be found in [2]. For simplicity, we illustrate the first two steps of the walk using the Hadamard operator (used in Equation 3), shown in Figure 3. If the walker is initially localised at a single node then this walk takes place over five nodes of the line. To account for the extra coin states which would occur were the line infinite, we add self

loops to the ends of the line of length five. The adjacency matrix for a line with five nodes is then:

$$\begin{pmatrix} 1&1&0&0&0 \\ 1&0&1&0&0 \\ 0&1&0&1&0 \\ 0&0&1&0&1 \\ 0&0&0&1&1 \end{pmatrix} \tag{4}$$

This requires the shift operator:

$$\begin{pmatrix} 0&1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1&0 \end{pmatrix} \tag{5}$$

As the protocol for generating the shift is identical for each node on the line, we only need to describe how to generate the shift for a single node, call this v. Say the first coin state of this node is called a_{-1} and the second is called a_{+1}, which join to nodes $v-1$ and $v+1$ respectively. The coin state a_{-1} moves amplitude to coin state a_{+1} of node $v-1$ as it has come from the node in the $+1$ direction, with the corresponding situation for node $v+1$. Coin state a_{-1} of v is joined to coin state of a_{+1} of $v-1$ by finding the index of a_{-1} in the shift operator. Using the bookkeeping lists it is easy to find out which node a_{-1} should join to, and we use them again to find out what the index of the correct coin state at that node should be in the shift operator.

On the line a two dimensional coin is required at each node, inducing the following operator over five nodes:

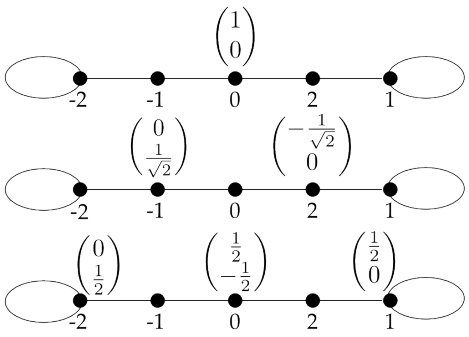


Fig. 3. The first two steps of the walk on the line

$$\frac{1}{\sqrt{2}}\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1
\end{pmatrix} \tag{6}$$

Starting at the center of the node in the 'moving left' state, after the first coin flip we have:

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \tag{7}$$

So after the shift operation equal portions of the amplitude have moved to the nodes to the left and right of the origin. In the second step half of the amplitude returns to the central node, with the rest propagating in the left and right directions, as in Figure 3. The fact that some amplitude goes away from the origin at each step is the reason why the quantum walk propagates linearly with the number of timesteps T, rather than with $\sqrt{T}$ in the classical case.

As there are an infinite number of possible graphs with an arbitrary number of nodes and edges, and uncountably infinite coin operators, it

is not possible to numerically study every possible walk. It is therefore necessary to narrow down the number of parameters that can be varied by choosing particular types of walk to study. These choices will depend on the reason why we are simulating the walk, and we now turn to two applications which could not have been investigated in a timely fashion without automated generation of the time evolution operators.

4 Applications

With code capable of generating appropriate shift and coin operators from an adjacency matrix, any discrete time quantum walk of reasonable size can be simulated. In this section, two applications of the outlined algorithm are described and the results briefly discussed. Both applications facilitate investigations into areas of current interest, namely perfect state transfer and the interpretation of quantum walks as quantum computers. The following numerical work was implemented in Python2.6, using built in functions from the `math` and `numpy` modules in addition to our own and the ones mentioned specifically below.

4.1 Perfect state transfer over small structures

In [4] we undertook a systematic investigation of quantum walks over specific types of small structures in order to find out which structures admit a particular type of quantum transport called perfect state transfer. In this case the 'state' is described by the wavefunction, usually denoted ψ but here denoted $|\psi\rangle = \sum_{v,c} \alpha_{v,c}|v,c\rangle$ to clarify that the state specifies both the position and coin states. The problem then is to find out how and when the entire state can be transported from one node of the graph to another. Therefore perfect state transfer is deterministic transport from some node v to another node w, and occurs after T steps when the following condition is met:

$$\sum_{c,d} \langle w,d|(\mathcal{SC})^T|v,c\rangle = 1. \tag{8}$$

It is clear from this definition that we are not concerned about whether the configuration of coin states at node w is the same as they were at node v, which enables perfect state transfer between nodes of different degrees. The summation is over coin states only, as typically in perfect state transfer scenarios the state is localised at a specific node.

The structures we investigated were based on small cycles, specifically C_4, C_6 and C_8. This choice was based on the fact that these cycles were known to admit perfect state transfer between antipodal nodes under

certain conditions [30]. Up to four nodes were added to each cycle, in the way described in [4]. In the case of C_4 his gave rise to 1042 graphs to test, of which many were isomorphic. Due to the fact that we ran our simulations from the same node of the graph each time, it was necessary to test the isomorphic cases too, this effectively tested between different pairs of nodes in the graph. Simulations were then run for three types of coin operator for 100 timesteps, using 1500 initial states. The initial states always had the walker initially localised at a specific node, and the coin states were chosen randomly such that they were uniformly distributed according to the Haar measure [23], this ensures all possible states have equal probabilities of arising. Using the formulae from [23] and a random number generator to obtain the necessary parameters, these initial states are easy to generate.

As we were looking for perfect state transfer to a specific node, the output from the simulation was simply a list of the structures, initial states, timesteps and probabilities for all cases where the probability was greater than 0.99. Further analysis of the walks returned by the simulation revealed that, in fact, perfect state transfer only occurred when the probability of the walker being at the designated node calculated using Python was exactly equal to one. This analysis was performed in MAPLE, by using it to solve for initial states such that perfect state transfer occurred on the graph investigated after the number of timesteps suggested by the Python simulations. Using MAPLE we were able to solve the relevant equations so we can be sure of this conclusion. MAPLE is not suitable for brute force searching, but it is simple to verify results highlighted by searches using their built in equation solving routines. Our approach to the analysis eliminates the possibility that initial states close to, but not identical to those tested gave rise to perfect state transfer on the graphs highlighted by the Python simulations. Despite preliminary investigations suggesting that increasing the number of initial states beyond our selected number did not affect the results, it is still possible that some cases of perfect state transfer were missed, however, as all previously known cases were identified by the simulation, this seems unlikely. Due to the infinite number of possible initial states, it is not possible to verify numerically that no initial state gives rise to a particular form of transport.

The overall conclusion from the simulations was that perfect state transfer is extremely rare, and the properties which preserve it are similar regardless of whether the graph is based on C_4, C_6 or C_8. However, some of the structures, shown in Figure 4, were found to admit perfect state transfer and could be generalised into families of graphs. The results for these infinite families were verified analytically. The robustness of this

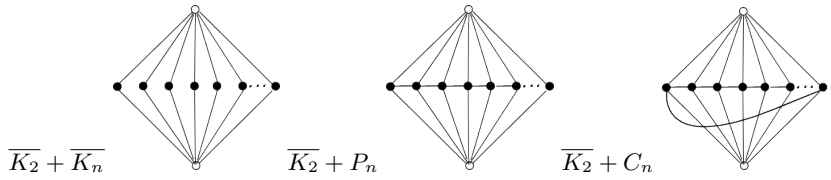


Fig. 4. The three families. The nodes highlighted with open dots indicate the initial and target nodes, due to the symmetry of the graphs, it does not matter which is which.

transfer and the transport properties of continuous time walks over these graphs were also examined. The continuous time walk also had perfect, or very high amplitude, state transfer and in one case it was possible to write down an analytic expression for the time evolution of the walk.

Python has many sophisticated packages which greatly facilitated further analysis of the graphs created. In particular, the `networkx` package can convert adjacency matrices, created as `numpy` arrays, into specific graph data structures. These can be tested for isomorphisms, which could then be removed. The isomorphic cases were needed for the perfect state transfer investigations, to simplify the simulation by allowing the walker to always start at the same node, in which case isomorphic graphs can give rise to different walk dynamics. For some purposes, such as visualising each graph created, removing the isomorphisms makes the problem far more tractable. With the isomorphisms removed we had 63 graphs from the original 1042 based on C_4. Finding graph isomorphisms is a notoriously tricky problem in computer science, as of yet we do not know what complexity class the problem is contained in. In general, the problem is NP-complete [9], but this can vary depending on the particular graphs being tested. There are many algorithms to test for graph isomorphism, some better suited to some graphs than others, and we used the VF2 algorithm [10] as it is conveniently implemented using `networkx`. Due to the relatively small size of our graphs, the isomorphism test is very quick. The `networkx` package also allows for automated visualisation of the graphs created, using its specialised graph data structure. There is a choice of visualisation options, which distribute the nodes in different ways, some examples of which can be seen in Figure 5.

4.2 Language recognisers

Quantum walks are known to be universal for quantum computation, because given appropriate graph structures and coin operators (in the discrete time case, the result also holds for the continuous time case)

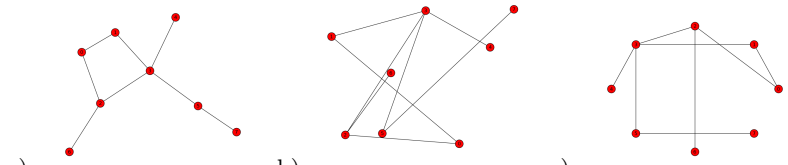


Fig. 5. Automated visualisation of a graph structure using `networkx` and various options for node distribution: a) standard b) random distribution of nodes c) nodes positioned on a circle

they can be used to simulate a universal gate set either using a single walker [8, 20] or multiple walkers [7]. In [3] we showed two ways in which they can be used to solve language recognition problems. These walks required a graph structure that was specified as a function of the length of the input, so having pre built functions to generate the shift operation from the adjacency matrix greatly facilitated the investigation of the walks for inputs of arbitrary lengths. The adjacency matrices of these graphs could be automatically generated easily using simple sequences, once a numbering scheme for the nodes of the graphs was decided on. A schematic diagram of the graph structure of a language accepting graph with a spatially distributed input can be seen in Figure 6. Coin operators designed in [20] to propagate amplitude forwards were used. The input specifies how the walk should be initialised, see Figure 6 for an example with a binary input alphabet. The setup described has a specific accepting node, so the probability of the walker being at this node after a specified number of timesteps determined whether or not the input was accepted by the walk.

The language recognition properties of the walks were verified by testing them on every binary string up to length 16. These strings were easy to generate using the Python `itertools` package. The probability of accepting each string was plotted against the Jaro distance [13, 14] between that string and the string of the appropriate length in the language accepted by the walk. The Jaro distance of strings w_1 and w_2:

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|w_1|} + \frac{m}{|w_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases} \tag{9}$$

with m being the number of matching characters, characters which occur in both strings, in the same order, within a distance which is determined by the length of the strings. The value t, the number of transpositions, is found by dividing the number of characters which differ by sequence order by 2. The three parts to the equation calculate the ratios of the

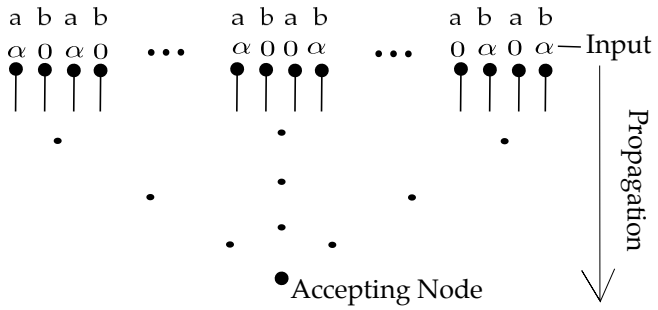


Fig. 6. Basic setup of a graph which enables quantum walks to accept formal languages by processing each input symbol simultaneously. The dotted segments join to graph structures whose form depends on which language the walk accepts.

number of matching characters to the lengths of w_1 and w_2 and then the ratio of non-transpositions to matching characters.

Whilst 65532 strings were tested and verified to have the desired properties, for clarity only the first 200 were plotted. The Jaro distance was selected as it is always between zero and one, with equal strings having distance one, so can easily be plotted against probability. An example of such a plot can be seen in Figure 7 (a), and, as required, the points where both curves go to one indicate indices of strings from the language accepted by the walk.

Some walks developed in [3] are particularly suitable for recognising languages which contain at most one word of each length. This is because the graph structure and coin operator, most notably the Grover operator, can then be specified to check for this specific word. By processing each input symbol simultaneously, this can be done in small number of timesteps. The walks were shown to accept the regular language $\mathcal{L}_{ab} = \{(ab)^m | m \in \mathbb{N}\} = (ab)^*$ and the context-free languages $\mathcal{L}_{eq} = \{a^m b^m | m \in \mathbb{N}\}$. The method can easily be used to accept the context sensitive language $\mathcal{L}_{abc} = \{a^m b^m c^m | m \in \mathbb{N}\}$ if the model is extended to process inputs from alphabets with more than two symbols. The authors have since used variations of the graphs from [3] to prove the language acceptance properties via induction. Also, we have since

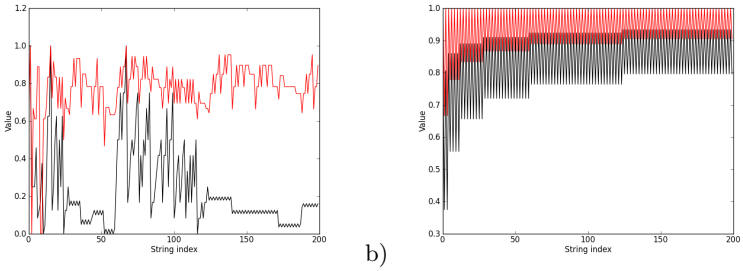


Fig. 7. Results from a walk accepting a) $\mathcal{L}_{eq}$ and b) $\mathcal{L}_{even}$ showing the probability of accepting a given string (black) and the Jaro distance between that string and one from $\mathcal{L}_{eq}$ of an appropriate size

used this type of walk to accept languages with more than one string of each length with bounded error, namely $\mathcal{L}_{even} = \{a, b\}^* a$, though the probability of accepting words not in this language is generally much higher, as can be seen in Figure 7 (b).

Whilst much further work is needed regarding these walks, they offer a novel way to consider quantum computation, in particular by allowing 'quantum inputs,' whereby instead of a specific word, the initial state is a superposition of words. Preliminary investigations suggest that the walks can also be interpreted as performing a type of quantum state discrimination, suggesting links between formal language theory and quantum metrology which could provide novel insights into both fields.

5 Discussion

We have described an algorithm which generates the operators required for the simulation of discrete time quantum walks over arbitrary structures, provided with the adjacency matrix for that structure. This has allowed the authors to investigate a large range of such walks over irregular structures to a very high degree of precision. The algorithm presented would not allow for efficient simulation of walks over regular structures, as in the case of regular graphs direct specification of the shift operation is possible. This bypasses the need for multiplication of large matrices, hence greatly reducing the complexity of the simulation.

Additionally, the algorithms are not optimal if we wish to simulate quantum walks over large structures, where the walker is initially localised. That is because the code takes into account the entire structure, whereas the number of timesteps limits the proportion of the structure

that the walker can have traversed at a given time. To efficiently simulate these walks, an iterative process, specifying only the portion of the structure that the walker can have reached rather than the entire adjacency matrix, is preferable. In light of this, the applications we have presented, using irregular structures and, in one case initially delocalised walkers, are particularly suited to the approach we have taken to the generation of the walks. Whilst clearly numerical simulation of models representing physical systems will always have shortcomings, the fact that our further analysis strongly corroborated the results of the simulations justifies the validity of simulation in the context of research into discrete time quantum walks.

Acknowledgements

KB is funded by the UK Engineering and Physical Sciences Research Council. VK was funded by a UK Royal Society University Research Fellowship.

References

[1] Andris Ambainis. Quantum walk algorithm for element distinctness. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 22–31. IEEE Computer Society, 2004.

[2] Andris Ambainis, Eric Bach, Ashwin Nayak, Ashvin Vishwanath, and John Watrous. One-dimensional quantum walks. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 37–49. ACM, 2001.

[3] Barr, K. and Kendon, V. Formal languages analysed by discrete time quantum walks. *arXiv preprint:1209.5238*, 2012.

[4] Barr, K., Proctor, T., Allen, D., and Kendon, V. Periodicity and perfect state transfer in quantum walks on three families of graphs. *arXiv preprint:1204.5937v3*, 2012.

[5] Milan Bašić, Marko D Petković, and Dragan Stevanović. Perfect state transfer in integral circulant graphs. *Applied Mathematics Letters*, 22(7):1117–1121, 2009.

[6] Sougato Bose. Quantum communication through an unmodulated spin chain. *Physical review letters*, 91(20):207901, 2003.

[7] A.M. Childs, D. Gosset, and Z. Webb. Universal computation by multiparticle quantum walk. *arXiv preprint arXiv:1205.3782*, 2012.

[8] Andrew M Childs. Universal computation by quantum walk. *Physical review letters*, 102(18):180501, 2009.

[9] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[10] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub) graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(10):1367–1372, 2004.

[11] Alex D Gottlieb, Svante Janson, and Petra F Scudo. Convergence of coined quantum walks on $\mathbb{R}_d$. *Infinite Dimensional Analysis, Quantum Probability and Related Topics*, 8(01):129–140, 2005.

[12] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[13] Matthew A Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[14] Matthew A Jaro. Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498, 1995.

[15] Julia Kempe. Discrete quantum walks hit exponentially faster. *Probability theory and related fields*, 133(2):215–235, 2005.

[16] V. Kendon. Where to quantum walk. *arXiv preprint:1107.3795*, 2011.

[17] Viv Kendon. Quantum walks on general graphs. *International Journal of Quantum Information*, 4(05):791–805, 2006.

[18] Viv Kendon and Ben Tregenna. Decoherence can be useful in quantum walks. *Physical Review A*, 67(4):042315, 2003.

[19] Vivien M Kendon and Christino Tamon. Perfect state transfer in quantum walks on graphs. *Journal of Computational and Theoretical Nanoscience*, 8(3):422–433, 2011.

[20] Neil B Lovett, Sally Cooper, Matthew Everitt, Matthew Trevers, and Viv Kendon. Universal quantum computation using the discrete-time quantum walk. *Physical Review A*, 81(4):042330, 2010.

[21] Ashley Montanaro. Quantum walks on directed graphs. *arXiv preprint quant-ph/0504116*, 2005.

[22] Cristopher Moore and Alexander Russell. Quantum walks on the hypercube. In *Randomization and Approximation Techniques in Computer Science*, pages 164–178. Springer, 2002.

[23] Kae Nemoto. Generalized coherent states for su (n) systems. *Journal of Physics A: Mathematical and General*, 33(17):3493, 2000.

[24] Christos H Papadimitriou. On selecting a satisfying truth assignment. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 163–169. IEEE, 1991.

[25] Nitin Saxena, Simone Severini, and Igor E Shparlinski. Parameters of integral circulant graphs and periodic quantum dynamics. *International Journal of Quantum Information*, 5(03):417–430, 2007.

[26] T Schoning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 410–414. IEEE, 1999.

[27] Shenvi, N., JKempe, J., and Whaley. K. Quantum random-walk search algorithm. *Physical Review A*, 67, 2003.

[28] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE, 1994.

[29] T. D. Mackay, S. D. Bartlett, L. T. Stephenson, and B. C. Sanders. Quantum walks in higher dimensions. *J. Phys. A: Math. Gen.*, 35, 2002.

[30] Ben C Travaglione and Gerald J Milburn. Implementing the quantum random walk. *Physical Review A*, 65(3):032310, 2002.

[31] Ben Tregenna, Will Flanagan, Rik Maile, and Viv Kendon. Controlling discrete quantum walks: coins and initial states. *New Journal of Physics*, 5(1):83, 2003.

[32] Salvador Elias Venegas-Andraca. Quantum walks for computer scientists. *Synthesis Lectures on Quantum Computing*, 1(1):1–119, 2008.

[33] John Watrous. Quantum simulations of classical random walks and undirected graph connectivity. In *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pages 180–187. IEEE, 1999.

Decision support for Complex Systems: a Smart Grid case

Jose Evora, Jose Juan Hernandez, and Mario Hernandez

SIANI, University of Las Palmas de Gran Canaria, Las Palmas, Spain,
jose.evora@siani.es, josejuanhernandez@siani.es,
mhernandez@siani.es

Abstract. Transitioning from traditional power grids to Smart Grids involves the use of a different approach based on complex systems to analyse the demand of power grids. This analysis provides information which supports the decision making when developing new policies for Smart Grids. These policies are designed and then tested through simulations since it is not possible to test them directly in a real power grid. Simulation output data can be analysed using a Business Intelligent approach in order to find out KPI (Key Performance Indicators) which support decisions that tune policies. The way in which the results management should be dealt with is through an OLAP (On-Line Analytical Processing) approach which enhances the capability of querying data.

1 Introduction

Climate change, the liberalisation of markets and other new requirements are pushing the energy sector towards a new paradigm known as the smart grid. This paradigm is characterised by the introduction of renewable energy sources (RES) in the power grids, new technologies such as storage mechanisms, massive integration of sensors and decision makers distributed along the grid or the introduction of a communication layer for the management and control of these technologies. The smart grid paradigm is also based on the use of Demand Side Management (DSM), the objectives of which include the minimisation of the peak demand and the system operation and planning improvement [3]. The system complexity is therefore increased and new tools are needed for the analysis and design of smart grids.

Due to the introduction of DSM in the Smart Grid, it is necessary to conceive new policies in order to perform this management which looks after the efficiency of power grids. This efficiency, among other

factors, is related to the efficient use of the energy available at all times, which fluctuates mainly because of RES. However, Smart Grid policies which manage power demand require an arduous analysis of individual consumers and their devices. For this reason, demand requires to be analysed in a disaggregated manner, leading to the usage of a complex system approach to represent the power grid.

Since Smart Grid policies need to be thoroughly tested before their exploitation. The procedure to test these policies is made through simulations, as it is not possible to experiment them in a real power grid. Hence, it is necessary to run complex system simulations where the power grid is represented to test the policies and thus provide feedback about them.

In figure 1, a first iteration of the life cycle of a policy design for the Smart Grid is presented. At this level, and taking into account some high-level considerations about how it should be, a policy is conceived. In order to find out whether the policy will work well or not it is needed to perform a test. To this end, Key Performance Indicators (KPI) [6] must be designed since they are required to support the decision making process which will modify the policy. These KPI are intended to make visible information which is hidden in the data provided by simulations. After this, the simulation must be designed and developed according to the test requisites. Once the simulation has been executed, results will be available. As this simulation can correspond to a big power grid where every single device is represented (complex system approach[1]), the results the simulation provides could be huge. This output must be managed in a way that enables a fast querying system so that KPI calculations can be performed and used for the decision making process. This process will involve some changes in the policy design which shall be tested afterwards when another iteration is initiated.

The complexity of a system from the point of view of Smart Grid simulations is measured in terms of the amount of entities that are in and the relationships among them which produce an emergent behaviour. Therefore, the larger the amount of elements (i.e. entities and relationships) is, the more complex the system is considered. This statement is totally transportable to the results side. The quantity of results in a complex system simulation increases proportionally to the increment of

[1] This representation could be regarded as agent-based. From our point of view, an element is considered an agent whenever it exhibits intelligence [11]. As devices have a mechanistic behaviour, we do not consider them agents. However, simulations that include intelligent elements (i.e. people switching devices in households or units that apply smart grid policies) are considered agent-based.

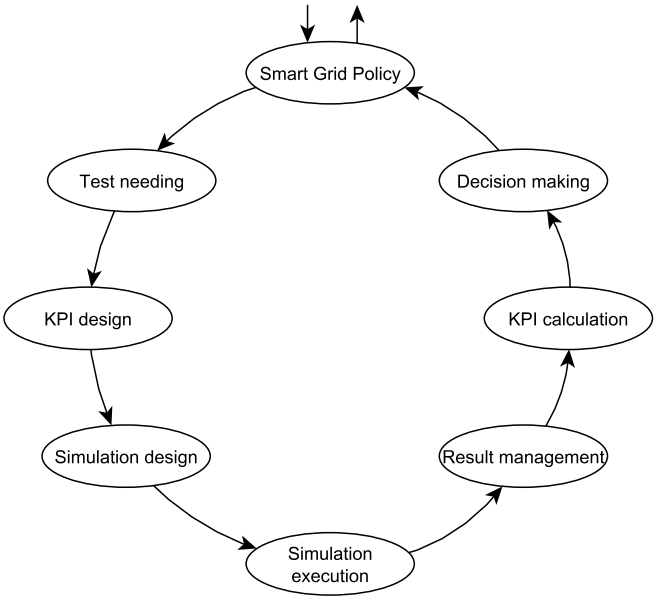


Fig. 1. Life cycle of a Smart Grid policy development

the system complexity. For example, a system that has 10 000 entities with an average of 5 state variables that have to be exported involves that, at each time step, the system will be providing 50 000 results. If the simulation is executed during 2 000 steps, the amount of results provided at the end of the simulation will be about 100 million items.

In another context of Smart Grids, hot topics are all problems related to Energy Data Management, such as the collection and exploitation for business processes of energy consumption data from smart meters installed in power grids [5]. These two examples correspond to problems related to the management of huge amounts of data.

When a Smart Grid simulation is performed, the results management is one of the most important issues as they will feed the design process of the policy. The experience we have had in this field is that it is not possible to perform manually a thorough analysis on large amounts of results. When facing such amounts of data, people usually focus on some details for a certain amount of entities and then conclusions are extrapolated. It has been empirically observed that this analysis may cover a very small percentage of the result set. This implies that many other conclusions could never be found out and extracted from data remaining hidden.

At this point, the use of tools which assist result analysis must be considered in order to deal with this issue. Business intelligence (BI) [4] techniques can play an interesting role in this stage, since it is considered the set of strategies and tools that focus on administration and knowledge creation through data analysis. Among these strategies, some of them encourage the use of technologies such as OLAP (On-Line Analytical Processing) (e.g. Saiku [1]), information visualisation (e.g. Gapmind [8]) and all the data mining corpus which helps to identify and extract hidden or non-evident knowledge (e.g. Weka [9]). These three groups of technologies are especially important for this kind of decision making.

In this paper, the problem of dealing with data exploitation will be further detailed. Then, the OLAP approach to deal with this issue will be exposed. Finally, an example of a Smart Grid case will be presented where results of a simulation are managed following the OLAP approach in order to identify how it helps the decision support when designing Smart Grid policies.

2 Smart Grid simulation issues

Simulations play a crucial role in the design of Smart Grid policies since they are a way to test them before their launch. However, the output provided by the simulations must be managed in a way that allows the policy designers to make decisions. This section explains the main concerns when analysing results obtained in a Smart Grid simulation. When facing a simulation of Smart Grids based on a complex system approach, the results analysis becomes a difficult stage since the amount of entities is huge.

All systems containing a large amount of entities and relations in simulation processes provide a large amount of results. The way in which these data are normally exported is through data files. These data files are usually designed according to the data that will be managed thus avoiding the possibility of querying this data beyond what was decided to export. Therefore, whenever we deem it convenient to extract data, which was not considered to export at the design phase, a new simulation must be configured and executed.

In order to exemplify this issue, a disaggregated model of a power grid system is used. This system only consists of the demand side, which is disaggregated at the device level. It is precisely at this level where we can find a layer consisting of heterogeneous elements, since the characteristics to extract from a radiator are not the same as the ones from a television (TV). If we want to preserve all variables that are not common to every

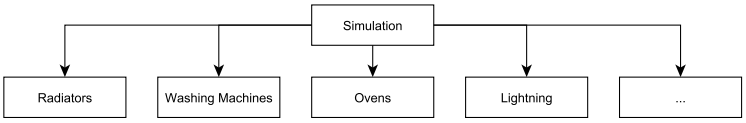


Fig. 2. Structure to export simulation results. Simulation outputs are exported to several files. Each file is related to the data produced by a device kind. For example, Radiators file contains information which regards the energy consumption of these devices

device, it will be necessary to export each device type into a different data sheet (Figure: 2). At this point, once the data exportation process has been defined, we can start thinking about querying it. The list below states some query examples and how they should be dealt with according to this data exportation structure:

- **Querying the consumption of all devices.** This query is very likely to be required. According to our data structure, firstly we calculate the total consumption at each device type. This would involve opening as many files as device types and making the calculations to obtain the total consumption per device type. Secondly, those columns which have the aggregated value at each device type must be moved into a new sheet where the final calculation would be performed obtaining the query result. The more device types there are, the trickier this process becomes.
- **Querying the consumption of all devices in a specific household.** This process would consist in gathering the columns belonging to all the devices contained in the household from the data files. Once they are all together in a new sheet, the query result can be obtained by adding up.
- **Querying the consumption of all devices in a specific district.** The process to obtain this query is really tricky. Firstly, all the devices belonging to a specific district must be listed. Next, all the columns which refer to the devices consumption must be gathered from the device type sheets following this list. Finally, all gathered columns can be moved to a new sheet where the query can be obtained.

Taking these examples into account, it is possible to imagine how tricky the results management of more complicated queries can get. Probably, some of these queries are easier to obtain by redefining the simulation results format and running it again. However, it would also be really tedious, and depending on the simulation kind, the results may

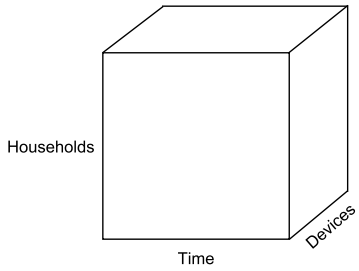


Fig. 3. An OLAP example of a cube for a power grid where every energy consumption is related to a household, device and time

differ from the previous simulation and in the end it would be necessary to start the result analysis from the beginning.

All these difficulties in querying the output of a simulation could involve that many other queries are not made due to the fact that they involve a strong and time consuming effort to perform them. Unfortunately, this usually leads to focus on a small subset of variables of the simulation neglecting much information and wasting too much time in performing simple queries.

The root of the problem behind the result analysis is that such results have a multi-dimensional and a multi-scale (namely temporal and spatial) nature which cannot be managed by using conventional data sheets. The example of the demand disaggregation is multi-dimensional and multi-scale. Multi-dimensional, since every data (for example, a power measure) is related to a specific device, location (household, building, district...) and time. Multi-scale, since the information can be aggregated at different time scales (per hour, per day, per month...) and at different spatial levels (device, household...).

3 OLAP

On-Line Analytical Processing (OLAP) is a solution used in BI, the aim of which is to accelerate querying large amounts of data. OLAP is based on cubes [2](Figure: 3), a multi-dimensional structure where data is stored. These cubes enable the insertion of data, namely facts, which are referred to several dimensions. For example, the measure of power taken from a washing machine can be referred to the device, the household where the device is and the time. Therefore, in this case, there would be three dimensions: devices, households and time.

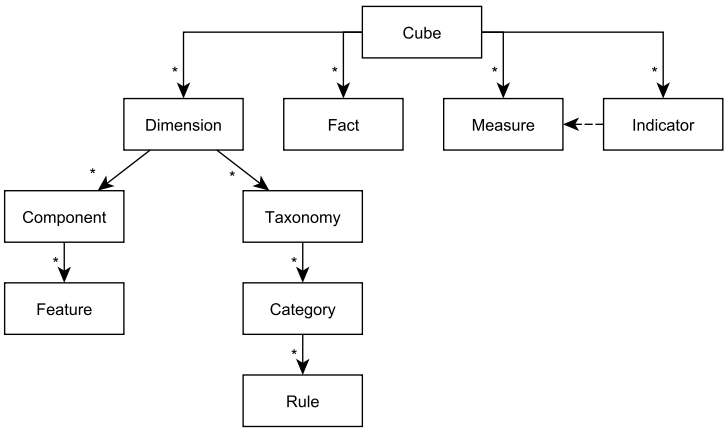


Fig. 4. An OLAP Cube structure. A cube contains dimensions, facts, measures and indicators.

The structure of an OLAP cube which addresses our problem is presented in the figure 4. Every cube consists of dimensions, measures and indicators. The list below describes every cube component.

- **Dimension:** it establishes a way to access the data inside the cube. Every single data is related to some elements such as when and where it happened. For example, a data of power consumption of a household would be related to the dimensions household and time.
  - **Component:** it is an element which is related to a dimension. For example, a dimension which concerns households would be filled by components which are households.
    - **Feature:** it is a property of the component. In case the components are households, a possible feature could be the number of square meters there is in each household.
  - **Taxonomy:** it is a way of categorizing a dimension. There are different ways to categorize the components inside a dimension. Each of these ways is known as taxonomy. In the example of the household dimension, a taxonomy could be the size or the orientation of the facade.
    - **Category:** it is a set of components that satisfy some specific conditions. For instance, possible categories for the size taxonomy could be small, medium or big. Therefore, each of these categories would contain a set of household components the relationship of which is having a similar size.

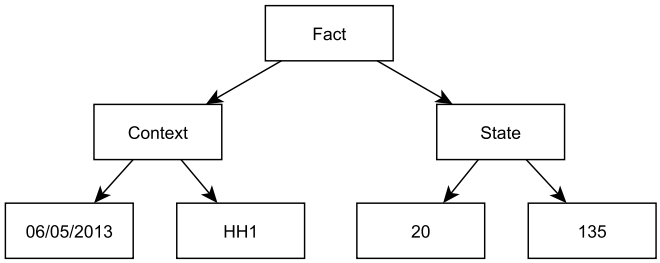


Fig. 5. A fact consists of context and state. The state is a set of measures which are related to components through the context

· **Rule:** it establishes the condition that a component must meet in order to fall into the category that owns the rule. In the case of the small category, a possible rule could be: all the household components the feature of which *number of square meters* is below $80m^2$

- **Measure:** it provides a semantic to the data inserted in the cube, e.g. the power of the household mentioned above is just a number. However, the power measure is what provides the semantic to this number. A measure is usually related to a metric which enables the comparison among measures that are in different cubes. In this case, the metric of the power measure would be Watts.
- **Indicator:** it designates the way in which a measure or a set of measures are aggregated. For example, the power measure could be aggregated using an average (AVG) function. This way of aggregating measures is known as indicator. It is possible to have several indicators for one measure, i.e. the integral operator over the power measure would provide a second indicator over this measure which could be designated as energy indicator.
- **Fact:** it relates the measures of a cube with the dimensions. A fact indicates that a certain combination of values (measures) took place for a specific combination of elements (components). In other words, a fact can be understood as a relation of a state to a context. The state is a set of measures and the context consists of components including time. In figure 5, the state contains 20 (centigrades) and 135 (Watts) as measures. These measures are related to a context which indicates the time and household where those measures were taken.

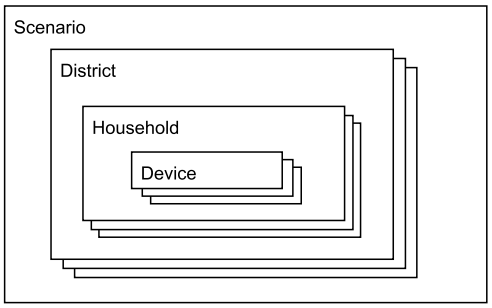


Fig. 6. Scenario composition. Several districts which contain several households and each household contains several devices.

4 An OLAP Smart Grid example

In this section, all concepts exposed previously will be used in a practical case. Assuming that a new Smart Grid policy is to be tuned, several simulations of power grid demand will be performed. To make decisions, these simulations must focus on the power demand and the temperature at the residential sector. Therefore, the scenario for those simulations consists of several districts with households (Figure: 6). Each household contains several devices and calculates the internal temperature.

To this end, several cubes have been designed so as to analyse the data coming from the simulation: first of all, the household cube which contains the facts regarding the temperature and, secondly, one cube per device type (TVs, Radiators and Washing Machines, among others) which contain facts about the devices. Since there are many kinds of devices in a household, in this example we are going to focus on two of them: TVs and radiators.

There are two dimensions in the household (HH) cube: one measure and one indicator (Figure: 7). The Time dimension is common to all cubes and configures a standard way of categorizing the timeline. Household dimension contains the households transformed into components which are described by features. The temperature of the household is the only measure that this cube is going to store and it will be aggregated using an average criteria according to the designation of the indicator.

The household dimension contains a taxonomy which concerns the locations (Figure: 8). This taxonomy is categorized following several levels: country, city and district. For instance, two household components have been included, both of which contain a feature which is their lo-

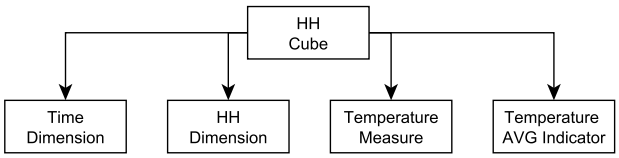


Fig. 7. Household cube. This cube contains two dimensions: time and household. Each fact will relate every temperature measure to a time and a household

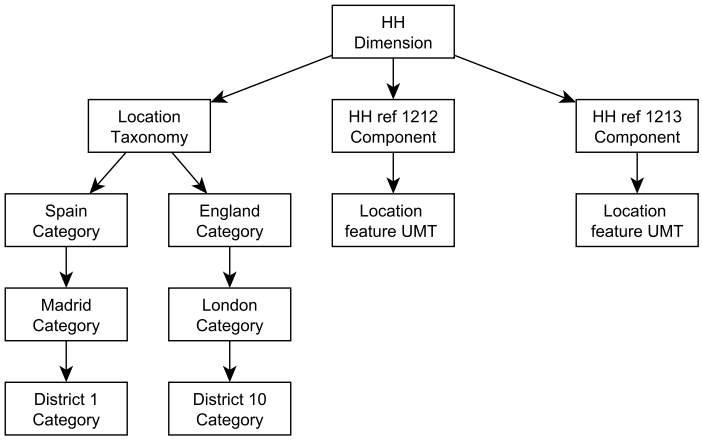


Fig. 8. Household dimension. This dimension contains a taxonomy that classifies the components in districts according to their location feature

cation using UTM coordinates. Therefore, these location features allow the dimension to identify which district each household is located in.

The TV and Radiator cases are exposed in order to demonstrate why devices must be disaggregated into separated cubes. The main reason for this separation is due to the fact that both devices do not share the same features and, therefore, their classification methods are different. This separation enhances the capacity of making queries since it is possible to filter components by features that are only present in a specific kind of device.

The TV cube registers data about power consumption as well as the TV mode (off, standby and on) (Figure: 9). Every set of measures (power and mode) is related to three dimensions: time, household and TV. Time and household dimensions are exactly the same dimensions as the ones detailed above. The TV dimension contains information about the TVs

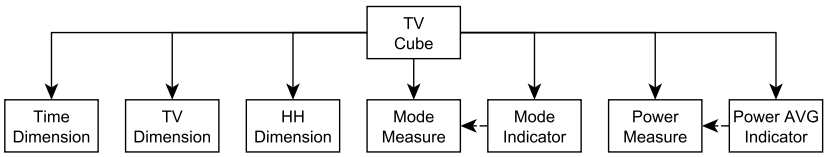


Fig. 9. TV cube. This cube contains three dimensions: time, TV and household. Therefore, each fact will relate a power and mode measures to a time and a TV which is located in a specific household

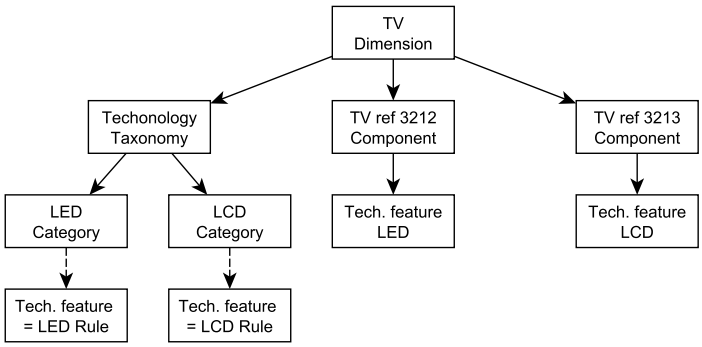


Fig. 10. TV dimension. In this case, the presented dimension has a taxonomy which classifies TV components according to their technology

in a component format. Furthermore, there are two indicators which are responsible for aggregating measures: the mode indicator, which performs a calculation that provides the percentage of TVs that are turned on, and the power indicator, which aggregates the power measures registered using an average formula.

The TV dimension, like the household dimension, focuses on specific features related to TV components (Figure: 10). In this case, the possibility of filtering TVs using a technological criteria is considered relevant. Therefore, two categories have been created so as to separate LED televisions from LCD televisions. This information will allow us to compare the consumption among the different TV technologies. Hence, TV components contain the technology feature which will be used to calculate whether a TV belongs to the LED or LCD category by using the rules that are related to these categories.

The radiator cube stores measures related to both the power consumption and the thermostat level (Figure: 11). These measures are related to three dimensions, as in the case of the TV cube. In this case,

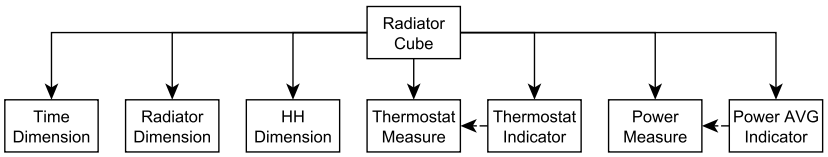


Fig. 11. Radiator cube. This cube contains three dimensions, as in the case of the TV cube. However, the measures are different since, in this case, the thermostat level of the radiator is stored. As it can be observed, devices are heterogeneous. This is the reason why devices have been separated according to a type criteria

apart from time and household dimensions, a new dimension has been designed: radiator dimension. This dimension contains components that represent radiators and their features. In addition, there are two indicators which aggregate the measures. On the one hand, the thermostat indicator aggregates the measures stored using a gradient function which shows big changes in the thermostat level in short periods of time. On the other hand, the power indicator aggregates the power measures using an average formula like in the TV cube.

The radiator dimension focuses on specific features which concern radiator components (Figure: 12). Since radiators are usually considered big consumers, a taxonomy to classify them into two groups has been designed. Indeed, this taxonomy will allow us to find out the amount of radiator components which are in what we consider a small consumer category (under 1kW installed power) or a big consumer category (over 1kW). Two components belong to this dimension and contain the feature installed power which is used to perform the classification in the installed power taxonomy.

4.1 N-Level indicators and Data mining

So far, some mechanisms which allow us to extract information based on the measures have been presented: indicators. These indicators are regarded as first level indicators since they are just based on measures. For this reason, it is possible to define, from first level indicators, a second level of indicators, which are computations carried out based on previous level indicators. This idea can be extended to the concept of N-Level indicators. Introducing this concept, data mining [7] procedures can be used in order to find out patterns.

An example of this is presented in figure 13. In this case, a data miner has been designed in order to identify consumption habits which concern

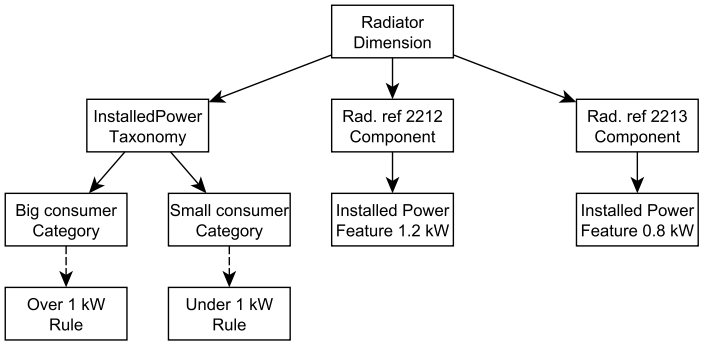


Fig. 12. Radiator dimension. This dimension is intended to store radiator components and classify them according to a criteria based on the installed power. Radiators are thus categorised into small or big consumers according to this feature

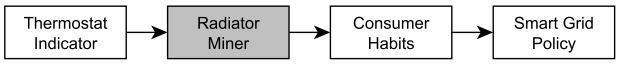


Fig. 13. Radiator data miner which intends to identify consumer habits

radiators. Using the thermostat indicator, which calculates the gradient based on the thermostat level measures, this data miner is able to identify habits throughout time. Therefore, common patterns of radiator usage could be identified and used to feed the Smart Grid policy design.

Moving onto low level details, this miner queries, for each radiator, its thermostat indicator throughout time. Based on this indicator, it uses techniques to extract habit patterns. These habits can be used by the policy in order to exert a more personalised control over the demand which enhances customer quality of service.

The list below presents two other cases where miners can be used in order to improve the design of a smart grid policy:

- Based on the technology feature of TV components, a miner can calculate the average time to amortise a TV based on a low-consumption technology by comparing them to the consumption of other technological kinds. According to these results, a smart grid policy could subsidise the purchase of TVs with a lower consumption. This kind of policy applies to other device kinds such as fridges or washing machines, among others.
- In the household cube, a miner can correlate the temperature and energy consumption of a household with its isolation features, sup-

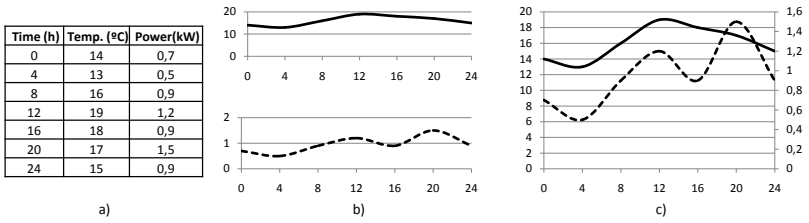


Fig. 14. Visual representations of both temperature (temp) and power

posing the information is available. Based on this correlation, the improvement of household isolation could be proposed.

4.2 Information visualisation

Information visualisation is the use of visual representations of data which step up the human cognition [10]. This is an important stage when analysing data since a proper visualisation may reveal information that could not be possible to extract using other visual representations. Figure 14 presents different visual representations which are discussed in the paragraph below.

Part a in figure 14 presents the data in a table format. From there, it is cognitively difficult to extract temperature or power trends throughout the day, especially when there are many rows. In order to deal with this issue, both series can be represented in separated charts to enable the extraction of trends (part b). These trends can be extracted at each series but relational effects among them are neglected. However, part c represents both individual trends and relational effects among them. It is possible to extrapolate the relation among high temperatures and high consumption at noon.

The example presented in the section is a simple case which intends to provide insights of what is known as information visualisation. In this case, the representation required to show the information correctly is too evident. However, there are cases in which finding out the proper way to represent the information requires a deeper study.

4.3 Decision making

Using this approach to perform the simulation analysis enhances the capability of making decisions. The way in which the data is structured facilitates the interaction. From now on, queries can be as complex as

needed in order to find out interesting conclusions which feed the decision making at the Smart Grid policy design. This structure is to be consumed using information visualisation patterns which could reveal interesting information that cannot be detected simply by analysing numbers.

In the previous example, important information can be extracted to be used in the Smart Grid policy design. The list below summarises some of the most relevant information:

- **Differences among districts:** Using the household dimension, it is possible to find differences among the districts located in the same city or, even, among cities. These differences can be noted in the way in which power is consumed, the devices are used or the temperature in the households. All of this could help in the design of a policy which provides enough flexibility in order to deal with these differences without losing efficiency when applied.
- **TV case:** it is possible to compare the differences in the consumption related to the TV technology. However, the TV dimension can be designed to take into account other aspects such as labelling and size. For instance, the labelling taxonomy could give information about whether it is worth promoting a Smart Grid policy which would subsidise the purchase of new high efficiency TVs.
- **Radiators:** using N-Level indicators allows us to identify consumption patterns that can be used to design more efficient Smart Grid policies which take into account customer usage. In other words, those patterns may be identified in order to build an intelligent control. On the one hand, this control could take note of the customer timetable in order to look after the quality of service. On the other hand, this control could take into account the grid state in order to reduce or increase consumption dynamically.

5 Conclusions and outlook

Transitioning from classical power grids to Smart Grids conveys a huge set of decisions to make. Among others, some of the most important are related to the management of demand. Therefore, an important analysis on the demand in a disaggregated manner is needed. This disaggregation involves the understanding of the power grid as a complex system.

Since consumption management policies in the context of the Smart Grids are not possible to experiment directly on the infrastructure, it is necessary to simulate them in order to make decisions. The complexity of the power grid system when it is disaggregated is so high that the results that the simulations return are huge. At this level, we have found

out that the way in which those results are handled is crucial for making decisions.

Using an OLAP approach has been really helpful as much important information hidden among the data results was discovered. Information visualisation studies have definitely been useful so as to detect relational effects among variables. These effects can be further studied so that modifications on the Smart Grid policy take them into account. Another important strategy to extract information which is hidden or not evident is data mining. Thanks to data mining, consumption patterns can be identified and used to modify the Smart Grid policy in order to take care of the quality of service.

OLAP solutions can be used in many other environments since they are meant to facilitate decision supporting at management positions. We have identified heterogeneous environments such as metrics to program code, product selling and public information systems. In other simulation environments, this method of data analysis can be really interesting, e.g. a set of simulations which run different configurations using the same scenario. Indeed, using an OLAP solution would make it possible to compare all of these configurations with each other.

6 Acknowledgment

This work has been partially supported by Agencia Canaria de Investigación, Innovación y Sociedad de la Información of Canary Islands Autonomic Government thanks to the PhD grant funding assigned to José Évora with reference TESIS20100095 and also the project ˜Framework para la Simulación de la Gestión de Mercado y Técnica de Redes Eléctricas Insulares basado en Agentes Inteligentes. Caso de la Red Eléctrica de Gran Canaria˝, with reference SolSub200801000137.

References

[1] Saiku. http://analytical-labs.com/.
[2] Surajit Chaudhuri and Umeshwar Dayal. An Overview of Data Warehousing and OLAP Technology. *ACM Sigmod Record*, 26(1), 1997.
[3] A. Gabaldon, A. Molina, C. Roldan, J.A. Fuentes, E. Gomez, IJ Ramirez-Rosado, P. Lara, JA Dominguez, E. Garcia-Garrido, and E. Tarancon. Assessment and simulation of demand-side management potential in urban power distribution networks. In *Power Tech Conference Proceedings, 2003 IEEE Bologna*, volume 4. IEEE, 2003.
[4] H. P. Luhn. A business intelligence system. *IBM Journal of Research and Development*, 2(4):314–319, Oct.

[5] Torben Bach Pedersen, Wolfgang Lehner, and Gregor Hackenbroich. Report on the First International Workshop on Energy Management Data. *Sigmod Record*, 42(1), 2013.
[6] Eric T. Peterson. *The big book of Key Performance Indicator*. 2006.
[7] Anand Rajamaran, Jure Keskovec, and Jeffrey D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2011.
[8] Hans Rosling, Ola Rosling, and Anna Rosling. Gapminder. http://www.gapminder.org/.
[9] The university of Waikato. Weka. http://www.cs.waikato.ac.nz/ml/weka/.
[10] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc, 2012.
[11] Michael Wooldridge. *An Introduction to MultiAgent Systems - Second Edition*. John Wiley and Sons, 2009.

Flattening Virtual Simulink Subsystems with Graph Transformation

Péter Fehér[1], Tamás Mészáros[1], Pieter J. Mosterman[2], and László Lengyel[1]

[1] Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
{feher.peter, mesztam, lengyel}@aut.bme.hu
[2] Design Automation Department, MathWorks
Natick, MA, USA
pieter.mosterman@mathworks.com

Abstract. Nowadays embedded systems are often modeled using MATLAB®, Simulink® and Stateflow® to simulate their behavior and facilitate design space exploration. As design progresses, models are increasingly elaborated by gradually adding implementation detail. An important elaboration is the execution order of the elements in a model. This execution order is based on a sorted list of all semantic relevant model elements. Therefore, it is fundamental to remove model elements that only have a syntactic implication such as hierarchical levels with no semantic bearing. The corresponding language construct in Simulink is the virtual subsystem. Thus, to create an implementation or to execute a model, Simulink performs a flattening model transformation that eliminates virtual subsystems. The work presented in this paper raises the level of abstraction of the model transformation by modeling the transformation itself in order to unlock the potential for reuse, platform independence, etc.. To this end, the transformation is implemented by applying graph transformation methods. An analysis of the solution shows the transformation model is proper (e.g., it terminates).

1 Introduction

Advances in electronics miniaturization combined with an understanding of computing are driving an ever-increasing complexity of technical

systems of truly all sorts (consumer electronics, defense, aerospace, automotive industry, etc.). Not only does the increasing capability of electronics enable more extensive logic to be implemented, the robustness and efficiency in communication protocols that it supports has been the driver of ever more network connected systems. The corresponding systems operate at the confluence of cyberspace, the physical world, and human participation. Recently, these systems have been termed Cyber-Physical Systems [1].

Raising the level of abstraction is an important tool to manage the enormous complexity of such Cyber-Physical Systems. To this end, Model-Based Design (e.g. [22], [21], [27]) introduces levels of abstraction in the form of computational models with executable semantics. At the various levels of abstraction, only concerns pertinent to the particular design task are included while implementation aspects are deferred to be addressed in more detailed models. Throughout the design of the embedded system part of a Cyber-Physical System, these models are then elaborated to include increasing implementation detail. The elaboration terminates when a level of detail is arrived at from which an implementation can be automatically generated. The implementation may be either in software by generating C code or in hardware by generating HDL.

The support for abstractions is important in formulating a design problem in the problem space. The design then concentrates on transforming the problem formulation into a solution formulation. In this context, it is of great value that the original problem formulation can be void of solution aspects. This is why domain-specific modeling is becoming increasingly popular to describe complex systems. It is a powerful, but still understandable technique. Its main strength lies in the application of the domain-specific languages. A domain-specific language is a specialized language that can be tailored to a certain problem domain; therefore, it is more efficient, than the general purpose languages that often are tailored to a solution domain (and domain specific in that sense) [18] [17].

Modern model transformation approaches are becoming increasingly valuable in software development because of the ability to capture domain knowledge in a declarative manner. This enables various steps in the software development to be specified separate from one another with apparent advantages such as reuse. In embedded system design, the computational functionality that is ultimately embedded moves through a series of design stages where different software representations are used. For example, before generating the code that is to run on the final target, code may be generated that includes additional monitoring functionality. As another example, the software representation may be designed in

floating point data types before being transformed into fixed point data types.

Today Simulink® [4] is a popular tool for control system design in industry. Therefore, applying model transformations for embedded software design purposes on Simulink models renders the developed technology easily adaptable and adoptable by industry. However, currently it is impossible to define and model declarative model transformations inside the Simulink environment. Therefore, a modeling and model processing framework is applied. The Visual Modeling and Transformation System (VMTS) [10] [7] framework has been prepared to be able to communicate with the Simulink environment. In this manner, with the help of modeled model transformation, problems can be solved at the most appropriate abstraction level. In this case the most appropriate abstraction level means that the required model optimization, modification, or traversing can be expressed in the Simulink domain. Thus Simulink users can use their well-known entities to define the required processing. This is the fundamental premise of Computer Automated Multiparadigm Modeling; to use the most appropriate formalism for representing a problem at the most appropriate level of abstraction [23] [24].

While operating at a given level of abstraction, two further mechanisms are often employed to scale system complexity: partitioning and hierarchy [20]. In the Simulink environment, hierarchy is supported as a purely syntactic construct by *virtual subsystems* and as a construct with semantic implications by *nonvirtual subsystems*. These subsystems are represented as blocks with input and output ports that are used to connect subsystem blocks. Subsystem blocks may contain other subsystem bocks or primitive blocks that represent behavior without being able to be further decomposed.

Before a Simulink model is executed, the engine creates an execution list with an order in which all of the blocks are executed. The execution list is computed from the sorted list, which is also generated by the Simulink engine based on the control and data dependencies that determine how the different blocks can follow each other in an overall execution. To create this list, the semantically superfluous hierarchical layers have to be flattened. So, the virtual subsystems that are only graphical syntax and that have no bearing on execution semantics are flattened before the sorted list is generated [5] [16].

This paper focuses on a novel solution to flattening virtual subsystems in Simulink models. This approach is based on a model transformation created in VMTS. Using model transformation to solve this issue helps raise the abstraction level of the transformation from the frequently used API programming to the level of software modeling. The solution

possesses all the advantageous characteristic of the model transformation, for example, it is reusable, transparent, and platform independent. The different attributes of the transformation are also examined in detail in this paper.

The remainder of the paper is organized as follows. Section 2 introduces related work. Section 3 briefly presents the VMTS and its graph rewriting-based model transformation capabilities. The basics of the communication between Simulink and VMTS are discussed in Section 4. Next, Section 5 introduces the flattening transformation. In Section 6, the properties of the flattening transformation are examined. Next, in Section 7, an example Simulink model is processed with the presented flattener transformation. Finally, concluding remarks presented.

2 Related Work

In [8] a formal description is given about a translation process that can convert a well-defined subset of Simulink block diagram models and Stateflow® [6] state transition diagram models into a standard form of hybrid automata [9]. This transformation is implemented with graph transformation. As a result, different verification tools for hybrid automata can operate on the industry-standard Simulink and Stateflow models.

To specify program transformation such as program optimizers, other work [13] developed a successful method. This method can be uniformly applied to analysis and transformation. The underlying technological solution is based on graph transformation.

Since the design patterns are valuable parts of the different phases of the software development, there is a necessity to specify them on a high level of abstraction instead of capturing this information informally. Other work [28] uses different graph transformation to support this necessity. With the help of this approach the design patterns can be specified on a higher abstraction level.

Another algorithm that works with Simulink models is presented in [14]. This algorithm is introduced for mapping discrete-time Simulink models to Lustre programs. Here, the transformation is not formally modeled as a declarative graph transformation, though.

In other related work [11], a new data model for tool integration is presented. This approach extends existing data models by an abstract graph model. Here, the manipulation is based on model transformation as formal graph transformations.

The work presented in this paper is different in that it does not address any semantic complications. A purely syntactic model transforma-

tion is developed. Moreover, in contrast to the aforementioned exogenous transformations, the transformation developed in the work in this paper is endogenous (i.e., no change of formalism) [19]. Finally, there are no restrictions to the Simulink modeling formalism necessary as the presented work applies to the full set of Simulink blocks.

In [26] and [15] novel approaches are proposed to represent Simulink models as directed, sparse graphs. Each subsystem graph is added to the highest layer graph.

3 VMTS, The Modeling Framework

The Visual Modeling and Transformation System (VMTS) is a general purpose metamodeling environment supporting an arbitrary number of metamodel levels. Models in VMTS are represented as directed, attributed graphs. The edges of the graphs are also attributed. The visualization of models is supported by the VMTS Presentation Framework (VPF) [25]. VPF is a highly customizable presentation layer built on domain-specific plugins, which can be defined in a declarative manner.

VMTS is also a transformation system. It uses a graph rewriting-based model transformation approach or a template-based text generation. Templates are used mainly to produce textual output from model definitions in an efficient way, while graph transformation can describe transformations in a visual and formal way.

In VMTS the Left-Hand Side (LHS) and the Right-Hand Side (RHS) of the transformation are represented together. In this manner, the transformation itself can be more expressive. In order to distinguish the LHS from the RHS in the presentation layer, the VMTS uses different colors. The elements represented with blue color are created by the transformation rule. This means that if the LHS and the RHS would be depicted in two separated graphs, these elements would be only part of the RHS graph. Similarly, the red color indicates that the given element will be deleted by the transformation rule. The yellow color is used when an edge between two elements will be replaced. In this case the type and the attributes of the edge will not change. The gray background means that the element will be modified. With the help of these colors, the transformation process is easily understandable. There is always an option to apply imperative constraints to each element, but this is not depicted separately.

The control flow language of the VMTS [12] contains exactly one start state and one or more end state objects. The applicable rules are defined in the rule containers. The rule containers determine which transformation rule must be applied at the given control flow state. This means that

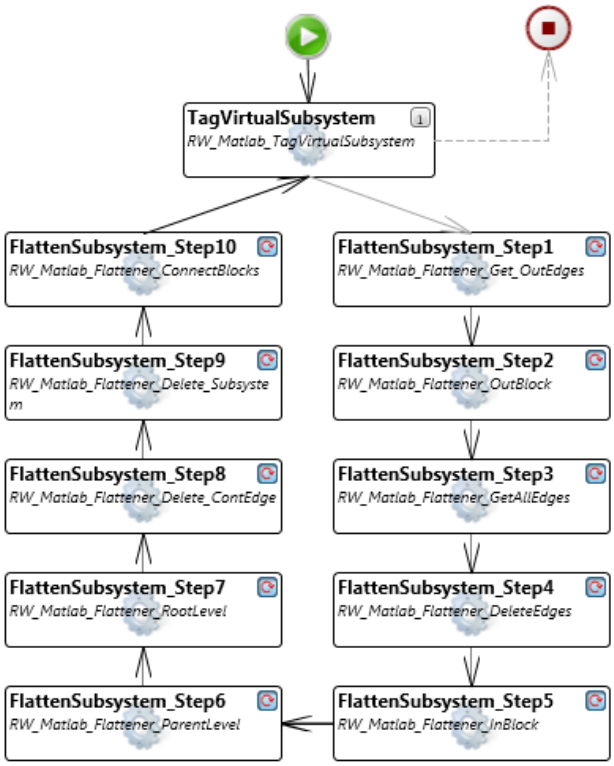


Fig. 1. The control flow of the TransFlattener transformation

exactly one rule belongs to each rule container. The application number of the rule can also be defined here. By default, the VMTS tries to find just one match for the LHS of the transformation rule. However, if the *IsExhaustive* attribute of the rule container is set to true, then the rule will be applied repeatedly as long as its LHS pattern can be found in the model. Figure 1 depicts an example control flow model; actually this is the control flow of the flattening transformation, which will be presented in detail in Section 5.

The edges are used to determine the sequence of the rule containers. The control flow follows an edge in order of the result of the rule application. In VMTS, the edge to be followed in case of successful rule application is depicted with a solid gray flow edge and in case of a failed rule application with a dashed gray flow edge. Solid black flow edges represent the edges that can be followed in both cases.

4 Communication between Simulink® and VMTS

Since the model is created in Simulink, which is part of the MATLAB® [3] environment and the transformation is created in another system (VMTS) there is a need to establish a communication method between the two systems.

To be able to represent Simulink models in VMTS, the metamodel of the Simulink languages, which are organized in various different libraries (also called *blocksets*), is required. Since in Simulink there is no hard boundary between the different languages, that is, a given block can be connected to almost everything else, a common Simulink metamodel was created. This metamodel contains all the elements of the Simulink library. The generation of this metamodel consists of the following two steps.

First, a *core* metamodel was created that contains the *Block* element, which is the common ancestor of all the nodes in Simulink models, and a descendant *Subsystem* node, which expresses the common ancestor of Simulink Subsystems. This metamodel also contains the *Signal* edge and a *Containment* edge to reflect containment hierarchy between nodes.

Then, by programmatically traversing the base Simulink library, this metamodel has been extended with the other nodes found in the different specialized libraries. For each Simulink element, exactly one node was generated. This resulted in several hundred new metamodel elements.

In addition to the metamodel, the VMTS must be prepared to read and write Simulink models. Thus, a new kind of data exchange layer was generated for communicating with MATLAB. To modify Simulink models, the P/Invoke technology [2] has been chosen. This has the advantage that the MATLAB interpreter can be called directly through DLL calls, instead of manipulating the textual model (mdl extension) files. This way the VMTS is independent of file format changes, and the changes performed on the VMTS model can be made visible, live during the transformation execution, on the Simulink diagrams as well. Furthermore, the values that are only available during simulation time of a Simulink model can be accessed also.

5 The Flattening Transformation

This section introduces and discusses the details of the transformation TransFlattener. The transformation is created in VMTS. As it was mentioned before, its goal is to process Simulink models and flatten its virtual subsystems.

For a better understanding the final control flow is presented first, which is shown in Fig. 1. This model defines how the transformation rules follow each other.

At first, the transformation checks if there is a virtual *Subsystem* block in the model. So the RW_Matlab_TagVirtualSubsystem transformation rule attempts to match a simple *Subsystem* block that has the *IsVirtualSubsystem* attribute set to true. If the transformation engine does not find a match for this rule, then there is no virtual *Subsystem* in the model, thus the transformation terminates. Otherwise, the transformation steps into the loop, which processes this *Subsystem*.

In Simulink, a block cannot be directly connected to a block on a different hierarchical level. When a block is moved out from a *Subsystem*, it loses all its edges automatically. This means that the transformation must take into account the connections between the blocks, and must take care of creating the necessary edges. Moreover, when a *Subsystem* is flattened, the *Inport* and *Outport* blocks of the *Subsystem* are deleted. So the transformation also must ensure that the blocks connected to the appropriate ports will be connected to each other after the processing.

The *Inport* and *Outport* blocks represent the input and output ports of the *Subsystem*. Each port of the *Subsystem* is associated with an appropriate block in the *Subsystem*. For example if a *Subsystem* has two input ports and three output ports, then there are two *Inport* and three *Outport* blocks in the *Subsystem*, and these blocks have a reference number to the port they belong to. This behavior is presented in Fig. 8.

In the aforementioned loop, which is responsible for processing the virtual *Subsystem*, the first applied rule is the RW_Matlab_Flattener_Get_OutEdges transformation rule. It is depicted in Fig. 2. In order to handle the deletion of the ports this rule matches the outgoing edges of a *Subsystem*, and deletes them if a match is found. In the meantime, the identifier of the block connected to the *Outport* port of the *Subsystem* (the *toNode* in Fig. 2) and the port number that the edge is connected to are stored in the appropriate *Outport* block (the *endNode* in Fig. 2).

After processing outgoing edges from all *Subsystem* type nodes, the transformation moves to the next transformation rule: RW_Matlab_Flattener_OutBlock (Fig. 3). This rule matches blocks that have outgoing edges to an *Outport* block and if a match is found deletes the edge connecting them. It also copies the information stored in the *Outport* block into the other one (the *simpleNode* in Fig. 3) and extends it with the port number where the edge starts. This way this block knows all the information about the edges the transformation must create after the block is moved out of the *Subsystem*. This information consists of the followings:

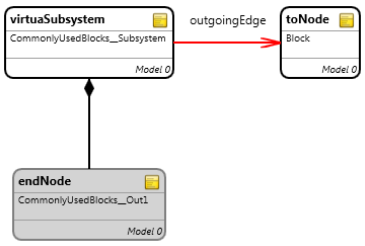


Fig. 2. The transformation rule RW_Matlab_Flattener_Get_OutEdges

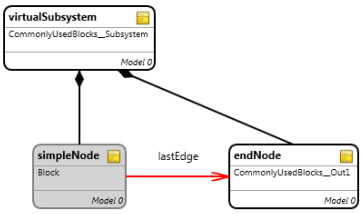


Fig. 3. The transformation rule RW_Matlab_Flattener_OutBlock

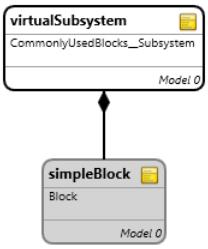


Fig. 4. The transformation rule RW_Matlab_Flattener_GetAllEdges

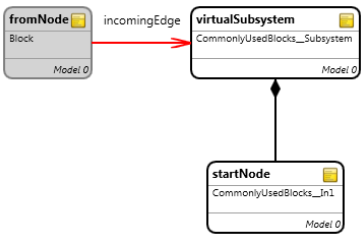


Fig. 5. The transformation rule RW_Matlab_Flattener_InBlock

- The port number the edge starts,
- The identifier the edge is connected to,
- The port number the edge ends,
- The necessary attributes of the edge.

In the previous two steps the transformation focused on the blocks connected to the *Out* ports and *Outport* blocks of the *Subsystem*. The next rule embodies the same functionality with every block in the *Subsystem*. The RW_Matlab_Flattener_GetAllEdges rule is depicted in Fig. 4. It does not differentiate based on the type of the block, which means that it matches *Inport* blocks as well as the blocks processed previously (i.e., the ones connected to the *Outport* blocks). The reason for this is the possibility that a block may have multiple outgoing edges and the transformation needs information about every edge. The rule is matched for every element in the *Subsystem* exactly once and stores all information about their outgoing edges.

After the first three rules of the loop, every block of the *Subsystem* knows the relevant information of their outgoing edges. Moreover, the blocks connected to the *Outport* blocks know which blocks are connected to the appropriate *Out* port as well. This means that the transformation can delete the edges inside the *Subsystem*, so if a block is moved out of it, then no dangling edges remains. The rule RW_Matlab_Flattener_DeleteEdges simply does that (i.e., deletes the edges between the blocks of the *Subsystem*).

In the next step the transformation deals with the blocks connected to *In* ports. The transformation rule RW_Matlab_Flattener_InBlock is shown in Fig. 5. It matches the incoming edges of *Subsystem* nodes and if a match is found deletes them. As it has been mentioned, the RW_Matlab_Flattener_GetAllEdges rule matches *Inport* blocks as well, so these elements know the necessary information about their edges. In this manner the current rule can copy this information to the blocks connected to the appropriate *In* ports. The information is also extended with the port number of the edge directed from the block to the *Subsystem*. Upon completion of this rule the blocks connecting to the *Subsystem* know the characteristic of the edges the transformation must create after the *Subsystem* is deleted.

At this point the blocks related to the *Subsystem* store the necessary information about their outgoing edges. The transformation also deleted the edges connecting to the *Subsystem* and the ones between its blocks. This means that the blocks are ready to be moved to a higher hierarchical level. The higher hierarchical layer means the *Subsystem* containing the *Subsystem* that is actually processed, if there is any. In case there is not, then the root layer is the higher layer. The transformation rule RW_Matlab_Flattener_ParentLevel looks for the parent *Subsystem*. If it finds a match, then the blocks contained by the processed *Subsystem*, except the *In* and *Out* ones, are replaced into the parent. The rule is depicted in Fig. 6. If the processed *Subsystem* still contains elements besides the *Inport* and *Outport* blocks after this rule, then it means, that the *Subsystem* is not contained by anything, it is at the root level. So the RW_Matlab_Flattener_RootLevel rule must delete only the *Containment* typed edges between the *Subsystem* and its blocks. In this manner the blocks are not contained by anything, they are moved to the root level as well.

Next, the transformation can safely delete the *Subsystem*. However, it cannot leave any dangling edges, so if it was contained by another *Subsystem*, then the transformation must delete the *Containment* edge after which the *Subsystem* is deleted.

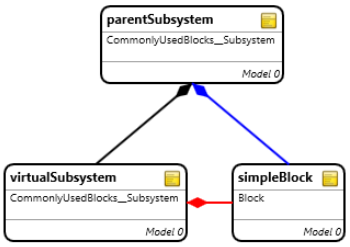


Fig. 6. The transformation rule RW_Matlab_Flattener_ParentLevel

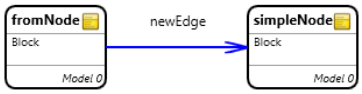


Fig. 7. The transformation rule RW_Matlab_Flattener_ConnectBlocks

Finally, the transformation must recreate the edges between the moved blocks. This is based on the information stored in the blocks. The transformation rule RW_Matlab_Flattener_ConnectBlocks creates the necessary edges (Fig. 7). The block storing the information is the source block; the identifier provides the target block. The appropriate ports are also saved along with other relevant information.

With this rule the loop ends and the transformation returns to the RW_Matlab_Flattener_TagVirtualSubsystem transformation rule, which checks for another virtual *Subsystem*. If there is one, then the engine steps into the loop again, otherwise it terminates.

The next section discusses the formal analysis of this transformation.

6 Analysis of the Transformation

The previous section has presented the transformation TransFlattener and its rules. Before its usage, it is advisable to perform the analysis of the transformation definition, which is the subject of this section. First, the functionality of the transformation is examined and then its further attributes, such as correctness and termination, are verified.

The coverage of Proposition 1 and Proposition 2 is depicted in Fig. 8. This picture may also help illustrate the relation of the different blocks.

Definition 6.1. *The **inner elements** of a Subsystem are all elements, except the Inport and Outport typed blocks, contained by the Subsystem.*

Proposition 6.1. *After the transformation* TransFlattener*, the inner elements of the processed Subsystem are connected if and only if they were connected before the transformation.*

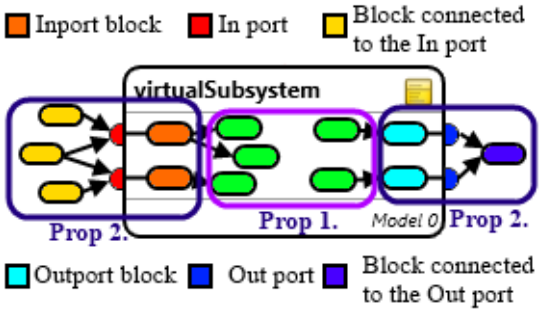


Fig. 8. The structure of a subsystem

Proof. Three transformation rules (RW_Matlab_Flattener_OutBlock, RW_Matlab_Flattener_GetAllEdges and RW_Matlab_Flattener_ConnectBlocks are responsible for the connection between the blocks. First, the RW_Matlab_Flattener_OutBlock deletes the edges pointing to the different *Outport* blocks, which represent the *Out* ports of the *Subsystem*. The rule also stores the identifiers of the blocks that had an edge pointing from the same *Out* port. (The identifiers of these blocks are already stored in the *Out* block because of the RW_Matlab_Flattener_Get_OutEdges rule.) Next, the RW_Matlab_Flattener_GetAllEdges stores for each block the identifier of those blocks that the given block has an edge pointing to. It also stores the details of the edges, that is, from which port point to which port. The rule examines every block exactly once. Next, the other rules of the transformation place the elements onto a higher level in the hierarchy. Since it is not possible in Simulink that elements in different hierarchy level are directly connected, it is ensured that after the replacing there is no edge pointing to or from the moved block. The transformation also deletes all edges between the inner elements to avoid the dangling edges. Finally, when the elements of the *Subsystem* are already placed onto a higher hierarchical level, the RW_Matlab_Flattener_ConnectBlocks creates the edges based on the stored information for each block. Since this is the only rule that generates edges and does this based on the stored information in the different blocks, there will be no new edges between the inner elements of the *Subsystem*. Note that the information about the edges is stored for each and every inner element that has outgoing edges, thus these edges will be regenerated.

In this manner the inner elements of the Subsystem are connected if and only if they were connected before the transformation execution. □

In order to not change the functionality of a Simulink model it is required that a block is reachable from another block if and only if it was reachable before the transformation.

Since the transformation cannot modify the functionality of a processed model, the following are required by the transformation:

- Let o denote the set of blocks outside of the *Subsystem* that have an outgoing edge to a given *In* port. Let i denote the set of inner elements that are connected to the *Inport* block related to the given *In* port. With this notation, after the transformation all elements in o must have an edge pointing to all elements in i.
- Regarding the *Out* ports we can state the same requirements. Let o denote the set of blocks outside of the *Subsystem* that have an incoming edge from a given *Out* port; and i denote the set of inner elements that are connected to the *Outport* block related to the given *Out* port. In this case, after the transformation all elements in i must have an outgoing edge to all elements in o.

Proposition 6.2. *After the transformation* TransFlattener *leaves the ports of the processed Subsystem the functionality of the Simulink model does not change.*

Proof. The RW_Matlab_Flattener_Get_OutEdges and the RW_Matlab_Flattener_OutBlock transformation rules are responsible for not changing the functionality of the model when the *Out* ports are removed. First, the RW_Matlab_Flattener_Get_OutEdges rule stores information in each *Out* block. This information contains the identifier of the blocks to which any edges point from the appropriate *Out* port. Moreover, the port attributes of these edges are also stored. The rule also deletes these edges. Next, the RW_Matlab_Flattener_OutBlock attempts to match those edges that are pointing from one of the inner elements of the *Subsystem* to one of the *Outport* blocks. For every match, the rule deletes the edge and copies the information stored in the *Outport* block to the matched element. It also extends this information with the number of the port from where the matched edge starts. With the help of these two transformation rules it is ensured that the blocks that have outgoing edges to one of the *Outport* blocks possess the proper information. The proper information means the identifiers of the block, where the edges from the appropriate *Out* port point to. The edges based on this information will be recreated by the RW_Matlab_Flattener_ConnectBlocks rule after the elements are placed to a higher hierarchical layer.

To ensure that the functionality of the model remains the same in case of the *In* ports, two rules are needed as well. These rules are

the RW_Matlab_Flattener_GetAllEdges and the RW_Matlab_Flattener_InBlock. As it was described in the proof of the Proposition 1, the RW_Matlab_Flattener_GetAllEdges rule stores for each and every block, so for the *Inport* blocks as well, to which port of which blocks it has outgoing edges. Next, the RW_Matlab_Flattener_InBlock matches each edge that points from a block outside of the *Subsystem* to one of its *In* ports. The rule deletes this edge, since after deleting the *Subsystem* there cannot be any dangling edge. The rule also copies the stored information from the appropriate *Inport* blocks to the matched block and extends it with the number of the port the matched edge starts from. With the help of these two transformation rules it is ensured that the blocks, which have outgoing edges to one of the *In* ports, have the information, where the appropriate *Inport* block has outgoing edges. The edges based on this information will be created by the RW_Matlab_Flattener_ConnectBlocks transformation rule after the elements are placed into a higher hierarchical layer.

It can thus be stated that after eliminating the ports of the Subsystem the functionality of the Simulink model does not change. □

Consequence of Proposition 2: The number of the edges in the Simulink model changes as follows: $\sum(-s_i - f_i + (s_i * f_i)) + \sum(-s_o - l_o + (s_o * l_o))$, where s_i stands for the number of edges going into the i^{th} *In* port of the *Subsystem*, f_i means the number of edges going out from the i^{th} *Inport* block, s_o stands for the number of edges going out from the o^{th} *Out* port of the *Subsystem* and l_o means the number of edges going into the o^{th} *Outport* block.

Proposition 6.3. *Proposition 1 and Proposition 2 together state that all inner elements of the processed Subsystem are connected if and only if they were connected before the iteration and the functionality of the model does not change. Considering the two propositions it can be stated, that the functionality of the model does not change after the Subsystem is flattened.*

Proof. The proof follows from the proofs of Proposition 1 and Proposition 2. □

Proposition 6.4. *Each iteration of the transformation* TransFlattener *moves the inner elements of the processed Subsystem exactly one level higher.*

Proof. The RW_Matlab_Flattener_ParentLevel and the RW_Matlab_Flattener_RootLevel transformation rules are responsible for this behavior. The RW_Matlab_Flattener_ParentLevel attempts to find a match, where the processed *Subsystem* is a child element

of another *Subsystem*. If such a match is found then the rule deletes the *Containment* edge between the processed *Subsystem* and its inner elements and also creates a *Containment* edge between the parent *Subsystem* and the aforementioned inner elements. If there is no match for this rule, then it can be stated that the processed *Subsystem* is at the root level. So the RW_Matlab_Flattener_RootLevel rule simply deletes the *Containment* edges of the processed *Subsystem*. This means that its inner elements are moved to the root level. Neither of these two transformation rules match for the *Inport* and *Outport* blocks of the *Subsystem*. □

Definition 6.2. *The* execution hierarchical layers *are layers created not for purely graphical purpose, but have a bearing on execution semantics.*

This means that since the *virtual Subsystems* are created to help organize and understand the models and do not have any additional role, the *execution hierarchical layers* are created only by *nonvirtual Subsystems*.

Proposition 6.5. *The transformation* TransFlattener *does not move elements between execution hierarchical layers.*

Proof. The RW_Matlab_TagVirtualSubsystem is the first rule of the transformation. The rule attempts to match virtual *Subsystems*. At the beginning of each iteration, a virtual *Subsystem* is tagged as the *Subsystem* under processing. As a consequence, only the elements of a virtual *Subsystem* are modified during the actual iteration. As a consequence none of the objects of a non-virtual *Subsystem* are moved to a higher hierarchical level. □

Proposition 6.6. *The transformation* TransFlattener *flattens all virtual Subsystems.*

Proof. The RW_Matlab_TagVirtualSubsystem transformation rule is the first rule of the iteration: This rule expresses the loop condition. At each time the rule is evaluated, it attempts to match a virtual *Subsystem*. If such a match is found then the found *Subsystem* will be processed. The other rules of the iteration move the inner elements of this *Subsystem* onto a higher hierarchical layer and also delete this *Subsystem*. None of the rules creates any type of *Subsystems*. This means that every iteration decreases the number of virtual *Subsystems* by one in the model. The iteration continuous till the RW_Matlab_TagVirtualSubsystem cannot find a successful match anymore. This means there are no remaining virtual *Subsystems* in the model. □

Proposition 6.7. *The transformation* TransFlattener *always terminates.*

Proof. To examine the termination of the transformation the following must be checked:

- The control flow cannot go into an infinite loop,
- The transformation rules, which are applied exhaustively, terminate in finite steps.

The control flow terminates if the RW_Matlab_TagVirtualSubsystem rule cannot find a match. This happens if and only if there are no virtual *Subsystems* in the model. In case there is no virtual *Subsystem* in the model at the starting point, the transformation terminates without stepping into the iteration. Otherwise a match is found and the transformation steps into the loop. Proposition 6 states that the transformation flattens all virtual *Subsystem*. Since in Simulink the *Subsystems* cannot be recursively defined (i.e. the containment loops are forbidden) there are finitely many *Subsystems* in the model. This means that after finite number of iteration there will be no remaining virtual *Subsystems* in the model, so the RW_Matlab_TagVirtualSubsystem rule cannot find a successful match anymore, thus the control flow terminates.

In the transformation each rule is applied exhaustively except the RW_Matlab_TagVirtualSubsystem rule. The exhaustively applied rules must be checked whether they terminate in finite steps:

- RW_Matlab_Flattener_Get_OutEdges rule: This rule matches an edge between the *Out* port of a *Subsystem* and other blocks. If a match is found the rule deletes this edge. This means that every application of the rule reduces the number of edges between the *Out* ports and other blocks, thus after finites steps it cannot be applied anymore.
- RW_Matlab_Flattener_OutBlock rule: This rule works by the same principle. The only difference between the two rules is that this one attempts to match an edge between an inner element and the *Outport* block of the *Subsystem*. The deletion of the matched edge, without creating any new, ensures its termination.
- RW_Matlab_Flattener_GetAllEdges rule: This rule marks the matched block at each application. After several steps there is no remaining unmarked inner element in the *Subsystem*. Since there is a condition in the LHS of the rule that the block cannot be marked before the rule is applied, the system cannot find a match.
- RW_Matlab_Flattener_DeleteEdges rule: The rule simply matches an edge between the blocks of the *Subsystem* and if a match

is found deletes it. After finite steps there are no edges left between the blocks, and the rule cannot be applied.
- RW_Matlab_Flattener_InBlock rule: This rule is equivalent to the RW_Matlab_Flattener_Get_OutEdges, but this one operates between the *In* ports and the *Subsystem*. It deletes the matched edges as well, therefore cannot be applied after a certain number of steps.
- RW_Matlab_Flattener_ParentLevel rule: The rule matches and deletes the containment edge between an inner element and the actual *Subsystem*. It is irrelevant that it creates a new edge between the parent *Subsystem* and the inner elements, since the LHS of the rule checks containment edges between the actual *Subsystem* and other blocks. This ensures that the rule cannot be applied indefinitely.
- RW_Matlab_Flattener_RootLevel rule: The rule simply deletes the containment edge between the actual *Subsystem* and its inner element. The deletion of the matched item without creating any new items ensures its termination.
- RW_Matlab_Flattener_Delete_ContEdge rule: The principle is the same. The rule attempts to match a containment edge between the actual *Subsystem* and a parent one. If it succeeds, then it deletes this edge.
- RW_Matlab_Flattener_Delete_Subsystem rule: The rule deletes the actual *Subsystem*. The LHS checks that the *Subsystem* must be marked. This marking occurs in the RW_Matlab_TagVirtual-Subsystem rule, which is applied exactly once before every iteration. In this manner there is only one element that can be a match for this rule.
- RW_Matlab_Flattener_ConnectBlocks rule: The rule matches blocks that store information about edges to create. After the rule creates such an edge, it removes the information from the block. Since the RW_Matlab_Flattener_GetAllEdges rule was applied for a finite number of times and this is the only transformation rule that stores information about the edges to create, the RW_-Matlab_Flattener_ConnectBlocks rule will be applied for a finite number of times as well.

Since both the control flow and the transformation rule terminate after finite number of steps, the transformation terminates as well. □

7 Experimental Results

The presented model transformation was applied on different Simulink models. One of these source models is depicted in Fig. 9. The root level is

shown in Fig. 9(a). This model contains a virtual Subsystem with one In port and two Out ports. The inner structure of this Subsystem is shown in Fig. 9(b). It can be seen, that each Inport/Outport block relates to exactly one In/Out port. This hierarchical layer also contains a virtual Subsytem, which is presented in Fig. 9(c).

After the transformation TransFlattener terminates, the structure of the model changes, as it is shown in Fig. 10. All inner elements were moved to the next level, and eventually, since the model did not contain any non-virtual Subsystem, to the root level. The Subsystem blocks were deleted with their In- and Outport blocks. The connection within the blocks were correctly maintained. The example also demonstrates that the transformation handles well when an Out port of a virtual Subsystem is connected to multiple blocks. In this manner, the transformation did not change the functionality of the source model.

The transformation was examined on simpler and more complex models as well, and the results always were found to be correct by inspection.

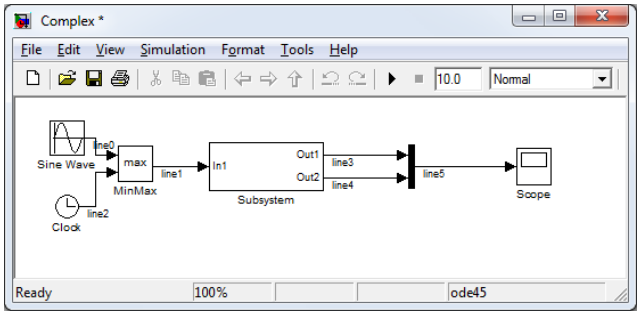
8 Conclusions and Future Works

As a popular tool for the design of embedded control systems, industry relies on Simulink models to support a level of abstraction much above the embedded implementation in, for example, C code. Design relies heavily on model elaboration to increasingly add detail to the design models. Such elaboration is a form of model transformation that currently is implemented in software as part of the Simulink code base or as external functionality based on the Simulink model API.

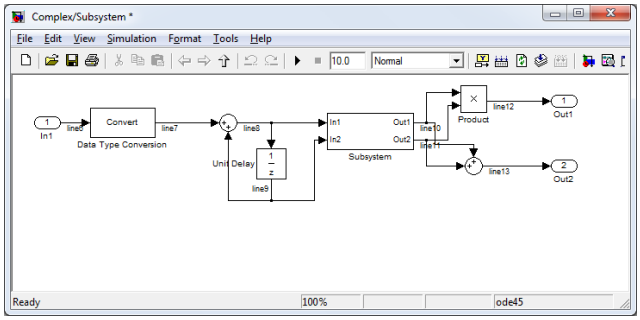
Part of the elaboration is removing hierarchical structures that have only a syntactic effect such as flattening of syntactic hierarchical layers. In this paper a detailed model transformation-based solution has been presented for flattening virtual subsystems in Simulink models. The approach enables taking advantage of benefits of modeled model transformation such as reusability and platform independence. In this manner, the abstraction level of the model transformation problem can be raised. Besides the transformation details, its formal analysis has been also discussed.

The transformation was implemented in the Visual Modeling and Transformation System. Therefore the modeling framework and its communication with the Simulink environment were briefly introduced as well.

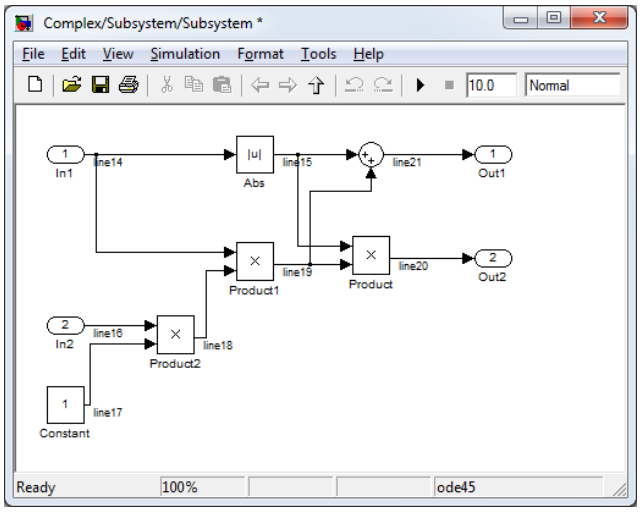
Future work intends to study whether with the help of this transformation, the sorted list and the execution list can also be implemented



(a) The root level of the Simulink® model



(b) The model contained by the fist Subsystem



(c) The containment of the nested Subsystem

Fig. 9. The example Simulink® model

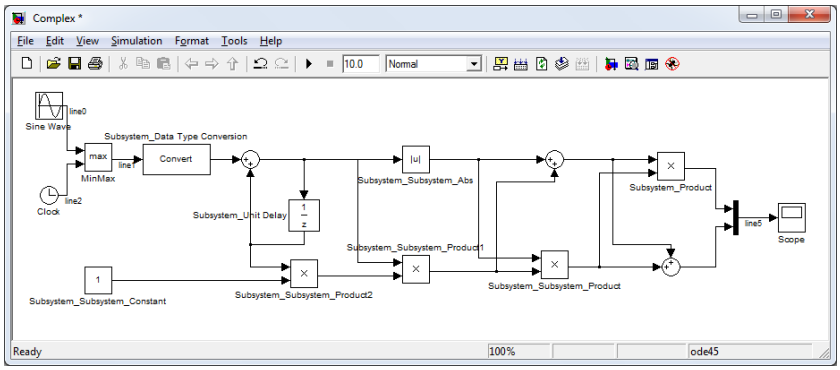


Fig. 10. The model after the TransFlattener transformation

via model transformation. In this manner the abstraction level could be raised even further and more benefits unlocked.

9 Acknowledgments

This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TMOP-4.2.2.C-11/1/KONV-2012-0013).

References

[1] PCAST document. http://varma.ece.cmu.edu/InfoCPS/Readings.html, 2007.
[2] An introduction to P/Invoke and marshaling on the Microsoft .NET compact framework. http://msdn.microsoft.com/en-us/library/aa446536.aspx, 2012.
[3] MATLAB® user's guide. http://www.mathworks.com/help/matlab/index.html, Sep 2012.
[4] Simulink®. http://www.mathworks.com/simulink/, 2012.
[5] Simulink® users manual. http://www.mathworks.com/help/simulink/index.html, 2012.
[6] Stateflow® user's guide. www.imec.be/elela/HK19/background/stateflow_users_guide.pdf, 2012.
[7] VMTS website. http://vmts.aut.bme.hu/, 2012.
[8] Aditya Agrawal, Gyula Simon, and Gabor Karsai. Semantic translation of simulink/stateflow models to hybrid automata using graph transformations. *Electron. Notes Theor. Comput. Sci.*, 109:43–56, dec 2004.

[9] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *THEORETICAL COMPUTER SCIENCE*, 138:3–34, 1995.

[10] L. Angyal, M. Asztalos, L. Lengyel, T. Levendovszky, I. Madari, G. Mezei, T. Mszros, L. Siroki, and T. Vajk. Towards a fast, efficient and customizable domain-specific modeling framework. In *Software Engineering*. ACTA Press, 2009.

[11] Raul Camposano Ansgar Bredenfeld. Tool integration and construction using generated graph-based design representations. In *Design Automation, 1995. DAC '95. 32nd Conference on*, pages 94–99, 1995.

[12] Mrk Asztalos and Istvn Madari. An improved model transformation language. In *Automation and Applied Computer Science Workshop 2009*, 2009.

[13] Uwe Amann. How to uniformly specify program analysis and transformation with graph rewrite systems. In *Compiler Construction (CC)*, pages 121–135. Springer, 1996.

[14] Paul Caspi, Adrian Curic, Aude Maignan, Christos Sofronis, and Stavros Tripakis. Translating discrete-time simulink to lustre. In *In: Third International ACM Conference on Embedded Software, Lecture Notes in Computer Science*, pages 84–99. Springer, 2003.

[15] Florian Deissenboeck, Benjamin Hummel, Elmar Jürgens, Bernhard Schätz, Stefan Wagner, Jean-François Girard, and Stefan Teuchert. Clone detection in automotive model-based development. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08, pages 603–612, New York, NY, USA, 2008. ACM.

[16] P. Fehr, P. J. Mosterman, T. Mszros, and L. Lengyel. Processing simulink models with graph rewriting-based model transformation. Model Driven Engineering Languages and Systems (MODELS 12) - Tutorials, 2012.

[17] M. Fowler. *Domain Specific Languages*. The Addison-Wesley Signature Series. Addison-Wesley, 2010.

[18] Steven Kelly and Juha-Pekka Tolvanen. *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley, 2008.

[19] Tom Mens, Krzysztof Czarnecki, and Pieter Van Gorp. A taxonomy of model transformation. In *Proc. Dagstuhl Seminar on "Language Engineering for Model-Driven Software Development". Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl*. Electronic, 2005.

[20] P. J. Mosterman, J. Sztipanovits, and S. Engell. Computer-automated multiparadigm modeling in control systems technology. *Control Systems Technology, IEEE Transactions on*, 12(2):223–234, march 2004.

[21] Pieter J. Mosterman, Jason Ghidella, and Jon Friedman. Model-based design for system integration. In *The Second CDEN International Conference on Design Education, Innovation, and Practice*, pages TB3–1–TB3–10, 2005.

[22] Pieter J. Mosterman, Sameer Prabhu, and Tom Erkkinen. An industrial embedded control system design process. In *Proceedings of The Inaugural CDEN Design Conference (CDEN'04)*, pages 02B6–1–02B6–11, 2004.

[23] Pieter J. Mosterman and Hans Vangheluwe. Computer automated multiparadigm modeling in control system design. *IEEE Transactions on Control System Technology*, 12:65–70, 2000.

[24] Pieter J. Mosterman and Hans Vangheluwe. An introduction to computer automated multi-paradigm modeling, 2004.

[25] Tams Mszros, Gergely Mezei, and Tihamr Levendovszky. A flexible, declarative presentation framework for domain-specific modeling. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '08, pages 309–312, New York, NY, USA, 2008. ACM.

[26] Hoan Anh Nguyen, Tung Thanh Nguyen, Nam H. Pham, Jafar M. Al-Kofahi, and Tien N. Nguyen. Accurate and efficient structural characteristic feature extraction for clone detection. In *Proceedings of the 12th International Conference on Fundamental Approaches to Software Engineering: Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009*, FASE '09, pages 440–455, Berlin, Heidelberg, 2009. Springer-Verlag.

[27] G. Nicolescu and P.J. Mosterman. *Model-Based Design for Embedded Systems*. Computational Analysis, Synthesis, and Design of Dynamic Models Series. CRC Press, 2010.

[28] Ansgar Radermacher. Support for design patterns through graph transformation tools. In *In Applications of Graph Transformation with Industrial Relevance (Intl. Workshop AGTIVE99, Proceedings), LNCS 1779*, pages 111–126. Springer, 1998.

Bursting a Bubble: Abstract Banking Demographics to Understand Tipping Points?

Philip Garnett

Department of Anthropology, Durham University, Dawson Building, South Road, Durham, DH1 3LE. UK philip.garnett@durham.ac.uk

Abstract. It has become popular to describe the behaviour of certain systems as "undergoing a tipping point". This is normally used as a description of a system that has rapidly changed from an apparently stable state to a new state with little or no warning. A wide range of complex systems can display tipping point behaviour, from climate systems to populations of people. Here we present preliminary work of using the British banking sector from 1559 to 2012 as a case study for the modelling of complex systems that show tipping point behaviour. We present a description of an abstract population model of the banking system. Once implemented we hope to use this model to test our assumptions about how systems undergo tipping points. In the future it might also help determine what the key drivers of the population trends seen in the British banking sector are, and what the possible implications were of past legislative interventions.

1 Introduction

The term tipping point is often used to describe a system that has undergone a rapid change in state. It is often applied when aspects of the change in state were not predictable before hand, such as the potential for very occurrence of the state change, the exact timing, or the nature of the final system state [3, 9]. We are interested in developing models and simulations of systems that could potentially experience a tipping point, avoiding the ever present danger of programming the tipping point into the model (and therefore the resulting simulation if implemented). Current best practise of building a simulation of this kind dictates that the system is simplified into a number of key interacting components. This is a difficult step as in a truly complex system it is difficult to identify the key causal components for an observed behaviour. Our informed

opinions of what is or is not important might be very accurate, but may also be focused on the wrong part of the system altogether. Once identified these components are then given simplified versions of their *real* behaviour, the simulation is then started from a suitable initial condition and the resultant system behaviour observed. However, the very nature of the modelling process biases the modeller towards selecting components of the larger system being modelled that have at least the potential to produce desired behaviour. This is somewhat inevitable as a modeller is not going to included bits of a system that he/she believes to be irrelevant. The process therefore has an aspect of self-selection. In our case as researchers we look for systems that display what we consider to be tipping point behaviour. We then, in the background of already having decided that the system has displayed what we define as a "tipping point", make assumptions about the behaviour of the components of that system. When the model is then built care is taken not to build the solution into the model, but it is impossible to operate in a completely unbiased way.

What the current best practice does give us is some indication of how good our assumptions about a system are. If we have identified likely key system components that when given reasonable behaviours do go on to produce the system behaviour that we are interested in seeing, then we have at the very least learnt something about our understanding of that system. This understanding can be compared with that of other researchers, and also considered in the wider background of the field of study. In short, we can make some assessment of how good we think that model is and how good we think its underlying assumptions are. This knowledge of the model can then be taken into account when the model is used. Models of tipping points have an additional problem that often we have only one example of a system going through a tipping point. Therefore we don't have a good understanding of how that system truly behaves; we do not know what constitutes its normal state. Therefore when making assumptions about key components and key behaviours we do so with the additional assumption that it can go through a tipping point. Therefore it would be helpful to the modelling process that the modeller had no knowledge of what we define as a tipping point.

In this paper we approach this from a slightly different angle. Rather than commissioning a modeller (free of the burden of 'tipping point' knowledge) to build a model of a system with only information that does not give away that it is capable of undergoing a tipping point, and then observing what they determine as important components and behaviours. We have chosen a model system where the important determinates of the global behaviour are not clear, but what is clear is that the

system has the potentail to undergo a tipping point. We have collected detailed population data on the banks present in British banking system from 1559 to 2012. The data includes useful demographic data, including the size of population of banks for each year, the number of bank failures, the number of bank creations, and also the number of mergers. Not only do we have the number of mergers but we are also able to track the flow of banks into one another by acquisition. The data suggests that in terms of population the banking sector undergoes a tipping point during this time period, but importantly it's a tipping point that we have so far being unable to satisfactorily discover the basis of. We therefore believe that we know a lot about the system, its components and behaviours, but we do not know which behaviours are producing the observed tipping point. Is this an opportunity to develop a simulation of a complex system and learn something about the modelling process itself, but also about the extent of our understanding of the banking system. The historical nature of the data also allows us to make predictions about how the population of banks could have responded to changes to the regulation of the sector, allowing the testing of alternative regulatory interventions. Once fully implemented we might also be able to predicted the effect of present day interventions on the future banking populaiton.

We have made extensive use of the CoSMoS process [1] to guide the development of a number of different simulations of biological and behavioural systems [6–8]. This paper applies the CoSMoS process to building an abstract model of the population demographics of the British banking sector from from 1600 to 2012. We focus on the first step of this process where assumptions are made about what parts of the British banking sector need to be included in the model.

2 Background

2.1 CoSMoS Process: The modelling lifecycle

The CoSMoS process used for this work is described in full by Andrews et al. [1], and used is the same as used in our earlier work [6–8]. Summarised in figure 1, the version of the process used here contains the following components (summarised from [1], and the description of the process is taken from [7]):

Research Context: the overall scientific Research Context. This includes the motivation for doing the research, the questions to be addressed, and the requirements for success.

Domain Model: conceptual "top-down" model of the real world system to be simulated. The Domain Model is developed in conjunction

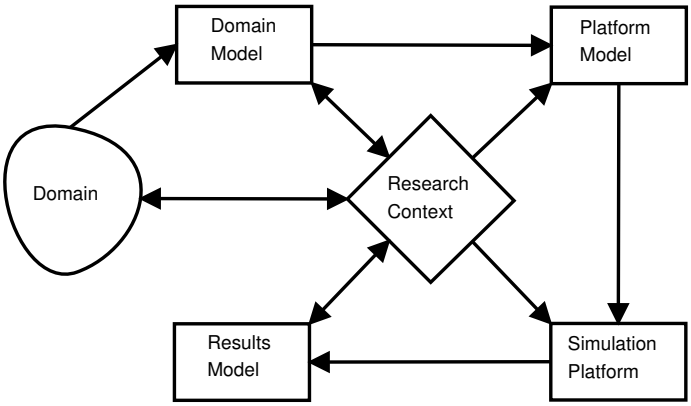


Fig. 1. The components of the CoSMoS process [1, fig.2.1]. Arrows indicate the main information flows during the development of the different components. There is no prescribed route through the process, in so far as going back a step at any point in the process is allowed and often useful.

with the domain experts, with its scope determined by the Research Context. The model may explicitly include various emergent properties of the system.

Platform Model: a "bottom up" model of how the real world system is to be cast into a simulation. This includes: the system boundary, what parts of the the Domain Model are being simulated; simplifying assumptions or abstractions; assumptions made due to lack of information from the domain experts; removal of emergent properties (properties that should be consequences of the simulation, rather than explicitly implemented in it).

Simulation Platform: the executable implementation. The development of the simulator from the Platform Model is a standard software engineering process.

Results Model: a "top down" conceptual model of the simulated world. This model is compared with the Domain Model in order to test various hypotheses. This part of the process is on-going research.

This work focuses on determined what parts of the Domain, the British banking sector, are included in the domain model. This is a particularly important part of the process for this model as we do not have a clear understanding of what causes the observed behaviour, but we believe we have a reasonable understanding of how the system is operating (on one level at least, Sect. 3).

3 The Research Context

The British banking sector is one of the oldest and most developed in the world. Starting in the 1550s it reached its maximal population of 1100 banks in 1810 before steadily declining to its current level of about 100 banks. Figure 2 shows data for the number of banks through time. The black line is the actual number, the long-dashed line is a exponential fit indicating a 2.7% increase year on year in the number of banks. The dashed-dotted line is a super exponential increase where an additional scaling factor is introduced to improve the fit to the real data. The short-dashed line as an exponential decrease of 1.5% year on year. Broadly the real data matches a exponential increase until the maximal population, after which the population decreases exponentially. The super exponential fit is interesting because these are often seen in situations where positive feedback is operating – perhaps indicating that creation of banks promoted the creation of more banks, discussed in Sect. 3.1. The last 200 years of the banking sector may have been dominated by a change in legislation and is discussed in Sect. 3.2.

3.1 The Banking Sector Pre 1810

The period of exponentially increasing numbers of banks could have a number of possible causes. During this time banks operated as partnerships; each bank had a number of partners and they brought with them the money that could be invested. During this period banks were limited to a maximum of six partners. This builds into the system a mechanism for the growth in the number of banks via an increasing population of available partners. Its is reasonable to assume that during this period of sustained economic growth there was a requirement for more banks, not only that there would also have been a supply of potential partners that wanted to invest money to make money. As the number of new potential partners increased so did the number of banks. The fact that the real growth of banks more closely matches a super exponential curve is interesting as it suggests that there was an element of positive feedback in the system. One possible explanation for this feedback is that people believed that there was money to be made in banking and therefore looked for opportunities to set up banks. They saw others making money by setting up banks and therefore copied that behaviour. Positive feedbacks (or herd behaviour) in financial systems can turn out to be unstable [2, 5, 11], creating a bubble that is destined to burst at some point in the future, and could be one possible cause for the eventual decline in the number banks.

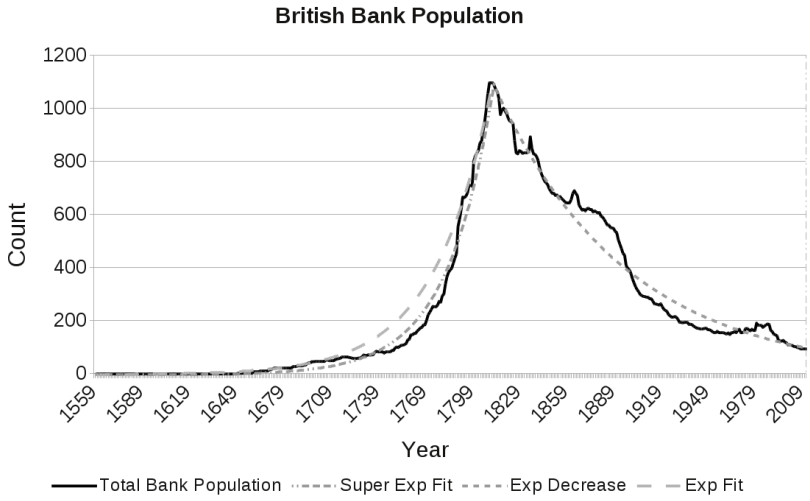


Fig. 2. The changing number of banks through time from 1559 to 2012. The black line is the actual number, the long-dashed line is a exponential fit indicating a 2.7% increase year on year in the number of banks. The dashed-dotted line is a super exponential increase where an additional scaling factor is introduced to improve the fit to the real data. The short-dashed line as an exponential decrease of 1.5% year on year.

3.2 The Banking Sector Post 1810

Post 1810 the number of banks starts to decline exponentially year on year. The actual date of the decline is interesting as it is close to a number of potentially significant historical events. The Napoleonic Wars ran from 1803-1815 and are likely to have been a source of economic disruption; there was also a significant financial crisis in 1825 [10]. Of particular interest is the Amalgamation Movement that describes a long peroid of banking history. In 1825 the rules governing banks changed and banks were able to expand via amalgamation, allowing the formaiton of joint stock banks [12]. This could explain a lot of the changes in the population of banks post 1810 as we have evidence that banks were rapidly increasing in size via amalgamation during this period, essentially by copying the behaviour of other banks in the population [4, 12]. The Amalgamation Movement was brought to a halt in 1925 is it was feared that the population of banks would fall too low [12].

There is evidence for a number of underlying processes at work in the British banking sector that might account for many of the trends

seen in the changing population of banks. A long period of growth in the British economy coupled with a restrictive policy limiting the size of banks suggests a mechanism for the expansion of the bank population. Add to that an interest in forming banks to make money increasing the population beyond what is strictly required and we are starting to identify possible components and behaviours to explain the growth in the population of banks. At some point (perhaps due to internal pressure or external drivers) new rules are introduced into the banking system that allow banks to increase in size via merging together. Once the rules are changed there follows a long period of bank amalgamation that results in an exponential decrease in the population of banks and dominating its development for the next 200 years. This poses a number of questions. Can we develop an abstract model of banking demographics and then implement a simulation based on these these basic rules? What behaviours will we observe using this simulation and how do they compare to the real banking population data? In order to see the observed tipping point in the banking system will we have to drive the system externally, or would the model require much more fine-grained detail about the economy and individual banks to reproduce the population trends through time?

4 The Domain Model: the banking sector

We intend to develop an agent-based model of an abstract banking sector based on the components and behaviours identified from studying the British banking sector. From the domain we can determine a number of key components to the model, we can also develop simplified behaviours for the components. These components and behaviours will be mapped to "agents" in the model and ultimately implemented in the simulation. We intend to evolve the components and their possible behaviours starting from a very simple initial set. This is to see how the introduction of new components affects the results from the simulations, allowing us to incrementally develop our understanding of the abstracted banking system.

Figure 3 shows the domain class diagram. A description of the agents (and their starting behaviours) and other components of the initial system follows:

Partners: Prior to 1825 Partners are central to the banking sector as they are the source of funds in the system. Agents representing Partners will have the following behaviours. New Partners enter into a pool of Partners; from here they can either join an existing Bank (initiated by the Bank) or form a new Bank. Existing Partners in a Bank

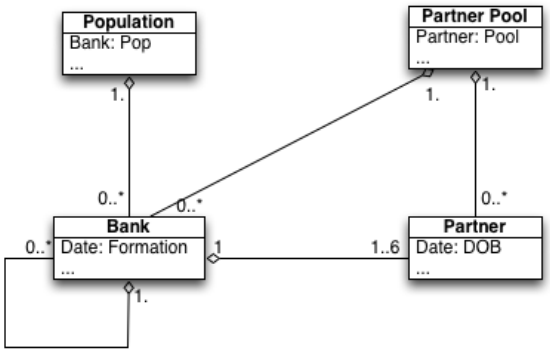


Fig. 3. Domain class diagram showing the relationship between the Bank and Partner classes. The Population starts with 0 Banks, supply of money causes the creation of Partners which are held in the Partner Pool. Banks are created by Partners and held in the Population. A Partner can be in only one Bank, each Bank can have a maximum of 6 Partners. A Bank can contain 0 or more acquired Banks.

can decide to leave their current Bank and form a new one; they can either do this individually or as a group. Partners can exit the population. Figures 4 and 5 represent the behaviour of the Partners.

Banks: Banks are container for Partners. A Banks can contain between 1 and 6 Partners. A Bank with less than 6 Partners can attempt to attract new Partners. Banks can also acquire other Banks to increase in size, they are therefore a contaion for Banks. A number of possible behaviours could be tested here. Including the effect of keeping the 6 Partner limit, this limit would block any merger of Banks that resulted in more than 6 Partners. Alternatively the Partners of the acquired banks could either exit the population, or return to the pool of Partners. Banks can only be formed by Partners and do not arise spontaneously. Banks can fail and exit the population, or they if a Bank's only Partner exits the population the Bank leaves too. Figure 6 shows the activity diagram for the basic bank behaviours. The size of a Bank could be determined by the number of Partners, the number of acuired Banks or a combination of both.

GDP: The simulation needs a method for introducing new Partners into the system. The population of Partners is increased in line with growth in estimated United Kingdom (UK) Gross Domestic Product (GDP).

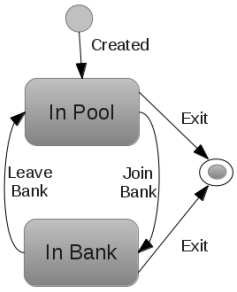


Fig. 4. State Diagram for the the Partners. Partners can occupy two different states, in the general pool of Partners, or in a Bank. The Partners can leave the simulation from both the In Bank state and the In Pool state.

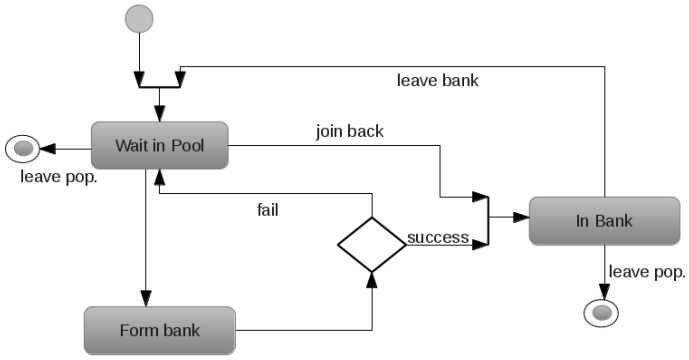


Fig. 5. Activity diagram for the Partners. The behaviours of the Partners drive the initial model.

There are few key differences between the domain model for the initial simulator implimentation and the domain. Firstly, Partners remain key to the formation of Banks throughout the simulation. In the real system post 1825, banks are not only owned by partners. We are making this alteration to the system to see if the changes that bring about the Amalgamation Movement are responsible for the declining population of banks. There is an approximate 10 year cap between the start of the decline of banks and the 1825 change in regulation, suggesting that the relationship between the obversed decline and the regulation is not clear. If the change in regulation had not been made what would the banking

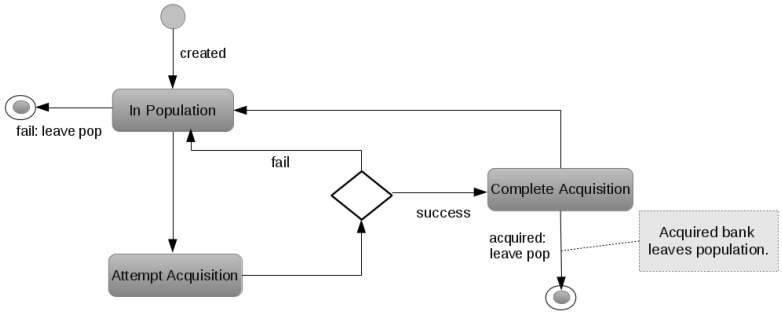


Fig. 6. Activity diagram for the Banks. Banks are formed by one or more Partner. Banks will also be able to acquire, or be acquired, by other banks. Acuired banks leave the general population but remain 'in side' the acquiring Bank.

sector look like based on our simple rules? Initially we will not introduce any external drivers to the system, such as economic disruption or regulatory change. This domain model represents the base model for the system to which additional processes will be added.

4.1 Drivers of Change

As it stands our domain model describes two distinct behaviours that look to dominate the simulated banking system at two different periods of time. During the early part of the development of the banking sector, from 1600s to 1810 the system is driven by the behaviour of partners. The supply of partners into the system should drive the formation of new banks, the growth phase. In the second time period, merger and acquisition dominate the system, the decline in population and the rise of super banks (banks that have acquired large numbers of other banks). These two systems are similar in some respects, they are both about generating sucessful banks that are as large as possible. In the case of the partner model, banks (created by partners) attempt to grow by attracting new partners from the pool. In the second phase, banks grow more by acquiring other banks. Modelling the swtich between these two methods of growth of banks present a challenge, and is largerly dependant on our assumptions about the evolution of the real banking system.

One possibility is to allow both behaviours to operate in parallel. Under this system it would be interesting to see under what conditions the behaviour of the simulation changes and how sensitive it is. When

there is an abundence of available Partners (stable money supply, condition of economic growth), is possible to produce a simulation where growth by attracting Partners from the pool dominates? However, if the economic conditions became more difficult (unstable money supply, poor or no economic growth), would a swtich to merger and acquisition behaviour take place? How stable would this switching be, and would the banks need to have the possibility of copying "successful" members of the population (follow the herd) for the behaviour to diffuse throughout the population? This would suggest that regulatory change might have legitimised a behaviour that was already starting to occur in the population of banks. Alternatively, to achieve the two distinct phases of population change might require external influence, indicating that the second phase was in response to regulation. What would be the effect of initially only allowing merger if the limit on six partners is respected?

5 Discussion

Developing a model and simulation of a tipping point is challenging as it is hard to take an unbiased view of a system as the system is often of interest because it seems to display tipping point behaviour. Here we approach a system, the population demography of the British banking sector, that appears to display tipping point behaviour but where the exact cause is unclear. We have identified the key aspects of the banking system for inclusion in the model that could be responsible for the general population trends seen in the population. The initial model is a highly simplified version of the real system. This is deliberate and is an attempt to produce a null model for the banking sector, with much of the complexity removed, that is still capable of matching the general trends. This model could be used to test the effect of internal drivers on the population of banks, but also if external drivers are required to match the general population trends.

We also hope to gain insight into modelling tipping points. The banking system appears to undergo a tipping point in its population in around 1810. Using the simulation we can test if when we model the components of the system as we assume them to work if the modelled system can undergo tipping points. We are also able to test the effect of introduced legislation on the behaviour of the modelled banking system. The two phase nature of the system could potentially help us understand how population of organisations might flip bewtween two possible but distinct behaviours. Under what conditions this flips occur and how often they occur. It is also possible that tipping points could be caused by behaviours no longer happening, forcing a system into one behaviour.

Acknowledgements

We gratefully acknowledge the financial support from the Leverhulme Trust who funds the Tipping Point project based in the Institute of Hazard, Risk and Resilience at Durham University. We would also like to thank the developers of the CoSMoS process. We also thank Dr Simon Mollan and Prof. Ranald Michie for useful discussions about how banks work.

References

[1] Paul S Andrews, Fiona A C Polack, Adam T Sampson, Susan Stepney, and Jon Timmis. The CoSMoS Process version 0.1: A process for the modelling and simulation of complex systems. Technical report, University of York, 2010.

[2] Sushil Bikhchandani and Sunil Sharma. Herd behavior in financial markets. *IMF Staff papers*, pages 279–310, 2000.

[3] William A Brock. *Tipping Points , Abrupt Opinion Changes , and Punctuated Policy Change by*. PhD thesis, University of Wisconsin, 2004.

[4] Paul J DiMaggio and Walter W Powell. The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields. *American Sociological Review*, 48(2):147–160, April 1983.

[5] Robert P Flood and Robert J Hodrick. Asset Price Volatility, Bubbles, and Process Switching. *The Journal of Finance*, 41(4):pp. 831–842, 1986.

[6] Philip Garnett. Going Around Again: Modelling Standing Ovations with a Flexible Agent-based Simulation Framework. In Paul Read Mark Stepney Susan Andrews, editor, *Complex Systems Simulation and Modelling Workshop*, pages 27–46, Orleans, France, 2012. Luniver Press.

[7] Philip Garnett, Susan Stepney, Francesca Day, and Ottoline Leyser. Using the CoSMoS Process to Enhance an Executable Model of Auxin Transport Canalisation. In S Stepney, P Welch, P. S. Andrews, and A. T Sampson, editors, *CoSMoS 2010*, pages 9–32, 2010.

[8] Philip Garnett, Susan Stepney, and Ottoline Leyser. Towards an Executable Model of Auxin Transport Canalisation. In Susan Stepney, Fiona Polack, and Peter Welch, editors, *Cosmos 2008 Complex Systems Modelling and Simulation*, pages 63–91. Luniver Press, 2008.

[9] Malcolm Gladwell. *The Tipping Point: How Little Things Can Make a Big Difference*. Little Brown, 2000.

[10] L Neal. The financial crisis of 1825 and the restructuring of the British financial system. *Review-Federal Reserve Bank of Saint Louis*, 80:53–76, 1998.

[11] Didier Sornette. *Why stock markets crash: critical events in complex financial systems*. Princeton University Press, 2004.

[12] J Sykes. *The Amalgamation Movement in English Banking, 1825-1924*. P.S. King and Son Ltd, London, 1926.

Understanding tissue morphology: model repurposing using the CoSMoS process

Ye Li, Adam Sampson, James Bown, and Yusuf Deeni

University of Abertay Dundee, DD1 1HG, UK,
ats@offog.org

Abstract. Drawing inspiration from the CoSMoS project structure, we consider the assumptions made during the design and implementation of a software simulation of physical interactions during the formation of vascular structures from endothelial cells. We show how the abstract physical model and its software implementation can be adapted for a different problem – the growth of cancerous tissue under varying physical conditions. By identifying the changes that must be made to adapt the model to its new context, along with the gaps in our knowledge of the domain that must be filled by wet-lab experimentation when recalibrating the model, we maintain confidence in the repurposed model and achieve a satisfactory degree of model reuse.

1 Introduction

The CoSMoS process [1, 11] describes a principled approach to scientific modelling and simulation: it provides a structure for managing and documenting the iterative development of a simulation, and gives scientists and simulation developers tools to reason – with an appropriate balance of confidence and scepticism – about how their simulation's results relate to the domain under study. CoSMoS is an agile approach based upon a pattern language: a user may organise their project entirely following the CoSMoS principles, or they may integrate some of the CoSMoS patterns as appropriate into an existing project.

Reusability of software components is a key concern of software engineering. Reusable components can – ideally – avoid the difficulty and expense of developing and validating substantial amounts of new software. But software developed for one purpose may not be reusable for a different purpose without substantial modification. In particular, a simulation component developed for one *in silico* experiment may rely

on assumptions (parameter values, model simplifications, etc.) that are only valid within the context of that experiment. Adapting such a component for reuse in a different context requires careful consideration of the assumptions made during its design.

In this paper, we use the general structure of projects outlined by CoSMoS to organise our thinking around how a model of physical interactions among cells can be adapted from one context – the formation of vascular structures from endothelial cells – to a different context – the effects of cancer treatment drugs on the growth of spheroid structures of cells. Neither of these projects was initially developed using a CoSMoS approach. To apply CoSMoS techniques, we must first effectively reverse-engineer our work to date, and attempt to organise the information we have about the systems under study and our models and simulations of them broadly in terms of the CoSMoS project structure. We expect that this step in itself will prove valuable.

Our objective is specifically to reuse the software components that implement this physical model within the simulation, as these required considerable development effort and are critical to the overall performance of the simulation. As the modes of physical interaction among cells are broadly similar between the two models, this seems *intuitively* to be an appropriate approach – but identifying and revalidating our assumptions will help to build our confidence in our simulation's results, and enable the future reuse of the physical model in other contexts.

In addition, we are now at the point in the development of our cancer model where it is clear that some wet-lab experimentation is required in order to recalibrate the parameters of the physical model. Since wet-lab experimentation is expensive and time-consuming (in this case, time-series imaging requires several days' commitment from a skilled researcher), we need to be confident that we are obtaining the correct data from the experiments to support our simulation development.

2 The Original Model: Vascular Formation

2.1 Research Context

The purpose of this simulation is to reproduce the results of an *in vitro* experiment from the literature, demonstrating the formation of capillary structures from endothelial cells [10].

The experiment explores how the physical interactions among the cells, and their low-level physical properties, affect the larger-scale structural patterns in the resulting capillary network. The effects of varying concentrations of growth factors – which have a direct effect on the low-level physical properties of the cells – are of particular interest.

2.2 Domain

Microvessels are formed within the body by the aggregation of endothelial cells, which themselves are formed by differentiation from stem cells. This formation process has three stages [4]:

- cell migration and early network formation;
- network remodelling, where cells connect to each other;
- further differentiation into tubular structures.

For the purposes of this experiment, we are only concerned with the first stage, which takes place between six and nine hours in vitro [10]. At the end of this stage, the basic network structure has formed, but cells have not yet begun to bind to each other or to differentiate further. All cells are similar in general terms during this stage, although their individual properties may vary – for example, we would expect to see a roughly normal distribution of cell sizes.

We believe that in this stage the most significant forces are those resulting from physical interactions: between pairs of cells, between cells and their surrounding medium, and between cells and the substrate (Matrigel film [10, p. 1778]). As the surrounding medium is relatively thin and the interactions with the substrate are strong, there is only limited potential for cell movement away from the substrate, and 2D imaging can be used effectively to capture cell positions in real-world systems. Time-series imagery can be used to characterise cell interactions – for example, [7] shows physical interactions among stem cells *in vitro*, including attraction between cells, and cell shape changes after differentiation and binding.

As cells follow growth factor gradients, the density of cells tends to be higher where there is a higher concentration of growth factors in the environment. The *in vitro* experiment examined the effects of an artificial reduction of growth factor levels across the environment, imaging control and reduced-factor experiments at 3 h intervals.

2.3 Domain Model

Fig. 1 shows the entities within the *in vitro* experiment that we are attempting to reproduce, and their interactions. This includes both the biological entities under study and their experimental environment. Vascular structures are also included here as an emergent behaviour of the cells. Note that we have used UML in a rather informal way here, as we have in later figures – for example, while growth factors are indeed individual molecules, we would not generally think about them that way when modelling the system. While the semantics of the diagram are not

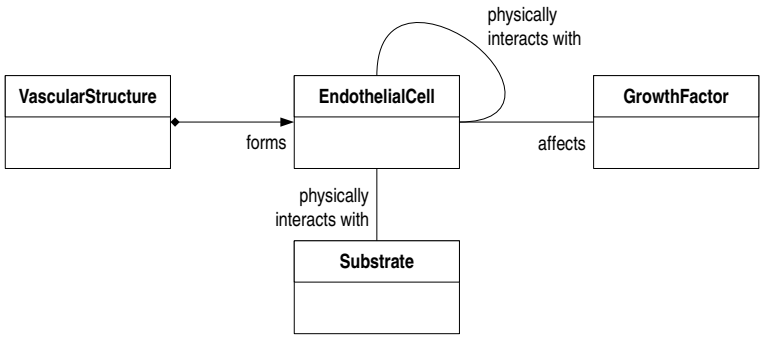


Fig. 1. Domain model: the entities of concern in the domain, shown as a UML class diagram

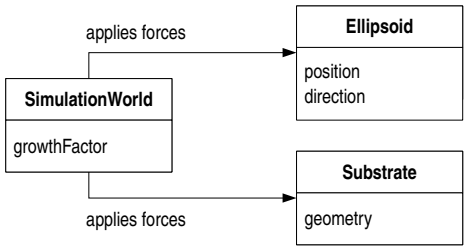


Fig. 2. Platform model: the entities of concern in the simulation, shown as a UML class diagram

correct, it is still useful as a "cartoon" in CoSMoS terms, capturing our (necessarily limited) understanding of the system in a convenient but loosely-specified notation.

2.4 Platform Model

Fig. 2 gives an overview of how the domain model has been simplified for the purposes of the simulation. We have chosen an agent-based modelling approach, so cells show a direct correspondence between the domain and platform models. This allows us to define interacting rules for single cells, and examine both the lower-level properties of individual cells and the higher-level behaviour of the system as a whole. The simulation proceeds in discrete timesteps, with all cells updating their positions and orientations atomically at the end of a timestep.

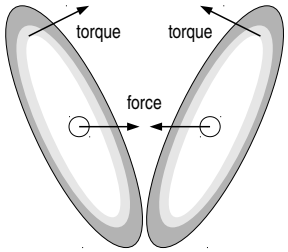


Fig. 3. Idealised ellipsoid cells within the platform model, showing torque and force

Vascular structures have been removed entirely, as these are the emergent property that we are attempting to reproduce. Other entities have been simplified, or introduced to allow implementation of the physical interactions within the system.

While cells can take a wide variety of shapes in the real world, we must model these as simpler shapes in order to practically simulate physical interactions at realistic scales. Modelling cells as simple spheres simplifies reasoning, but it does so by discarding information about the orientation of the cell, which limits the types of physical interactions that are possible. Initial prototyping showed that it was difficult to reproduce vascular formation behaviour using spheroid cells.

We therefore represent cells as ellipsoids. (Fig. 3). The shapes of cells observed in the *in vitro* experiment are roughly ellipsoidal (in the first phase). An ellipsoid has three orthometric semi-axes, which can be used as a local coordinate system. The rotation of an ellipsoid can be represented by the change of this local coordinate system, and the direction of an ellipsoid can be represented by the transformation from the local coordinate system to the global coordinate system. The position of the centre of an ellipsoid represents the position of the whole ellipsoid.

We are only interested in simulating the first phase of vascular formation, during which cells do not divide or measurably change their physical properties. We do not therefore need to simulate cell differentiation or the cell cycle, and can assume that cells' sizes and shapes are constant over time.

We assume that the density of cells is even, so forces can be modelled as acting on the centre of the cell, and changes in cell orientation can be modelled as torques acting on the cell. This is a modelling convenience

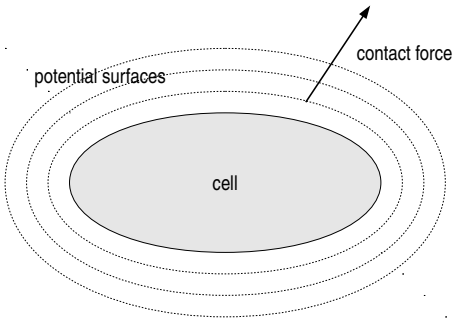


Fig. 4. An ellipsoidal cell showing potential surfaces, and the vector along which contact force is computed

and difficult to validate against experimental data, as cell rotations are hard to distinguish in 2D time-series images.

We model physical interactions between cells in terms of forces between them. The *adhesion force* attracts cells to each other; the *contact force* repels them and prevents them from overlapping; there is also a *resistance force* resulting from cells' interactions with the surrounding medium. For each force, there is an corresponding torque that is computed in an analogous way.

The contact force only takes effect when cells are in physical contact; the greater the overlap, the greater the contact force. As the ellipsoid is not an isotropic shape, we cannot simply use the distance between two ellipsoids to calculate the contact force and torque. Instead, we compute a potential for each interaction: a path-independent potential energy. In Fig. 4, dotted lines represent potential surfaces around the cell – the potential is constant for any point on the same surface, although the distance to the cell centre will vary as a result of the cell shape. The potential is calculated following Perram and Wertheim's approach [8], using the direction, position and length of the semi-axes of the interacting ellipsoids.

The potential is then transformed into energy using the Hertz formula. The magnitude of the resulting force or torque is the same for all points on a potential surface; the direction is computed based on the partial derivative of the energy field towards the centre of the interacting ellipsoid (Fig. 4).

The adhesion force, however, is modelled as a constant force attracting the centres of every pair of cells in the same way, provided they are

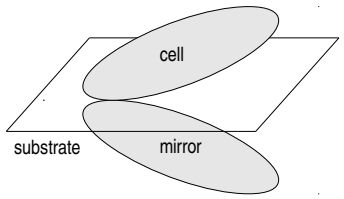


Fig. 5. Cell-substrate interaction, modelled as interaction with a copy of the same cell, mirrored in the substrate plane

within a minimum distance of each other. This is the simplest approach that reproduces the behaviour observed in time-series images of the *in vitro* experiments. If cells are beyond the minimum distance they have no physical interactions; if they are within range, they move towards each other, until they become close enough to overlap, stopping at the point at which the adhesion force and contact force balance each other.

The relative strengths of the two forces may be calibrated so that this balance happens at a potential corresponding to that observed in cells *in vitro*. The potential will depend on the elasticity of the cells, with higher balancing potential levels indicating more rigid cells. Some elasticity is necessary to obtain realistic cell interactions: an early prototype of the model used a simpler approximation to the Hertz function which effectively gave inelastic collisions between cells, and resulted in cells visibly "bouncing off" each other – which did not match what we see in time-series images!

As cells move at relatively low speeds within the medium, their acceleration can be approximated as zero – which means the sum of the forces upon them is also zero:

$$\sum \boldsymbol{F} = 0 = \boldsymbol{F}_{\text{contact}} + \boldsymbol{F}_{\text{adhesion}} + \boldsymbol{F}_{\text{resistance}} \tag{1}$$

We can therefore compute the resistance force in terms of the contact and adhesion forces – and, from this, compute the velocity of the cell using Stokes' law, based on the known size and shape of the cell and the properties of the medium. The angular velocity can be found using a similar technique; from these, the position and orientation of the cell on the next timestep can be computed.

The substrate itself is modelled as a plane. The physical interaction between a cell and the substrate is modelled as the interaction between a cell and its mirror image in the plane (Fig. 5). However, the adhesion force between a cell and its mirror image is scaled up to account for the

stronger interactions between cells and the substrate than between cells and other cells.

The model is dimensionless, being defined in terms of a unit time (the simulation timestep) and a unit length (the radius of a typical cell). These two quantities are related, in that computing the velocity of a cell within the fluid medium depends on both the timestep and the shape of the cell. However, making an assumption about the maximum velocity of a cell allows us to find reasonable bounds for one unit knowing the other, and in our case choosing a unit timestep of 1 s gives a physically-plausible maximum velocity for endothelial cells.

To summarise, we have made the following assumptions when constructing the platform model:

- Cells can be represented as ellipsoids.
- Cell size and shape do not change during the experiment.
- Matter is evenly distributed within a cell.
- Only contact force, adhesion force and resistance force are significant.
- Contact force can be computed using the Perram-Wertheim approach.
- Adhesion force can be modelled as a step function on distance (i.e. the growth factor gradient does not have a significant effect on attractive force).
- Contact and adhesion forces balance at a defined point when cells are in contact, and the strengths of the forces can be calibrated based on this.
- Cells move at very low speed, so their acceleration approaches zero and the forces upon them are balanced.
- Resistance force can be computed using Stokes' law, and the known properties of the fluid medium.
- Interactions with the substrate can be modelled as interactions with mirrored cells.

The physical parameters of the model (the unit time and length, and the constants involved in computing the forces) depend on the following values:

- the typical size of a cell;
- the range of ellipsoidal shapes a cell may adopt;
- the mean density of a cell;
- the dynamic viscosity of the fluid medium;
- the maximum speed at which a cell may move in the medium.

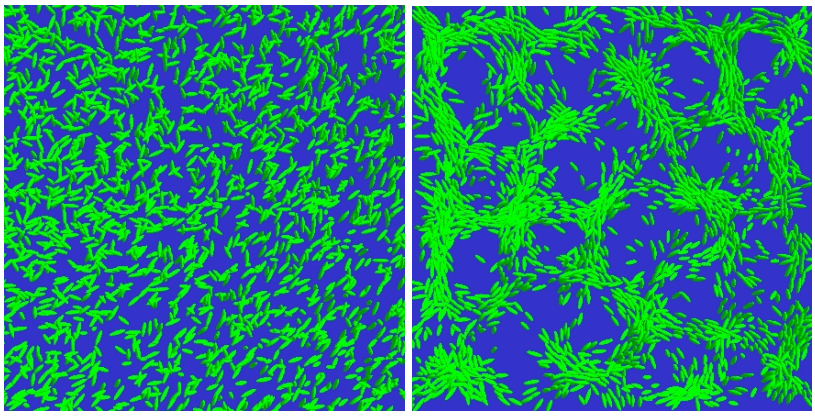


Fig. 6. Visualisation showing cell positions and orientations at the start (left) and end (right) of the simulation; "unstable" pattern

2.5 Simulation Platform

The simulation implementation follows the structure described in the platform model (Fig. 2). The simulation world object maintains the set of agents, and computes and applies the forces among them. In addition, it provides the ability to import simulation parameters, and to export the state of the simulation to a file for visualisation and analysis by external tools.

Model parameters were calibrated as described above. However, testing the simulation with these constants resulted in cells moving unrealistically rapidly. Reducing the strengths of the cohesion force and adhesion force by an order of magnitude resulted in more realistic cell movement – but the cause of this has not yet been traced back to the model.

2.6 Results Model

Fig. 6 shows the starting and ending conditions of the simulation. This certainly resembles the vascular network we are trying to reproduce – but we need a quantitative measure of this, in order to relate the results back to the changes in the level of growth factor.

There is a quantised method to describe the pattern of this structure, which is called the radial distribution function. The radial distribution function is a tool to describe space distribution of a system that consists of particles, by describing the chance of finding another particle within an arbitrary distance from the reference particle. In the form of the

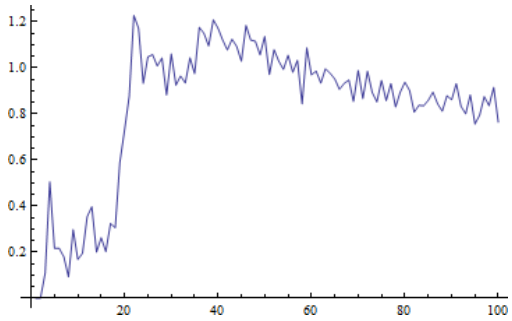


Fig. 7. Radial distribution of cells in Fig. 6 (right); X axis is distance between cells in simulation units, and Y axis is normalised probability of finding another cell at that distance

distribution curve, normally the X axis is distance, and the Y axis is the function value. If the function value is bigger than 1.0 at a certain distance, it means the cell density is higher than average at that distance; if the function value is smaller than 1.0, it means the cells are more sparse at this distance. Fig. 7 shows the radial distribution of cells at the endpoint of the simulation.

The minimum near distance 0 shows that cells tend not to have very close neighbours; the second minimum near distance 100 shows the typical size of hole in the net-shaped structure. This minimum corresponds to the typical net-size in [10], which is determined by the concentration of growth factor. As the distance from the reference cell increases, the value of the distribution function varies around 1.0, which means over longer distances the cells tend to be distributed evenly. Comparing with the distribution curve obtained from the *in vitro* experiment [10], we can say that our physical interaction has similar effects to the growth factor in the experiment.

If we allow the simulation to continue past the state shown in Fig. 6 – i.e. past the period of time covered in the original model design – the pattern will collapse into a few large clusters of cells. Fig. 8 shows the results of an simulation where the physical parameters have been adjusted to produce a stable pattern that does not collapse; while some network structure is visible, it is not as clear as the original model. This is echoed in the radial distribution, shown in Fig. 9, which no longer shows a clear minimum.

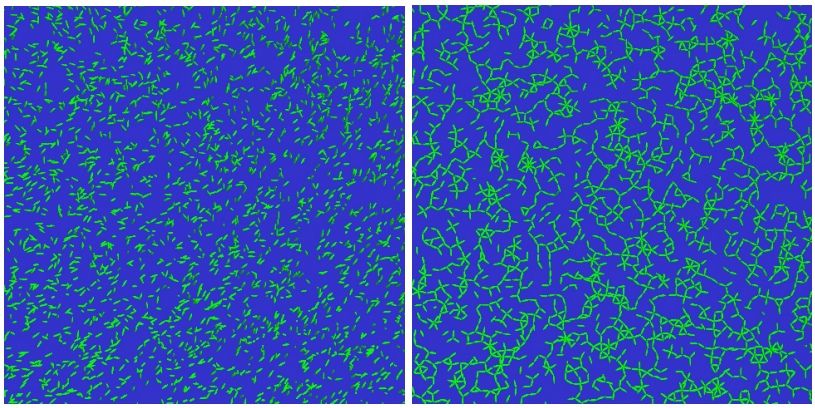


Fig. 8. Visualisation showing cell positions and orientations at the start (left) and end (right) of the simulation; "stable" pattern

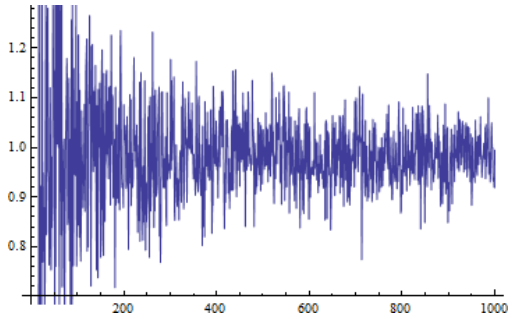


Fig. 9. Radial distribution of cells in Fig. 8 (right); X axis is distance between cells in simulation units, and Y axis is normalised probability of finding another cell at that distance

3 The New Model: Spheroid Growth

3.1 Research Context

As with the vascular development model, our objective is to relate lower-level physical interactions to higher-level structural behaviours: we want to explore the effects of

- certain cancer treatment drugs,
- hypoxia (low concentrations of oxygen), and
- different cell lines (types of cell grown for experimental purposes)

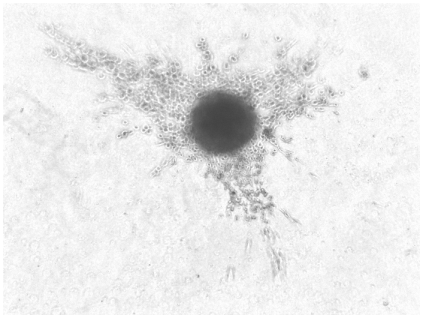


Fig. 10. 2D side-view image of a three-dimensional spheroid growing within a gel medium

upon the growth of tumours. This work forms part of a wider programme of activity developing techniques for cancer drug discovery and development [2]. Our domain experts are cancer researchers who are interested in making use of models and simulations to direct experimentation.

Tumours develop distinctive patterns of cells, which can be classified by domain experts either manually or using automated image processing. It is specifically these spatial patterns that we are interested in reproducing within a simulation.

Our existing physical model has already demonstrated the ability to reproduce spatial patterns of cell growth resulting from physical interactions within an agent-based simulation, and we have existing tools to visualise and analyse the output from the model. We would like to reuse as much of this infrastructure – both the model and the simulation code – as possible to reduce development time, but to do this we must identify the changes that need to be made by reevaluating our original assumptions within the new research context.

In addition, we must identify what information necessary for reengineering and calibrating the model needs to be obtained by wet-lab experimentation. We aim to maximise the value obtained from this experimentation.

3.2 Domain

In the real domain, cancer cells develop and grow into tumours within surrounding tissue [3]. In the lab, growth experiments may be conducted on a Petri dish – in which case cells can grow into a flat structure – or in a larger volume of gel, in which case spheroid structures can form (Fig. 10).

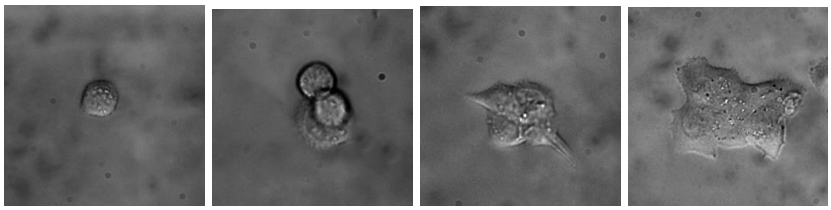


Fig. 11. HCT-116 (p53+/+) cells, growing on a glass plate, imaged at 6 h intervals. The diameter of the initial cell is 10 μm.

Petri-dish experiments are easier to collect data from, since 2D images can be taken non-destructively; spheroids must be sectioned before imaging in order to obtain data at a cellular resolution. A typical Petri-dish experiment contains around 5,000 cells; a spheroid contains on the order of 10^6 cells. A single section through a spheroid is comparable in size to a Petri-dish experiment.

Experiments are conducted using cell lines: cells grown for experimental use which have well-understood properties, such as the activation of particular oncogenes, or the ability to form structures such as spheroids.

The shape and volume of cells varies as they progress through their developmental cycle (Fig. 11); the rates at which the cycle progresses varies somewhat among cell lines. The HCT-116 cells we are using typically have diameter 10 μm immediately after division, and can be observed to grow over a period of approximately 24 h before dividing. Cells only remain healthy under experimental conditions for a limited period of time; it is therefore impractical to run experiments for more than 72 h, and images are typically taken every 6–8 h.

Some cancer drugs limit cell growth by arresting the cell cycle at a particular stage [6]. The progression of the cell cycle within the individual cells is therefore important when understanding the effects of drugs upon a tumour: if the cell cycles are synchronised (as can happen under experimental and *in silico* conditions), then a drug can arrest many cells simultaneously, whereas cells at a mix of developmental stages will be less strongly affected.

For spheroid structures, we are particularly interested in the effects of hypoxia, which can have a suppressive effect on cell growth [5]. The high density of cells within a spheroid structure means that cells become increasingly hypoxic towards the centre of the spheroid.

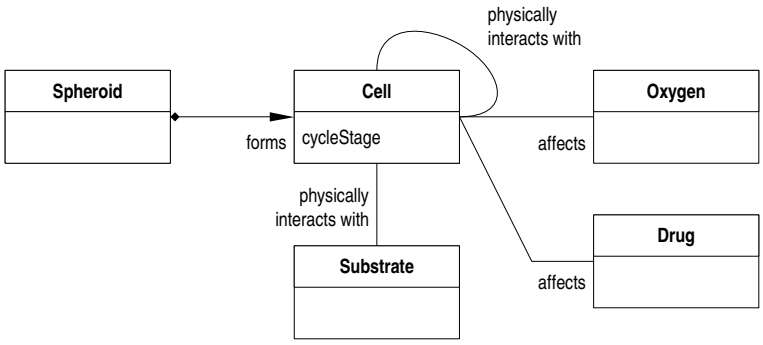


Fig. 12. Domain model: the entities of concern in the domain, shown as a UML class diagram

3.3 Domain Model

Fig. 12 shows the entities within the domain model. While the domain is substantially different from the previous one, the way cells are modelled retains a level of similarity, because the emergent behaviour of interest still results from physical interactions among cells. However, the physical properties of the cells themselves are somewhat different from our previous model – in particular, the cells' properties are known to change over time, and we are interested in the effects of this on the emergent properties.

The drugs and hypoxia condition are added to environment conditions, forcing cells to enter or quit certain cell cycle phases. For the two-dimensional experiment, we still need to consider the substrate. But for the spheroid experiment, as the tumour cells grow in 3D in agar gel, we will no longer consider the substrate.

3.4 Platform Model

We know from the domain model that the individual development of the agents – e.g. the growth of cells over time – will be important to the behaviours we are trying to replicate, and must be taken into account in the simulation. As a result, we have chosen again to use an agent-based modelling approach. Fig. 13 shows the entities within the simulation platform.

Fig. 14 shows the state machine that models a simplified cell cycle and drives the behaviour of the simulated cell. This represents the observed behavioural modes of the cell – growth, reproduction, apoptosis – rather

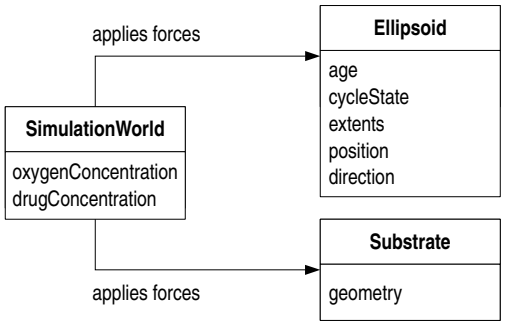


Fig. 13. Platform model: the entities of concern in the simulation, shown as a UML class diagram

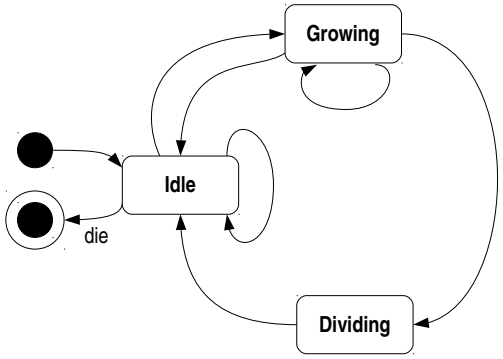


Fig. 14. Platform model: simplified cell cycle, shown as a UML state diagram

than the biological markers that would normally be used to describe cell cycle stages.

In the construction of the physical aspects of this platform model, we aim to reuse, as far as possible, our previous approach. In order to evaluate whether this is appropriate, we must reconsider our previous assumptions, listed in Sect. 2.4, based on our knowledge about the new research context.

While the physical properties of the cells and medium are somewhat different, we believe that most assumptions remain valid. One assumption, however, is no longer reasonable: cell size and shape *do* change during the simulation. This requires changes to how the interaction po-

tentials and their resulting forces and torques are computed, since these must now take changes to cell size and shape into account.

We must also ensure that we have sufficient information to allow calibration of the physical parameters. We no longer just need the typical size and shape of a cell: we need a profile showing how cell size and shape can change as the cell cycle progresses. This information will need to be obtained by time-series imaging under the experimental conditions we wish to simulate, as in Fig. 11. We will then give each simulated cell an interpolated growth curve based on the measured points. The other information we need for calibration is available in the literature (e.g. the dynamic viscosity of the medium).

The choice of timestep size (i.e. unit time in the model) is a concern. The timestep must be short enough to obtain results at a comparable temporal resolution to the *in vitro* experimental data. However, smaller timesteps require more calculation steps to simulate the same length of real-world time; the 1 s timesteps used in the previous simulation would result in *in silico* experiments taking an impractically long time to run with typical simulation sizes (10^3–10^6 cells).

In the *in vitro* experiment, the typical treatment time is 48 h to 72 h, and the sampling rate varies with different phrases of the experiment. For example, in the first 2 h, the cells may be imaged every 10 min, then the time between each sample increases as the experiment goes on. The simulation timestep does not therefore need to be any less than 10 min, and 1 h would probably be reasonable. The other constants will need to be adjusted to suit – for example, this results in the simulation's unit length being considerably smaller (which does not affect the outcome of the simulation).

3.5 Simulation Platform

The simulation platform is currently under development, following the structure described in the platform model. As we have reused the physical aspects of the platform model, we have similarly been able to reuse much of the code from the previous simulation platform – with the adaptations we have just described.

3.6 Results Model

We are primarily interested in the shape of structure tumour cells can form. We can expect cell density changes across a slice through the spheroid. Typically as hypoxia often happens in the centre of spheroids, the cell density in centre part should be lower than the cell density near

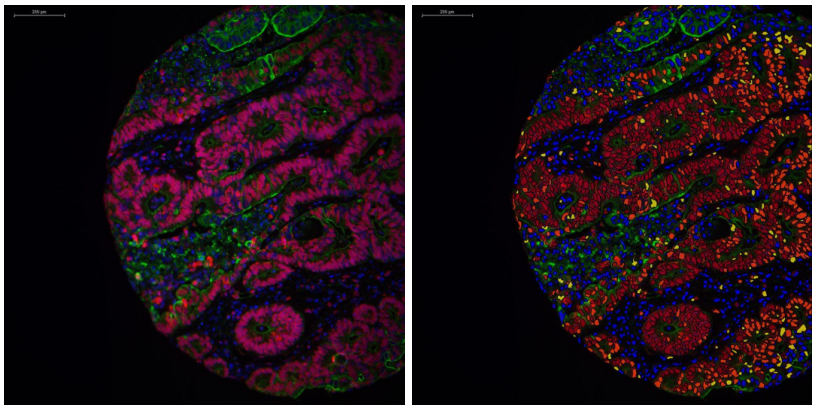


Fig. 15. Left: Ki67 expression in colorectal carcinoma tissue microarray data. Right: cell outlines and activity levels automatically identified from the previous image using Definiens.

the surface of the spheroid. If we slice the tumour tissue, the cell density should be lowest in the middle of the slice.

We are also interested in the overall shape of the spheroid. There are existing image analysis tools that can be used for this. We will use them to extract information from wet-lab experimental imagery, including the position and direction of all the cells. We will then have directly compatible data from both wet-lab experiments and our simulation that can be analysed using a consistent approach. With this information we may use methods such as fractal geometry [9] to analyse the overall structure of both experimental and simulation data. Based on our experience with our physical model, we expect to be able to obtain reasonably good correspondence for 2D data – but we suspect that extending the spatial interactions into three dimensions will require further elaboration of the model.

4 Conclusion

Through initial analysis guided by the CoSMoS project structure, we have identified that the physical aspects of the new domain do indeed have considerable similarities with those of the original domain – so reuse of the physical model should be appropriate, provided that the assumptions in the model – which we have explicitly identified – are reevaluated appropriately within the new context.

Other aspects of the two domains are substantially different; for example, the cancer simulation requires an implementation of the cell cycle in order to accurately simulate the effects of different treatments and environmental conditions, whereas this was unnecessary in the vascular simulation owing to the initial stage limitation.

We have also identified the gaps in our knowledge about the domain, necessary to appropriately calibrate the physical model, which must be filled by web-lab experimentation. This gives us confidence that we are asking the right questions when conducting experiments in support of calibration.

What we have ended up with is emphatically *not* "a CoSMoS project" – we have simply made use of a few aspects of CoSMoS to structure our thinking about model reuse. In effect, we have only made use of some of the large-scale CoSMoS patterns that describe concepts such as "domain model", and that in a rather sketchy and informal way – but we feel that even this first step towards CoSMoS has been valuable in terms of forcing us to think in a principled way about our existing work. As this work continues, we intend to make increased use of CoSMoS techniques; for example, to more effectively structure our interactions with domain experts during the calibration of the spheroid model. We feel, in general, that the ability to adopt patterns as appropriate is a significant strength of the CoSMoS approach in terms of adoption by existing projects – as it is for other pattern languages.

While it is important to emphasise that this project is still work in progress, we feel that we have achieved a satisfactory degree of model and software reuse – and, more importantly, we are confident that this reuse has been achieved in a way that is *appropriate* and *useful* within our new research context. In the future, we would like to consider strategies and patterns for this kind of reuse within the CoSMoS process – in particular, how a validity argument might be constructed and updated as a model is reused.

In addition, by documenting this process, we now have a framework in place that would allow us to reuse the physical model within new research contexts in future projects. Once the cancer cell growth model has been demonstrated in two dimensions, we plan to extend it to simulate three-dimensional spheroid structures – which will require further reevaluation of the physical model, particularly relating to interactions with a gel medium.

5 Acknowledgements

The authors would like to thank their colleagues who kindly provided data and illustrations for this paper. Hilal Khalil ran the experiments and provided the images in Fig. 11. Fig. 10 appears courtesy of Simon Langdon. Fig. 15 appears courtesy of Peter Caie.

References

[1] Paul S. Andrews, Fiona A. C. Polack, Adam T. Sampson, Susan Stepney, and Jon Timmis. The CoSMoS process version 0.1: A process for the modelling and simulation of complex systems. Technical Report YCS-2010-453, Department of Computer Science, University of York, March 2010.

[2] James Bown, Paul S. Andrews, Yusuf Deeni, Alexey Goltsov, Michael Idowu, Fiona A. C. Polack, Adam T. Sampson, Mark Shovman, and Susan Stepney. Engineering simulations for cancer systems biology. *Current Drug Targets*, 13(14), December 2012.

[3] Antonio Brú, Sonia Albertos, José Luis Subiza, José López García-Asenjo, and Isabel Brú. The universal dynamics of tumor growth. *Biophys J.*, 85(5):2948–2961, November 2003.

[4] Judah Folkman and Christian Haudenschild. Angiogenesis in vitro. *Nature*, 288:551–556, 1980.

[5] A. Kaida and M. Miura. Visualizing the effect of hypoxia on fluorescence kinetics in living HeLa cells using the fluorescent ubiquitination-based cell cycle indicator (Fucci). *Exp Cell Res.*, 318(3):288–297, February 2012.

[6] David O. Morgan. *The cell cycle: principles of control.* New Science Press, 2007.

[7] Lane Niles. Human cultured neural stem cells, September 2012. `<https://www.youtube.com/watch?v=x_e3PEJgrFY>`.

[8] John. W. Perram, John Rasmussen, Eigil Præstgaard, and Joel L. Lebowitz. Ellipsoid contact potential: Theory and relation to overlap potentials. *Phys. Rev. E*, 54:6565–6572, December 1996.

[9] Anne Savage, Elad Katz, Alistair Eberst, Ruth E. Falconer, Alasdair Houston, David J. Harrison, and James Bown. Characterising the tumour morphological response to therapeutic intervention: an ex vivo model. *Dis Model Mech.*, 6(1):252–260, January 2013.

[10] Guido Serini, Davide Ambrosi, Enrico Giraudo, Andrea Gamba, Luigi Preziosi, and Federico Bussolino. Modeling the early stages of vascular network assembly. *EMBO J.*, 22(8):1771–1779, April 2003.

[11] Susan Stepney et al. *Engineering simulations as scientific instruments.* Springer, 2013. To appear.

CoSMoS simulation experiment reproducibility and the ODD protocol

Susan Stepney

Department of Computer Science, University of York, UK

Abstract. CoSMoS is a defined approach for designing, building, and arguing fit for purpose, a scientifically rigorous simulation of a complex real world system. The ODD (Overview, Design concepts, Details) protocol is a standardised way of describing simulation models, defined to support reproducibility. This paper demonstrates that use of the CoSMoS approach to build a simulation supports the presentation of the simulation results using the ODD protocol. It does so by presenting a mapping from the various ODD components, to where the required information for those components is located in a CoSMoS project's documentation. All the information is present, but is distributed through the project documentation. Moreover, a project developed using the CoSMoS approach documents additional information, which is necessary not merely for reproducibility, but for understanding of and confidence in the simulations.

1 Introduction

Computer-based simulation is a key tool in many fields of scientific research. In silico experiments can be used to explore and understand complex processes, to guide and complement in vitro and in vivo experiments, to suggest new hypotheses to investigate, and to predict results where experiments are infeasible. Simulation is an attractive, accessible tool: producing new simulations of simple systems is relatively easy. But it is also a dangerous tool: simulations are often complex, buggy, and difficult to relate to the real-world system.

A simulation needs to be both scientifically useful to the researcher, and scientifically credible to third parties; it needs to have the properties of a well-designed *scientific instrument* [4]. The CoSMoS (Complex Systems Modelling and Simulation infrastructure) project has established an approach to simulation of complex systems that supports principled

development of simulations as scientific instruments. The CoSMoS approach [2, 23] is generic: it does not mandate a particular modelling technique, or particular implementation language. What it does mandate is the careful and structured use of models and arguments, to ensure that the simulation is both well-engineered, and seen to be well-engineered. In order to help developers through this careful and structured approach, we have developed a pattern language [22, 23] to help guide development, promote good simulation engineering practice, and warn of potential pitfalls.

Since CoSMoS is generic, it can work in harmony with various other approaches that are used for building scientific simulations. For example, there are well-established third party simulation implementation libraries and frameworks. NetLogo [26] is a multi-agent programmable modelling environment used for prototyping multi-agent simulations. It provides a programming language and user-interface widgets to support rapid prototyping of relatively simple agent-based simulations. It has a large user-base and a large library of example simulations. MASON [12], from George Mason University, is a discrete-event multi-agent simulation library, which can be used as the foundation for Java simulations. FLAME (Flexible Large-scale Agent Modelling Environment) [6], from the University of Sheffield, is an agent-based modelling system. From an extended finite state machine model, FLAME generates a complete agent-based application, which can be targetted to computing systems ranging from laptops to super computers. Libraries and frameworks such as these can be exploited as part of a simulation project being developed with the CoSMoS approach. They can be used as the implementation basis of the CoSMoS Simulation Platform (see later) or of a Prototype. Their integration into the overall project can be argued fit-for-purpose using the CoSMoS Argumentation approach [15–17]. Thus a CoSMoS user can benefit from the existing significant investment in such implementation platforms.

In addition to specific implementation technologies, researchers have also developed approaches to making the resulting simulations more scientifically acceptable. One essential requirement of a scientific result is that it be *reproducible* [18, §22]. This requires sufficient description of the experimental details that third parties can replicate the setup, and reproduce the result. If simulation is to be used as a scientific instrument, credible to the scientific community, then it is necessary for individual simulation experiments to be described in sufficient detail for their reproducibility. Grimm and co-workers have devised the ODD (Overview, Design concepts, Details) protocol [7, 8] as a standard way of describing simulations using Agent Based Models (ABMs), with the explicit aim

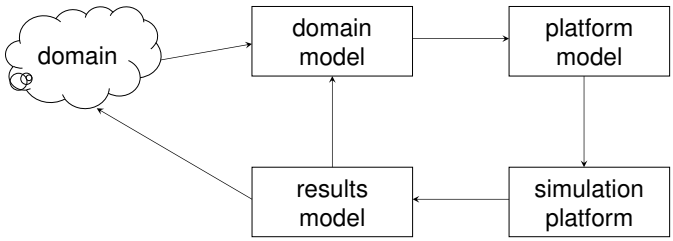


Fig. 1. Relationship between simulation components; arrows represent flows of information. These are all framed by the Research Context.

of making them reproducible: "ODD is expected to lead to more complete model descriptions, making ABMs easier to replicate and hence less easily dismissed as unscientific" [8].

The Open ABM Consortium [13] provides support for the ODD protocol. It has a discussion forum "to allow the ABM community to collaboratively develop this protocol into a widely useable communication standard for describing ABMs in the social sciences" [11]. As part of its support, it defines CoMSES (Computational Modeling for SocioEcological Science) Modeling Standards, which include (among other things) the requirement to be "fully documented using the ODD standard for model documentation, or an equivalent documentation protocol" [14].

In the same way that existing libraries and frameworks can be exploited by the CoSMoS user for implementation, the existing ODD protocol can be followed for presenting CoSMoS simulation experiments and results. This provides the CoSMoS user with a well-established and recognised means to ensure third party reproducibility of their scientific simulation results. Contrariwise, it provides the ODD-compliant simulation developer with the broader CoSMoS approach within which to develop a scientifically credible simulator.

In order to support such reuse of the ODD protocol within CoSMoS, this paper presents a mapping from the various ODD components, to where the required information is located in CoSMoS components.

2 CoSMoS in a nutshell

The CoSMoS approach is documented through a pattern language [22, 23]. These patterns are based on a collection of models (figure 1) [3] and other components. The patterns referenced in this paper are:

Research Context: the statement of the overall scientific context, goals and scope of the simulation-based research being conducted, includ-

ing the Simulation Purpose, resources, constraints, assumptions, and success criteria.

Domain: the part of the real world that is the system of study, that the simulation project is "about".

Domain Model: a model encapsulating the scientific understanding of appropriate aspects of the Domain. It provides the agreed scientific basis and assumptions for the development of a Simulation Platform; simulation implementation details are not considered in this model.

Platform Model: a model providing the high level specification of the Simulation Platform, comprising design and implementation details, incorporating relevant Domain Model scientific concepts, Research Context experimental requirements, and implementation constraints and assumptions.

Simulation Platform: the encoding of the Platform Model into a Calibrated software and hardware platform with which various Simulation Experiments can be performed.

Results Model: a model that encapsulates the understanding of outputs and results from Simulation Experiments, in Domain terms, enabling comparison with results from domain experiments.

Simulation Purpose: a component of the Research Context. It documents the scientific purpose for which the Simulation Platform is being built and used. The *purpose* of a simulation exercise is the single most important concept. Without an agreed purpose, it is impossible to scope the research context, or to arrive at a consensus over fitness for purpose. The purpose of the simulation constrains the appropriate levels of abstraction for modelling, the appropriate implementation languages and platform, and the appropriate analysis and interpretation of results. A simulator that is designed for one purpose may or may not be modifiable for another purpose.

Modelling Approach: a component of the Domain Model and Platform Model. The choice of an appropriate modelling approach and notation that can capture the relevant aspects of the Domain and Simulation Purpose in a form that can be implemented in the Simulation Platform.

Argumentation: the demonstration that the simulation is fit for purpose (as defined in the Simulation Purpose). The fitness-for-purpose argument exposes the rationale for scientific and engineering credibility of the simulator, the assumptions made, the limitations in the purpose and scope of the simulator, and in the potential use of its results. While CoSMoS's rigorous model-based approach ensures that the simulator *is* fit for purpose as a scientific instrument, the accompanying argumentation ensures that the simulator *is seen to be* fit for purpose.

Simulation Experiment: a specific experiment executed on the Simulation Platform. This should be designed and documented by analogy to a domain experiment, including the hypothesis to be tested, initial conditions and parameter values, experimental process, and results analysis.

Data Dictionary: a component of the Domain Model, Platform Model, Results Model, and Simulation Experiment. It provides a common definition of the modelling data (parameter values such as sizes, scales, rates) used to build the Simulation Platform, calibration data, and the experimental data (initial values and results) both from Domain experiments and the corresponding Simulation Experiments.

Calibration: tuning the Simulation Platform parameter values so that simulation results match the experimental calibration data provided in the Data Dictionary [1, 20].

There are many more patterns that make up the CoSMoS approach. The ones summarised here are those that are relevant to mapping the ODD protocol.

3 ODD and CoSMoS

The ODD (Overview, Design concepts, Details) protocol was introduced in [7] and refined in [8] as a standard way of describing simulation models (particularly in the ecological domain), specifically to aid reproducibility of implementation of such models. By 2010 the original formulation had been used in more that 50 publications [8]; experience gained from this use led to its update.

ODD is more narrowly focussed than CoSMoS: it is concerned with reproducibility of simulation models from their given descriptions (although a noted side-effect of its use is "a more rigorous formulation of models" [8]). Hence the ODD protocol information is only part of the entire CoSMoS model: it is mostly contained in the Platform Model, and some of the Research Context. ODD explicitly does not cover any Results Model features of experimentation and sensitivity analysis (it considers these to be the "methods" part of a description; the ODD protocol covers only the "materials" part of a standard scientific article [7]). Also, it is more concerned with the implemented model, so it has only a little that correspond to Domain Model elements.

In addition to the explicit material noted in the summary below, the ODD protocol also recommends making the source code available (that is, the Simulation Platform and Simulation Experiment), and using code comments to highlight the various parts of the protocol information [7].

The CoSMoS approach has places for all the information needed to give an ODD protocol description. This section describes the various components of the updated ODD protocol (as defined in [8, §3]), and explains where the corresponding material can be found in a CoSMoS-based simulation.

ODD quotations in this section are taken from [8].

3.1 ODD: purpose

ODD component: "a concise summary of the overall objectives(s) for which the model was developed"

CoSMoS location: this maps naturally onto the Simulation Purpose.

Simulation purpose is closely tied to criticality (for example, will the results be applied directly, or be used to generate hypotheses that can be tested by other means) and impact (for example, will the results affect small-scale research, or large scale policy decisions) [15]. If the purpose is to provide critical evidence for high-impact research, then the approach to simulation development should access state-of-the-art software engineering methods, argumentation, and documentation approaches; the simulator and its fitness-for-purpose must be capable of international expert scrutiny. However, if the purpose is to provide a test-bed for hunches and a generator of hypotheses, all of which will then be subject to conventional laboratory analysis and confirmation before publication, then the development and argumentation need to be just good enough: internal consensus and internal documentation are sufficient.

3.2 ODD: entities, state variables and scales

This part of the ODD protocol specifies the structure of the modelled world in terms of its components. Like CoSMoS, ODD claims not to be wedded to any one implementation approach: "Although the protocol was designed for [Agent Based Model]s, it can help with documenting any large, complex model" [8]. Nevertheless, the terminology in this part of the ODD protocol is ABM-specific. The Modelling Approach is assumed to be Agent Based from now on; UML is a notation that may be used to capture aspects of ABMs [5].

Entities

ODD component: "An entity is a distinct or separate object or actor that behaves as a unit and may interact with other entities or be affected by external environmental factors."

[8] distinguishes the following types of entities: (i) agents/individuals; (ii) spatial units (grid cells); (iii) environment; (iv) collectives (groups of agents that can have their own group-level identity and behaviours)

CoSMoS location: the implemented entities are captured as part of the Platform Model; if UML is the notation used, then the entities could be partially captured as the classes in a class diagram.

Most Platform Model entities are derived from, but may differ from, those in the Domain Model. For example, simplifications may be made; platform entities may be surrogates for more complex domain entities. In particular, the Domain Model may include emergent entities that are not explicitly included in the Platform Model, since determining their (hypothesised) emergence may be part of the Simulation Purpose. Non-implemented emergent entities are distinct from ODD "collectives", which are explicitly modelled and implemented hierarchical entities. It is important to distinguish these cases in order to ensure that the desired emergent answer is not hard-coded into the simulation. This is one reason why CoSMoS makes a careful distinction between the Domain Model and Platform Model [3].

There may be additional entities in the Platform Model, such as implementation-specific entities (such as instrumentation and interface entities needed to support Simulation Experiments, or proxy entities needed to implement a large-scale distributed simulator). Again, this is a reason to separate the Domain Model and Platform Model: the scientific Domain Model is not polluted with implementation details, and may be an appropriate basis for diverse implementations.

State variables

ODD component: "A state variable or attribute is a variable that distinguishes an entity from other entities of the same type or category, or traces how the entity changes over time. ... [they] can contain both numerical variables and references to behavioural strategies. ... If state variables have units, they should be provided."

CoSMoS location: the implemented state variables are captured as part of the Platform Model. If UML is the notation used, these would be captured, for example, as the instance variables (for "numerical variables")

and methods (for "behavioural strategies") in a class diagram. The CoSMoS Data Dictionary component pattern contains some of the detail, including the units.

Scales

ODD component: "spatial and temporal scales and extents (the amount of space and time represented in a simulation), ... what the model's units represent in reality."

CoSMoS location: this information is documented in the Data Dictionary, both the Domain Model part for physical scales, and the Platform Model part for any surrogates and abstractions of these scales. Tuning these and other experimental parameter values, particularly where experimental values refer to Domain entities and the corresponding Platform Model entities are surrogates, is part of Calibration. The validation of the mapping from real world units to simulation units, and of the appropriateness of the spatial and temporal discretisation (including spatial grid size and time-step size) is part of the Argumentation process.

3.3 ODD: process overview and scheduling

ODD component: this covers the dynamical behaviour of the model: what the agents do, in what order (the order in which state variables are updated, synchronous or asynchronous, concurrency); the behaviour of any controller object; how time is modelled (discrete, continuous, or hybrid).

"one should use pseudo-code to describe the schedule in every detail, so that the model can be re-implemented from this code."

CoSMoS location: the Platform Model contains this information, at various levels of detail. If UML is the notation used, the highest level model might be expressed as activity diagrams and state diagrams. Lower level models, developed on the way to implementation, might be expressed in pseudo-code.

3.4 ODD: design concepts

The final part of the ODD protocol is a list of specific concepts that need to be defined for reproducibility. See also [9, ch.5], [19] for further discussion. These concepts are particularly important for ecological models where the agents have complex behaviours, where the agents may change

their behaviours, and where emergent properties are prominent. Not all these concepts are important in every CoSMoS simulation, but when they are, there is a place for them to be captured in the approach, listed below.

ODD: basic principles

ODD component: the general concepts, theories, hypotheses and modelling approaches underlying the model's design.

CoSMoS location: these are variously captured in the Research Context and the Domain (general concepts), the Domain Model (theories and hypotheses), and the Modelling Approach. The various components may be brought together during Argumentation.

ODD: emergence

ODD component: the system level phenomena that emerge from the individual behaviours, rather than being built in to the model

CoSMoS location: emergent properties are captured in the Domain Model, and removed from Platform Model. In general, given a hypothesis under consideration, components in the Domain Model that are *outcomes* of hypothesised mechanisms, whether emergent or not, should not appear in the Platform Model, to ensure that the answer is not be explicitly coded into the Simulation Platform [3]. The Results Model is used to capture emergent properties of Simulation Experiments; the simulated properties captured in the results model can be compared to the real-world properties in the Domain Model.

ODD: adaptation, objectives, learning, prediction

ODD component: the rules the entities have for making decisions and changing behaviour in response to changes in themselves or the environment; measures of an entity's adaptive success (such as fitness, utility), and the criteria used to measure this success; how entities change their adaptive traits; how an entity predicts future consequences in order to make decisions.

CoSMoS location: the real world versions of these can all be captured in the Domain Model as a specific kind of entity behaviour or property. The appropriate abstractions and surrogates are then all captured in the Platform Model. If UML is the notation used, then a stereotype can be used to highlight the specific kinds of behaviours and properties of interest.

ODD: sensing, interaction

ODD component: what state variable values (of itself, and of the environment) an entity has access to, that it can exploit to guide or influence its behaviour; any direct and indirect interactions between agents; representation of communications

CoSMoS location: setting the scope of entity interactions, in terms of sensing capabilities and levels of detail, is part of the Domain Model. The information access across entities required for simulation is captured in the Platform Model. If UML is the notation used, then the relationships between entities (including the environment entity) a class diagram can document the information they can sense about each other.

ODD: stochasticity

ODD component: how and where randomness is used to model real world variability that is unimportant to model in detail

CoSMoS location: the Domain Model captures the real world variability at some level of abstraction; the Platform Model implements this as randomness; the Argumentation of the appropriateness of this particular form of implementation demonstrates that the approximation is fit for purpose.

ODD: collectives

ODD component: collections of agents that behave as entities in their own right; which are emergent, and which are explicitly modelled

CoSMoS location: the Platform Model captures explicitly-modelled collectives; emergent collective entities are in the Domain Model but removed from the Platform Model (see the discussion under the emergence heading earlier in this section).

ODD: observation

ODD component: a definition of the data samples collected from the ABM for testing, understanding, and analysing it.

CoSMoS location: the Platform Model includes specification of the instrumentation used to produce the output data; the Data Dictionary (Results Model component) includes the specification of the output data and its analysis; the Simulation Experiment includes experiment-specific details.

3.5 ODD: initialisation

ODD component: the initial state of simulation, the number and state of agents and the environment, at time $t = 0$.

CoSMoS location: state variable values, and other parameter values, for specific Simulation Experiments are held in the Data Dictionary.

3.6 ODD: input data

ODD component: the data that drives environmental variables (such as rainfall or harvesting regimes) [7], imported from external files or models.

CoSMoS location: raw data for specific Simulation Experiments is held in the Data Dictionary. If the data is produced on the fly by an external model, that model and its interfaces will be captured, at some level of abstraction, in the Platform Model. Initialisation data for that model is also held in the Data Dictionary.

3.7 ODD: submodels

ODD component: the detailed sub-models, mathematical equations, rules, and parameters that define the processes (§3.3).

CoSMoS location: the Platform Model contains this information, at various levels of detail. For components such as mathematical equations, the Platform Model may contain a reference to where the equation appears in the Domain Model, plus the extra definitions necessary to implement the equation under the prevailing model assumptions such as the implementation of time and space. Parameters are stored in the Data Dictionary.

4 Summary and Conclusions

As demonstrated, the documentation resulting from a simulation project developed under the CoSMoS approach contains places for all the information needed to document an ABM using the ODD protocol. Hence CoSMoS can be used to develop ODD-compliant simulations. However, the material is scattered through several CoSMoS artefacts: the Platform Model and its Data Dictionary; the Research Context and Simulation Purpose; the Domain Model; the Argumentation; and more. If the ODD protocol is to be used to present the results of a CoSMoS-developed Simulation Experiment, then it would be sensible to devise some project standards that specifically tag the ODD-relevant information in each of the CoSMoS artefacts.

A CoSMoS project's artefacts contain much more than the information required by the ODD protocol. Specifically, much of the ODD information is found in the CoSMoS Platform Model, yet much of the information in the Platform Model is itself derived from the Domain Model. Additionally, there is information in the Domain Model explicitly not carried over to the Platform Model: the emergent properties and other hypothesised outputs of the Simulation Experiments. This extra, crucial, information has a formal home in the CoSMoS approach.

Furthermore, a key part of the CoSMoS approach is Argumentation: the explicit demonstration that the simulation is fit for purpose [15–17]. This exposes a difference in the goals of ODD and CoSMoS: use of the ODD protocol makes simulation experiments more *reproducible*; use of the CoSMoS approach additionally helps to ensure that the simulated world has a clear link to the real world (the experiments *make domain sense*), and that the simulator as a scientific instrument is fit for purpose (the experimental results are *credible*).

The developers of ODD are not trying to capture everything in their protocol: in particular, ODD covers only the "materials" part of a standard scientific article, and not the "methods" part [7]. The ODD researchers are developing TRACE (Transparent and Comprehensive Ecological Documentation) [10, 21] for more fully documenting models and their analyses. The OpenABM CoMSES Modeling Standards requires that a model "Correctly simulates the processes it claims to simulate" [14]. Future work for the CoSMoS approach is to demonstrate how it is compliant with the requirements of TRACE and CoMSES.

Acknowledgements

This work is part of the Complex Systems Modelling and Simulation (CoSMoS) project, partially funded by EPSRC grants EP/E053505/1 and EP/E049419/1. I would like to thank my CoSMoS project colleagues, and pattern book [23] co-authors, for providing much of the raw material on which this paper is based. Particular thanks go to Paul Andrews, Jim Bown and Fiona Polack, for comments on earlier versions of this paper. Thanks also to Teodor Ghetiu for bringing the ODD protocol to the CoSMoS team's attention.

CoSMoS project documentation is available from the CoSMoS project website www.cosmos-research.org.

References

[1] Kieran Alden, Mark Read, Jon Timmis, Paul S. Andrews, Henrique Veiga-Fernandes, and Mark Coles. *Spartan*: A comprehensive tool for understanding uncertainty in simulations of biological systems. *PLoS Comput Biol*, 9(2):e1002916, 2013.

[2] Paul S. Andrews, Fiona A. C. Polack, Adam T. Sampson, Susan Stepney, and Jon Timmis. The CoSMoS process, version 0.1: A process for the modelling and simulation of complex systems. Technical Report YCS-2010-453, Department of Computer Science, University of York, March 2010.

[3] Paul S. Andrews, Susan Stepney, Tim Hoverd, Fiona A. C. Polack, Adam T. Sampson, and Jon Timmis. CoSMoS process, models, and metamodels. In Stepney et al. [25], pages 1–13.

[4] Paul S. Andrews, Susan Stepney, and Jon Timmis. Simulation as a scientific instrument. In Stepney et al. [24], pages 1–10.

[5] Bernhard Bauer and James Odell. UML 2.0 and agents: How to build agent-based systems with the new UML. *Journal of Engineering Applications of Artificial Intelligence*, 18:141–157, 2005.

[6] FLAME website: www.flame.ac.uk.

[7] Volker Grimm, Uta Berger, Finn Bastiansen, Sigrunn Eliassen, Vincent Ginot, Jarl Giske, John Goss-Custard, Tamara Grand, Simone K. Heinz, Geir Huse, Andreas Huth, Jane U. Jepsen, Christian Jørgensen, Wolf M. Mooij, Birgit Müller, Guy Pe'er, Cyril Piou, Steven F. Railsback, Andrew M. Robbins, Martha M. Robbins, Eva Rossmanith, Nadja Rüger, Espen Strand, Sami Souissi, Richard A. Stillman, Rune Vabø, Ute Visser, and Donald L. DeAngelis. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198(1-2):115–126, 2006.

[8] Volker Grimm, Uta Berger, Donald L. DeAngelis, J. Gary Polhill, Jarl Giske, and Steven F. Railsback. The ODD protocol: A review and first update. *Ecological Modelling*, 221(23):2760–2768, 2010.

[9] Volker Grimm and Steven F. Railsback. *Individual-based Modeling and Ecology*. Princeton University Press, 2005.

[10] Volker Grimm and Amelie Schmolke. How to read and write TRACE documentations, 1st draft. Technical report, Helmholtz Centre for Environmental Research, Leipzig, Germany, 2011.

[11] Marco A. Janssen, Lilian Na'ia Alessa, Michael Barton, Sean Bergin, and Allen Lee. Towards a community framework for agent-based modelling. *Journal of Artificial Societies and Social Simulation*, 11(2):6, 2008.

[12] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. MASON: A multi-agent simulation environment. *Simulation: Transactions of the society for Modeling and Simulation International*, 82(7):517–527, 2005. Website: cs.gmu.edu/~eclab/projects/mason.

[13] OpenABM website: www.openabm.org.

[14] OpenABM model certification: www.openabm.org/faq/what-model-certification-and-how-does-it-work.

[15] Fiona A. C. Polack. Arguing validation of simulations in science. In Susan Stepney, Peter H. Welch, Paul S. Andrews, and Adam T. Sampson, editors, *2010 CoSMoS workshop*, pages 51–74. Luniver Press, 2010.

[16] Fiona A. C. Polack, Paul S. Andrews, Teodor Ghetiu, Mark Read, Susan Stepney, Jon Timmis, and Adam T. Sampson. Reflections on the simulation of complex systems for science. In *ICECCS 2010*, pages 276–285. IEEE Press, 2010.

[17] Fiona A. C. Polack, Alastair Droop, Philip Garnett, Teodor Ghetiu, and Susan Stepney. Simulation validation: exploring the suitability of a simulation of cell division and differentiation in the prostate. In Stepney et al. [25], pages 113–133.

[18] Karl Popper. *The Logic of Scientific Discovery*. Hutchinson, 1959.

[19] Steven F. Railsback. Concepts from complex adaptive systems as a framework for individual-based modelling. *Ecological Modelling*, 139(1):47–62, 2001.

[20] Mark Read, Paul S. Andrews, Jon Timmis, and Vipin Kumar. Techniques for grounding Agent-Based Simulations in the real domain: a case study in Experimental Autoimmune Encephalomyelitis. *Mathematical and Computer Modelling of Dynamical Systems (MCMDS)*, 18(1):67–86, 2012.

[21] Amelie Schmolke, Pernille Thorbek, Donald L. DeAngelis, and Volker Grimm. Ecological models supporting environmental decision making: a strategy for the future. *Trends in Ecology & Evolution*, 25(8):479–486, 2010.

[22] Susan Stepney. A pattern language for scientific simulations. In Stepney et al. [24], pages 77–103.

[23] Susan Stepney, Kieran Alden, Paul S. Andrews, James L. Bown, Alastair Droop, Teodor Ghetiu, Tim Hoverd, Fiona A. C. Polack, Mark Read, Carl G. Ritson, Adam T. Sampson, Jon Timmis, Peter H. Welch, and Alan F. T. Winfield. *Engineering Simulations as Scientific Instruments*. Springer, 2013. in preparation.

[24] Susan Stepney, Paul S. Andrews, and Mark Read, editors. *Proceedings of the 2012 Workshop on Complex Systems Modelling and Simulation, Orleans, France, September 2012*. Luniver Press, 2012.

[25] Susan Stepney, Peter Welch, Paul S. Andrews, and Carl G. Ritson, editors. *Proceedings of the 2011 Workshop on Complex Systems Modelling and Simulation, Paris, France, August 2011*. Luniver Press, 2011.

[26] Uri Wilensky. NetLogo. Website: ccl.northwestern.edu/netlogo.

Understanding Self-Organized Regularities: AOC-Based Modeling of Complex Healthcare Systems

Li Tao and Jiming Liu*

Department of Computer Science, Hong Kong Baptist University
ltao,jiming@comp.hkbu.edu.hk

Abstract. A healthcare system, as a well recognized complex system, exhibits certain types of self-organized regularities, such as the statistical distribution of wait-time variations. What remains to be a challenge in understanding a complex healthcare system is how to model and characterize emergent self-organized regularities by taking into account the underlying individual-level behavior (e.g., patient hospital selection behavior) with respect to various impact factors (e.g., geographic accessibility to services, hospital resourcefulness, and service performance). In this paper, we present an Autonomy-Oriented Computing (AOC) approach to modeling and simulating service utilization in the context of a cardiac surgery system, so as to characterize the self-regulating patient arrivals and wait time. By experimenting with the AOC-based cardiac surgery system model (AOC-CSS), we are able to explain how the global-level regularities in patient arrivals and wait time may emerge from the individual-level patient hospital selection behavior and its interrelationship with hospital wait time.

1 Introduction

A healthcare system has been well recognized as a complex system [24] [17]. Some interesting self-organized regularities in healthcare service utilization, such as the power-law distribution of specialists' waiting-list variations (i.e., the variations of mean time that patients spend in specialists' waiting lists) [26], have been reported. However, it is still unclear what and how individual-level patient behavior (e.g., hospital selection

* Corresponding author

behavior) and underlying factors (e.g., distance from homes to services, hospital resourcefulness in terms of operating room capacity and physician supply, and service performance represented as wait time) account for such emergent global-level regularities.

Dynamically-changing patient arrivals and wait time may be directly or indirectly affected by various factors, as schematically illustrated in Fig. 1, including, but not limited to, demographics, socioeconomics, environment (e.g., weather), human behavior [3], service delivery behavior [29], and geographic accessibility to services [28]. For instance, old age (physiological age at a biological scale) is an important risk factor for cardiovascular diseases, while the patient hospital selection behavior at a personal scale may heavily influence the distribution of actual patient flows to various hospitals. Furthermore, the factors of demand and supply may have complex interrelationships, coupled interactions, and/or feedback loops [23]. For instance, as shown in Fig. 1, the hospital performance (a quality measurement for hospitals, e.g., wait time), may affect the patient hospital selection behavior via a feedback loop, which will in turn influence the distribution of patient flows and further exert effects on the performance of hospitals.

In view of this, to understand the global-level self-organized regularities in healthcare service utilization from a complex systems perspective, it would be essential to address the following issues:

- **Scope**: What factors, variables, processes, and hierarchical levels (e.g., services in a hospital level or in a regional level) are relevant, and hence should be investigated and modeled?
- **Coupling relationships and/or interactions**: What are the interrelationships among the impact factors and variables? Identifying their local feedback loop(s) would be crucial for understanding global-level self-organized regularities.
- **Heterogeneity**: The behavior of patients in choosing hospitals may be heterogeneous due to their differences in personal profiles, socioeconomic backgrounds, and service distributions in and around their residence areas. Hospitals may also be heterogeneous in delivering healthcare services because of their variations in equipped resources, management strategies, and dynamically-changing patient arrivals. Thus, capturing the heterogeneity of patients and hospitals will be essential in modeling and simulating a real-world complex healthcare system.

In this paper, we present an Autonomy-Oriented Computing (AOC)-based, "bottom-up" approach [19] to understanding the global-level self-organized regularities relating patient arrivals and wait time in a cardiac

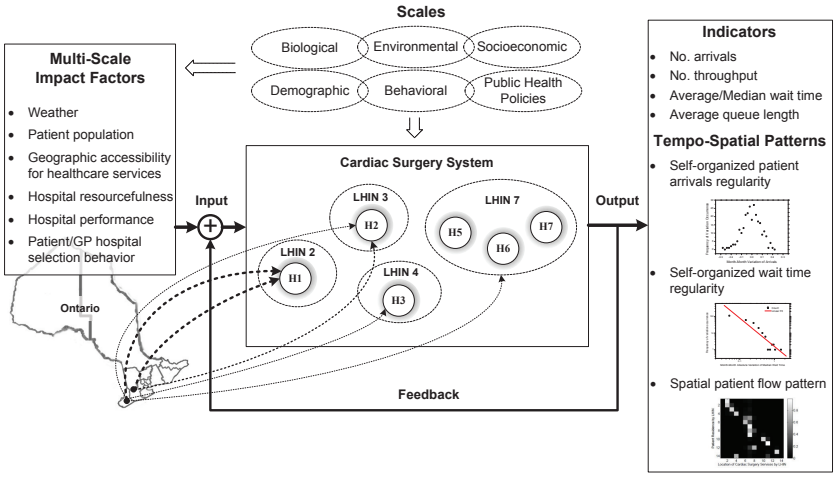


Fig. 1. An illustration of the complex cardiac surgery system in Ontario, Canada. The illustrated tempo-spatial patterns are observed from secondary data about cardiac surgery service utilization between January 2005 and December 2006. LHIN represents Local Health Integration Network, a geographic-location-based health authority responsible for planning and determining healthcare service needs and priorities in a certain area of Ontario, Canada [6]. There are 14 different LHINs in total. LHIN 2 to LHIN 7 are exemplified LHINs in Ontario (refer to http://www.lhins.on.ca/FindYourLHIN.aspx for the corresponding geographic areas in Ontario). H denotes a hospital.

surgery system. In essence, "an AOC system is a multi-agent system (MAS)" but different from a traditional MAS in that it is aimed to address "the issues of self-organization, self-organized computability, interactivity..." [18, p.9] in solving problems as well as in modeling complex systems. It is not only scalable and efficient in problem solving, but also capable of characterizing the properties of complex systems, such as self-organized regularities and emergent behaviors, by means of modeling the underlying individuals' behavior and their interactions and hence uncovering the essential mechanisms of positive-feedback-based aggregation and collective regulation in the systems.

Specifically, in this work, we utilize the AOC-by-prototyping approach [19] to construct an AOC-based cardiac surgery system model (AOC-CSS). In modeling the real-world cardiac surgery system in Ontario, Canada, we consider multi-scale factors impacting patient arrivals (as shown in Fig. 1), such as weather, demographics of cities/towns in Ontario, geographic accessibility for cardiac surgery services, resource-

fulness of physicians in a hospital, hospital performance in terms of wait time, and patient hospital selection behavior. Our goal is to characterize the global-level self-organized regularities in service utilization, as emergent from the individual-level behaviors of patients and hospitals with respect to multi-scale impact factors.

The remainder of this paper is organized as follows. Section 2 presents the global-level self-organized regularities in healthcare systems as reported in previous studies as well as in our work. Section 3 briefly states the research problem and related issues. Section 4 shows the detailed formulation of our AOC-CSS model. Section 5 presents and discusses simulation-based studies and results on characterizing the regularities of patient arrivals and hospital wait time. We finally summarize our findings and consider future work in Section 6.

2 Empirical Regularities in Complex Healthcare Systems

Prior studies have empirically identified self-organized regularities in healthcare systems. For instance, Smethurst and Williams found that the monthly absolute variations of wait time that patients spent in specialists' waiting lists (as calculated by the changes of mean wait time w at time steps t and $t-1$ $(w_t - w_{t-1})/w_t$) followed a power-law distribution [26], and thus concluded that hospital waiting lists were self-regulating.

In this work, we have analyzed the month-to-month variations of patient arrivals and median wait time for 11 hospitals providing cardiac surgery services in Ontario, Canada, over a 2-year period from January 2005 to December 2006. The investigated data was provided by Cardiac Care Network of Ontario (http://www.ccn.on.ca/ccn_public/FormsHome/HomePage.aspx; accessed in February 2011). The month-to-month variations of patient arrivals (or median wait time) are calculated as follows:

$$v_{t+1} = \frac{x_{t+1} - x_{min}}{x_{max} - x_{min}} - \frac{x_t - x_{min}}{x_{max} - x_{min}} \tag{1}$$

where v_{t+1} denotes the variation of patient arrivals (or median wait time) at time $t+1$. In this work, each time step t corresponds to a month. x_t denotes the number of patient arrivals (or median wait time) at time t. x_{min} and x_{max} are the minimum and the maximum values of patient arrivals (or median wait time) over the two-year period, respectively.

Accordingly, the absolute month-to-month variations of patient arrivals (or median wait time), v'_t, can be obtained as follows:

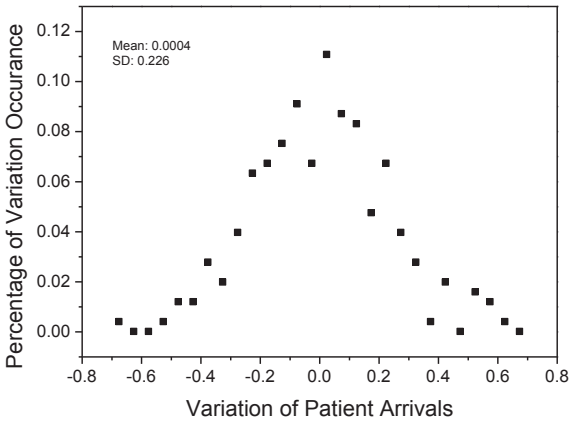


Fig. 2. The statistical distribution of patient-arrival variations for cardiac surgery services in Ontario, Canada, between January 2005 and December 2006. The distribution follows a normal distribution; the normality of the distribution has passed the Kolmogorov-Smirnov test.

$$v'_t = |v_t| \quad (t > 1) \tag{2}$$

By observing the (absolute) variations of patient arrivals and median wait time, two types of self-organized regularities can readily be discovered, as shown in Figs. 2 and 3.

As shown in Fig. 2, the monthly variations of patient arrivals against the percentages of variation occurrences follow a normal distribution with mean value of 0.004 and standard deviation (SD) of 0.226. More interestingly, as shown in Fig. 3, the monthly absolute variations of median wait time follow a power-law distribution with power of -1.36 and standard deviation of 0.28 ($p < 0.001$), implying that there exists a statistical regularity in month-to-month variations.

3 Problem Statement

In this work, as aided by AOC, we will build a white-box model (i.e., AOC-CSS model), which not only considers the input and output of a real-world cardiac surgery system, but also addresses the underlying interaction mechanisms among the entities in the system (e.g., patients

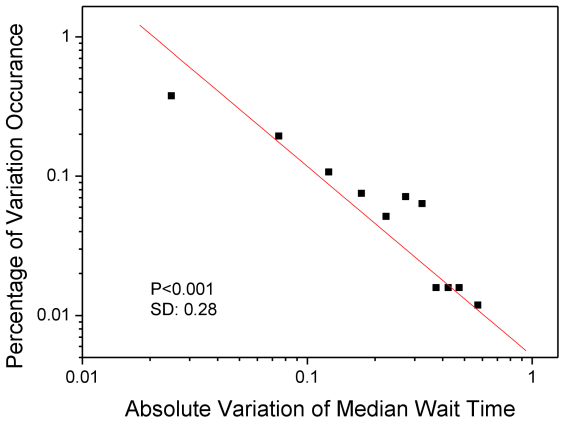


Fig. 3. The statistical distribution of absolute variations of median wait time for cardiac surgery services in Ontario, Canada, between January 2005 and December 2006. The distribution follows a power law with power of -1.36 ($p < 0.0001$).

and hospitals), to explain how the global-level regularities (as shown in Figs. 2 and 3) emerge from the individual-level behavior and interactions.

Specifically, in designing the AOC-CSS model, we will address the following issues:

- **Major impact factors**: What factors, by and large, affect the patient hospital selection behavior?
- **Behavioral rules**: How to formulate the behavioral rules of the entities in the focal cardiac surgery system that govern the patient hospital selection behavior, while taking into account the identified impact factors and the heterogeneity of patients and hospitals?
- **Local interactions and feedback loop(s)**: How to capture the local interactions and feedback loop(s) of the entities in the system?
- **Simulation-based validation**: Can the empirically observed global-level regularities emerge from the individual-level behavior (e.g., the behavior of patients with respect to the wait time of a hospital) through simulation?

In what follows, we will describe in detail how we build the AOC-CSS model and address the above-mentioned issues.

4 The AOC-Based Cardiac Surgery System Model (AOC-CSS)

As a case study to understand self-organized regularities by means of AOC-based modeling and simulation, we present the following three steps in constructing an AOC-based cardiac surgery system model (AOC-CSS):

1. Identifying the participating heterogeneous entities in the system, major impact factors, and local feedback loop(s).
2. Modeling the system based on the AOC methodology where special attention is paid to deriving entities' behavioral rules that incorporate (1) the heterogeneity of the entities, (2) the identified impact factors, and (3) the local feedback loop(s) (e.g., the feedback loop between the patient hospital selection behavior and hospital wait time). In this step, the spatial pattern of patient flow distributions by LHINs (as shown in Fig. 1), which reflects aggregated effects of the patient hospital selection behavior, will be considered.
3. Capturing the self-organized regularities by means of simulating the constructed AOC-CSS model.

4.1 Identifying Entities, Major Factors, and Local Feedback Loop(s)

In Ontario, general physicians (GPs) are the "gatekeepers" for referrals to cardiac surgery services since 93% of the population in Ontario have GPs [5]. When a patient needs a cardiac surgery, he/she will first make a mutual decision with his/her GP on choosing a hospital for the required treatment [1]. In most cases, the patient, no matter what his/her age and socioeconomic background is, will select a hospital following his/her GP's recommendation [3] [11]. For this reason, modeling the patient hospital selection behavior is equivalent to modeling the patient-GP mutual decisions on selecting a hospital. After the patient and the GP make a mutual decision on hospital selection, the GP will refer the patient to the selected hospital, in which the patient will register and wait for receiving the treatment [1]. Finally, the patient will leave the hospital after finishing the treatment.

Based on the above process, we can readily identify three types of autonomous entities in the cardiac surgery system; they are: *GP*, *patient*, and *hospital*. For each patient entity, he/she and his/her GP will make a mutual decision on hospital selection based on (1) the released information about the hospitals and (2) the applicable behavioral rules for hospital selection taking into account certain impact factors.

According to the previous literature, the key factors affecting the decisions of patients and GPs on choosing hospitals may include distance (from homes to services), the resourcefulness of a hospital (referred to as hospital resourcefulness hereafter), and hospital performance (e.g., wait time). First of all, it has been well recognized that the geographic distance from homes to services is negatively associated with the likelihood to select the services (i.e., selection probability) for patients [25] [15] and GPs [16] [12] [15] because patients are more likely to visit hospitals close to their homes. The resourcefulness of a hospital, as represented by the number of physicians [29], has been found to be positively correlated with the probability that patients and GPs select the specific hospital [29] [13] [27] because more hospital resources may attract more patient arrivals [26]. In addition, wait time is also a major concern for patients [3] and GPs [15] [22], who are usually in favor of hospitals with short wait time [3] [15] [22].

The aforementioned interrelationships among these factors and patient-GP mutual decisions on hospital selection may form a certain feedback loop between patient arrivals and hospital wait time (as illustrated in Fig. 4), which may result in nonlinear phenomena (e.g., self-regulating patient arrivals and wait time) in the complex cardiac surgery system. For instance, the long wait time in a hospital may weaken the probability of patients/GPs selecting the hospital, which will in turn decrease the number of patient arrivals, and thus the wait time in the hospital will be reduced soon afterwards.

In what follows, we will describe the detailed formulation of the AOC-CSS model, defining the three types of entities, the environment in which they reside and receive the released information, and their behavioral rules.

4.2 Environment

In our work, the geographic relationship/structure between cities and hospitals is conceptualized as a weighted bipartite network defined as follows:

Definition 1: City-Hospital Network $CH = (C, H, D)$, where $C(N) = \{c_i\}$ $(i \in [1, N])$ and $H(M) = \{h_j\}$ $(j \in [1, M])$ are two node sets, $H \cap C = \emptyset$; $D = \{d_{ij}\}$ $(i \in [1, N], j \in [1, M])$ is a set of weighted edges.

In our research context, each node c_i $(\forall c_i \in C)$ represents a sampled city/town, which has more than 40,000 population in 2006 according to the census data in Ontario, Canada. Each node h_j $(\forall h_j \in H)$ denotes a hospital that provides cardiac surgery services in Ontario, Canada.

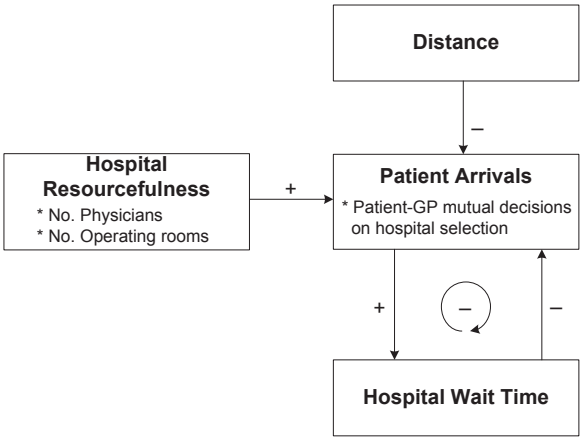


Fig. 4. The effects of impact factors on patient-GP mutual decisions on hospital selection and the interacting feedback loop. In this figure, "+" or "–" means a positive or a negative relationship between two factors, respectively.

Finally, each weighted edge d_{ij} ($\forall d_{ij} \in D$) represents the driving time from a city/town c_i ($\forall c_i \in C$) to a hospital h_j ($\forall h_j \in H$) which is estimated by using the "Get directions" function in Google Maps [4].

The environment E in the AOC-CSS model records the released information about hospitals. We formally define environment E as follows:

Definition 2: Environment E for the AOC-CSS model is represented as a bipartite network as defined in Definition 1. E maintains information that could be accessed by patients and GPs. We define environment E as a tuple: $< D, R, W >$, where the elements are given as follows:

- D: Distance information $D = \{d_{ij}\}$. Each d_{ij} records the driving time between city/town c_i ($\forall c_i \in C$) and hospital h_j ($\forall h_j \in H$).
- R: Hospital resourcefulness information $R = \{r_j\}$, where r_j records the number of physicians in h_j ($\forall h_j \in H$) .
- W: Wait time information $W = \{w_{j,\tau}\}$. Each $w_{j,\tau}$ records the wait time information for hospital h_j ($\forall h_j \in H$) at time round τ. Here, a unit time round τ to review hospital operations (e.g., one month or one quarter) includes T number of unit time steps t (a unit of time to record the hospital operational information, e.g., one day), i.e., $\tau = T * t$. In this paper, $w_{j,\tau}$ records the median wait time of h_j over the past time round $\tau - 1$.

4.3 Entities

(1) General physician (GP)

In the AOC-CSS model, patients come to a hospital selected upon patient-GP mutual decisions and the released information in the environment E. As most cardiac surgery patients are referred by GPs, we define entities $GP[N]$ to record and represent patient-GP mutual decisions on hospital selection.

Definition 3: $GP[N]$ records the information for patients and GPs to make hospital selection decisions. We assume that GPs are homogeneous in a city/town, so that all GPs in city/town c_i can be regarded as one GP entity, GP_i. Each entity GP_i maintains a record: $< cityID, rule, P, \lambda_k, A_k >$, where the elements are given as follows:

- $cityID$: The unique identity for a city/town that GP_i resides in.
- $rule$: The behavioral rules representing how GP_i and his/her patients make mutual decisions on selecting hospitals. The specific behavioral rules will be introduced in Section 4.4.
- P: The hospital preference information, $P = \{p_j\}$ $(\forall j \in [1, M])$. Each p_j presents GP_i's preference for hospital h_j $(h_j \in H)$. The preference for a hospital is estimated based on GP_i's behavioral rule for hospital selection and the accessed environmental information.
- λ_k: The mean number of type k patients at each time step.
- A_k: The patient flow information for type k $(k \in K)$ patients, $A_k = \{a_{k,j}\}$. Each $a_{k,j}$ records the number of type k $(k \in K)$ patient arrivals for hospital h_j $(h_j \in H)$ at each time step.

(2) Patient

At each time step t, city/town c_i generates a number of patient entities following a poisson distribution, $\sum_{\forall h_j \in H} a_{k,j} \sim P(\lambda_k)$. The mean patient number varies from one city/town to another due to the differences in demographic characteristics, such as population size and age profile. A patient entity can be defined as follows:

Definition 4: A patient entity, *patient*, records its information in a hospital. Each patient entity maintains a record: $< patientID, cityID, hospitalID, type, joinTime, endTime, \tilde{w} >$, where the elements are given as follows:

- $patientID$: The unique identity represented by a constant for a patient.

- $cityID$: The unique identity for the city/town that a patient comes from.
- $hospitalID$: The unique identity for the hospital that a patient arrives at.
- $type$: The patient type that represents the heterogeneity of patient entities, $\forall k \in [1, K]$ $(K \geq 1)$.
- $joinTime$: The time step that a patient joins in the queue of a hospital.
- $endTime$: The time step that a patient has been served in a hospital.
- $\tilde{w}$: The wait time of a patient, $\tilde{w} = endTime - joinTime$.

(3) Hospital

In this work, we model the operations of a hospital entity that provides cardiac surgery services in the same way as the existing studies on queueing theory based modeling and simulation of hospital operations (i.e., hospital service behavior) [9]. Since operating rooms for cardiac surgery services in a hospital are, to a certain extent, homogeneous, it is reasonable to regard hospital j as one server (i.e., one operating room) with service rate μ_j, and thus assume that each hospital is an M/M/1 queueing model [14]. A hospital entity can be defined as follows:

Definition 5: $Hospital[M]$ records the information of all the hospitals. Each hospital entity h_j $(\forall h_j \in H)$ maintains a record: $< hospitalID, cityID, \mu, \tilde{A}_k, w, queue >$, where the elements are given as follows:

- $hospitalID$: The unique identity for a hospital.
- $cityID$: The unique identity for the city/town in which a hospital is located.
- $\tilde{A}_k$: The patient arrival information for type k $(k \in K)$ patients, $\tilde{A}_k = \{\tilde{a}_{i,k}\}$. Each $\tilde{a}_{i,k}$ records the number of type k $(k \in K)$ patients coming from city/town c_i at each time step.
- μ: The hospital service rate.
- w: The wait time information of hospital h_j in a past time period, which will be released in environment E. In this work, the wait time information w released at time round τ is the average median wait time for the past time round $\tau - 1$.
- $queue$: The queue including all the patient entities waiting for cardiac surgery services at each time step.

4.4 Behavioral Rules for Selecting Hospitals Based on Stylized Facts

By reviewing the literature, some stylized facts about the effects of key impact factors, i.e., distance, hospital resourcefulness, and wait time,

on patient-GP mutual decisions on hospital selection are identified as follows:

- ***Stylized fact 1***: The probability for a patient selecting a hospital is exponentially and inversely associated with the distance from his/her home to the hospital [25].
- ***Stylized fact 2***: Distance has a certain threshold effect [3] for patients and GPs to select hospitals. That is to say, patients and GPs will, more or less, consider visiting a hospital within a certain distance threshold. If the distance from homes to a specific hospital is longer than a distance threshold, patients and GPs are less likely to visit the hospital. For instance, patients may consider to visit a hospital within two-hour driving time. Here, two-hour driving time could be regarded as a distance threshold.
- ***Stylized fact 3***: Patients usually prefer to visit resourceful hospitals, in terms of human resources (e.g., physicians) and physical resources (e.g., ORs) [29] [13] [27]. In other words, the hospital resourcefulness and the number of patient arrivals are positively correlated in a cardiac surgery system [20].
- ***Stylized fact 4***: Patients usually prefer to visit a hospital with shorter wait time [3] [15] [22]. However, certain patients, especially the elderly will less likely or never consider the wait time factor when they select hospitals [3].

In view of this, two preferred categories of hospitals that patients and GPs may consider to visit can be defined as follows:

Definition 6: The first preferred category of hospitals G_{i1} ($i \in [1, N]$, $G_{i1} \subseteq H$) represents the most preferred hospitals with respect to driving distances for patients and GPs residing in city c_i. G_{i1} contains the hospital(s) that the patients and the GPs can reach within a distance $\hat{d}$ ($\hat{d} > 0$); or the nearest hospital(s) if there does not exist any hospitals within $\hat{d}$ from city c_i.

Definition 7: The second preferred category of hospitals G_{i2} ($i \in [1, N]$, $G_{i2} \subseteq H$) denotes the hospitals that are farther than those in G_{i1}, but can be reached within a distance $\beta * \hat{d}$ for patients and GPs residing in city c_i. Here β ($\beta > 1$) is a multiplying factor to determine the distance threshold for G_{i2} based on that for G_{i1}.

The three different behavioral rules for patients and GPs residing in city c_i to determine the probability a_{ij} for selecting hospital h_j are defined as follows:

***Behavioral rule 1*: D-rule (Distance).** Patients and GPs select a hospital h_j based only on the distance d_{ij} from their residing city/town c_i to the hospital. The hospital selection probability is calculated by using Equation 3:

$$a_{ij} = \begin{cases} \delta * \frac{(d_{ij})^{-\alpha}}{\sum_{h_m \in G_{i1}} \delta * (d_{im})^{-\alpha} + \sum_{h_m \in G_{i2}} (d_{im})^{-\alpha}}, & h_j \in G_{i1} \\ \frac{(d_{ij})^{-\alpha}}{\sum_{h_m \in G_{i1}} \delta * (d_{im})^{-\alpha} + \sum_{h_m \in G_{i2}} (d_{im})^{-\alpha}}, & h_j \in G_{i2} \\ 0, & else \end{cases} \tag{3}$$

where δ ($\delta \geq 1$) is a bias factor to indicate the preference of patients or GPs for hospitals in G_{i1} over those in G_{i2}, α is an exponent factor to scale the hospital selection probability with respect to driving distances.

***Behavioral rule 2*: DH-rule (Distance and Hospital resourcefulness).** Patients and GPs choose a hospital h_j based on the distance d_{ij} from their residing city/town c_i to the hospital, and the hospital resourcefulness r_j as represented by the number of physicians. The hospital selection probability is calculated by using Equation 4.

$$a_{ij} = \begin{cases} \frac{\delta * (d_{ij})^{-\alpha}}{\sum_{h_m \in G_{i1}} \delta * (d_{im})^{-\alpha} + \sum_{h_m \in G_{i2}} (d_{im})^{-\alpha}} * f(r_j), & h_j \in G_{i1} \\ \frac{(d_{ij})^{-\alpha}}{\sum_{h_m \in G_{i1}} \delta * (d_{im})^{-\alpha} + \sum_{h_m \in G_{i2}} (d_{im})^{-\alpha}} * f(r_j), & h_j \in G_{i2} \\ 0, & else \end{cases} \tag{4}$$

$$f(r_j) = \frac{r_j}{\sum_{h_m \in (G_{i1} + G_{i2})} r_m}$$

***Behavioral rule 3*: DHW-rule (Distance, Hospital resourcefulness, and Wait time).** Patients and GPs select a hospital h_j based on the distance d_{ij} from their residing city/town c_i to the hospital, the hospital resourcefulness r_j, and the released wait time information $w_{j,\tau}$ at time round τ. The hospital selection probability is calculated by using Equation 5.

$$a_{ij} = \begin{cases} \frac{\delta * (d_{ij})^{-\alpha}}{\sum_{h_m \in G_{i1}} \delta * (d_{im})^{-\alpha} + \sum_{h_m \in G_{i2}} (d_{im})^{-\alpha}} * f(r_j) * f(w_{j,\tau}), & h_j \in G_{i1} \\ \frac{(d_{ij})^{-\alpha}}{\sum_{h_m \in G_{i1}} \delta * (d_{im})^{-\alpha} + \sum_{h_m \in G_{i2}} (d_{im})^{-\alpha}} * f(r_j) * f(w_{j,\tau}), & h_j \in G_{i2} \\ 0, & else \end{cases} \tag{5}$$

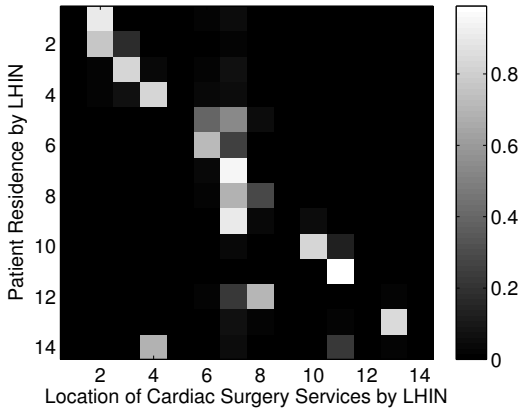


Fig. 5. The distribution of operated cardiac surgery patients with respect to their residence by LHINs in the year of 2007-2008 in Ontario, Canada.

$$f(r_j) = \frac{r_j}{\sum_{h_m \in (G_{i1}+G_{i2})} r_m}$$

$$f(w_{j,\tau}) = \frac{(w_{j,\tau})^{-1}}{\sum_{h_m \in (G_{i1}+G_{i2})} w_{j,\tau}^{-1}}$$

Due to the differences in factors, such as geographic accessibility to cardiac surgery services, patients and GPs in different cities may use different behavioral rules for selecting hospitals. The spatial pattern of real patient flows (as shown in Fig. 5), which represents the aggregated effects of patients' and GPs' hospital selection behaviors, will be utilized to find out the suitable behavioral rule for patients and GPs in each city/town. During this process, we are also able to estimate the values of the distance threshold $\hat{d}$, the multiplying factor β, the exponent factor α, and the bias factor δ from real-world data.

Based on our experiments, it has been found that when $\hat{d} = 0.5$, $\alpha = 2.8$, $\beta = 1.5$, and $\delta = 1.5$, we can get relatively small values of mean and standard deviation of absolute errors. Here, the absolute error is defined as $|e_{ij}| = |\hat{a}_{ij} - \hat{a}'_{ij}|$, where e_{ij} is the error between the percentage of patients residing in LHIN l_i coming to hospitals in LHIN l_j in the year of 2007-2008 in Ontario, and that obtained from our simulation. The best-fit behavioral rules found for patients and GPs residing in cities/towns within different LHINs are summarized in Table 1. The estimated hospital selection probabilities and errors with respect to the best-fit behavioral rule for each LHIN are shown in Fig. 6.

Table 1. A summary of best-fit behavioral rules for patients and GPs residing in cities/towns within different LHINs

LHIN	Hospital	Behavioral Rule
1	–	DH-rule
2	London Health Sciences Centre	DHW-rule
3	St. Mary's General Hospital	D-rule
4	Hamilton Health Sciences	DH-rule
5	–	DH-rule
6	Trillium Health Centre	DH-rule
7	St. Michael's Hospital, Sunnybrook Hospital, University Health Network	DHW-rule
8	Southlake Regional Health Centre	DH-rule
9	–	DHW-rule
10	Kingston General Hospital	DH-rule
11	University of Ottawa Heart Institute	D-rule
12	–	D-rule
13	Hôspital Régional de Sudbury	DHW-rule
14	–	D-rule*

∗: Since the driving time from any city/town in LHIN 14 in Ontario to any cardiac surgery services is longer than 10 hours, the service accessibility as well as the hospital selection behaviors of patients and GPs in LHIN 14 are apparently different from those in other LHINs. In this case, the best-fit behavioral rule may not adequately characterize the selection behavior for patients and GPs.

–: No hospital providing cardiac surgery services.

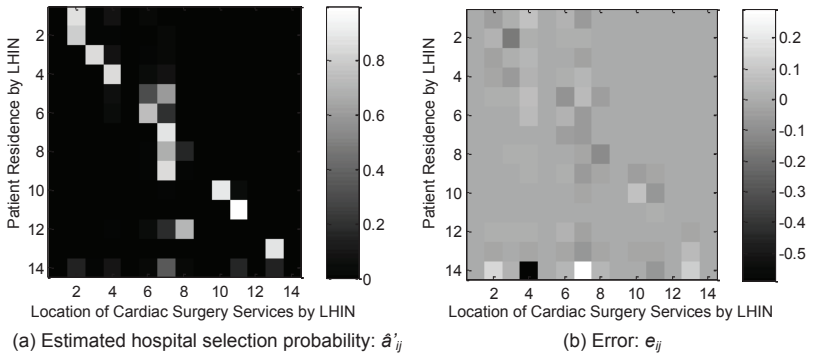


Fig. 6. (a) Estimated hospital selection probabilities for patients residing in each LHIN, and (b) errors between the percentages of patients residing in an LHIN coming to hospitals in its own LHIN or other LHIN(s) in the real-world and those as obtained from the simulation, with respect to the best-fit rules as shown in Table 1.

5 Simulation and Discussion

In this section, we will describe simulations based on our AOC-CSS model with an aim to understand the global-level self-organized regularities in cardiac surgery service utilization.

5.1 Simulation Settings

The parameters in the AOC-CSS model are initialized by using publicly available data. Cardiac Care Network of Ontario (CCN) published monthly statistical reports on cardiac surgery service utilization in Ontario hospitals in the years between January 2005 and December 2006 (we accessed the data in February 2011). In the statistical reports, the average number of treated cases, the median wait time, and the queue length in a month for each hospital were reported. Therefore, the service rate μ_j for hospital h_j can be approximated as the average number of served cases in a day. The arrival rate for each patient type in city/town c_i can be approximated by the following Equation 6:

$$\sum_{k \in K} GP_i.\lambda_k = s_i * m_i \tag{6}$$

where s_i is the patient-generation probability, i.e., the probability of a person in city/town c_i being a patient who needs a cardiac surgery service, m_i is the size of total population in city/town c_i. The parameter s_i represents the heterogeneity of city/town c_i in producing patient population requiring cardiac surgery services with respect to its demographics and socioeconomic factors. In this work, the patient-generation probabilities for cities/towns in each LHIN could be inferred from [8]. The total population m_i for each city/town is gathered from the 2006 Canada Census data [7].

There are two types of patients in our simulation, i.e., $K = 2$. One patient type is urgent, and the other is non-urgent. According to the data reported in [8, p.71], the arrival rate of urgent patients versus that of non-urgent patients is set to 0.23:0.77. Urgent patients have a higher priority in receiving cardiac surgery services than non-urgent patients.

Since seasonal weather is an important contributing factor influencing patient arrivals [21], the arrival rate will be adjusted seasonally in our simulation. For a relatively warm season (i.e., from May to October in a year in Ontario), the arrival rate is approximately 15% lower than that of a cold season (i.e., from January to April, and from November to December in a year in Ontario) according to the reported CCN data.

In accordance with the real-world monthly service utilization data from January 2005 to December 2006, we therefore set the same time

period, i.e., two years, to run simulations. At each time step, the simulation runs 100 times and generates an average value of patient numbers for each city/town.

5.2 Global-Level Patterns of Patient Arrivals and Wait Time

In this section, we examine the global-level service utilization regularities in our modeled cardiac surgery system. Fig. 7 shows the comparison between the distribution of patient arrival variations in the real world (represented as square dots in the figure) and that as obtained from the simulation (represented as star dots in the figure). From Fig. 7, we can note that the shape of the distribution relating real-world patient-arrival variations, by and large, has been reproduced by our simulation. The mean value and the standard deviation (SD) of real-world patient-arrival variations are 0.0004 and 0.226, respectively, while those of simulated patient-arrival variations are -0.015 and 0.243, respectively. The relative entropy or the Kullback-Leibler (KL) divergence (which is a measure of the difference between two probability distributions) [10] of the statistical distribution of simulated patient-arrival variations from that of real-world patient-arrival variations is 0.1398. The small value of KL divergence implies that the distribution of patient-arrival variations as obtained from the simulation may be approximate to that from the real world.

Fig. 8 presents the statistical distribution of absolute variations of median wait time as obtained from our simulation. From Fig. 8, we can note that the absolute variations of median wait time exhibit a power-law distribution ($p < 0.0001$), indicating that a self-organized regularity has emerged.

Fig. 9 compares the statistical distribution of absolute variations of median wait time as obtained from the simulation to that from the real world. The Kullback-Leibler (KL) divergence of the distribution of simulated absolute wait-time variations (represented as star dots in the figure) from that of real-world absolute wait-time variations (represented as square dots in the figure) is 0.1227. The small value of KL divergence implies that the two distributions are similar to a certain extent.

5.3 Discussion

Based on our AOC-CSS model and simulation-based experiments, we are able to characterize the self-organized regularities as observed in the real-world complex cardiac surgery system. This is partially due to the local feedback loop between patient arrivals and hospital wait time as shown in Fig. 4.

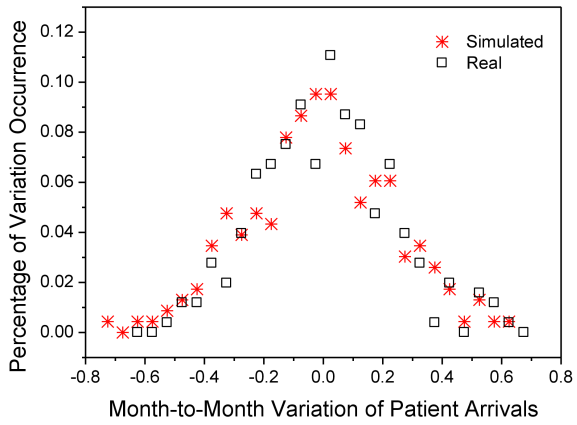


Fig. 7. Distributions of simulated and real-world arrival variations in cardiac surgery services.

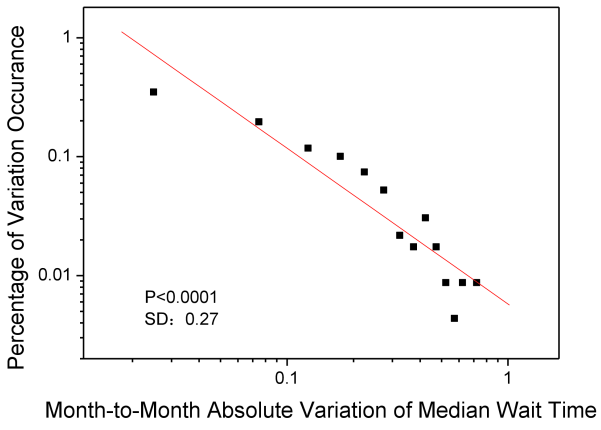


Fig. 8. The distribution of absolute wait-time variations in cardiac surgery services as obtained from the simulation. The distribution follows a power law with power of -1.31 ($p < 0.0001$).

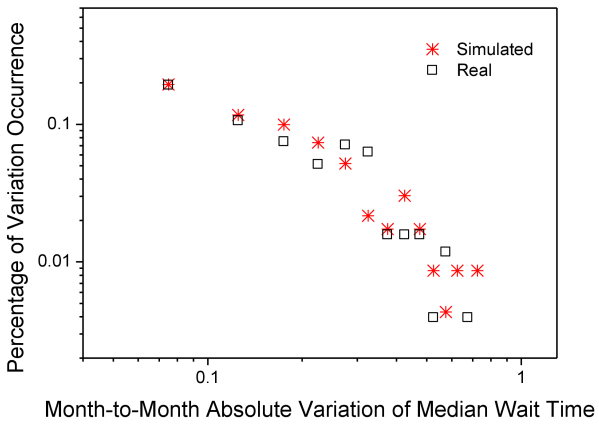


Fig. 9. Distributions of simulated and real-world wait-time variations in cardiac surgery services.

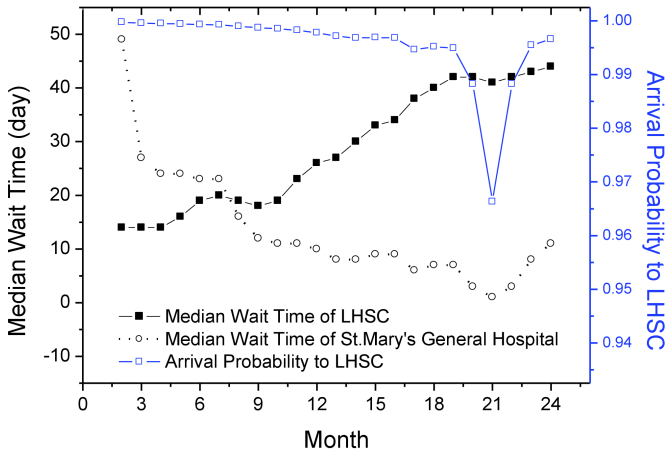


Fig. 10. The dynamically-changing preference of patients residing in the city of London to London Health Sciences Centre (LHSC) along with wait time changes in LHSC and St. Mary's general hospital over time.

Let's take the city of London, Ontario, as an example to illustrate the self-organizing process at an individual level. For patients residing in

this city, there exist two nearest hospitals, i.e., London Health Sciences Centre (LHSC) and St. Mary's general hospital that offer cardiac surgery services. Before 2005, the wait time in St. Mary's general hospital was longer than that in LHSC. However, in 2005, the wait time in St. Mary's general hospital was notably reduced (as shown in the first 12 months in Fig. 10) as two new operating suits for cardiac surgery became operational [2]. As a result, as shown in Fig. 10, during the initial time rounds (i.e., months) in our simulation, almost all patients living in London prefer LHSC, because (1) the driving distance from London to LHSC is shorter than that to St. Mary's general hospital since LHSC is exactly located in the city of London, (2) LHSC also has more physicians than St. Mary's general hospital, and (3) the wait time in LHSC is shorter than that in St. Mary's general hospital. As nearly all the patients in London come to LHSC, the wait time in LHSC will increase while that in St. Mary's general hospital will decrease afterwards. As a consequence, the preference of subsequent patients to LHSC will decrease and thus some patients may choose St. Mary's general hospital (from month 1 to month 21 in Fig. 10), which will in turn result in the increase of wait time in St. Mary's general hospital and the decrease of wait time in LHSC (from month 21 to month 24 in Fig. 10). This self-regulating process is initiated by autonomous patient/GP entities according to their hospital selection behavioral rules, which incorporates the feedback loop (between the wait time and the hospital selection behavior) that may account for the observed global-level self-organized patterns.

6 Conclusion

In this paper, we have presented an AOC-based modeling and simulation approach to characterizing self-organized regularities in a real-world cardiac surgery system. In particular, we have described three types of entities, i.e., patient, GP, and hospital, as well as the environment that they reside in and access information from. Based on the identified major impact factors of distance, hospital resourcefulness, wait time, as well as their interaction relationships and the local feedback loop, we have derived three types of behavioral rules for patient-GP mutual decisions on hospital selection. Drawing on the spatial pattern of real-world patient flows, we have discovered the best-fit behavioral rule for patient and GPs residing in each LHIN. Intuitively, the identified best-fit behavioral rule for an LHIN is, to a certain extent, associated with its accessibility to cardiac surgery services. In general, patients and GPs tend to exhibit D-rule behavior in selecting hospitals (i.e., taking into account only the distance factor) when they reside in LHINs with few hospital choices

within a certain distance. On the other hand, if patients and GPs live in LHINs with good service accessibility, they will be inclined to exhibit DH-rule behavior (i.e., considering the factors of distance and hospital resourcefulness concurrently), and even DHW-rule behavior (i.e., involving the factors of distance, hospital resourcefulness, and wait time), in selecting hospitals.

Through simulation-based experiments, we have observed that the constructed white-box AOC-CSS model produces global-level regularities similar to those found in the real-world cardiac surgery system. This indicates that the patient-GP mutual hospital selection behavior and its interrelationship with hospital wait time may account for the self-regulating service utilization. It also reveals that the AOC-based modeling approach can effectively provide a means for explaining the self-organized regularities and investigating emergent phenomena in complex systems. In our future study, it would be promising to study the applications of the presented approach to other real-world complex healthcare systems, so as to better understand how global-level regularities emerge from individuals' collective behavior and their closely coupled interactions.

References

[1] Advanced adult cardiac care patient access management process: Better access to quality cardiac care. http://www.ccn.on.ca/ccn_public/uploadfiles/files/Patient%20Access%20Mgmnt%20diagram.pdf.

[2] Cardiac care network of ontario: Cardiac surgery in ontario: Ensuring continued excellence and leadership in patient care. http://www.ccn.on.ca/ccn_public/uploadfiles/files/Surgical_Report_October31_2006_BOARD.pdf.

[3] Cardiac care network of ontario: Patient, physician and ontario household survey reports: Executive summaries. http://www.ccn.on.ca/ccnpublic/UploadFiles/files/CCNSurveyExecSum200508.pdf.

[4] Google maps. http://maps.google.com.

[5] Ontario freezing doctor pay to invest in more community care for families and seniors. http://www.health.gov.on.ca/en/news/release/2012/may/nr_20120507_1.aspx.

[6] Ontario's local health integration networks. http://www.lhins.on.ca/home.aspx.

[7] Statistics canada: 2006 census database. http://estat.statcan.gc.ca/cgi-win/CNSMCGI.EXE?Lang=E&C91SubDir=ESTAT&DBSelect=FSA06IN.

[8] D.A. Alter, E.A. Cohen, X. Wang, K.W. Glasgow, P.M. Slaughter, and J.V. Tu. Cardiac procedures. In J.V. Tu, S.P. Pinfold, P. McColgan, and A. Laupacis, editors, *Access to Health Services in Ontario*, pages 55–95. 2006.

[9] C. Brecht, D. Erik, and J. Belien. Operating room planning and scheduling: A literature review. *Eur. J. Oper. Res.*, 201(3), 2010.
[10] K.P. Burnham and D.R. Anderson. *Model Selection and Multi-Model Inference: A Practical Information-Theoretic Approach.* Springer-Verlag, New York, 2002.
[11] B. Chan. Access to physician services and patterns of practice. In D. Naylor and P. Slaughter, editors, *Cardiovascular Health and Services in Ontario: An ICES Atlas Toronto*, pages 301–318. Institute for Clinical Evaluative Sciences, 1999.
[12] S.L. Grace, S.G. Witte, J. Brual, N. Suskin, L. Higginson, D. Alter, and D.E. Stewart. Contribution of patient and physician factors to cardiac rehabilitation referral: A prospective multilevel study. *Nat. Clin. Pract. Cardiovasc. Med.*, 5(10), 2008.
[13] K.S. Kincben, L.A. Cooper, D. Levine, N.Y. Wang, and N.R. Powe. Referral of patients to specialists: Factors affecting choice of specialist by primary care physicians. *Ann. Fam. Med.*, 2(3), 2004.
[14] L. Kleinrock. *Queueing Systems: Volume I – Theory.* Wiley Interscience, New York, 1975.
[15] S.F. Lakha, B. Yegneswaran, J.C. Furlan, V. Legnini, K. Nicholson, and A. Mailis-Gagnon. Referring patients with chronic noncancer pain to pain clinics. *Can. Fam. Physician.*, 57(3), 2011.
[16] G.R. Langley, S. Minkin, and J.E. Till. Regional variation in nonmedical factors affecting family physicians' decisions about referral for consultation. *CMAJ*, 157, 1997.
[17] L.A. Lipsitz. Understanding health care as a complex system: The foundation for unintended consequences. *JAMA*, 308(3), 2012.
[18] J. Liu. Autonomy-oriented computing (aoc): The nature and implications of a paradigm for self-organized computing. In *The Fourth International Conference on Natural Computation*, pages 3–11, 2008.
[19] J. Liu, X. Jin, and K.C. Tsui. *Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling.* Springer, 2004.
[20] J. Liu, L. Tao, and B. Xiao. Discovering the impact of preceding units' characteristics on the wait time of cardiac surgery unit from statistic data. *PLoS One*, 6(7), 2011.
[21] Mensah G. Mackay, J. *The Atlas of Heart Disease and Strokes.* World Health Organization, Geneva, 2004.
[22] Wakefield P.A., G.E. Randall, and G.M. Fiala. Competing for referrals for cardiac diagnostic tests: What do family physicians really want? *JMIRS*, 43(3), 2012.
[23] P.E. Plsek and T. Greenhalgh. The challenge of complexity in health care. *BMJ*, 323, 2001.
[24] W.B. Rouse. Health care as a complex adaptive system: Implications for design and management. *The Bridge*, 38(1), 2008.
[25] J.E. Seidel, C.A. Beck, G. Pocobelli, J.B. Lemaire, J.M. Bugar, H. Quan, and W.A. Ghali. Location of residence associated with the likelihood of patient visit to the preoperative assessment clinic. *BMC Health Serv. Res.*, 6(13), 2006.

[26] D.P. Smethurst and H.C. Williams. Self-regulation in hospital waiting lists. *J. R. Soc. Med.*, 95, 2002.

[27] L. Tao and J. Liu. An integrated analytical method for uncovering complex causal relationships in healthcare: A case study. In *ACM SIGKDD Workshop on Health Informatics*, 2012.

[28] L. Tao, J. Liu, and B. Xiao. Effects of neighborhood geodemographic profiles on healthcare service wait time: A case study on cardiac care. *in review: BMC Health Serv. Res.*, 2013.

[29] H.C. Wijeysundera, T.A. Stukel, A. Chong, M.K. Natarajan, and D.A. Alter. Impact of clinical urgency, physician supply and procedural capacity on regional variations in wait times for coronary angiography. *BMC Health Serv. Res.*, 10(5), 2010.