Modeling multi-cellular behavior in epidermal tissue homeostasis via finite state machines in multi-agent systems

Thomas Süttelin1, Simone Huber2, Hartmut Dickhaus1 and Niels Grabe1,∗

1Medical Informatics Department, Institute of Medical Biometry and Informatics, University Hospital Heidelberg, Im Neuenheimer Feld 267, 69120 Heidelberg and 2Department of Informatics, Heilbronn University, Max-Planck-Str. 39, 74081 Heilbronn, Germany

Received on December 2, 2008; revised on May 18, 2009; accepted on June 9, 2009
Advance Access publication June 17, 2009
Associate Editor: Troy Ideker

ABSTRACT

Motivation: For the efficient application of multi-agent systems to spatial and functional modeling of tissues flexible and intuitive modeling tools are needed, which allow the graphical specification of cellular behavior in a tissue context without presuming specialized programming skills.

Results: We developed a graphical modeling system for multi-agent based simulation of tissue homeostasis. An editor allows the intuitive and hierarchically structured specification of cellular behavior. The models are then automatically compiled into highly efficient source code and dynamically linked to an interactive graphical simulation environment. The system allows the quantitative analysis of the morphological and functional tissue properties emerging from the cell behavioral model. We demonstrate the relevance of the approach using a recently published model of epidermal homeostasis as well as a series of cell-cycle models.

Availability: The complete software is available in binary executables for MS-Windows and Linux at tiga.uni-hd.de
Contact: niels.grabe@bioquant.uni-heidelberg.de

1 INTRODUCTION

The study of epithelial homeostasis offers an excellent opportunity for developing systems biological models which realize the link from the molecular to the cellular and further on to the tissue level (Galle et al., 2005; Walker et al., 2004). Previously, we demonstrated an initial simulation of tissue homeostasis and its application to psoriasis using a multi-agent system (Grabe and Neuber, 2005, 2007). Each of these agents realizes the functional and spatial behavior of an individual cell. We observed that the direct use of algorithms for describing cellular behavior enables very fast multi-cellular simulations. Furthermore, we learned that multi-agent systems provide an ideal conceptual approach for this purpose. Multi-agent systems experience an increasing popularity in bioinformatics (Merelli et al., 2004, 2007; Walker et al., 2004; Webb and White, 2006) as they are directly intended for modeling emergent behavior of interacting entities. Unfortunately, the direct coding of parameterized algorithms in software allows the non-expert user only to affect a small subset of the model parameters. This appears in particular problematic for highly complex tissue models. Therefore, we set out to find a more flexible approach for modeling cellular behavior for multi-agent simulations.

Graphical modeling tools like CellDesigner (Kitano et al., 2005) have become quite popular in the field of systems biology. CellDesigner process diagrams represent the state transitions of individual molecules like p53 or EGFR. Here, we carry on the intuitive idea of using process diagrams and deploy the approach to describe the behavior of whole cells. To this end, we adapted the theoretical concept of finite state machines (FSM). Indeed, one of the first applications of Turing’s theoretical automata theory which introduced the concept of FSMs (Turing, 1936) was the neuronal cell (McCulloch and Pitts, 1943). It was subsequently refined in the 1960s (Minsky, 1967). FSMs have been used in the context of numerous publications concerning cellular automata (Alarcón et al., 2003; Cown, 2004). Unfortunately, the grid structure of cellular automata renders them unsuitable for spatially dynamic simulations of epithelial homeostasis. We therefore adapt the theoretical concept of FSMs to our problem. In this respect, we suggest to use process diagrams as an abstract description of cellular behavior instead of directly coded algorithms. Interpreted in the context of FSMs, these graphical models describe conditional transitions between individual states of the cell. General cellular states may comprise e.g. proliferation, differentiation, migration and apoptosis. To each of these general states a hierarchically structured set of sub-states is conceivable, each corresponding to a different cellular function like, for example, changes in polarity or rearrangements of adhesion complexes.

When combined with graphically specified FSMs, we suppose multi-agent systems to become an ideal approach for specifying multi-cellular models. When specified in the form of FSM, graphical depictions of cellular behavior can be translated relatively straightforward into executable code. Furthermore, during such an automatic code generation, automatic code optimization can take place, producing high-performance minimal code. Especially, when aiming at the simulation of tissue, composed of hundreds or thousands of cells, the generation of minimal code will prove to be of utmost importance.

Other general FSM modeling tools are available like e.g. Matlab/Stateflow and HyVisual (Lee and Zheng, 2005). However, the requirement for a tight integration of graphical modeling, highly specific code generation and incorporation of the generated code
into multi-agent simulations of tissues rendered the use of presently available software practically impossible.

Starting from these conceptual considerations, we developed a graphical modeling system (GMS) for the intuitive modeling and efficient simulation of individual behavior of cells in a tissue context. The specified models are hierarchically structured thus allowing the specification of complex cellular functions. At the push of a button, the models are automatically translated into highly efficient minimal executable code being then automatically and dynamically bound to a multi-agent simulation environment which executes the model in discrete time steps and in a semi-parallel manner for each cell. Thereby cells interact via transport processes and signaling. We demonstrate the efficiency of the resulting multi-cellular simulation at the example of our previously presented model of keratinocyte differentiation as well as various cell cycle models.

2 METHODS

2.1 Graphical modeling system

We developed our GMS on the basis of the open source software platform Eclipse that was originally developed by IBM. Eclipse in turn is based on the Open Services Gateway initiative (OSGi) Framework Equinox that allows Eclipse to be dynamically extended by plug-ins. We realized the GMS as a set of plug-ins for Eclipse in the programming language Java. We used the Java Development Tools (JDT), the Eclipse Modeling Framework (EMF) and the Graphical Editing Framework (GEF). The GMS is subdivided in three independent components: (i) Variable-Sheet Editor (described in Section 3.1), (ii) Graphical Model Editor (GME) (Section 3.2) and (iii) Function Library (Section 3.3).

Following the paradigm of model-driven software development, a meta-model for each component was developed using the Unified Modeling Language (UML). These meta-models are instances of the meta-meta-model Ecore provided by the EMF. The EMF in turn is able to derive source code out of the XMI files using the technique ‘Templates and Filtering’ (Stahl et al., 2006). The source code generated from the model is translated into Java byte code using Sun Microsystems’ java compiler. Automatic validation of graphical models is performed by a parser using the open source Java Compiler Compiler (JavaCC). JavaCC is a parser generator that allows to generate a LL(k) parser on the basis of a given grammar (Copeland, 2007).

A user-specified graphical model controls the behavior of each agent in the multi-agent simulation. For executing the compiled models we developed a multi-agent based graphical simulation environment (GESE) which is built on the Georgia Mason University’s multi-agent simulation framework MASON (Multi Agent Simulation Of Neighborhoods) (Luke et al., 2005).

3 SYSTEM

3.1 Graphical model editor

We denote a graphical ‘cell behavioral model’ (CBM) as an abstract description of a FSM. The CBM formalizes all possible forms of behavior of a cell as a process graph. CBMs are specified using the GME. They are composed of the available predefined model elements depicted in Figure 1. The available model elements comprise:

- **Edge**: connects two model elements using their input port and their output port, respectively. Input ports and output ports are depicted in Figure 1 as small triangles.
- **State**: represents a state element in the model. Each state has one input port and one output port.
- **Connection point**: in case multiple transitions shall lead to a single common model element, a connection point is used to join the incoming edges. Therefore, a connection point always has just one output port.
- **Submodel**: submodels allow to implement a hierarchical structure in a CBM. Technically, the overall model is thereby physically distributed over different model files with unambiguous file names. Semantically, each file realizes a certain self-contained functional aspect of the overall model. The number of submodels is not limited.
- **Transition point**: in case multiple transitions shall lead to a single common model element, a connection point has to be used to join the incoming edges. Therefore, a connection point always has just one output port.
- **Comment**: textual descriptions or explanations can be added as a comment anywhere in the model.

Using these elements complex CBMs can be built. In each model the user denotes exactly one model element as the starting and one model element as the end point.

3.2 Variable-sheet editor

The Variable-Sheet Editor allows the intuitive creation, definition and modification of all variables and constants of the CBM which are composed of the available predefined model elements as described above.
are stored in the variable-sheet file. Variables or constants can be interactively dragged from the Variable-Sheet Editor and dropped into a certain model element in the GME where needed. All variables and constants of a CBM belong to one of the four categories:

- **Cell type:** each cell occurring in the simulation is assigned an abstract ‘cell type’ in form of an unambiguous string constant. For example, keratinocytes and fibroblasts are different cell types. However, multiple differentiation stages of a cell may also be introduced as distinct cell types.

- **Constants:** In this category, constant values of a CBM are defined which are of general use. Each constant consists of a name and a value constrained by a data type to be selected.

- **Global tissue property:** These variables specify global properties of the whole tissue. Contrary to ‘constants’, the value of a global tissue property can be modified interactively prior to simulation but also at runtime in the graphical SE. All cells of the multi-cellular simulation adhere to the same values of the specified global tissue properties.

- **Cell property:** cell properties comprise the variables of the CBM whose values are to be modified in every ‘state’ during the model simulation. At run-time each cell occupies these variables with individual values, thus realizing a personal state for each cell. Therefore, the state of an individual is defined by the values of its ‘cell properties’—variables at a particular point in time.

For each ‘global tissue property’ and ‘cell property’ a name, a data type and a scientific unit have to be defined. A range of typical data types is provided. For a ‘cell property’, these attributes are complemented by a maximum, a minimum and a default value. During the specification of graphical models, the value of a variable or constant can be addressed by the respective name as defined in the variable-sheet file. A neighboring cell’s property can be addressed by adding the prefix _n_ to the respective cell property’s name. This triggers a loop which is executed over all direct neighbors. Likewise, the allowed maximum and minimum value of a cell property can be accessed by adding the suffix _max_ and _min_, respectively.

### 3.3 Function library

The ‘Function Library’ provides a set of predefined functions for formulating mathematical expressions as well as specific functions for initiating cell proliferation and interaction with the cell’s neighborhood respectively. The function calls can be incorporated into ‘action strings’ which are specified by the user in the model element ‘state’. Power (Pow), square root (Sqrt), sine (Sin) and cosine (Cos) are the currently available mathematical functions. Additionally, the mathematical constants \( \pi \) (\( \pi \)) and \( e \) (\( e \)) are provided.

The functions \( \text{Send}(\text{var, val, toNeighbor}) \) and \( \text{Receive}(\text{var, val, fromNeighbor}) \) allow the specification of a certain fraction of material and intercellular signaling molecules to be transported to and from neighboring cells or the intercellular space respectively in a discrete amount of time. Thus, intercellular communications modeled through these functions are executed at each discrete time step. It is up to the modeler to interpret a discrete time step as a specific period of time. Equilibrating processes can be modeled when considering a sufficiently large number of time steps. Both functions implement a heuristic intercellular communications approach. Therefore, intercellular communication is modeled indirectly without addressing directly a respective neighboring cell. This offers the possibility to abstract away from the exact number of neighboring cells as well as from their spatial distribution. In detail, the abbreviation \( \text{var} \) denotes the respective ‘cell property’ usually representing a molecule to be exchanged. \( \text{val} \) denotes the amount of units to be sent or received, whereas \( \text{toNeighbor} \text{ or fromNeighbor} \) is either of the value true (\( T \)) or false (\( F \)). In case of false, a ‘cell property’ is just altered but there is no respective transfer to neighboring cells. For example \( \text{Send}([\text{var} \rightarrow \text{val}, \text{toNeighbor}]) \) initiates the transport of 10% of the intracellular Ca\(^{2+} \) ions to the intercellular space.

Formally, the function \( \text{Send}([\text{var} \rightarrow \text{val}, \text{toNeighbor}]) \) works as follows. The variable \( \text{var} \) corresponds to the name of a ‘cell property’ defined in the Variable-Sheet Editor. Initially, the maximal amount of the molecule or material specified in \( \text{val} \) is determined which can be sent without \( \text{var} \) falling below the variable-sheet specified minimal value. In case of transport to the neighboring cells (i.e. \( \text{toNeighbor} \text{ is true} \) ) the resulting amount is equally distributed among the neighbors of the actual cell. For each neighboring cell it is checked that it does not exceed the given minimal or maximal values. The \( \text{Receive} \) function works analogously to the \( \text{Send} \) algorithm with inverse checks. Both implemented algorithms allow an undirected as well as a directed propagation of material and signaling molecules among neighboring cells and thus through the whole tissue. For a directed propagation of material the graphical modeling element \( \text{transition condition} \) has to be used for defining respective preconditions. The equilibration of a particular signaling molecule or material among all neighboring cells is achieved via the iterative execution of the whole model over a sufficiently large number of discrete simulation time steps.

Finally, the Function Library provides the function \( \text{NewCell}(\text{var1}=\text{val1}, \text{var2}=\text{val2}, \ldots) \) which performs the proliferation of a cell. Computationally, this means that a new cell is instantiated. The initial values for the new cell’s properties are to be specified in between the parentheses.

### 3.4 Model validation

Automatic translation of the graphically specified models into executable source-code requires structurally validated models. Our GMS performs this for ensuring the following:

- **Completeness:** A model is complete if all output ports of all model elements are connected to an input port of another model element and every state element comprises at least one action string. Furthermore, every transition condition is to be specified by a Boolean expression and every transition choice must comprise a choice expression. Lastly, each model file needs exactly one model element marked as the start point and one model element marked as the end point.

- **Uniqueness:** A variable or constant is not unique if the same name is used by another variable or constant. This is checked during creation of a new variable or constant in the variable-sheet. Alternative names are suggested if a variable or constant name is already in use.

- **Syntactical correctness:** All action strings, Boolean expressions or choice expressions of any graphical element are checked for basal syntactical correctness.
3.5 Code generation and model simulation

The validated graphical models can be transformed into executable code. As a result of the transformation process we obtain a code archive which is stored at the user’s desired location.

For performing the simulation the resulting code archive is loaded into a newly developed SE depicted in Figure 2b. Every cell of the simulation is represented by an independent agent who is embedded in a continuous toroidal spatial environment. This spatial environment does not possess a predefined grid structure as it is the case for cellular automata. The simulation environment has been specifically designed for the simulation of human epidermis. A starting set of cells (i.e. agents) is positioned on a basal lamina with idealized rete ridges. Each agent executes an instance of the graphical CBMs generated and compiled code. Internally, the SE provides the CBM with an array containing all neighboring agents as an input parameter. In turn, the CBM returns an array containing all newborn cells which have to be included as new agents in the SE. The simulation is executed using a simplified biomechanical model which was described earlier (Grabe and Neuber, 2005, 2007).

4 RESULTS

4.1 Keratinocyte model

We set out to demonstrate the suitability of the here proposed approach for simulating epidermal homeostasis. We therefore re-implemented the previous algorithmic simulation of the epidermal homeostasis described earlier (Grabe and Neuber, 2005, 2007) using our new GMS. This allowed us to principally assess whether our previous algorithmic approach of modeling can be realized using the graphical FSM-based modeling approach presented here. We were able to graphically rebuild the former hard coded model without limitations. We compared the performance of the hard coded and of the automatically generated source code for 10,000 simulation steps. The average execution time for one simulation step is 1.73 ms in the originally hard coded simulation and 2.16 ms in the simulation using the dynamically loaded generated source code while producing the same simulation results. Thus, the automatically generated source code yields comparable performance against the original hard-coded version.

Figure 3 shows an example how complex intercellular transport or communication processes can be implemented relatively easy. The depicted submodel represents a part of the re-implemented keratinocyte CBM. It describes the upwards transport of molecules in the tissue with the transepidermal water flux towards the stratum corneum. Initially it is checked whether there are neighboring cells (step 1). The next condition (step 2) limits the number of molecules transported between two cells in one simulation step to 1 unit. The prefix \( n_\) induces a loop around the following graphical models which is conditionally executed for all neighboring cells if the according Boolean expression is fulfilled. This is the case if the respective neighboring cell did not already exchange molecules with one of its neighbors in the current simulation step. In step 3 the cell receives a certain percentage of each neighbor’s Ca\(^{2+}\) ions. This percentage is defined in the global tissue property \( \text{epidermalDiffusion} \). The position of each neighboring cell is checked in step 4. If a neighboring cell’s position is below the position of the current cell, Ca\(^{2+}\) and lamellar bodies are requested from the neighbor according to the value specified in the global tissue property \( \text{epidermalWaterflux} \).

4.2 Cell-cycle model

To illustrate the principal difference between state-based modeling and modeling using ordinary differential equations (ODEs), we created three different implementations for a well-known cell-cycle
According to the BioModels database (Le Novere et al., 2006)—by using our GMS (Fig. 4)—we entered the two ODEs first in the form of Euler-Cauchy (cellcycle_tyson_euler) and then according to Runge-Kutta in fourth order (cellcycle_tyson_range_kutta). As a third form of the cell cycle we implemented the linear mitogen model (cellcycle_epiSim) which triggers mitosis in a sawtooth wave form in dependence of time (Tyson and Kauffman, 1975). During runtime of the simulation the user can switch between the three different forms (cellcycleMode). The last implementation is the form which was used in the initial epidermal simulation (see above). Elements (1) and (2) trigger mitosis. For modeling delayed mitosis due to space constraints the mitotic state can be preserved (3).

The results are depicted in Figure 5 accordingly. Both ODE realizations (Euler-Cauchy and Runge Kutta) generate the characteristic spontaneous oscillations (Fig. 5b) which are on a higher level of abstraction equivalent to the linear mitogen model (Fig. 5a). The phase diagram shows the corresponding limit-cycle of the ODE system (Fig. 5c). In ‘excitable switch’ state the ODE system is responsive to sufficiently large perturbations (Fig. 5d) but not to smaller ones (Fig. 5d and e).

For illustration we further implemented a simple five-phase model of the cell cycle (Fig. 4b). In G0 state the cell cycle rests while otherwise cycling through G1, S, G2, M. Time constants have been scaled to map the estimated 60 h cell cycling time. Checkpoints are omitted for brevity but could be easily added. During S-phase DNA content linearly doubles. The time-averaged distribution of DNA content shows the characteristic profile well known from FACS analysis with the large fraction of cells in G1, a medium fraction in M-, and a small fraction of actively DNA synthesizing cells in S-phase (Fig. 5f).

5 DISCUSSION

We previously demonstrated the general feasibility of algorithmic coding of cellular behavior in multi-agent systems for simulating epidermal homeostasis and its perturbations (Grabe and Neuber, 2005, 2007). We now developed a novel system for the graphical modeling and simulation of cellular behavior in a multi-cellular, tissue-like environment and applied it to the previously presented (hard-coded) model of epidermal homeostasis. Particularly, this example shows how material exchange and cell–cell communication can be efficiently modeled in a multi-cellular ensemble.

To further illustrate the principles and advantages of state-based modeling of cell behavior we implemented different variants of the original cell-cycle model of Tyson. We chose this model as it is not only an example of a mitotic oscillator but also a modified version of the famous Brusselator which as a reaction-diffusion system is a model of chemical oscillations in general. The examples show that our multi-agent system is able to generate the complex patterns of reaction diffusion systems already in single cells. In the same time it shows that hybrid models containing ODE elements can be nicely integrated with the state-base modeling approach using our GMS. The results also point out how signaling networks on the timescale of hours can be easily implemented using our graphical interface. A further illustration of the state-based approach is the five-phase model of the cell cycle which demonstrates how a biological model can be conveniently implemented using the proposed graphical state based notation.

The given examples show how the user is provided with different abstraction levels he can freely choose to work on depending on his needs: during sole biochemical modeling, the according reactions are modeled explicitly, while the states of the system emerge implicitly by simulation. In behavioral modeling, the states are modeled explicitly, while the tissue behavior emerges implicitly. Therefore, it is important to note the main objective of our work is not to compete with ODE modeling, but to provide a simulation environment on a higher level of abstraction than the pure biochemical reaction modeling, thus allowing the direct specification of cellular behavior.

We here graphically modeled cellular behavior as a process diagram of a FSM. The cell models are automatically translated into high performance source code and dynamically linked to a multi-agent graphical simulation environment where the source code is then executed for each agent. By proliferation and interaction of the cells (i.e. the agents) the tissue structure then emerges according to the specified CBM. The term ‘cellular behavior’ in our view explicitly comprises cellular decision making but furthermore includes all cellular aspects necessary for modeling tissues like cellular interactions and cytoskeletal remodeling. Tissue in this respect is produced by the emergent behavior of individual cells.
Fig. 4. Implementation of the two-variable cell-cycle model (Tyson, 1991) in the GME. (a) The differential equations were implemented using Euler, Runge-Kutta or saw-tooth function. Mitosis is triggered by thresholding. In delayed mitosis, the mitotic state of the cell is preserved. (b) Exemplary state-based implementation of a 5 phase (G1, S, G2, M, G0) cell-cycle model. DNA content linearly increases during S-phase.

FSMs allow for the modeling of cell behavior and decision processes by conditional transitions between states. It is important to understand that FSMs use the term ‘state’ in a more general way than it is commonly used in the modeling of molecular networks. The four main states of a cell are ‘proliferation’, ‘differentiation’, ‘apoptosis’ and ‘migration’. These main states are accompanied by a multitude of sub-states which imply certain cellular functions (e.g. change of polarity, decomposition of adhesion complexes) or characteristics (e.g. growth arrest). Transitions between states are realized biochemically by e.g. cellular signaling and effector processing accordingly (Miller-Jensen et al., 2007). On the contrary, the states of FSMs are of a more abstract, conceptual nature: in theory they comprise all possible occupations of a cell’s ‘variables’. In practical modeling a user will only specify those states of cellular behavior which are relevant to him. FSM thus allow modeling on a broad range of cellular abstraction levels, e.g. decision trees (Kharait et al., 2007).

The question is, whether FSMs are generally a suitable approach for multi-cellular modeling, or if cellular decision models should be based directly on mechanistic biochemical reaction networks. In our view both approaches are not mutually exclusive. The multi-agent approach itself does not prohibit modeling using ODEs or obtaining accurate quantitative time-resolved results. In fact, we graphically specified models were translated into source code numerically solving two ODEs by Euler-Cauchy and Runge-Kutta. The key concept of the multi-agent approach is that it simulates a cellular ensemble as a clocked system. The resolution of the time scale, which is chosen by the modeler, thereby determines whether the simulated system is of more continuous or of more discrete nature. As such the time-accuracy of the simulation lies in the hands of the user and is not a-priori constrained by the simulation environment.

For tissue simulation multi-time scale modeling is of central relevance. This could be achieved by hierarchical modeling (Asthagiri and Lauffenburger, 2000), which here could be implemented using submodels running with different clock times. A simpler solution in our view is to approximate the outcome of the submodels by states. For example: a decision tree is implemented as an abstraction of a fast intracellular network. In this sense, we offer a novel opportunity for complexity reduction: fast running decision processes like e.g. intracellular signal transduction are approximated by modeling their outcome. Thus not all details of the complex system have to be modeled to achieve accurate results at a higher level. This is in line with the ‘slaving principle’ that indicates that the cell’s long-term phenotypic response can be expressed in terms of its slowest evolving functional elements (Busch et al., 2008; Haken, 2004).

We studied the performance of our system using the key modeling concepts ‘state’, ‘condition’ and ‘submodel’ as well as regular intercellular communications. The results show a linear complexity of our implementation for up to 100 000 model elements (data not shown). This demonstrates the principal suitability of the approach for modeling even extremely complex multi-cellular ensembles.

The technical idea of the presented approach was the use of automatic code generation originating from the field of software engineering for translating graphically depicted decision models automatically into source code. This source code is then dynamically linked into a multi-cellular simulation environment. In this way, we provide a way for the computationally efficient realization of multi-cellular models. Technically, our system is based on the well-known multi-agent framework MASON. To our knowledge, our system is the first implementation coupling a flexible graphical modeling tool to the source code generation for a multi-agent framework. The GMS presented here was primarily designed for creating models for simulating epidermal homeostasis. We think, that independent of this matter of subject, our approach could also be applied to modeling other tissues. Particularly, as systems biology is currently
in a transition towards medically oriented applications, modeling of multi-cellular ensembles will become essential. Our results indicate that especially for this purpose, behavioral modeling of cells using FSMs in multi-agent systems could become a promising approach.

ACKNOWLEDGEMENTS

The manuscript is dedicated to the memory of Prof. Dr. Karsten Neuber for his visionary and passionate contributions to this project. We thank Dr. Michael Grabe for critical reading of the manuscript.

Funding: BMBF FORSYS-PARTNER and BMBF MEDSYS.

Conflict of Interest: none declared.

REFERENCES


