УДК 004.942, 004.434

Д. С. Мигинский [1, 2], **В. С. Тимонов** [1, 3]

[1] Институт систем информатики им. А. П. Ершова СО РАН
пр. Акад. Лаврентьева, 6, Новосибирск, 630090, Россия

[2] Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия

[3] Институт цитологии и генетики СО РАН
пр. Акад. Лаврентьева, 10, Новосибирск, 630090, Россия

E-mail: dmiginsky@gmail.com; vladimir.timonov@gmail.com

ПРИМЕНЕНИЕ СЕТЕВЫХ ОПИСАНИЙ ЭКОСИСТЕМ ДЛЯ АВТОМАТИЗИРОВАННОГО ПОСТРОЕНИЯ ИМИТАЦИОННЫХ МОДЕЛЕЙ

Рассматриваются вопросы автоматизации построения имитационных дискретно-событийных моделей экосистем на основе сетевых описаний. Необходимой предпосылкой для построения динамической модели является формализация понятийного аппарата. Вместе с тем отличительной чертой экосистем является вариативность понятийного аппарата, что создает сложности при разработке универсального инструмента моделирования, применимого для широкого класса экосистем. Частично проблема вариативности предметной области решается применением формальных онтологий, однако это решение применимо только для декларативного уровня, т. е. уровня сетевых описаний. В работе предложен способ построения проблемно-специфичного языка, предназначенного для дискретно-событийного моделирования и настраиваемого на предметную область с помощью онтологии.

Ключевые слова: имитационное моделирование экосистем, сетевые модели, формальные онтологии, проблемно-специфичные языки.

Введение

Живые системы на любом уровне своей организации обладают высокой степенью сложности, которая выражается в большом количестве структурных элементов и их типов, а также поведенческих отношений между ними. Эти особенности живых системы делают необходимым и крайне важным применение компьютерных методов анализа и моделирования в биологических и фармакологических исследованиях. Вместе с тем, попытки создания в достаточной степени универсальных программных инструментов сталкиваются с проблемой сложности формализации предметной области. Под формализацией понимается выделение четко определенного, достаточно компактного и вместе с тем эффективного для решения задач системного моделирования понятийного аппарата. Для ряда биологических объектов эта проблема в настоящее время неразрешима. Это делает неприменимыми традиционные подходы программной инженерии, когда программный инструмент достаточно жестко привязан к предметной области, определяя ее понятия в виде классов используемого языка программирования, таблиц базы данных или других аналогичных программных единиц.

Решением является представление модели данных в форме онтологии и использование метамодели вместо жестко заданной модели данных при разработке программных инструментов. Были предложены методы представления живых систем, основанные на сетевых (графовых) моделях. Элементы сети (дуги и вершины) типизированы элементами онтологии,

которая также является динамической структурой данных с точки зрения программной системы. При этом функциональность по изменению онтологий доступна пользователю.

Сетевые модели живых систем являются промежуточным результатом при проведении исследований и представляют собой ценность только при условии их использования для дальнейшего анализа и моделирования динамики. Предложена методика автоматизированной генерации дискретно-событийных моделей на основе сетевых, применимая для некоторого класса экосистем. Методика включает автоматическую генерацию существенной части динамической модели, задания ряда параметров модели (вычисляемых исходя из сетевой модели и онтологического описания) на декларативном уровне, а также описание поведения с помощью автоматически генерируемого на основе онтологического описания проблемно-специфичного языка.

Особенности компьютерного представления живых систем

Живые системы имеют несколько уровней организации: систему можно рассматривать и моделировать на уровне биохимических взаимодействий (молекулярно генетические системы, МГС), взаимодействия клеток (ткани, органы), организма (взаимодействие органов и подсистем организма – пищеварительной, нервной и т. д.), взаимодействия организмов и их сообществ (экосистемы). Наиболее изученным в плане формализации понятийного аппарата следует считать МГС, т. е. самый нижний уровень организации живого. Для МГС существует достаточно много баз данных [1] с жестко заданной структурой, отражающей достаточно устоявшиеся ключевые понятия МГС, такие как ген, белок и т. д. Эти базы постоянно актуализируются и используются в том числе и в прикладных фармакологических исследованиях, что свидетельствует об их эффективности. Это касается и архитектурных решений, в большинстве вполне традиционных, т. е. основанных на фиксированной модели данных, и не позволяющих ее менять без изменения программного кода. Вместе с тем следует отметить, что часто между такими базами бывают существенные расхождения на уровне семантики данных (например, понятия «белок»), что вызывает существенные проблемы при семантическом объединении и совместном использовании данных из разных источников [2].

В качестве основного метода компьютерного представления МГС используются сетевые (графовые) модели. Следует отметить, что в отличие от баз данных большинство современных программных инструментов для сетевого моделирования и анализа МГС [3; 4] не привязаны жестко к предметной области, позволяя пользователю до определенной степени вводить свои понятия и свойства, т. е. изменять онтологию. Как правило, для МГС нет необходимости полной модификации модели данных, так как базовую терминологию в значительной степени следует считать устоявшейся. Это несколько упрощает разработку программного инструмента, снимая ряд проблем, связанных с ПО, полностью основанном на метамодели [5].

Поднимаясь на уровень экосистем, т. е. самый «высокий» уровень организации живого проблема формализации предметной области становится более существенной. Представление экосистемы в общих терминах, таких как трофические и информационные взаимодействия, биотические и абиотические факторы, дает в основном описательную модель, не позволяющую применять к ней эффективные методы анализа и тем более моделировать динамику. Уточнение же терминологии ведет к тому, что для каждой экосистемы или небольшого класса экосистем требуется своя собственная онтология, практически не пересекающаяся с другими классами экосистем.

Вместе с тем следует отметить, что компьютерное представление экосистем не требует такого объема исходных данных, как представление МГС. Это связано с тем, что экосистемы редко достигают размера в сотни элементов / отношений, тогда как для МГС характерны размеры в сотни и тысячи [1]. Программный инструмент для моделирования экосистем, таким образом, становится качественно проще, так как отпадают ряд требований, присущих обычно промышленным приложениям (и также касающихся ПО для моделирования МГС [2; 5]):

[1] Вероятно, это отражает не столько специфику экосистем, сколько текущее состояние исследований экосистем в сравнении с МГС.

обязательное хранение данных в базе, гарантия целостности данных на всех уровнях, конкурентный многопользовательский доступ.

Задача сетевого моделирования экосистем

Задача, в частности, состоит в разработке универсального представления экосистем, которое бы позволяло:

- описывать формальный понятийный аппарат для различных экосистем, т. е. строить онтологии;
- строить описательные модели экосистем с различной степенью детализации;
- верифицировать модель экосистемы на основе онтологии;
- использовать данное представление в качестве основы для построения динамической модели.

За основу было выбрано сетевое представление данных, так как оно позволяет удовлетворить всем перечисленным требования. Действительно, элементы сетевой модели могут состоять в отношении типизации к элементам онтологии, причем тип онтологии (т. е. конкретная метамодель, лежащая в основе) заранее не фиксируется, что дает высокую гибкость при разработке ПО. Кроме того, онтология является инструментом верификации данных. Глубина возможной верификации в существенной степени зависит от принятой метамодели, а также качества проработки онтологии.

Опыт применения сетевых моделей для описания МГС, а также в других областях (схемотехника, бизнес-процессы и т. д.) показывает, что сетевую модель можно развивать эволюционно, т. е. постепенно, от «набросков» можно переходить к более детализированным моделям, обладающим не только качественными, но и количественными характеристиками. Также достаточно легко поддерживать версионность не только моделей, но и их элементов, причем, не выходя за рамки самой сетевой модели. Достаточно ввести дополнительные служебные типы отношений на уровне онтологии. Пример такого подхода продемонстрирован в Rational Unified Process [6], где язык UML может рассматриваться отчасти как онтология, отчасти как метамодель данных, а UML-модели – как сетевые модели. Для отслеживания происхождения элементов моделей используются соответствующие стереотипы отношения «зависимости», что позволяет восстановить, например, из каких требований или аналитических классов проистекает тот или иной программный модуль.

Возможность построения динамической модели на основе сетевой в общем случае является очень сложно проблемой. Однако если не говорить о полной универсальности и автоматизации этого процесса, то задача становится решаемой.

Программный инструмент для сетевого моделирования экосистем

Было разработано программное обеспечение, позволяющее пользователю описывать и создавать собственные онтологии и, на основе их, сетевые модели экосистемы (потенциально и других типов систем). Метамодель представления данных в упрощенном виде приведена на рис. 1.

Здесь связка метаклассов *Type*, *Property*, *Object*, *Value* представляет собой классическую объектную метамодель (классификация свойств для простоты на диаграмме не представлена). Остальные метаклассы представляют собой ее расширения. *Element* является экземпляром «второго уровня», представляя собой некоторый объект в определенном окружении / среде и характеризующийся дополнительными свойствами, имеющими смысл только в рамках моделируемой экосистемы. Например, в онтологии может быть задан тип (*Type*) «Species» (биологический вид), его экземпляром является «Mus Musculus» (мышь домовая), которая в рамках рассматриваемой экосистемы может образовывать популяцию, являющуюся уже экземпляром *Element*, и характеризуется дополнительно численностью популяции и рядом других параметров, имеющих смысл только в контексте экосистемы.

Метаклассы *Layer* (слой) и *Scheme* (схема) являются единицами представления и декомпозиции сетевых моделей. Слой непосредственно представляет собой часть сетевой модели

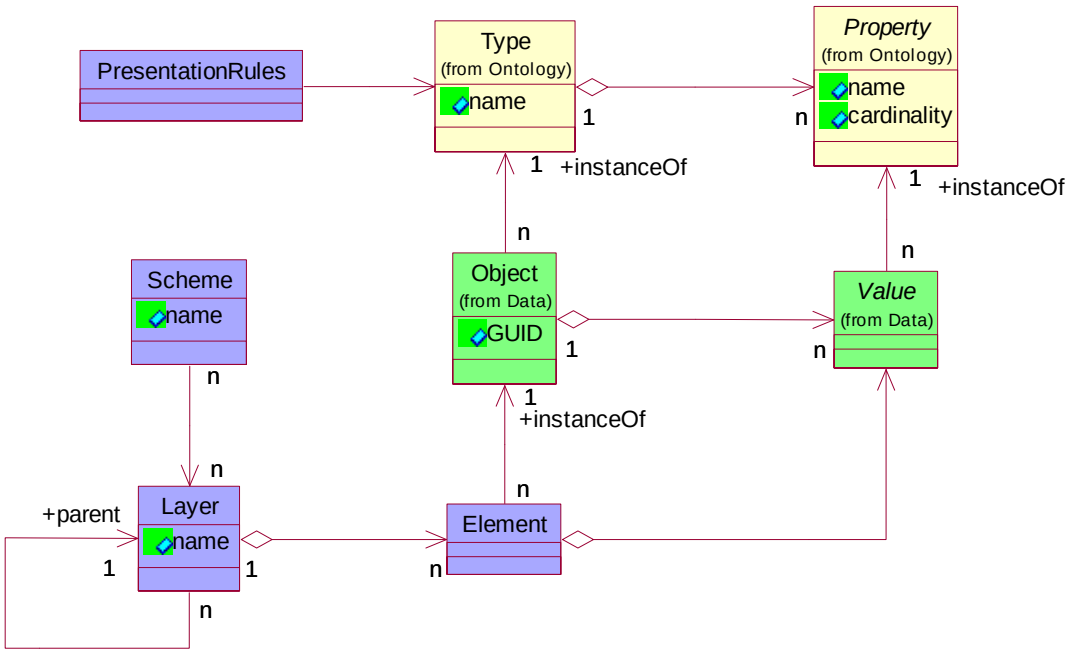

Рис. 1. Упрощенный вид метамодели данных

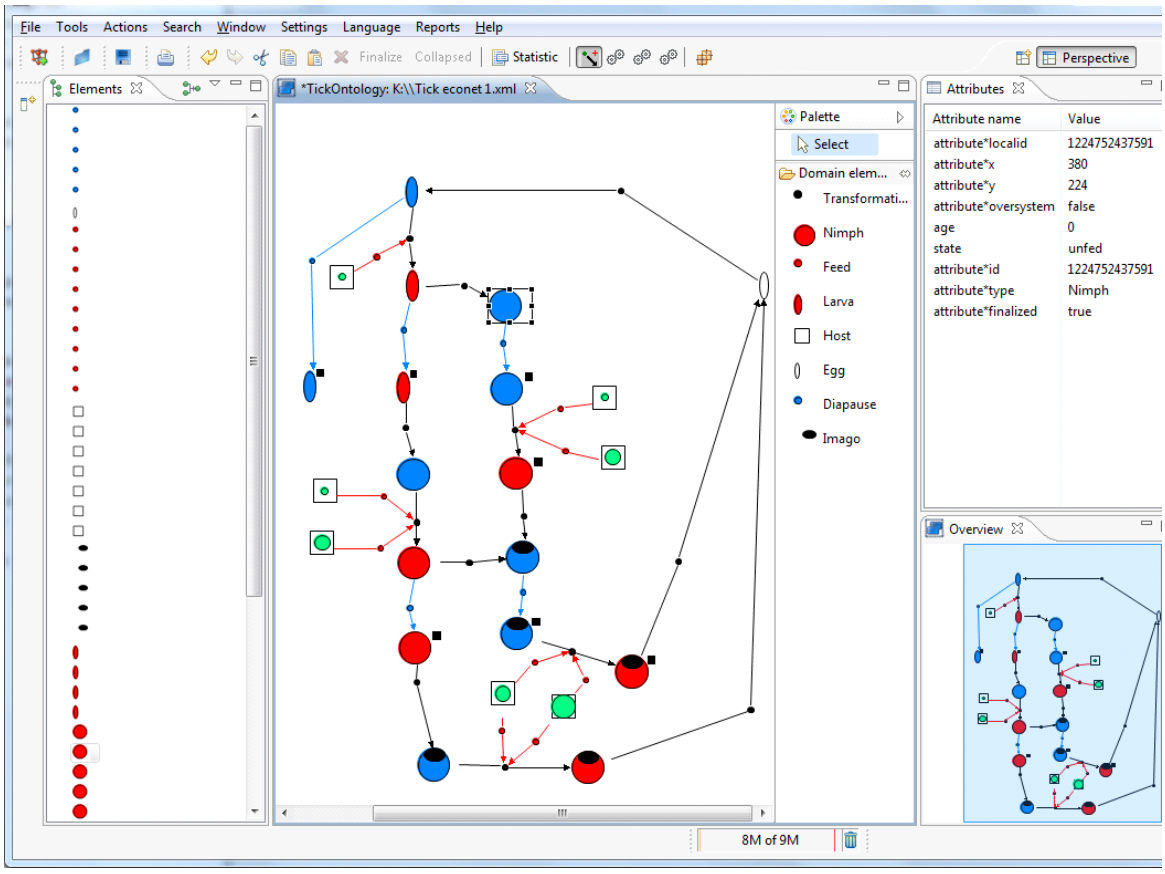

Рис. 2. Редактор экосетей

(как правило, некоторый поведенческий аспект изучаемой экосистемы), может быть зависимым от другого слоя, что позволяет его элементам ссылаться на элементы другого слоя. Зависимости между слоями образуют лес. На практике это необходимо для повторного использования одних и тех же частей сетевой модели. Например, сетевые модели, описывающие поведение одной и той же экосистемы в разные сезоны (зимой и летом), состоят, как правило, из одних и тех же объектов, но отношения между ними могут существенно отличаться. Таким образом, структуру экосистемы имеет смысл определить в едином базово слое, а поведение в различные сезоны в виде отдельных зависимых слоев, содержащих отношения между объектами. Схема является объединением некоторого набора слоев из одного дерева и, по сути, является единицей пользовательского представления и визуализации.

Метакласс *PresentationRules* отвечает за визуализацию отдельных элементов сетевой модели, что необходимо для наглядности (немаловажный фактор для больших сетевых моделей). Помимо того, что объекты различных типов целесообразно визуализировать различными пиктограммами, для некоторых типов удобно изменять вид пиктограммы в зависимости от свойств объекта, т. е. параметризовать пиктограмму.

Общий вид редактора сетевых моделей представлен на рис. 2: используется онтология для экосистемы иксодового клеща, представлен его жизненный. Все типы, свойства и пиктограммы заданы с помощью другого разработанного программного инструмента – редактора онтологий.

Имитационное моделирование экосистем

В качестве модельной системы для построения динамической модели была рассмотрена экосистема иксодового клеща (переносчика энцефалита, далее – клеща). Был выбран имитационный дискретно-событийный метод моделировании, так как: 1) имитационная модель, как и сетевая, описывает систему в естественных терминах, что упрощает генерацию; 2) дискретно-событийная модель наиболее адекватно отражает специфику данной экосистемы, поскольку вся динамика рассматриваемой экосистемы имеет именно дискретной-событийный, а не непрерывный характер.

Приведем краткое описание жизненного цикла клеща. Клещ имеет четыре стадии развития: яйцо, личинка, нимфа, имаго (взрослый клещ). В каждой из стадий клещ находится год, иногда два. Переход между стадиями (кроме превращения яйца в личинку) осуществляется после кормления, т. е. после укуса клещом какого-либо животного (прокормителя). Во время укуса также может произойти передача заболевания (энцефалита и / или боррелиоза) от клеща к прокормителю или наоборот (см. рис. 2). Существуют также и более сложные случаи, в частности передача заболевания между клещами, укусившими одного прокормителя. Учитывая, что взрослый клещ после укуса откладывает яйца, имеем замкнутый цикл развития клеща продолжительностью в среднем 4 года. Также в литературе [7] описан ряд отклонений от этого цикла, влияющих на его продолжительность. Кроме того известно поведение клеща, в частности его перемещения, активность в зависимости от параметров окружающей среды, смертность от различных факторов и т. д.

На рис. 3 приведен фрагмент жизненного цикла клеща в форме описательной сетевой модели, а также сетевой модели, описывающей структуру дискретно-событийной модели. Дискретно-событийная модель оперирует такими понятиями, как «состояние системы» (некоторая модель данных в форме набора объектов), «событие» и «триггер». Событие может изменить состояние и инициировать триггер(-ы), триггер ставит новое событие(-я) в очередь с приоритетом по времени исполнения. Элементарный цикл при таком моделировании – это отработка следующего события и связанных с ним триггеров. При этом модифицируются состояние системы и очередь событий.

Ставится задача автоматизации процесса построения дискретно-событийной модели на основе подготовленной сетевой. Полностью автоматизировать этот процесс не удается, однако частичная автоматизация возможна. Рассмотрим основные элементы дискретно-событийной модели с этой точки зрения.

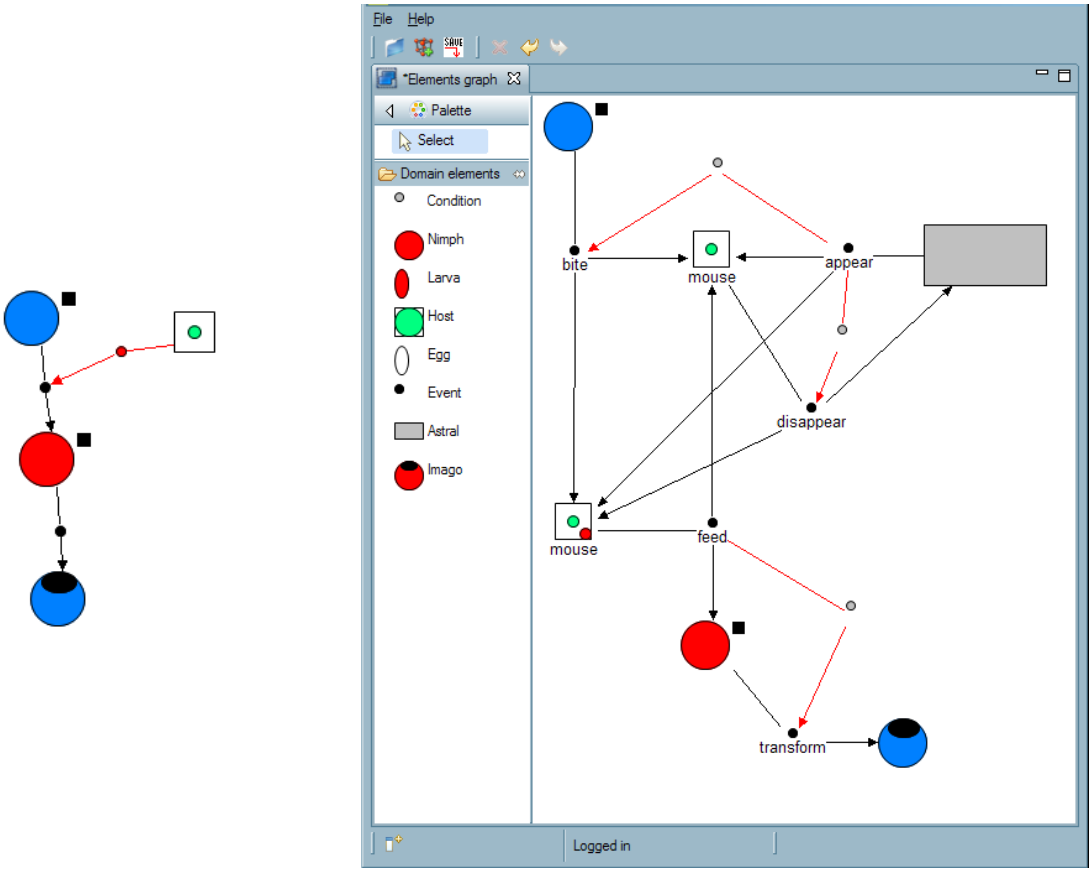

Рис. 3. Фрагмент базовой модели жизненного цикла клеща (слева) и тот же фрагмент, подготовленный для генерации дискретно-событийной модели

Состояние. Сетевая модель в явном виде не определяет структуру состояния и начальное состояние ввиду того что отношение между вершинами сетевой модели и объектами в динамической модели неоднозначно. Тем не менее генерация начального состояния может быть в существенной степени автоматизирована. Вершина сетевой модели интерпретируется, как множество объектов заданного типа с частично заданным состоянием, т. е. по части свойств состояние может быть фиксировано, по части – нет. По незафиксированным свойствам генератору должны быть заданы дополнительные указания, с каким распределением их создавать. Одним из таких параметров является мощность указанного множества объектов. Вырожденным случаем является ситуация, когда одной вершине соответствует ровно один объект.

Событие. Характеризуется участием некоторых объектов (классы которых задаются сетевой моделью), а также влиянием, которое оказывает событие на эти объекты. Участники событий однозначно задаются сетевой моделью. С точки зрения поведения события удобно разделить на несколько типов: 1) перевод объекта из одного состояния в другое; 2) порождение объекта; 3) изменение состояния объекта с учетом его текущего состояния, а также состояния других участников. Первый тип является достаточно частным случаем, однако весьма значимым (отвечает за представление системы, фактически, в терминах автомата), поэтому его имеет смысл рассматривать отдельно. Его поведение практически однозначно задается сетевой моделью, от пользователя требуется только указать, какого из участников трактовать как входное состояние, какого – как выходное. Примером является переход клеща из одной стадии развития в другую. Для третьего случая имеем функцию преобразования вектора старого состояния участников в вектор нового состояния. Здесь от пользователя требуется каким-либо образом определить поведение этой функции. Второй случай также требует задания функции-конструктора, параметрами которой являются другие участники.

Триггер. Характеризуется событием, которое инициирует данный триггер, а также функций преобразования участников события и, вообще говоря, текущего состояния системы, в множество пар (событие, время), которые ставятся в очередь событий. На практике триггер чаще всего выставляет событие в очередь однозначно, конструируя его на основе участников события, которое повлекло за собой срабатывание данного триггера. Вычисляемым параметром оказывается только время постановки такого события. Например, при условии успешного кормления клеща он однозначно переходит на другую стадию развития (либо оставляет потомство, что однозначно определяется его текущей стадией развития), однако это может произойти в разное время, в том числе и в следующем году. Таким образом, функцию вычисления времени, а в некоторых случаях и функцию порождения нового события / событий должен задать пользователь.

В случае событий и триггеров, таким образом, требуется определение некоторого поведения системы в форме программного кода. Целесообразно написание такого коде не на базовом языке разработки системы (Java в данном случае), а некотором проблемно-специфичном языке (DSL, domain-specific language), который генерируется автоматически на основе онтологии. Это существенно упрощает для пользователя генерацию динамической модели, так как позволяет разрабатывать такие сценарии в терминах предметной области, кроме того, декларативная часть может быть сгенерирована автоматически. Например, на основании описания типа события в онтологии можно вычислить его участников, а также тип события (при наличии механизма стереотипирования или подобных ему).

Выводы

В работе рассмотрены вопросы разработки программных инструментов для построения и применения сетевых моделей экосистемам. Предложена и реализована архитектура таких инструментов. Ключевой особенностью архитектуры является метамодель данных, позволяющая настраивать систему на предметную область, также представлять специфические для сетевых моделей данные, в том числе проводить их декомпозицию. Возможности задания онтологии, а также способов визуализации основанных на ней сетевых моделей вынесены на пользовательский уровень в форме специализированного редактора.

Показана возможность применения сетевых моделей для автоматизации построения дискретно-событийных моделей. Часть целевой модели может быть сгенерирована полностью автоматически, часть – путем дополнительных деклараций со стороны пользователя (что может быть реализовано в форме набора диалогов или «мастера» и не требует от него навыков программирования). Некоторые поведенческие аспекты не могут быть сгенерированы автоматически, однако может быть построен DSL, существенно упрощающий их описание пользователем. На данный момент методика продемонстрирована на модели иксодовых клещей. Целесообразно продолжение исследований, разработка полноценного программного инструмента для генерации и исполнения дискретно-событийных моделей, а также расширения набора модельных задач.

Список литературы

1. *Galperin M. Y., Cochrane G. R.* Nucleic Acids Research annual Database Issue and the NAR Online Molecular Biology Database Collection in 2009 // Nucleic Acids Research. 2009. Vol. 37 (Database issue). P. D1–4.

2. *Мигинский Д. С., Лабужский В. В., Лаврентьев М. М., Морозов А. В., Соколов С. А.* Технология семантической интеграции баз данных в системной биологии // Вычислительные технологии. 2008. Т. 13, № 6. С. 103–119.

3. *Sivachenko A. Y., Yuryev A.* Pathway Analysis Software as a Tool for Drug Target Selection, Prioritization and Validation of Drug Mechanism // Expert Opin. Ther. Targets. 2007. Vol. 11 (3). P. 411–421.

4. *Shannon P., Markiel A., Ozier O., Baliga N. S., Wang J. T., Ramage D., Amin N., Schwikowski B., Ideker T.* Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks // Genome Res. 2003. Vol. 13 (11). P. 2498–2504.

5. *Miginsky D. S., Suslov V. V., Timonov V. S., Rasskazov D. A., Sournina N. Yu., Podkolodny N. L.* Approaches to the Computer Reconstruction of the Biological Networks // Intelligent Data Analysis. 2008. Vol. 12. No. 5. P. 463–479.

6. *Рамбо Дж., Якобсон А., Буч Г.* Унифицированный процесс разработки программного обеспечения. СПб.: Питер, 2002.

7. *Никитин А. Я., Антонова А. М.* Учеты, прогнозирование и регуляция численности таежного клеща в рекреационной зоне г. Иркутска. Иркутск, 2005.

Материал поступил в редколлегию 18.08.2011

D. S. Miginsky, V. S. Timonov

APPLICATION OF ECOSYSTEM NETWORK MODELS FOR AUTOMATION OF SIMULATING

The paper considering the problem of ecosystems discrete-event simulation based on network models. For simulation of system of any kind it is essential to formalize the terminology for current domain. On the other hand the trait of ecosystems is the high variability of the domain. It makes it difficult to develop the unified tool that is applicable for relatively wide class of ecosystems. Partially the problem could be solved by application of formal ontologies. But ontology itself is only applicable for descriptive modeling, e.g. network modeling in this case. To solve this problem the method for developing the domain-specific language is proposed. Such language is applicable for discrete-event modeling and capable to be adjusted for particular domain by use of ontology.

Keywords: ecosystem simulation, network modeling, formal ontologies, domain-specific-languages.