УЧЕНЫЕ ЗАПИСКИ КАЗАНСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Том 148, кн. 3 Естественные науки *2006*

УДК 573.4

ПРИМЕНЕНИЕ ПРИНЦИПОВ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ К ОПИСАНИЮ И КЛАССИФИКАЦИИ БИОЛОГИЧЕСКИХ ОБЪЕКТОВ НА ПРИМЕРЕ ГРИБОВ РОДА *TRICHODERMA*

Д.С. Тарасов, Р.И. Тухбатова, Н.И. Акберова, Ф.К. Алимова

Аннотация

В статье рассматривает возможность приложения объектно-ориентированного программирования к проблемам биологической систематики на примере описания и классификации *Trichoderma* – рода грибов, систематика которого в настоящее время вызывает споры. Показано, что информация об организме для целей его идентификации, классификации и систематики может быть представлена не в форме словесных описаний, а в виде программ онтогенетического развития, записанных на объектно-ориентированных языках программирования, что отражает особенности динамического развития живых организмов. Описания таксонов и связи между отдельными таксонами, равно как и отношения между отдельными организмами могут быть представлены с помощью таких механизмов объектно-ориентированного программирования, как классы, наследование и полиморфизм. Объектно-ориентированная классификация лучше отражает биологическую действительность, поскольку допускает множественное наследование, различную длину деревьев наследования и дает однозначное определение концепции вида. В рассмотренном примере классификации грибов рода *Trichoderma* отмечено, что объектно-ориентированная классификация позволяет более точно отразить взаимоотношения между отдельными организмами, используя при этом меньшее число абстракций (таксонов), чем это сделано в существующей классификации данного рода.

Введение

В биологии, в отличие от физических дисциплин, приобретаемое знание хранится и представляется с использованием преимущественно естественных языков. В течение долгого времени эта форма была адекватна для решения большинства теоретических и прикладных задач, стоящих перед биологическими науками. Тем не менее, если рассмотреть существующие на сегодняшний тенденции более внимательно, неизбежным оказывается вывод о необходимости создания и внедрения в научный обиход современных форм представления биологических знаний.

Прежде всего обращает на себя внимание рост объемов экспериментальных данных. В настоящий момент гигабайтные объемы уже не являются необычными для результатов, получаемых в ходе одного эксперимента [1]. В будущем, с широким внедрением беспроводных датчиков, собирающих данные двадцать четыре часа в сутки, можно ожидать многократного увеличения информационных потоков. Даже данные «скромных» размеров в десятки и сотни

килобайт невозможно обрабатывать без средств автоматизации, таких как распространенные электронные таблицы Excel и некоторые пакеты статистического анализа. Однако эти средства автоматизации и предоставляемые ими возможности являются рудиментарными и обречены на исчезновение в ближайшем будущем.

Централизованные базы данных биологической информации будут наращивать свои объемы, и их роль будет постоянно возрастать. В то же время, очевидно, что типичная на сегодня стратегия работы с этими базами «найти информацию → загрузить к себе на компьютер → проанализировать» становится с ростом объемов данных непрактичной, поскольку существуют ограничения на размер информации, которая может быть передана через интернет. Соответственно, в базы данных будут посылаться не запросы на определенные сведения, а программы, которые будут производить вычисления и анализ данных на месте и передавать только результаты. Эти операции невозможны в том случае, если данные представлены исключительно в форме публикаций на естественном языке.

В случае, когда размер исходных данных превышает определенную величину, неизбежно возникает ситуация, когда даже объем промежуточных обобщений и гипотез, необходимых для продолжения исследования, становится больше того, который может быть эффективно обработан человеком в разумные сроки. Это приводит к неправильному выбору направления дальнейшей работы, напрасному расходу средств и усилий и в итоге к выводам, находящимся в противоречии с уже собранными данными. Традиционные формы представления результатов в форме графиков и таблиц с текстовыми пояснениями не дают практической возможности произвести исчерпывающую проверку на внутреннюю согласованность исходных данных, промежуточных обобщений и окончательных выводов.

В то же время появление средств автоматизированного анализа данных и даже интеллектуальных систем автоматизированного выдвижения и проверки гипотез становится причиной «отчуждения» научного работника от собственно научного процесса, когда собственно сам процесс обработки данных и формирования заключений оказывается скрытым от исследователя внутри «черного ящика» программного пакета. В результате уровень научной работы резко снижается, поскольку, во-первых, снижается степень понимания исследователем методов обработки данных, а во-вторых, сами эти методы часто применяются неправильно, поскольку компьютерные программы, не знакомые с контекстом проводимых исследований, не способны сами выбрать наиболее подходящие средства анализа информации. Для преодоления сложившегося противоречия необходимо расширить степень знакомства исследователей с методами компьютерной обработки данных, одновременно направив усилия на усовершенствование средств общения ученых-биологов с компьютерными системами. Абсолютно необходимы новые средства представления знаний и процессов, которые могли бы дать исследователю возможность заглянуть внутрь «черного ящика» обработки данных.

Наконец, необходимо отметить важную роль формы представления знаний в процессах междисциплинарной интеграции. Сегодня можно нередко наблю-

дать ситуацию, когда даже исследователи-биологи не способны найти общий язык между собой.

В настоящей работе впервые предпринимается попытка использования существующей методологии программирования для решения задачи описания и систематики биологических объектов. Нами представляется специализированный язык описания информации об организме, получаемой в ходе эксперимента, для его идентификации, классификации и систематики. Данный язык одновременно является языком программирования общего назначения и предназначен для создания программ обработки филогенетической информации, в том числе проверки внутренней согласованности построенной систематики, автоматизации идентификации видов и предварительной обработки данных эксперимента. Кроме того, он служит средством обмена информацией о характеристиках и классификации организмов между исследователями. Язык и способ его применения представлены на примере создания описаний и классификации грибов рода *Trichoderma*.

Объектно-ориентированное программирование и систематика

В задачах систематики можно выделить три основных компонента:

- *Описание:* запись информации об имеющемся организме либо таксоне в соответствии с некоторыми правилами.
- *Идентификация*: определение принадлежности данного организма к тому или иному таксону на основании описаний.
- *Классификация*: на основании имеющихся описаний определение новых таксонов и взаимоотношений между ними.

Некоторая *система* живых организмов может считаться определенной, если установлен унифицированный способ описания, идентификации и классификации организмов. Система может быть описана в различных формах (на естественном языке, математически и т. п.), а также иметь различное содержание (различные отношения между таксонами).

Основным способом спецификации организма или таксона до настоящего времени является описание, записанное на естественном языке (т. е. в форме последовательности предложений, записанных на русском, английском или другом языке общения людей). В настоящее время многие области науки, в том числе и биологии, отказались от естественного языка в пользу той или иной формальной системы. Недостатки естественного языка для научных приложений вообще и для описаний биологических объектов в частности включают неоднозначность (нет четкого определения понятий), локализованность (перевод с одного языка на другой может привести к искажению смысла), излишнюю громоздкость конструкций, недостаточную адаптированность к предметной области.

Описание вида должно включать его состояние на разных стадиях.

Известно, что любое описание характеристик вида, претендующее на некоторую полноту, должно включать в себя описания характеристик различных стадий жизненного цикла. Большинство способов и форматов описания напоминают мгновенные фотографические снимки – срезы всех характеристик на

определенный момент времени, что, конечно, не отражает достаточно полно специфику изучаемого организма. Даже включение описания различных стадий не исправляет положения, т. к. в данном случае описание представляет собой как бы серию снимков, сделанных, как правило, через значительные промежутки времени. Такие спецификации не естественны и недостаточно представительны.

Развитие живого организма можно представить как вычислительный процесс.

Живая клетка может быть рассмотрена как вычислительное устройство [2]. В этом случае последовательную смену жизненных стадий можно описывать как развертывание вычислительного процесса, выполняемого под управлением клеточного компьютера.

Наиболее подходящим способом описания вычислительного процесса является программа.

Для задания последовательности действий (последовательности вычислений) используются программы, записанные на специальных языках. Программа является более компактным и естественным способом представления вычислительного процесса, нежели последовательное описание всех состояний системы. Кроме того, содержание программы не зависит от конкретной стадии развития.

Живая клетка уже содержит программу в форме последовательности нуклеотидных звеньев ДНК. Однако по ряду практических соображений использование такой формы описания вида является неудобным. Проводя аналогию с компьютерами, можно сказать, что ДНК содержит своеобразные «машинные коды», описывающие низкоуровневые операции клеточных механизмов. Перевод морфологических данных в такой формат является сложной задачей, кроме того, непосредственное чтение и анализ нуклеотидных последовательностей человеком практически невозможно.

Тем не менее, основываясь на этом, мы можем построить язык более высокого уровня, выражения которого в равной степени пригодны для чтения человеком и компьютером. Такое описание будет естественным для рассматриваемой системы и способно отражать многие закономерности, незаметные или полностью теряемые в обычных системах описания.

Наиболее подходящим способом выражения биологических программ является объектно-ориентированное программирование (ООП).

Многие задачи лучше описывать и решать в терминах объектов, состояние которых изменяется со временем. Если рассматривать описание гриба как систему объектов (среда, гифы, споры и т. д.), взаимодействующих с целью изменения состояния среды и друг друга, задачу можно значительно упростить и систематизировать. Объектно-ориентированные языки поддерживают такой подход к решению задачи, при котором задача разбивается на набор взаимодействующих между собой объектов. Они обладают состоянием, которое может изменяться со временем, и множеством функций, определяющих их поведение.

Объектно-ориентированное программирование (ООП) – современный метод составления и структурирования компьютерных программ [3, 4]. Первый

ОО-язык SIMULA был создан в рамках реализации проекта по моделированию дискретных событий [5].

Многие языки программирования компьютеров поддерживают объектно-ориентированный подход, в том числе Java [6], SmallTalk [7], C++ [8]. Кроме того, система представления биологических знаний использует элементы объектно-ориентированного подхода. Методы ООП нашли широкое применение в описании и моделировании экологических [9] и молекулярно-биологических систем [10].

Чтобы называться объектно-ориентированным, язык должен обеспечивать три возможности: *инкапсуляцию, полиморфизм* и *наследование*. Рассмотрим эти возможности более детально и то, как их можно применить при описании и систематике грибов.

Инкапсуляция. Объектно-ориентированная инкапсуляция – это сочетание в одной структуре элементов данных и процедур для их обработки. Такие структуры называются классами. Например, класс TColony (колония) может объединять в себе информацию о радиусе колонии (ColonyRadius) и процедуру, определяющую его зависимость от времени (рис. 1). Класс имеет название, *атрибуты* и *методы*. *Атрибуты* – это некоторые характеристики класса, например: цвет колонии, наличие выростов, форма, запах и т. п. *Методы* определяют реакцию всех принадлежащих к данному классу объектов на внешние и внутренние события – последовательность и условия проявления различных признаков.

Рис. 1. Класс Колония включает атрибуты Радиус, Край и Прирост, а также метод Рост, который задает процедуру роста колонии

Наследование. Наследование – это механизм поддержки языком абстрактных классов. Оно позволяет определить общие классы и задать структуру и поведение их специализацией. Наследование классов не имеет прямого отношения к наследственности живых организмов, хотя некоторые аспекты последнего могут быть представлены с помощью первого. Наследование позволяет определять новые классы, которые подобны исходным классам во всем, кроме специфического набора отличий (рис. 2).

Полиморфизм. Слово полиморфизм произошло от двух корней: «поли», означающего много и «морф» – форма. Полиморфизм обозначает возможность определения нескольких классов с функционально различными, но одинаково названными методами или свойствами. Так метод Деление (рис. 2) может быть определен для всех клеток, но результаты деления могут различаться в зависимости от типа клеток.

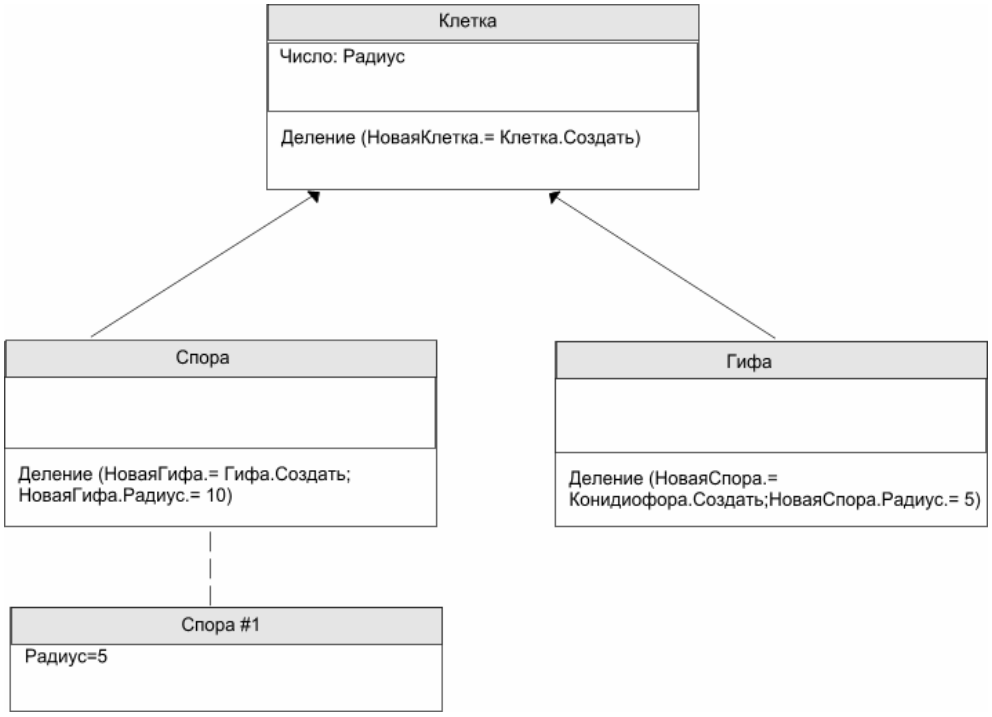

Рис. 2. Полиморфизм и наследование. Суперкласс «Клетка» определяет атрибут «Радиус» и метод «Деление». Спора и гифа являются подклассами класса «Клетка» и, соответственно, наследуют атрибут «Радиус». Спора#1 является экземпляром класса «Спора» с конкретным значением радиуса. Метод деления определен в классе «Клетка», как свойственный всем клеткам. Спора и Гифа переопределяют данный метод таким образом, что результатом деления споры является гифа, а при делении гифы образуется спора

Табл. 1

Соответствие понятий биологической классификации и ООП

Биологическая классификация	ООП
Таксон	Класс
Вид	Не абстрактный класс, который может иметь экземпляры
Таксон высших порядков	Абстрактный класс
Организм	Экземпляр класса

Если мы сопоставим ООП и биологическую систематику, то можем идентифицировать некоторые полезные соответствия. Любая существующая биологическая система может быть представлена средствами ООП. Чтобы понять эти соответствия, обратимся к рис. 3, на котором представлен фрагмент систематики в форме диаграммы классов. Любому таксону может быть сопоставлен класс, определяющий общие характеристики данного таксона (табл. 1).

ООП может быть оболочкой, «оберткой» для любой существующей систематики. Конвертация любой систематики в ООП дает ей все преимущества использования формального языка без ущерба гибкости подхода. С другой сторо-

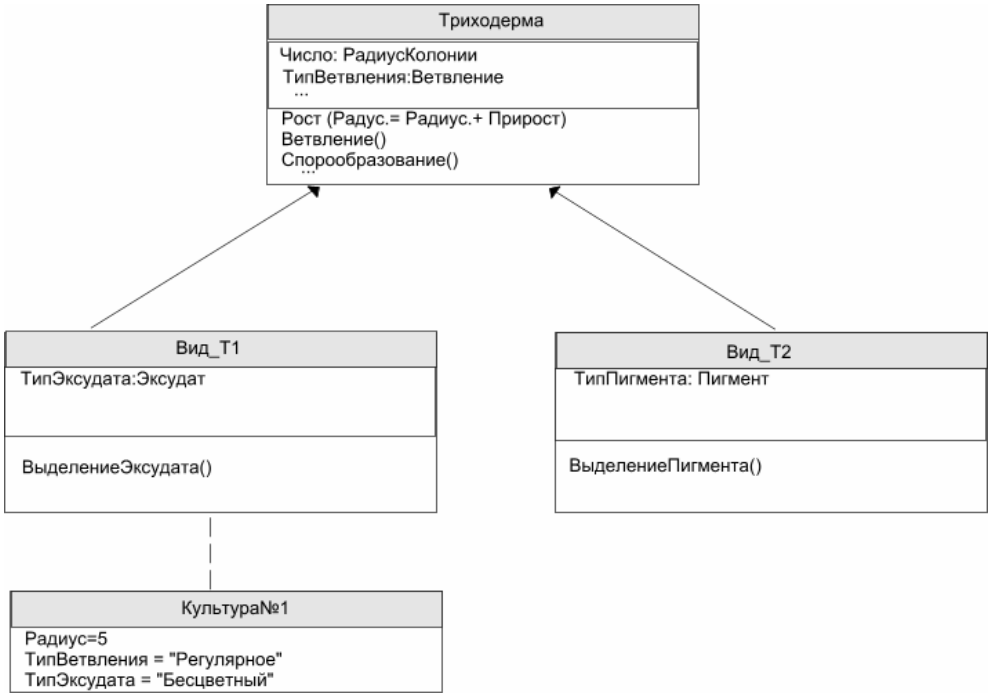

Рис. 3. Наследование классов и биологическая систематика. «Триходерма» является абстрактным классом (не может иметь экземпляров), в котором определены атрибуты и методы, свойственные всем организмам рода *Trichoderma*. Классы Вид_Т1 и Вид_Т2 являются подклассами «Триходерма», определяющими дополнительные атрибуты. Классы Вид_Т1 и Вид_Т2 являются *видами* с точки зрения биологической систематики, поскольку из них могут образовываться конкретные экземпляры (Культура № 1)

ны, возможно построение чистых ОО-систематик, основывающихся на принципах подхода и использующих возможности ОО, для которых в биологических систематиках нет аналогов (например, множественное наследование). Более подробно сравнение свойств традиционной и ОО-систематики приведено в табл. 2.

Сравнительная оценка качества ОО-систематик

Очевидно, что возможно создание различных ОО-систематик. В связи с этим возникает вопрос о том, по каким критериям можно сравнивать различные ОО-систематики. Поскольку ОО-систематика представляет собой систематику программ, то разумно попробовать применить к ней критерии, использующиеся при оценке компьютерных программ.

1. Цикломатическая сложность и число методов на класс. Цикломатическая сложность [11] впервые предложена для оценки сложности программы в 1976 г. Томасом МакКейбом и остается до сих пор одной из наиболее широко применяемых метрик (в различных вариантах). Цикломатическая сложность есть число линейно-независимых путей через программный модуль. В ООП

Табл. 2

Сравнение свойств объектно-ориентированной и традиционной систематики

	Традиционная классификация	ОО-классификация
Дополнительная информация	Опирается на внешнюю информацию о биологии организмов, прямо не включаемую в систематику	Содержит всю информацию, необходимую для понимания
Точность спецификации	Определения могут быть расплывчатыми и пониматься разными исследователями по-разному	Однажды специфицированная не допускает неоднозначных прочтений
Последовательность в именовании таксонов	Разные таксоны могут именоваться одинаково, что вносит путаницу	Одинаковые названия разных таксонов невозможны
Численные характеристики	Классификация может быть оптимизирована с помощью кластерного или парсимонного анализа, при условии сопоставления признаком условных весовых факторов, выбор которых является абсолютно произвольным	К ОО-классификации применимы все численные характеристики, применимые к обычной классификации. Кроме того, дополнительно доступны специфические алгоритмические и ОО-метрики, такие, как WMC, LCOM (см. раздел «сравнение ОО-систематик»)
Определение вида	Неопределенное, дается разными авторами по-разному, не следует из контекста самой классификации	Однозначно (см. табл. 1)
Деревья наследования	Теоретически все деревья наследования имеют одинаковую длину. Практически это требование часто нарушается	Деревья наследования могут иметь различную длину, что лучше отражает биологическую реальность
Множественное наследование	Запрещено	В принципе допустимо. Отражает такие биологические процессы, как половое размножение, горизонтальный перенос генов и др.
Устойчивость к ошибкам	При конструировании классификации систематик может допустить любую ошибку или непоследовательность, которая может быть выявлена лишь при тщательном ручном анализе. Это создает почву для существования внутренне противоречивых классификаций	Существование формального языка и правил ООП заставляет систематика более четко формулировать свои мысли. Многие ошибки и внутренние противоречия могут быть выявлены компилятором на этапе конструирования систематики
Способ спецификации организма	Конкретный организм или таксон специфицируется с помощью словесного описания	Спецификацией организма является программа его развития, не допускающая неоднозначных толкований. Для того чтобы лучше понять программу, ее можно запустить (целиком или в интерактивном режиме) и рассмотреть результат выполнения

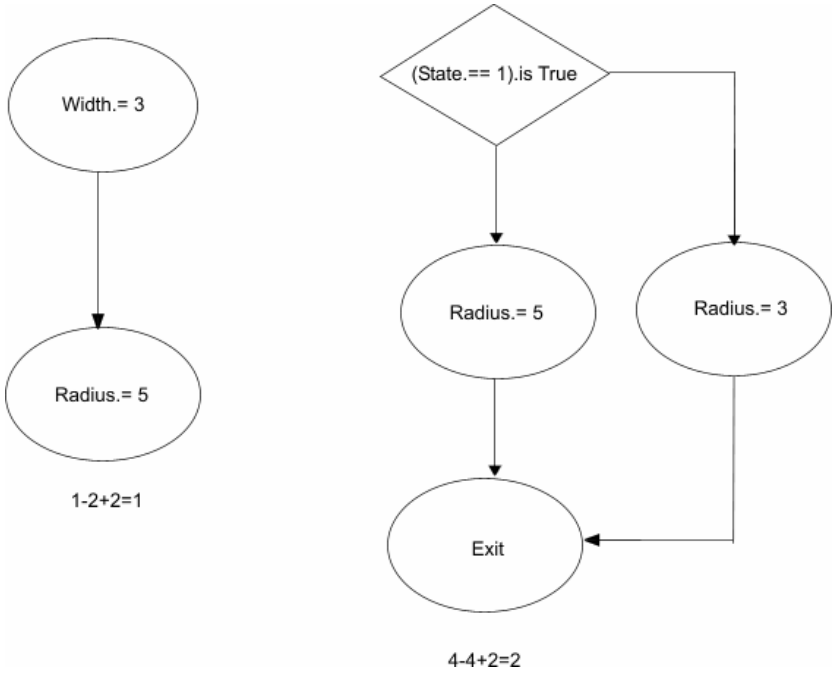

Рис. 4. Примеры расчета цикломатической сложности для методов систематики

цикломатическая сложность используется для оценки сложности реализации методов и определяется как *число узлов – число ребер* плюс 2 (см. рис. 4).

Число методов на класс – это количество методов, реализованных внутри класса, или сумма сложностей этих методов.

Сумма сложностей всех методов всех классов, входящих в ОО-систематику, позволяет сравнивать две систематики при условии, что для их создания использовался один набор признаков. Систематика с меньшей общей сложностью, очевидно, является лучшей, поскольку выделяет наибольшее число общих признаков, и следовательно, определение принадлежности организма к определенному классу будет давать максимальную информацию о данном организме.

Разница в цикломатической сложности двух классов (без учета наследуемых методов) может служить мерой расстояния между двумя классами, что позволит оценить удаленность друг от друга различных таксонов.

2. Отсутствие связности методов. Связность является важным понятием в ООП. Она показывает, представляет ли класс одну абстракцию или несколько абстракций. С точки зрения биологической систематики, связность должна показывать, представляет ли данный таксон единое целое, или он должен быть разделен на несколько таксонов.

Существует несколько метрик определения отсутствия связности.

Чидамбер и Кемерер [12] определяют отсутствие связности как число пар методов в классе, которые не используют хотя бы одного общего атрибута, минус число пар методов в классе, которые используют хотя бы один общий атрибут. Если результат отрицателен, значение метрики равно нулю.

Хендерсон-Селлерс [13] определяет отсутствие связности следующим образом.

Если M – множество методов, определяемых классом; F – множество атрибутов, определяемых классом; $P(f)$ – число методов, которые обращаются к атрибуту f, где f входит в F; $\langle p \rangle$ – среднее значение $p(f)$ на F; LC – значение метрики отсутствия связности, тогда

$$LC = \frac{\langle p \rangle - |\mathrm{M}|}{1 - |\mathrm{M}|}.$$

Высокая связность показывает правильное деление на классы. Отсутствие или низкая связность увеличивает сложность. Классы с низкой связностью могут, скорее всего, быть разделены на два или более подкласса с большей связностью.

ConceptSystem: ОО-язык для биологических приложений

Для работы с описаниями биологических объектов нами разработан специальный формальный язык ConceptSystem, похожий на язык программирования, но имеющий ряд специфических особенностей. Сравнение разработанного языка с другими объектными языками, а также детальное его описание находится за пределами рамок данной работы. В настоящем разделе приводится краткий обзор разработанного языка.

Общие положения.

Все в языке является объектами. Объекты взаимодействуют друг с другом только при помощи сообщений.

Переменные.

Переменные в ConceptSystem также сами являются объектами, связывающими некоторое символьное название с тем или иным объектом-значением. Связывание названия с объектом происходит с помощью сообщения «=». Сообщения, передаваемые переменным и не обрабатываемые ими, передаются связанным объектам.

Методы.

Каждый объект реализует интерфейс, определяющий набор сообщений, которые могут обрабатываться данным объектом. Описание сообщения вместе с набором процедур, описывающих реакцию объекта на прием данного сообщения, называется методом объекта.

Описание сообщения включает в себя название сообщения и перечень классов, которые могут быть переданы в качестве аргументов. Само сообщение состоит также из названия и списка ссылок на объекты-аргументы. В результате выполнения метод возвращает объект.

Метод включает в себя фрагмент кода. Вызов метода приводит к получению объекта, возвращаемого методом. Единственный способ заставить объект сделать что-либо – это вызвать его метод. Таким образом, программа состоит из последовательности различных указаний различным объектам.

Структура методов.

Метод состоит из фрагмента кода и аргументов.

Фрагмент кода – последовательность вызовов методов, разделенных символом ‘;’ и заключенных в скобки. Фрагмент кода рассматривается как объект класса CodeFragment.

Пример:

(console.Write ‘Hello!’;console.Write ‘This is me!’)

Аргументы являются локальными переменными методов, к которым привязываются аргументы, употребленные при вызове метода. Аргументы причисляются через запятую и отделяются от тела метода знаком |.

Каждый метод имеет локальную переменную Result, которая используется для возращения результата работы метода:

(Number ToAdd|Result.= ToAdd.+ 5)

Некоторые предопределенные классы

Class

Класс – это особый объект. Класс содержит спецификацию для создания какого-либо объекта. Класс содержит спецификацию методов и спецификацию свойств, методы, обеспечивающие редактирование спецификаций и создание объектов по заданным спецификациям. Класс специфицирует сам себя. Кроме того, класс отвечает за создание других классов, реализацию механизма наследования.

Методы класса:

New создает экземпляр объекта данного класса.

KindOf (Class Parent|) создает новый класс – наследник от класса-аргумента Parent.

Method (String Name, CodeFragment Body*)* добавляет метод Name с кодом Body к спецификации объекта класса.

Contains (Class Type, Identifier Id|) добавляет новый атрибут типа Type с названием Id к спецификации объекта класса.

Object

Базовый класс. Всякий другой класс является наследником класса object (кроме class, bindingsite).

Методы:

is (Class Classname, CodeFragment, YES CodeFragment NO) – определяет, является ли объект наследником класса class, выполняет код YES или NO в зависимости от результата.

== (Object ToComp)

Определяет отношение объекта к объекту-аргументу ToComp. Возвращает объект типа Boolean.

Pool

Пул – это особый объект. Он отвечает за хранение находящихся в нем объектов, классов и глобальных переменных.

Методы:

Contains (Class Type, Identifier Name) – объявление новой переменной типа Type с именем Name.

Using (FileName String) читает внешний файл и добавляет содержащиеся в нем объекты в пул.

Item (Number Index) возвращает объект содержащийся в пуле под номером Index.

Count возвращает общее число объектов в пуле.

Number

Класс-число

Методы:

+ (Number X) возвращает сумму данного числа и числа-аргумента.

– (Number X) возвращает разность данного числа и числа-аргумента.

* (Number X) возвращает произведение данного числа и числа-аргумента.

/ (Number X) возвращает результат деления данного числа и числа-аргумента.

> (Nunber X) сравнение чисел. Возвращает True или False.

< (Number X) сравнение чисел. Возвращает True или False.

== (Number X) сравнение чисел. Переопределяет метод == Object.

String

Класс-строка

Методы:

== (String S) сравнивает две строки. Возвращает значение типа Boolean.

+ (String S) выполняет объединение двух строк.

Модификаторы

В большинстве наиболее распространенных языков программирования можно провести аналогию между классом и существительным естественного языка. Классы соответствуют определенным предметным сущностям (Колония, Стол, Файл и т. п.). Для того чтобы использовать определенные признаки-прилагательные, можно создать классы-наследники базового класса (КруглаяКолония, КруглыйСтол, ЗеленаяКолония). В то же время иногда возникает необходимость абстрагировать признак от объекта и оперировать такими понятиями, как Зеленый, Круглый и т. п. Это обосновано, если признаки-прилагательные несут набор свойств, которые более или менее единообразно преобразуют классы-существительные. Для реализации этой возможности в Concept-System служат модификаторы.

Модификаторы можно применять совместно с идентификаторами классов для быстрого конструирования новых классов. Результатом применения модификаторов являются новые классы. Модификатору можно сопоставить произвольное преобразование класса. Например, можно создать модификатор Круглый, который можно применять для создания классов (Круглый Колония, Круглый Стол), при этом новые классы будут иметь дополнительно специфические методы и атрибуты, определяемые модификатором (например, Радиус).

Реализация модификаторов опирается на динамические классы.

Динамические классы.

В систематике часто возникает необходимость «сузить» класс путем наложения определенных ограничений (например, Вид1 есть Род1, у которого значение радиуса колонии лежит в диапазоне [3..5]). Идентификация организмов часто решает обратную задачу – определение принадлежности экземпляра к классу по значениям атрибутов. Таким образом, желательно существование классов, принадлежность к которым определяется динамически как функция значений атрибутов.

Для того чтобы сделать класс динамическим в ConceptSystem, необходимо реализовать у этого класса метод defineas. Код данного метода выполняется для каждого экземпляра класса с целью определения его принадлежности.

Динамические классы позволяют автоматизировать процедуру идентификации организмов по описаниям.

Trichoderma: описание и систематика

Существует множество групп живых организмов, систематика которых находится в процессе постоянного изменения. Одной из таких групп являются грибы рода *Trichoderma*. Грибы рода *Trichoderma* представляют ценность для применения в различных областях. В частности, грибы этого рода используются для биологической защиты растений против грибных возбудителей болезней [14, 15].

Жизненный цикл грибов рода *Trichoderma* представлен на рис. 5. Он включает в себя половую и бесполую стадию. В данной работе основной интерес для нас будет представлять бесполая стадия.

Бесполая стадия начинается с прорастания конидий (спор) в состоящий из гиф субстратный мицелий, из которого затем развивается воздушный мицелий. Кроме того, в субстратном мицелии можно обнаружить хламидоспоры, утолщения и анастомозы. Воздушный мицелий состоит из конидиеносцев, из которых развиваются фиалиды. Фиалиды несут на себе конидии.

Ряд признаков, обычно используемых при описании, включает цвет колонии, присутствие или отсутствие запаха, развитие колонии, присутствие или отсутствие пигмента, диффундирующего в среду и т. п.

Описание.

Важным элементом в систематике вообще и в систематике *Trichoderma* в частности является (наряду с другими подходами) классификация по морфологическим признакам [16]. При классификации по морфологическим признакам сначала составляют по некоторым правилам описание объекта, а затем объект согласно описанию относят к одной из существующих систематических групп либо выделяют новые группы.

При описании организмов возникает ряд проблем:

1. описания часто содержат неточности и могут интерпретироваться неоднозначно;

2. описания трудно сравнивать между собой, что является источником возникновения разногласий;

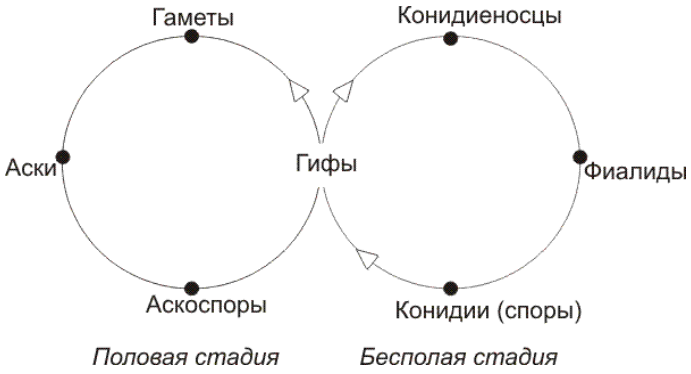

Рис. 5. Жизненный цикл *Trichoderma*

3. различия иерархической значимости признаков не находят прямого отражения в описаниях;

4. описания слабо отражают факторы, связанные с развитием организма во времени.

Ранее нами предлагался метод преодоления проблем, возникающих при создании описаний [17].

Предлагаемый в настоящее работе способ описания базируется на том, что развитие организмов рода *Trichoderma* можно рассматривать как последовательность деления и дальнейшей специализации отдельных клеток. В колонии *Trichoderma* клеточные перегородки могут формироваться не до конца или разрушаться, однако это несущественно для исходного предположения.

Описание представляет собой программу, которую необходимо задать исходной клетке (споре), чтобы получить ее развитие в колонию *Trichoderma* и в итоге снова в спору. Такое описание является естественным, поскольку отражает реальное положение вещей и позволяет разделить признаки на *свойственные единичным клеткам* и *появляющиеся в результате взаимодействия большого числа клеток*. Описание-программа следует логике развития самого организма, а не логике проведения наблюдений, особенностей экспериментов и исторических обстоятельств, нередко оказывающих влияние на обычные текстовые описания.

Таким образом, основные сущности, с которыми оперируют предлагаемые описания, – это клетки, способные делиться в различных направлениях, специализироваться в клетки гифы, конидиеносцы, фиалиды и споры, а также обладающие различными свойствами – размерами, цветом, способностью к выделению пигмента и т. п.

Такие сущности, как «субстратный мицелий», «воздушный мицелий», «реверзум – обратная сторона колонии» и т. п., не являются естественными для организма: они появляются в результате взаимодействия перечисленных выше элементов и их свойства являются следствием структуры программы отдельных клеток. Вводить эти понятия в описания-программы как отдельные сущности нет никакой необходимости.

Рассмотрим следующий фрагмент описания:

```
(Pool.Contains Class Trichoderma). = Class.KindOf Object
    (
        Contains Number State;
        Contains Number Branches;
        Contains Trichoderma Branch1;
        Contains Trichoderma Branch2;
        Contains Trichoderma Branch3;
        Contains Number Diameter;
        Contains Number Length;
        Contains Number Width;
        Method Grow (
                (State.== 0).is True (BranchConidia) ();
                (State.== 1).is True (BranchGifa) ();
                (State.== 2).is True (BranchConidiophore) ();
                (State.== 3).is True (BranchPhialida) ();
        );
        Method BranchConidia (
            (Branches.== 0). is True (Branch1.= Trichoderma.New;
                Branch1.Initialize 1;
                Branches.= 1;
                ) ();
        );
                 Method BranchGifa (
            (Branches.== 0). is True (Branch1.= Trichoderma.New;
                Branch1.Initialize 2;
                Branches.= 1;
                ) ();
        );
        Method BranchConidiophore (
            (Branches.== 0). is True (Branch1.= Trichoderma.New;
                Branch1.Initialize 3;
                Branches.= 1;) ();
        );

        Method BranchPhialida (
            (Branches.== 0). is True (Branch1.= Trichoderma.New;
                Branch1.Initialize 0;
                Branches.= 1;
            ) ();
        );
        Method Initialize (Number NewState|
            State.= NewState;
            Branches.= 0;
        );
)
```

Первая строка:

(Pool.Contains **Class** Trichoderma).= **Class**.KindOf Object

создает новый класс и ассоциирует с ним имя – Trichoderma. Этот класс содержит общий план развития и характеристики, свойственные всем организмам данного рода. Он будет служить опорой для всех остальных классов, представляющих виды и другие систематические единицы.

Строки:

Contains **Number** State;
Contains **Number** Branches;
Contains Trichoderma Branch1;
Contains Trichoderma Branch2;
Contains Trichoderma Branch3;
Contains **Number** Diameter;
Contains **Number** Length;
Contains **Number** Width

определяют атрибуты класса. Каждый из этих атрибутов относится к одной клетке колонии.

Contains **Number** State;

атрибут-число, определяющий специализацию (состояние) данной конкретной клетки. Клетка может *спорой* (0), *клеткой гиф* (1), *конидиеносцем* (2), *фиалидой* (3) и т. п.

Contains **Number** Branches

атрибут-число, определяющий число соседей (ветвей) данной клетки.

Contains Trichoderma Branch1;
Contains Trichoderma Branch2;
Contains Trichoderma Branch3;

атрибуты, представляющие ветви (клетки-соседи), произошедшие из данной клетки и также являющиеся клетками Trichoderma.

Contains **Number** Diameter;
Contains **Number** Length;
Contains **Number** Width;

атрибуты, представляющие линейные размеры клетки – диаметр, длину, ширину.

Method Grow (
(State.== 0).is True (BranchConidia) ();
(State.== 1).is True (BranchGifa) ();
(State.== 2).is True (BranchConidiophore) ();
(State.== 3).is True (BranchPhialida) ();
);

Метод (процедура), определяющий последовательность развития клетки в зависимости от ее состояния. Так, если клетка является спорой (состояние 0), то в дальнейшем ее развитие будет определяться методом BranchConidia.

Method BranchConidia (
(Branches.== 0). is True (Branch1.= Trichoderma.New;

```
            Branch1.Initialize 1;
            Branches.= 1;
        ) ();
                    );
```

Метод BranchConidia, определяющий развитие споры, указывает, что спора должна прорастать в одном направлении – в гифу. В методе создается новая клетка (Branch1.= Trichoderma.New;), указывается, что данная клетка будет специализироваться в гифу (Branch1.Initialize 1) и увеличивается число ответвлений исходной клетки-споры (Branches.= 1).

```
Method BranchGifa (
        (Branches.== 0). is True (Branch1.= Trichoderma.New;
            Branch1.Initialize 2;
            Branches.= 1;
        ) ();
                );
```

Метод BranchGifa, определяющий развитие клетки гифы. В данном случае указано, что гифа растет в одном направлении и образует клетки конидиеносца (состояние 2).

```
Method BranchConidiophore (
        (Branches.== 0). is True (Branch1.= Trichoderma.New;
            Branch1.Initialize 3;
            Branches.= 1;) ();
                            );
```

Метод BranchConidiophore определяет, каким образом будет развиваться клетка представляющая конидиеносец, а именно то, что она способна образовывать фиалиду.

```
Method BranchPhialida (
            (Branches.== 0). is True (Branch1.= Trichoderma.New;
            Branch1.Initialize 0;
            Branches.= 1;
            ) ();
                    );
```

Метод BranchPhialida определяет, каким образом будет развиваться клетка, представляющая фиалиду. Можно видеть, что фиалида образует спору, которая может в дальнейшем опять развиваться – прорастать в гифу и т. д.

```
Method Initialize (Number NewState|
        State.= NewState;
        Branches.= 0;
        );
```

Метод Initialize – начальный метод, который вызывается при создании новой клетки. В нем определяется, какого типа это будет клетка (в зависимости от того, из какой клетки она происходит и других условий), а также указывается, что новая клетка первоначально не имеет ответвлений (Branches.= 0).

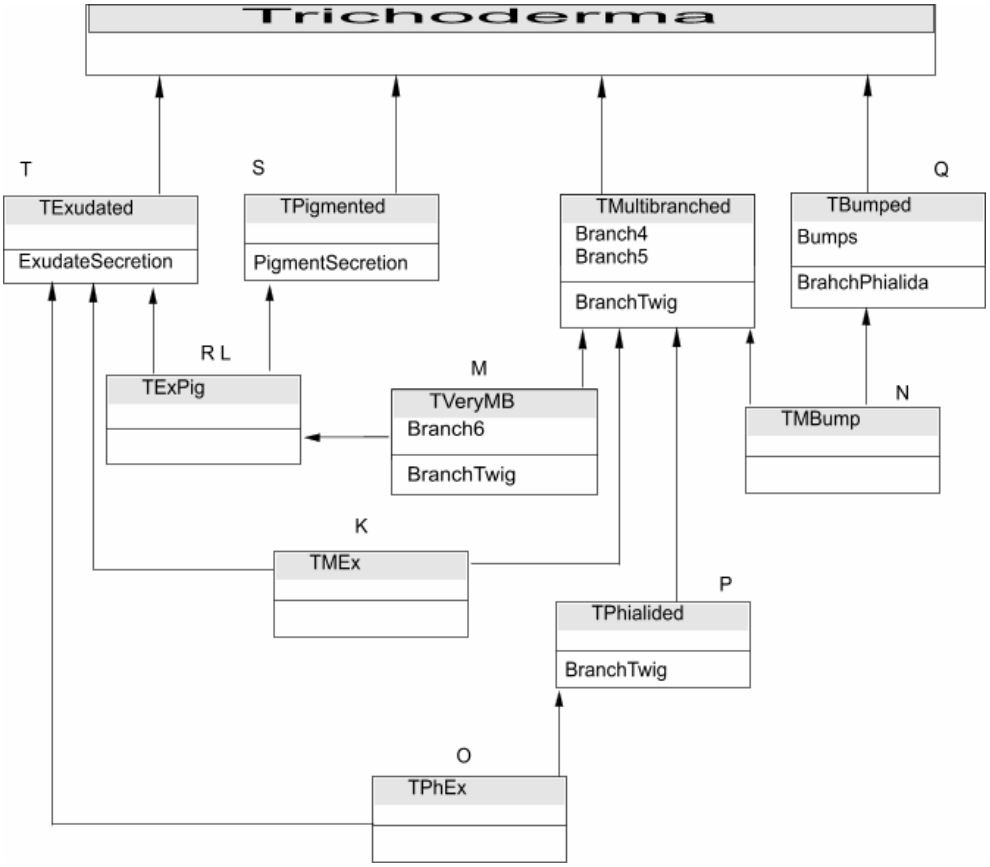

Рис. 6. ОО-версия фрагмента классификации *Trichoderma*. Рядом с каждым классом обозначены буквами (K–T) соответствующие экземпляры (изоляты) *Trichoderma*

Нами было рассмотрено описание класса Trichoderma, который содержит методы и атрибуты общие для всех грибов, относящихся к роду *Trichoderma.* Мы не касались вопросов описания систематических единиц более низкого ранга, в т. ч. видов, организации систематики на базе предложенных описаний. Эти вопросы рассмотрены далее.

Классификация.

На рис. 6 приведен пример фрагмента систематики, полученной с использованием *описаний-программ* и принципа множественного наследования.

Буквами K, L, M, N, O, P, Q, R, S, T обозначены конкретные изоляты *Trichoderma*. Соответствие их видам традиционной систематики показано на рис. 7. Изоляты в данной версии рассматриваются как экземпляры классов.

Рассмотрим в качестве примеров образование классов TPigmented и TExudated. Это наиболее простой пример, иллюстрирующий основные принципы. Класс TPigmented отличается от описания абстрактного класса Trichoderma тем, что он способен выделять в среду пигмент. Точнее, выделять пигмент способны гифы этого класса.

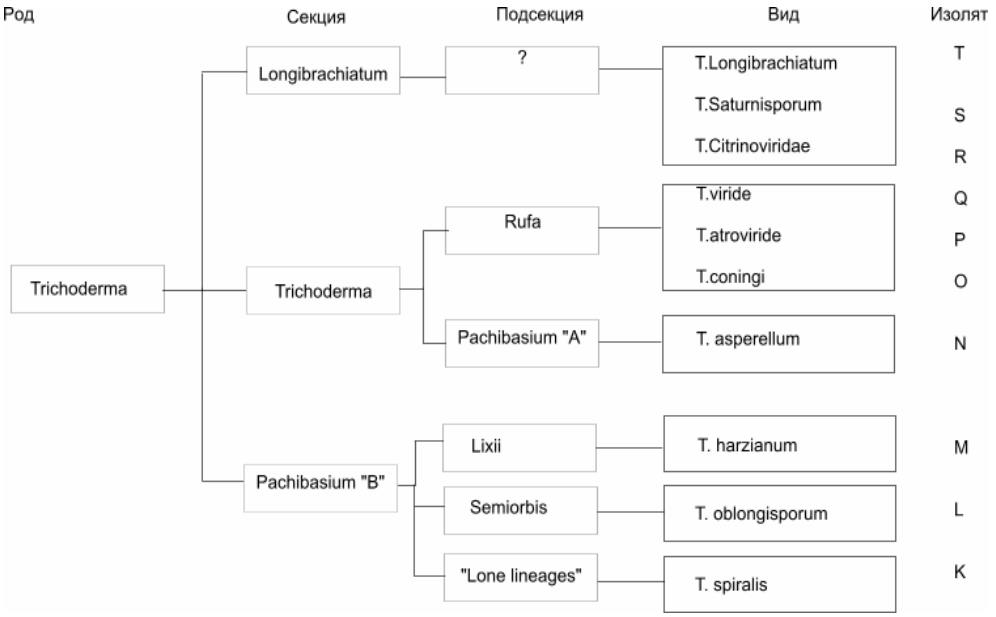

Рис. 7. Фрагмент одной из «традиционных» классификаций рода *Trichoderma* [16]. Справа приведено систематическое положение изолятов, которые использовались при построении ОО-систематики

Это описывается следующим образом:

```
(Pool.Contains Class TPigmented).= Class.KindOf Trichoderma
(
        Method PigmentSecretion (
                                        (State.== 1). is True ((Pool.Contains
                                        Pigment Pigment1).= Pigment.New;)
                    );
        Method Grow (Grow#Parent; PigmentSecretion);
)
```

Класс TPigmented происходит из абстрактного класса Trichoderma (TPigmented.= **Class**.KindOf Trichoderma) и, соответственно, наследует все его характеристики. Поэтому в этом классе специфицируются только его отличительные особенности – способность к образованию пигмента, определяемая методом PigmentSecretion.

Класс TExudated отличается способностью к образованию экссудата. Это описывается следующим образом:

```
(Pool.Contains Class TExudated).= Class.KindOf Trichoderma
(
        Method ExudateSecretion (
                                        (State.== 1). is True ((Pool.Contains Exudate
                                        Exudate1).= Exudate.New;)
                                );
        Method Grow (Grow#Parent; PigmentSecretion);
)
```

Наконец, класс TExPig способен к образованию как пигмента, так и экссудата. Для задания этих свойств необходимо указать, что этот класс является наследником одновременно TExudated и TPigmented:

(Pool.Contains **Class** TExudated).= **Class**.KindOf TExudated TPigmented ()

Таким образом, спецификация этого нового класса состоит всего из одной строки. Чтобы задать описание изолята, обладающего способностью к выделению пигмента и экссудата, достаточно указать, что этот изолят является экземпляром класса TExudatedPigmented:

R.= TExudatedPigmented.New

Для сравнения на рис. 7 приведен один из существующих вариантов «стандартной» биологической классификации *Trichoderma*. Эта версия классификации содержит 19 абстракций на 10 экземпляров, из них 10 видов, 1 род и 8 промежуточных таксонов. Различные по смыслу таксоны имеют одинаковые названия (*Trichoderma* – одновременно род и секция). При этом рисунок не содержит никакой информации о смысле различий между таксонами.

ОО-версия классификации, приведенная ранее (рис. 6), содержит 11 абстракций на 10 экземпляров, из которых 9 видов, 1 род и 1 промежуточный таксон. Между тем, как можно видеть из рисунка, ОО-классификация описывает взаимоотношения между рассматриваемыми организмами более подробно. В частности, в «стандартной» версии виды, к которым относятся изоляты T, S и R, поставлены в один ряд, и дальнейшая информация об их взаимоотношениях отсутствует, в то время как в ОО-версии класс TExPig (R) является наследником класса TPigmented (S) и TExudated (T), из чего сразу следует, что R имеет признаки как S, так и T.

Заключение

Нами была рассмотрена возможность использования подходов, развитых в рамках объектно-ориентированного программирования для классификации биологических объектов. В результате данного рассмотрения мы нашли, что:

1. Организм или таксон могут быть специфицированы не в принятой обычно форме словесного описания, а с помощью программы его развития, записанной на объектно-ориентированном языке программирования. Такая спецификация является более строгой, не допускает неоднозначных толкований, отражает факт динамического развития живых организмов, а кроме того, в отличие от пассивного текста, оно интерактивно.

2. С помощью объектно-ориентированного подхода возможно специфицировать не только отдельные организмы, но и целые таксоны, которые могут рассматриваться как классы ООП. Механизм наследования позволяет установить связи между этими таксонами. Таким образом, можно построить объектно-ориентированную классификацию.

3. ОО-классификация лучше отражает биологическую действительность, нежели «стандартная» биологическая систематика, поскольку допускает множественное наследование, различную длину деревьев наследования, дает однозначное определение концепции вида. ОО-классификация основывается на

программах, которые могут включать в себя любое количество качественных и количественных признаков.

4. Расстояние между таксонами в ОО-классификации могут быть оценены с помощью метрик, разработанных в рамках объектно-ориентированного программирования. Эти же метрики могут быть использованы для оценки необходимости выделения того или иного таксона, а также для сравнения качества построения различных ОО-классификаций.

Пример ОО-описания и классификации продемонстрирован на примере грибов рода *Trichoderma.* Мы нашли, что для рассмотренного набора видов ОО-классификация требует введения меньшего числа абстракций (таксонов) и в то же время лучше демонстрирует взаимоотношения между организмами.

Summary

D.S. Tarasov, N.I. Akberova, R.I. Tuhbatova, F.K. Alimova. Object-oriented programming and taxonomy.

The article describes a possibility of application of object oriented programming concepts to the problems of microbial taxonomy, using classification of *Trichoderma* microscopic fungi genus as example. The work shows how information about individual isolate can be described in form of ontogenetic development program written in object-oriented programming language and proposes subsequent use of such programs for the purpose of species classification and identification. Arguments are presented for proving that object-oriented taxonomy is closer to biological reality because of its support for multiple inheritance, inheritance trees of different length and because it gives a non-ambiguous definition for the concept of specie.

Литература

1. *Muggleton S.H.* Exceeding human limits // Nature. – 2006. – V. 440. – P. 409–410.
2. *Ji S.* The cell as the smallest DNA-based molecular computer // Biosystems. – 1999. – V. 52, No 1–3. – P. 123–133.
3. *Booch G.* Object-oriented design with applications. – Benjamin-Cummins, Redwood City, CA, 1991. – 580 p.
4. *Booch G.* Object Solutions: Managing the Object-Oriented Project, Pub. Addison – Wesley, Menlo Park, 1996.
5. *Birtwistle G.M., Dahl O.-J., Myhrhaug B., Nygaard B.* SIMULA Begin. – Philadelphia, PA: Auerbach Publishers Inc, 1973.
6. *Gosling J, Joy B, Steele G, Bracha G.* The Java language specification, second edition. – Addison-Wesley, 2000. – 505 p.
7. *Goldberg A., Kay A.* Smalltalk-72 instruction manual // Technical Report SSL 76-6, Learning Research Group, Xerox Palo, Alto Research Center, 1976.
8. *Stroustrup B.* The C++ Programming Language. – Addison Wesley, 1985.
9. *Sutherst R.W.* Implications of global change and climate variability for vector-borne diseases: generic approaches to impact assessments // Int. J. Parasitol. – 1998. – V. 6. – P. 935–945.
10. *Roux-Rouquie M., Caritey N., Gaubert L., Rosenthal-Sabroux C.* Using the Unified Modelling Language (UML) to guide the systemic description of biological processes and systems // Biosystems. – 2004. – V. 1–3, No 75. – P. 3–14.

11. *McCabe T.* A complexity measure // IEEE Trans. On Softw. Eng. – 1976. – V. 2, No 4. – P. 308–320.
12. *Chidamber S. R., Kemerer C.F.* Metrics suite for object oriented design // IEEE Trans. On Softw. Eng. – 1994. – V. 20, No 6. – P. 476–493.
13. *Henderson-Sellers B.* Object-oriented metrics: measures of complexity. – Prentice-Hall, 1996. – P. 142–147.
14. *Дьяков Ю.Т., Сергеев Ю.В.* Новое в систематике и номенклатуре грибов – М.: Изд-во Нац. акад. микологии; Медицина для всех, 2003. – С. 219–222.
15. *Kubicek C.P., Harman G.E.* Trichoderma and Gliocladium // Basic Biology, Taxonomy and Genetics. – London: Taylor and Francis, 1998. – V. 1. – P. 278.
16. *Chaverri P., Castlebury L.A., Overton B.E., Samuels G.J.* Hypocrea-Trichoderma: species with conidiophore elongations and green conidia // Mycologia. – 2003. – V. 95, No 6. – P. 1100–1140.
17. *Тарасов Д.С., Тухбатова Р.И., Акберова Н.И., Алимова Ф.К.* Формат представления биологических описаний гриба и его применение на примере рода Trichoderma // Вестн. Татарстанского отд. рос. эколог. акад. – 2005. – Т. 24, № 2. – С. 44–49.

Поступила в редакцию
19.05.06

Акберова Наталья Ивановна – кандидат биологических наук, старший научный сотрудник кафедры биохимии Казанского государственного университета.
E-mail: *nakberov@ksu.ru*

Алимова Фарида Кашифовна – кандидат биологических наук, доцент кафедры микробиологии Казанского государственного университета.
E-mail: *falim@ksu.ru*

Тарасов Денис Станиславович – аспирант кафедры генетики Казанского государственного университета.
E-mail: *dtarasov@mntech.v.rusonyx.ru*

Тухбатова Резеда Ильгизовна – студентка кафедры микробиологии Казанского государственного университета.