УДК 681.3

А.Е. ДОРОШЕНКО, О.М. ОВДЕЙ, Е.А. ЯЦЕНКО

ОНТОЛОГИЧЕСКИЕ И АЛГЕБРОАЛГОРИТМИЧЕСКИЕ СРЕДСТВА АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ «ОБЛАЧНЫХ» ПЛАТФОРМ

Аннотация. Предложен подход к автоматизированной разработке программ, основанный на использовании средств онтологий и алгеброалгоритмического инструментария проектирования и синтеза программ. Применение подхода проиллюстрировано на примере разработки параллельной программы из области метеорологического прогнозирования, а также приложения, предназначенного для выполнения созданной программы в «облачной» среде.

Ключевые слова: онтология, алгебра алгоритмов, проектирование и синтез программ, параллельная программа, «облачные» вычисления.

ВВЕДЕНИЕ

В последнее время «облачные» вычисления (cloud computing) [1] становятся все более востребованными. Параллельные вычисления с применением мультипроцессорных средств являются основным источником обеспечения высокой производительности вычислений при решении сложных научно-технических проблем на «облачных» платформах, в частности, из предметной области (ПрО) метеорологического прогнозирования. Программные приложения «облачных» систем используют Интернет-технологии и развернуты на множестве вычислительных узлов и сервисов, которые постоянно изменяются. Разработка таких приложений является достаточно сложной задачей, в связи с чем возникает необходимость создания специальных высокоуровневых средств генерации высокопроизводительных программ для «облачных» платформ.

В работах [2, 3] предложен подход к автоматизированному проектированию программ на основе совместного использования онтологий и средств алгебры алгоритмов. Создана онтология, содержащая основные понятия (структуры данных, операторы и их взаимосвязи), необходимые для описания прикладных алгоритмов и программ. На базе онтологического описания программы выполнена автоматическая генерация начального варианта высокоуровневой спецификации программы с помощью разработанного интегрированного инструментария проектирования и синтеза (ИПС) программ [4, 5]. Спецификация представлена в виде схемы в модифицированных системах алгоритмических алгебр Глушкова (САА-М) [4]. Затем проведено конструирование схемы и генерация текста программы на целевом языке программирования (C++, Java) в системе ИПС.

Целью настоящей статьи является дальнейшее развитие интегрированных онтологических и алгеброалгоритмических средств для автоматизации проектирования программ из ПрО метеорологического прогнозирования для выполнения на «облачной» платформе. Использование разработанных средств проиллюстрировано на примере создания параллельной программы для численного решения задачи конвективной диффузии [6, 7]. Рассмотрено также проектирование приложения, предназначенного для выполнения данной программы в «облачной» среде.

Предлагаемый подход близок к приведенному в работах, посвященных использованию онтологий для описания и разработки программ [8–11]. В частности, в [8] онтологическое представление применяется для моделирования системных требований и автоматической генерации инфраструктуры веб-приложений,

а в [9, 10] онтологии используются для семантического представления сервисов, ресурсов и проектирования приложений для «облачных» платформ. В работе [11] рассматривается автоматическая генерация шаблонов проектирования, используемых в предметно-ориентированных языках программирования, в соответствии с заданной онтологией предметной области.

Преимущество предложенного далее подхода состоит в генерации на основе онтологии промежуточных высокоуровневых алгеброалгоритмических спецификаций (схем) программ, представленных в естественно-лингвистической форме, что облегчает понимание алгоритмов и достижение необходимого качества программ. Другим преимуществом разработанных средств является применение метода диалогового конструирования синтаксически правильных программ [4], который исключает возможность возникновения ошибок в процессе проектирования схем.

1. ПОДХОД К ПРОЕКТИРОВАНИЮ ПРОГРАММ НА ОСНОВЕ ОНТОЛОГИЙ И СРЕДСТВ АЛГЕБРЫ АЛГОРИТМОВ

Для проектирования программ в настоящей работе применяется разработанный в [2, 3] подход, основанный на совместном использовании онтологии проектирования программ, языка САА-М [4] и ИПС [5]. Онтология в [2, 3] построена таким образом, что нижний уровень включает концепции из различных ПрО, а при необходимости ее можно расширить понятиями из новых ПрО. В соответствии с разработанным подходом основными этапами проектирования программы, относящейся к некоторой ПрО, являются следующие:

- добавление в онтологию проектирования программ концепций, необходимых для описания прикладных задач из выбранной ПрО, а также установление связей с соответствующими представлениями в САА-М;
- подготовка онтологического описания разрабатываемой программы;
- генерация начальной высокоуровневой схемы программы в САА-М по ее онтологическому описанию и дальнейшая модификация схемы в системе ИПС;
- генерация текста программы в системе ИПС на целевом языке программирования на основе сконструированной схемы.

Далее в подразд. 1.1 рассмотрена разработанная онтология проектирования программ и кратко изложено ее расширение новыми концепциями. В подразд. 1.2 приведены средства алгебры алгоритмов, которые используются для генерации схем программ на основе созданной онтологии.

1.1. Онтология проектирования программ. Рассматриваемый подход к проектированию программ основан на понятии онтологии. Формальная модель онтологии [12, 13] представляет собой кортеж $O = \langle C, H_C, P_C, I, A \rangle$, где C — конечное множество понятий, называемых также концепциями или классами; H_C — иерархия понятий, т.е. рефлексивное, транзитивное и антисимметричное бинарное отношение $H_C \subseteq C \times C$; $H_C(C_1, C_2)$ означает, что класс C_1 является подклассом C_2; $P_C = \{P \mid P \subseteq C \times C\}$ — множество бинарных отношений между понятиями, называемых свойствами; область определения свойства $P \in P_C$ — множество понятий $Dom(P) = \{C_D \mid (C_D, C_R) \in P\}$, а область значений свойства $P \in P_C$ — множество понятий $Range(P) = \{C_R \mid (C_D, C_R) \in P\}$; I — множество экземпляров понятий из C; A — множество аксиом онтологии.

В работах [2, 3] построена онтология, которая предназначена для описания данных и операторов, а также общей структуры программ из ПрО сортировки массивов. В настоящей статье выполнено дальнейшее расширение разработанной онтологии для проектирования программ из области метеорологического прогнозирования с применением «облачных» вычислений. В качестве инструментального средства разработки онтологии выбрана система Protégé [14], которая использует

язык описания онтологий OWL (Web Ontology Language) [15]. На рис. 1 в виде ориентированного графа представлена иерархия классов разработанной онтологии. Узлами графа являются концепции, а дугами показаны отношения наследования между ними. Рассмотрим назначение основных концепций.

Подклассы класса Data представляют различные структуры данных, используемые в программах: параметры (Parameter), поля данных (DataField), простые переменные (Variable), массивы (Array) и др. Свойствами этих подклассов являются: hasName — идентификатор переменной; isOfType — тип данных, представляющий собой экземпляр класса DataType. Класс DataField также имеет дополнительное свойство hasAccessModifiers, в котором указываются модификаторы доступа к данным.

Класс Operation содержит операции, применяемые к данным в программе, — операторы и предикаты. Операции могут быть базисными (BasicOperation) или составными (CompoundOperation). Понятие составного элемента соответствует понятию подпрограммы или метода класса в программировании. Каждый базисный элемент соответствует определенному базисному элементу в базе данных системы ИПС [2, 3]. Все операции имеют такие свойства: hasName — идентификатор операции; hasParameter — атрибут, связывающий операцию с экземплярами класса Parameter; hasOutputOfType — тип возвращаемого значения. Кроме того, базисные операции имеют дополнительное свойство hasSAAText — запись операции в САА-М, а составные операции — дополнительное свойство usesOperation, в котором указываются используемые базисные или составные операторы.

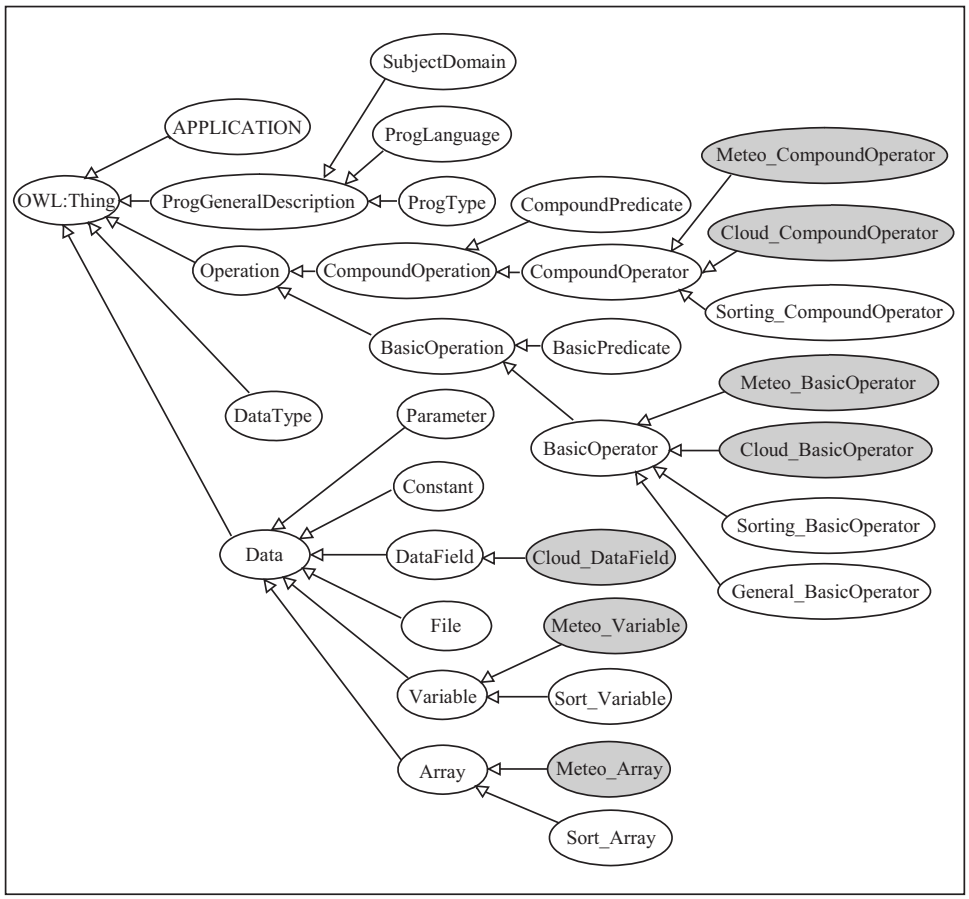

Рис. 1. Иерархия классов онтологии проектирования программ

Таблица 1. Основные свойства класса APPLICATION

Имя свойства	Область значений	Тип отношения
hasName	string	1 : 1
usesData	Data	1 : *N*
hasMethod	CompoundOperation	1 : *N*
hasProgType	ProgType	1 : 1
targetProgLanguage	ProgLanguage	1 : 1
hasSubjectDomain	SubjectDomain	1 : 1

Класс APPLICATION предназначен для спецификации экземпляров программ, которые используют данные, определенные подклассами класса Data, и операции — экземпляры класса Operation. Класс APPLICATION имеет следующие свойства (табл. 1): hasName — имя программы; usesData — совокупность глобальных переменных или полей данных; hasMethod — подпрограммы (составные операторы); hasProgType — тип программы (последовательная, многопоточная или распределенная); targetProgLanguage — целевой язык (Java или C++); hasSubjectDomain — название предметной области. (Примеры экземпляров класса APPLICATION рассмотрены в разд. 2.)

Серым цветом на рис. 1 обозначены добавленные в онтологию новые концепции. Понятия Meteo_Variable, Meteo_Array, Meteo_BasicOperator и Meteo_CompoundOperator представляют данные и операторы, предназначенные для численного решения задач, относящихся к предметной области метеорологического прогнозирования. Экземпляры этих концепций ориентированы на решение задачи конвективной диффузии [6, 7], возникающей при математическом моделировании процессов в атмосфере (см. подразд. 2.1).

В онтологию также добавлены понятия, ориентированные на «облачные» вычисления [1], т.е. модель распределенной обработки данных, в которой вычислительные ресурсы (например, серверы, устройства хранения данных, приложения) предоставляются пользователю как сервис через сеть Интернет. Одной из основных технологий, положенных в основу «облачных» платформ, является виртуализация [16] — предоставление вычислительных ресурсов, не зависящее от аппаратной реализации и обеспечивающее логическую изоляцию вычислительных процессов, выполняемых на одном физическом ресурсе. В настоящее время в разработанную онтологию включены концепции Cloud_DataField, Cloud_BasicOperator и Cloud_CompoundOperator, представляющие необходимые данные и операторы, предназначенные для выполнения программных приложений на виртуальных машинах (ВМ) «облачной» платформы. (Примеры экземпляров данных концепций рассмотрены в подразд. 2.2.)

Отметим, что при построении онтологии в Protégé для установления связи между элементами онтологии и соответствующими элементами языка САА-М используется свойство «rdfs:comment» (комментарий к концепции) [2, 3].

1.2. Средства алгебры алгоритмов и генерация схем программ на основе разработанной онтологии. Онтологическое описание программ, а именно экземпляры класса APPLICATION рассмотренной ранее онтологии, используется для автоматической генерации с помощью системы ИПС [5] соответствующей начальной спецификации программы, представленной в виде схемы в модифицированных САА. Данные САА-М [4] предназначены для формализованного проектирования последовательных и параллельных программ и являются двухосновной алгеброй $\langle \mathrm{Op}, \mathrm{Pr}; \Omega \rangle$, где Op и Pr — множества операторов и логических условий (предикатов) соответственно; Ω — сигнатура логических и операторных операций. Основными операторными операциями являются следующие:

• *operator*1; *operator*2 — последовательное выполнение операторов (композиция);

• IF (*predicate*) THEN *operator*1 ELSE *operator*2 END IF — операция ветвления;

• WHILE (*predicate*) *operator* END OF LOOP — операция цикла;

• PARALLEL($i = 0, \ldots, n-1$) (*operator*) — параллельное выполнение n операторов (потоков), где i — номер потока;

• WAIT *predicate* — синхронизатор, выполняющий задержку вычислений до тех пор, пока значение условия *predicate* не станет истинным.

Приведенные операции представлены в естественно-лингвистической форме. Спецификации алгоритмов — суперпозиции рассматриваемых операций, а также базисных операторов и предикатов, называются САА-схемами. В работе [17] приведены дополнительные операции САА-М, предназначенные для проектирования параллельных программ для графических процессоров.

Система ИПС используется для автоматизированного конструирования САА-схем и синтеза соответствующих программ на целевом языке программирования [4, 5]. Данная система также выполняет генерацию САА-схем на основе онтологического описания программы. Для генерации схемы на вход ИПС передается OWL-файл онтологии, построенной в Protégé [14]. Генерация схемы осуществляется на основе одного из экземпляров концепции APPLICATION. Вначале обрабатываются значения свойства usesData, т.е. используемые в программе экземпляры данных. Для каждого экземпляра генерируется соответствующий базисный элемент определения данных, который включается в блок описания глобальных переменных схемы алгоритма. Затем обрабатывается свойство hasMethod. Названия, параметры и начальная алгоритмическая реализация составных операций, перечисленных в этом свойстве, вставляются в схему. Под начальной алгоритмической реализацией понимается последовательность вызовов базисных и составных операторов, перечисленных в свойстве usesOperation составного элемента. Подробнее процесс генерации САА-схем рассмотрен в [2, 3].

Сгенерированная начальная САА-схема используется затем как основа для дальнейшего, более детального, конструирования схемы программы в системе ИПС. Построение схем осуществляется путем суперпозиции языковых конструкций САА-М, которые пользователь выбирает из списка, на основе применения метода диалогового конструирования синтаксически правильных программ (ДСП-метода) [4]. В соответствии с упомянутым методом на каждом шаге проектирования система ИПС предоставляет пользователю только те конструкции САА-М, вставка которых в дерево алгоритма не нарушает синтаксической правильности схемы. По сконструированному таким образом алгоритму ИПС выполняет генерацию программы на целевом языке программирования (Java или C++). Описание операций САА-М и базисных элементов, а также их отображение в язык программирования содержится в базе данных системы ИПС [5].

2. ПРИМЕНЕНИЕ ОНТОЛОГИЧЕСКИХ И АЛГЕБРОАЛГОРИТМИЧЕСКИХ СРЕДСТВ ДЛЯ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ПРОГРАММЫ МЕТЕОРОЛОГИЧЕСКОГО ПРОГНОЗИРОВАНИЯ

В данном разделе рассмотрено использование созданной онтологии, средств САА-М и системы ИПС на примере разработки параллельной программы из области метеорологии, а также проектирование программного приложения, предназначенного для выполнения параллельной программы на «облачной» платформе.

Таблица 2. Значения свойств экземпляра программы model

Имя свойства	Значение
hasName	model
usesData	TmKnotsPer12h; M_prm; Subdomains; TmLimCalc; u; v; w; T; d; p
hasMethod	main; CalcParallelPart
hasProgType	Multithreaded
targetProgLanguage	C_plus_plus
hasSubjectDomain	Meteorology

2.1. Проектирование прикладной параллельной программы. В основу рассматриваемой программы метеорологического прогнозирования положена параллельная численная реализация трехмерной задачи конвективной диффузии, возникающей при математическом моделировании атмосферных процессов [6]. Используемая математическая модель представляет собой систему [7] из пяти уравнений в частных производных. Решением системы являются значения u, v, w составляющих вектора скорости ветра, а также абсолютной температуры T и плотности воздуха d. В программе применяется численный модифицированный аддитивно-усредненный метод (МАУМ) [6]. Распараллеливание вычислений осуществляется на трех уровнях: по уравнениям модели, количество которых задается в константе EVL_ MT_ VAL ≤ 5; пространственным направлениям — долготе λ, широте φ и высоте над уровнем моря z; по подобластям, на которые разделяется каждое пространственное направление. Количество подобластей задается в переменной Subdomains ≥ 1. Количество параллельных потоков в программе устанавливается в соответствии с формулой NmbThreads $= 3 *$ Subdomains $*$ EVL_ MT_ VAL.

Для проектирования рассматриваемой программы в разработанной онтологии создан экземпляр класса APPLICATION (см. разд. 1, табл. 1) с помощью системы Protégé [14]. Значения свойств данного экземпляра перечислены в сокращенном виде в табл. 2. В свойстве hasName для экземпляра программы задано имя model.

В свойстве usesData приведены примеры основных переменных программы, представляющие собой экземпляры классов Meteo_Variable и Meteo_Array построенной онтологии, а именно: TmKnotsPer12h — количество точек временной сетки для 12-часового периода; M_prm — m-параметр МАУМ [6]; Subdomains — количество подобластей; TmLimCalc — период времени (в секундах), на который вычисляется прогноз; u, v, w, T, d, p — двумерные массивы для хранения значений составляющих вектора скорости ветра u, v, w, температуры T, плотности воздуха d и атмосферного давления p. Свойство hasMethod содержит перечисление подпрограмм проектируемой программы, которые представляют собой экземпляры класса Meteo_CompoundOperator.

В табл. 3 в качестве примера приведены значения основных свойств подпрограммы main. В свойстве usesOperation перечислены используемые в подпрограмме операции (экземпляры классов Meteo_BasicOperator и Meteo_CompoundOperator). Назначение подпрограммы main состоит в инициализации данных, вызове подпрограммы выполнения параллельных вычислений CalcParallelPart, а также сохранении результирующих данных в выходные файлы.

Таблица 3. Значения основных свойств экземпляра подпрограммы main

Имя свойства	Значение
hasName	main
usesOperation	Set_initial_parameters; Load_the_data_to_arrays_for_meteorological_values; CalcParallelPart; Comparing_interpolated_and_calculated_values; Save_data_to_files_for_3D_convective_diffusion_task

На основе онтологического описания программы model с помощью системы ИПС сгенерирована приведенная далее начальная САА-схема, в которой названия составных операторов от их детализаций отделены цепочками символов =, базисные операторы отмечены двойными кавычками, а базисные условия — одинарными:

```
SAA-SCHEME model ====
     General parallel scheme for solving 3D convective diffusion problem
END OF COMMENTS

"GlobalData"
==== "Declare variables (TmKnotsPer12h, M_prm, Subdomains, TmLimCalc)
      of type (int)";
      "Declare dynamic size two-dimensional arrays (u, v, w, T, d, p) of type (double)";

"main"
==== "Set initial parameters (Subdomains, M_prm, TmKnotsPer12h, TmLimCalc)";
      "Load the data to arrays for meteorological values (u, v, w, T, d, p)";
      "CalcParallelPart";
      "Comparing interpolated and calculated values";
      "Save data to files for 3D convective diffusion task";

"CalcParallelPart"
==== "Initialization of data for parallel part";
      "Initialization of variables and arrays for current thread (proc)";
      "Set actual equality values for (u, v, w, T, d, p) and boundary conditions";
      "Compute the subtasks for current space direction (lambda, fi, z)";
      "Compute the average on the basis of the results for each direction";
      "Storing the results to global arrays";
      "Deallocate the memory for arrays for 3d convective diffusion task";

END OF SAA-SCHEME model
```

Далее разработчик продолжает конструирование приведенной схемы с помощью инструментария ИПС, а именно исправляет подсхему, детализирующую составной оператор CalcParallelPart, т.е. добавляет операторы присваивания, операцию параллельного выполнения, цикл и синхронизаторы. Результирующая подсхема имеет следующий вид:

```
 "CalcParallelPart"
==== "Initialization of data for parallel part";
      "NmbThreads := 3 * Subdomains * EVL_MT_VAL";
       PARALLEL(proc = 0,..., NmbThreads-1)
       (
         "Initialization of variables and arrays for current thread (proc)";
         "(j) := (M_prm)";
         WHILE '(j * tau) <= (TmLimCalc)'
         LOOP
           "Set actual equality values for (u, v , w, T, d, p) and boundary conditions";
           "Compute the subtasks for current space direction (lambda, fi, z)";
           WAIT 'All threads completed work';
           "Compute the average on the basis of the results for each direction";
           WAIT 'All threads completed work';
           “Storing the results to global arrays";
```

```
    WAIT 'All threads completed work';
    "Increase (j) by (M_prm)";
  END OF LOOP;
 "Deallocate the memory for arrays for 3d convective diffusion task"
);
```

На основе построенной САА-схемы model в системе ИПС выполнена генерация текста параллельной многопоточной программы на языке программирования C++ с использованием библиотеки OpenMP [18].

2.2. Проектирование приложения для выполнения параллельной программы на «облачной» платформе. Для автоматизации выполнения созданной параллельной программы model на «облачной» платформе спроектировано программное приложение, названное CloudApplication. Предполагается, что программа model уже размещена и скомпилирована на одной или нескольких ВМ, предоставленных «облачной» платформой. Данные ВМ также называются серверами (servers). Назначение приложения CloudApplication состоит в аутентификации пользователя на упомянутой платформе, запуске параллельной программы на одном из активных серверов, а также предоставлении пользователю результатов выполнения программы в виде архива с выходными файлами программы. Подход проиллюстрирован для случая, когда «облачная» система реализована на основе OpenStack [19] — платформы для построения «облачных» инфраструктур с открытым исходным кодом.

Начало проектирования приложения CloudApplication предполагает создание экземпляра класса APPLICATION и заполнение значений его свойств (табл. 4).

Таблица 4. Значения основных свойств экземпляра программы CloudApplication

Имя свойства	Значение
hasName	CloudApplication
usesData	novaApi; provider; endPoint; identity; credential; serverIPAddress; commandToExecute; outputPath; destPath
hasMethod	main_CloudApplication; initialize_CloudApplication; executeProgramOnFirstActiveServer
targetProgLanguage	Java

В свойстве usesData созданного экземпляра CloudApplication перечислены поля данных основного класса программы — экземпляры класса Cloud_DataField разработанной онтологии. Такими полями являются следующие: novaApi — переменная, предназначенная для доступа к Nova (сервису, управляющему вычислительными ресурсами «облачной» платформы OpenStack); provider — имя сервиса, предоставляемого «облачной» платформой (openstack-nova); endPoint — унифицированный указатель информационного ресурса (URL) для аутентификации пользователя в платформе; identity — имя проекта OpenStack и имя пользователя, credential — пароль для доступа пользователя к платформе; serverIPAddress — IP-адрес активной ВМ (сервера); commandToExecute — команда для выполнения параллельной программы model на ВМ; outputPath — имя архива с выходными файлами программы model; destPath — путь к локальному каталогу пользователя, в который необходимо скопировать архив после выполнения программы.

В свойстве hasMethod перечислены подпрограммы приложения CloudApplication, а именно: main_ CloudApplication — метод main в программе на языке Java; initialize_CloudApplication — метод, в котором выполняются инициализации данных и аутентификация пользователя; executeProgramOnFirstActiveServer — метод,

выполняющий программу на ВМ. Эти подпрограммы являются экземплярами класса Cloud_CompoundOperator созданной онтологии. В табл. 5 в качестве примера приведены значения свойств подпрограммы main_CloudApplication. В свойстве usesOperation перечислены используемые в подпрограмме операции (экземпляры классов Cloud_BasicOperator и Cloud_CompoundOperator).

Таблица 5. Значения основных свойств подпрограммы main_CloudApplication

Имя свойства	Значение
hasName	main_CloudApplication
usesOperation	initialize_CloudApplication; executeProgramOnFirstActiveServer; Release_resources

На основе онтологического описания программы CloudApplication с использованием системы ИПС сгенерирована следующая начальная САА-схема:

```
SAA-SCHEME CloudApplication

"GlobalData"
==== "Declare a variable (novaApi) of type (NovaApi) with access modifiers (private
     static)";
     "Declare variables (provider, endPoint, identity, credential, serverIPAddress,
     commandToExecute, outputPath, destPath) of type (String)
     with access modifiers (private)";

"main_CloudApplication"
==== "initialize_CloudApplication";
     "executeProgramOnFirstActiveServer";
     "Release resources (novaApi)";

"initialize_CloudApplication"
==== "Read input parameters from config file (provider, endPoint, identity, credential,
     serverIPAddress, commandToExecute, outputPath, destPath)";
     "Authenticate against the cloud provider based on parameters
     (provider, endPoint, identity, credential)";

"executeProgramOnFirstActiveServer"
==== "Find the first active server (activeServer)”;
     "Set parameters (serverIPAddress, host, password) for connecting to the server
     (activeServer)";
     "Connect to server (activeServer) with parameters (host, password)";
     "Execute command (commandToExecute) on server (activeServer)";
     "Return archived output files (outputPath, destPath) from server
     (activeServer)";
     "Disconnect from server (activeServer)";

END OF SAA-SCHEME CloudApplication
```

Далее разработчик продолжает конструирование приведенного алгоритма в системе ИПС, а именно, в подсхему для составного оператора executeProgramOnFirstActiveServer добавляет две операции ветвления и оператор вывода сообщения на экран. Исправленная подсхема имеет следующий вид:

```
"executeProgramOnFirstActiveServer"
==== "Find the first active server (activeServer)";
     IF 'Found active server'
     THEN
        "Set parameters (serverIPAddress, host, password) for connecting to the server
        (activeServer)";
```

```
"Connect to server (activeServer) with parameters (host, password)";
IF 'Connected to a server (activeServer)'
THEN "Execute command (commandToExecute) on server (activeServer)";
        "Return archived output files (outputPath, destPath) from server
        (activeServer)";
        "Disconnect from server (activeServer)"
END IF
ELSE "Output the message ("No active server found.")"
END IF
```

На основе построенной САА-схемы CloudApplication в системе ИПС выполняется генерация текста программы на языке программирования Java с использованием библиотеки с открытым кодом jclouds [20]. Данная библиотека представляет собой интерфейс программирования приложений (API) для различных «облачных» платформ — OpenStack, Amazon Web Services, CloudStack и др.

2.3. Результаты численного эксперимента. Разработанная программа CloudApplication применяется для выполнения параллельной программы метеорологического прогнозирования model на ВМ, входящей в состав тестовой «облачной» системы, реализованной на основе OpenStack [19]. Данная ВМ имеет такую конфигурацию аппаратного и программного обеспечения: восемь виртуальных процессоров с частотой 2 ГГц; объем оперативной памяти 1 Гбайт; операционная система Scientific Linux 7.1. Виртуализация основана на использовании KVM [16], являющегося одним из основных программных средств мониторинга ВМ в OpenStack. Описанная ВМ выполнялась на физическом вычислительном узле «облачной» платформы с двумя четырехъядерными процессорами Intel Xeon CPU E5335 с частотой 2 ГГц.

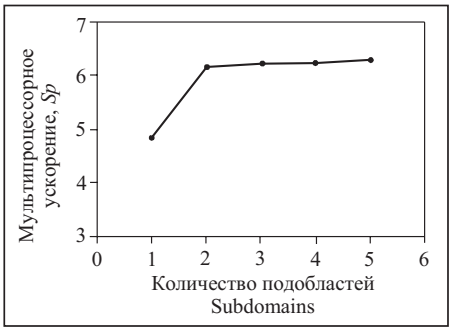

Рис. 2. Зависимость мультипроцессорного ускорения *Sp* от количества подобластей Subdomains для параллельной программы model

В процессе эксперимента принимались такие значения входных параметров программы model (см. подразд. 2.1): период времени, на который осуществлялся прогноз TmLimCalc = 43200 с (12 час); количество точек временной сетки TmKnotsPer12h = 4320; параметр модифицированного аддитивно-усредненного метода [6] M_ prm = 10. Количество подобластей Subdomains изменялось от одной до пяти, а количество параллельных потоков NmbThreads — от девяти до 45 в соответствии с формулой NmbThreads = $= 9 * \text{Subdomains}$. На рис. 2 показаны полученные значения мультипроцессорного ускорения $Sp = T_1 / T_8$, где T_1 и T_8 — среднее время выполнения программы model на одном и восьми виртуальных процессорах соответственно. Максимальное значение ускорения *Sp* получено при Subdomains = 5 и составило 6.31. Эффективность использования процессоров при этом $E = Sp / 8 = 0.79$.

ЗАКЛЮЧЕНИЕ

Предложено развитие ранее разработанного авторами подхода к проектированию программ, основанного на использовании онтологии и инструментальных средств алгебры алгоритмов. Онтология расширена новыми концепциями из ПрО метеорологического прогнозирования, а также программирования «облачных» вычислений. Онтология позволяет описать основные объекты разра-

батываемых программ из выбранной ПрО — структуры данных, обрабатывающие их операторы и взаимосвязи между ними. На основе онтологического описания программы выполняется автоматическая генерация начальной высокоуровневой схемы программы и ее модификация с помощью разработанного инструментария проектирования и синтеза программ. В инструментарии осуществляется также генерация программы на целевом языке программирования. Применение подхода проиллюстрировано на примере разработки параллельной программы для решения трехмерной задачи конвективной диффузии и приложения, предназначенного для выполнения созданной программы в «облачной» среде. Проведен эксперимент по выполнению разработанной параллельной программы на мультипроцессорном узле «облачной» платформы, результаты которого продемонстрировали хороший показатель эффективности распараллеливания вычислений.

СПИСОК ЛИТЕРАТУРЫ

1. Saurabh K. Cloud computing: insights into new-era infrastructure. New Delhi: Wiley India, 2011. 236 p.
2. Дорошенко А.Е., Яценко Е.А. Средства автоматизации разработки параллельных программ на основе онтологий и алгебр алгоритмов. *Проблемы программирования*. 2008. № 4. С. 94–103.
3. Doroshenko A., Yatsenko O. Using ontologies and algebra of algorithms for formalized development of parallel programs. *Fundamenta Informaticae*. 2009. Vol. 93, N 1–3. P. 111–125.
4. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. Алгеброалгоритмические модели и методы параллельного программирования. Киев: Академпериодика, 2007. 631 с.
5. Яценко Е.А. Интеграция инструментальных средств алгебры алгоритмов и переписывания термов для разработки эффективных параллельных программ. *Проблемы программирования*. 2013. № 2. С. 62–70.
6. Прусов В.А., Дорошенко А.Е., Черныш Р.И. Метод численного решения многомерной задачи конвективной диффузии. *Кибернетика и системный анализ*. 2009. № 1. С. 100–107.
7. Черниш Р.І. Паралельна реалізація моделі макромасштабної циркуляції атмосфери. *Вісник Київського національного університету імені Тараса Шевченка: Серія фізико-математичні науки*. 2009. № 2. С. 155–158.
8. Solis J., Pacheco H., Najera K., Estrada H. A MDE framework for semi-automatic development of Web applications. *Proc. 1st International Conference on Model-Driven Engineering and Software Development*. Barcelona, Spain (19–21 February, 2013). Lisbon: SciTePress, 2013. P. 241–246.
9. Martino B.D., Cretella G., Esposito A., Carta G. Semantic representation of cloud services: a case study for OpenStack. *Proc. 7th International Conference on Internet and Distributed Computing Systems (IDCS 2014)*. Calabria, Italy (22–24 September, 2014). Cham: Springer International Publishing, 2014. P. 39–50.
10. Gonidis F., Paraskakis I., Simons A.J.H. On the role of ontologies in the design of service based cloud applications. *Proc. Euro-Par 2014 International Workshops*. Porto, Portugal (25–26 August, 2014). Revised Selected Papers, Part II. P. 1–12.
11. Ojamaa A., Haav H.-M., Penjam J. Semi-automated generation of DSL meta models from formal domain ontologies. *Proc. 5th International Conference on Model & Data Engineering (MEDI 2015)*. Rhodes, Greece (26–28 September, 2015). Cham: Springer International Publishing, 2015. P. 3–15.
12. Maedche A., Zacharias V. Clustering ontology-based metadata in the semantic Web. *Proc. 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2002)*. Helsinki, Finland (19–23 August, 2002). Berlin: Springer, 2002. P. 348–360.
13. Ehrig M., Sure Y. Ontology mapping an integrated approach. *Proc. 1st European Semantic Web Symposium (ESWS 2004)*. Heraklion, Crete (10–12 May, 2004). Berlin: Springer, 2004. P. 76–91.
14. Horridge M. A practical guide to building OWL ontologies using Protégé 4 and CO-ODE tools. Manchester: The University of Manchester, 2011. 107 p.
15. OWL 2 Web ontology language primer (Second edition), URL: https://www.w3.org/2012/pdf/REC-owl2-primer-20121211.pdf.

16. Chen G. KVM open source virtualization for the enterprise and OpenStack clouds. URL: http://openvirtualizationalliance.org/sites/ova/files/resources/files/251810.pdf.
17. Андон Ф.И., Дорошенко А.Е., Бекетов А.Г., Иовчев В.А., Яценко Е.А. Инструментальные средства автоматизации параллельного программирования на основе алгебры алгоритмов. *Кибернетика и системный анализ*. 2015. Т. 51, № 1. С. 162–170.
18. OpenMP Application program interface. Version 4.0. July 2013. URL: http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf.
19. OpenStack open source cloud computing software. URL: http://www.openstack.org.
20. Apache jclouds. The Java multi-cloud toolkit. URL: http://jclouds.apache.org.

Надійшла до редакції 13.05.2016

А.Ю. Дорошенко, О.М. Овдій, О.А. Яценко

ОНТОЛОГІЧНІ ТА АЛГЕБРОАЛГОРИТМІЧНІ ЗАСОБИ АВТОМАТИЗАЦІЇ ПРОЕКТУВАННЯ ПАРАЛЕЛЬНИХ ПРОГРАМ ДЛЯ «ХМАРНИХ» ПЛАТФОРМ

Анотація. Запропоновано підхід до автоматизованого розроблення програм, що ґрунтується на використанні засобів онтологій та алгеброалгоритмічного інструментарію проектування і синтезу програм. Застосування підходу проілюстровано на прикладі розроблення паралельної програми у сфері метеорологічного прогнозування, а також програмного застосунку, призначеного для виконання створеної програми в «хмарному» середовищі.

Ключові слова: онтологія, алгебра алгоритмів, проектування і синтез програм, паралельна програма, «хмарні» обчислення.

A.Yu. Doroshenko, O.M. Ovdii, O.A. Yatsenko

ONTOLOGICAL AND ALGEBRA-ALGORITHMIC TOOLS FOR AUTOMATED DESIGN OF PARALLEL PROGRAMS FOR CLOUD PLATFORMS

Abstract. We propose an approach to automated development of programs, which is based on the use of ontological facilities and algebra-algorithmic tools for design and synthesis of programs. The approach is illustrated on the example of developing a parallel program in the meteorological forecasting domain, as well as software application to execute the developed program on a cloud computing platform.

Keywords: ontology, algebra of algorithms, design and synthesis of programs, parallel program, cloud computing.

Дорошенко Анатолий Ефимович,
доктор физ.-мат. наук, профессор, заведующий отделом Института программных систем НАН Украины и профессор кафедры Национального технического университета Украины «Киевский политехнический институт имени Игоря Сикорского», e-mail: anatoliy.doroshenko@gmail.com.

Овдей Ольга Михайловна,
младший научный сотрудник Института программных систем НАН Украины, Киев,
e-mail: olga.ovdiy@gmail.com.

Яценко Елена Анатольевна,
кандидат физ.-мат. наук, старший научный сотрудник Института программных систем НАН Украины, Киев, e-mail: oayat@ukr.net.