УДК 004.4'23

О.В. МАЛЫШЕВ

РЕЗЕРВЫ ПРОГРАММИРОВАНИЯ ДЕЯТЕЛЬНОСТИ. СИСТЕМА ПОДДЕРЖКИ СОЗДАНИЯ И ИСПОЛЬЗОВАНИЯ ПРОГРАММ ДЕЯТЕЛЬНОСТИ

Abstract. *Programming is one of the types of simulation an activity. There are the followings possibilities to increase the efficiency of programming the activity: 1) revision and clarification of concepts and terminology in the examined subject domain; 2) expansion of possibilities of creation and using of programmatic activity models due to application of special systems and 3) creation and using of the n-dimensional programmatic activity metamodels. The article is devoted to consideration of the second of indicated possibilities.*
Key words: *activity, action, model, metamodel, process, operation, operand, operator, method of execution of operation, program, activity programming.*

Анотація. *Програмування є одним із видів моделювання діяльності. Існують такі можливості підвищення ефективності програмування діяльності: 1) ревізія і уточнення понятійно-термінологічного апарата в даній предметній області; 2) розширення можливостей створення і використання програмних моделей діяльності за рахунок застосування спеціальних систем; 3) створення та використання n-мірних програмних метамоделей діяльності. Стаття присвячена розгляду другої із зазначених можливостей.*
Ключові слова: *діяльність, дія, модель, метамодель, процес, операція, операнд, оператор, спосіб виконання операції, програма, програмування діяльності.*

Аннотация. *Программирование является одним из видов моделирования деятельности. Имеются следующие возможности повышения эффективности программирования деятельности: 1) ревизия и уточнение понятийно-терминологического аппарата в рассматриваемой предметной области; 2) расширение возможностей создания и использования программных моделей деятельности за счет применения специальных систем; 3) создание и использование n-мерных программных метамоделей деятельности. Статья посвящена рассмотрению второй из указанных возможностей.*
Ключевые слова: *деятельность, действие, модель, метамодель, процесс, операция, операнд, оператор, способ выполнения операции, программа, программирование деятельности.*

1. Введение

Программирование деятельности – понятие, которое впервые было введено в научно-технологический обиход в работе [1] и которое, как нам представляется, адекватно отражает суть моделирования аспектов человеческой деятельности, касающихся ее процедурной регламентации, в отличие от понятия «моделирование бизнес-процессов».

В статье [2] были определены направления, работая по которым можно увеличить практическую отдачу от программирования деятельности:

совершенствование понятийно-терминологического аппарата в области создания и использования высокоуровневых программ, регламентирующих осуществление технологических процессов в самых разных сферах человеческой деятельности;

поддержка создания и использования программ деятельности соответствующими инструментальными средствами;

исследование, создание и использование типовых метамоделей систем деятельности, основанных на программировании.

Там же, в [2], были рассмотрены вопросы совершенствования понятийно-терминологического аппарата в области создания и использования программ деятельности. В частности, было показано, что использование понятия «бизнес-процесс» приводит к некоторым неудобствам и противоречиям и предложено использовать термины «операция» (действие), «операнд» (то, над чем производится действие) и «процесс» (процесс осуществления действия). Кроме

того, были введены в рассмотрение некоторые важные типы операций.

Данная статья продолжает начатую тему исследования резервов программирования деятельности и посвящена рассмотрению второго из указанных направлений повышения его эффективности.

2. Феномен программирования

Строго говоря, было бы неправильным пересматривать объем понятия «программа», отправляясь от компьютерных программ в сторону его расширения. Ведь феномен программирования, по-видимому, возник вовсе не в связи с появлением компьютера, а с первыми попытками человека сознательно[1] зафиксировать в виде информации[2] способы осуществления тех или иных типовых действий для собственного употребления и для передачи себе подобным. Так, программами, например, можно считать:

– кулинарный рецепт, если в нем не только перечисляются ингредиенты блюда, но и устанавливается последовательность их подготовки и интеграции в конечный продукт;

– метод изготовления дамасской стали;

– правила совершения какого-либо обряда (ритуала).

Мы, к примеру, с привычным пониманием воспринимаем такие словосочетания, как «программа развития отрасли», «программа телепередач», «программа партии» и т.д. Более того, мы готовы даже согласиться с тем, что, например, «… Библия – это описание программы жизни Человечества…» [3].

Процесс осуществления деятельности видится как продукт взаимодействия реальных объектов, которые можно назвать «участниками» процесса. Они могут иметь различную природу и участвовать в процессе в различных ролях. Например, можно различать «пассивные» объекты (над которыми производится действие)[3] и «активные» объекты (собственно производящие действие)[4]. Но в процессе могут участвовать и информационные объекты – программы, задающие способ осуществления деятельности и являющиеся руководством для «активного» исполнительного органа: человека или компьютера.

Весь вопрос лишь в том, насколько четкой и конкретной является информация, претендующая на роль «программы». И если в обыденной жизни мы зачастую готовы простить «программам» их нечеткость и неконкретность, то компьютеру мы вынуждены давать только предельно формализованные инструкции относительно того, какими информационными объектами он должен манипулировать и в каком порядке.

Представляется необходимым также указать на различие между такими сущностями, как «программа» и «план». Программой мы называем описание способа выполнения операции в терминах вспомогательных операций, в котором могут быть заданы безусловные и условные перехо-

---

[1] Мы вполне допускаем, что человек на подсознательном уровне имеет и использует «программы», определяющие его поведение в типовых ситуациях, однако так глубоко забираться мы (пока) не будем.

[2] Мы по-прежнему [4] считаем, что информация порождается и используется человеком на сознательном уровне.

[3] Если речь идет о типовом действии – операции, то, в нашей терминологии, «пассивные» участники процесса ее выполнения – это операнды.

[4] Такие объекты можно назвать «преобразователями».

ды, параллелизмы, циклы, без указания, кто (что) должен (должно) выполнять эти операции и за какое время. План – это описание «однонаправленной» цепочки действий или параллельных цепочек действий с указанием исполнителей и сроков.

В общем случае план строится на основании программы и в течение своего жизненного цикла имеет разные степени определенности и статусы. Так, в самом начале, когда в план внесена лишь последовательность выполняемых работ, его можно назвать «каркасом», который требует последующей «застройки» внесением различной дополнительной информации: о предоставленных ресурсах, сроках или ограничениях на время выполнения и т.д.

С течением времени план постепенно выполняется (или не выполняется). В него можно вносить отметки о выполнении отдельных работ, их фактические характеристики, ссылки на их результаты, оценки результатов и т.д. Однако, если учесть, что не всегда работы выполняются успешно, могут появиться отметки и о неудачном завершении, о принудительном приостановлении и т.д. Это может повлечь необходимость внесения изменений в план, его модификации. При этом не стоит, по-видимому, «выбрасывать» предыдущие версии плана в корзину – сама история эволюции плана может представлять интерес.

Полностью выполненный план, с первого раза или с модификациями, либо план, попытки выполнения которого предпринимались, но потерпели полную или частичную неудачу, это уже не план, а, так сказать, «факт» – описание (протокол) процесса выполнения работы, выполненное на определенном уровне детализации.

Разумеется, вопросы о том, какими программами деятельности мы располагаем, как соответствуют наши реальные планы нашим программам, как соответствует наша реальная деятельность нашим планам и до какой степени мы за собой «записываем» историю нашей деятельности, могут решаться по-разному. Мы же будем исходить из того, что для многих сфер человеческой деятельности было бы весьма полезным применить программирование в строгом смысле этого слова, обеспечив полное соответствие планов программам, реальной деятельности – планам, а также обеспечив себя в итоге протоколами деятельности, по которым мы бы могли, при необходимости, уточнять не только источники наших неожиданных побед, но и причины неожиданных неудач. Причем, все сказанное должно быть рассмотрено еще и с точки зрения автоматизации.

Соответственно, наша первоочередная цель – повысить степень формализованности программ, устанавливающих порядок выполнения операций, для которых (в отличие от компьютерных программ) входами/выходами (операндами) и вспомогательными объектами являются объекты любой природы (в т.ч. информационные), а средствами преобразования – также объекты любой природы (в т.ч. компьютеры).

В связи с этим возникает необходимость обсудить вопросы, касающиеся взаимоотношений программ с объектами реального мира.

3. Работа с реальными объектами

Для того, чтобы обеспечить возможность работы с конкретным реальным объектом, необходимо его идентифицировать, определить и отслеживать его местоположение, а также организовать управление данным объектом на протяжении всего периода его существования (жизненного цик-

ла).

Любая попытка описать действие (операцию) или процесс выполнения действия (операции) сопряжена с необходимостью идентифицировать объекты-участники процесса и их роли. А идентификация объектов неразрывно связана с их типизацией. Так, фраза «Возьмем стакан с водой» основана на различении типовых объектов «стакан», «вода» и «стакан с водой». Разумеется, возникает вопрос, насколько детально определены эти типы: бывают разные стаканы и разные воды. И сколько нужно налить в стакан воды, чтобы считалось, что он «с водой».

Количество различаемых современным человеком типов объектов вообще огромно, но, если речь идет об описании деятельности о какой-нибудь конкретной предметной области, то проблем с типизацией в ней реальных объектов возникать не должно. Так, «стакан» и «вода» могут быть актуальными типами для одной предметной области (например, кулинарии) и совершенно невостребованными в другой.

Очевидно, что для некоторой предметной области, в которой программируется, планируется, осуществляется и протоколируется деятельность, должен быть налажен учет как различаемых типов объектов, так и самих объектов различаемых типов. Кроме того, должны быть обеспечены возможности «связывания» объектов с типами, распознавания принадлежности объектов к типам, а также расширения множества различаемых типов объектов (в случае необходимости).

Как мы уже установили, операндами могут быть как информационные объекты, так и любые объекты материального мира. Информационные объекты могут иметь как электронные, так и любые другие носители (например, бумажные).

Для информационных объектов, наряду с возможностями описания «обычных» типов данных (integer, float, character, string, boolean и т.д.), структур данных и т.п., должны присутствовать возможности описания типовых документов, в частности, включающих бланки для бумажных документов, формы для электронных документов, а также средства манипулирования с ними.

Для типизации объектов материального мира в рамках конкретной предметной области должны использоваться соответствующие классификаторы.

У читателя, по-видимому, может возникнуть закономерный вопрос, а не поддерживает ли работу с реальными объектами хорошо известное объектно-ориентированное программирование (ООП)? Ответ отрицательный – ООП основан на создании и использовании информационных моделей реальных и абстрактных объектов. В нашем же случае речь идет о манипулировании реальными объектами, а не их моделями.

Пример. Система программирования деятельности для предметной области разработки автоматизированных систем может содержать определение типа объекта «Техническое задание (ТЗ) на создание автоматизированной системы (по ГОСТ 34.602-89)». Это определение, кроме всего прочего, может содержать файл с комментированным шаблоном документа. Программа, регламентирующая процесс разработки ТЗ, создает «пустой» экземпляр данного типа или экземпляр, заполненный шаблоном, доступный для ее вспомогательных операций.

Одной из важных характеристик объекта, ожидающего свое участие в процессах деятель-

ности или уже участвующего в каком-то процессе[5], является место его размещения. В связи с этим возникает задача создания и использования в конкретной предметной области для определенного типа объектов или группы типов «координатной сетки», в терминах которой можно было бы задавать «адрес» объекта и актуализировать текущий адрес в случае его возможного изменения. Это может быть система идентифицированных контейнеров (например, шкафов, полок, папок, ячеек склада), площадок, помещений и т.п.

Программа деятельности может предусматривать итерации - последовательное создание версий одного и того же объекта. При этом может понадобиться, чтобы последующие версии не замещали предыдущие, которые при определенных условиях могут понадобиться в будущем. Таким образом, возникает задача поддержки версий объекта.

Жизненный цикл экземпляра объекта определенного типа, от его создания до уничтожения/утилизации, должен быть управляемым по определенным правилам. В этом случае должна быть разработана программа управления, реализующая эти правила. Такая программа должна предусматривать взаимодействие с другими программами деятельности, регламентирующими использование экземпляра объекта в определенных целях. Исходя из того, что каждый отдельный тип объекта обладает определенной спецификой, для него должен быть разработан соответствующий «словарь», в терминах которого и будет описываться управление.

4. Библиотека типов объектов (операндов, преобразователей)

Итак, мы приходим к тому, что при создании программ деятельности для конкретной предметной области в состав средств программирования должна входить библиотека типов объектов, характерных для данной области. Основным компонентом библиотеки является совокупность описаний (дескрипторов) типов объектов. Дескриптор типа объекта содержит исчерпывающую информацию о данном типе, например:

название и назначение;

код типа, который должен использоваться в программах для ссылки на данный тип;

«словарь» типа – перечень аспектов, существенных для данного типа;

правила, по которым осуществляется создание экземпляра объекта (конструктор);

правила, по которым осуществляется уничтожение/утилизация объекта (деструктор);

правила, по которым осуществляется управление объектом и/или взаимодействие с ним на протяжении его жизненного цикла, а также различного рода другая информация, специфичная для данного типа объекта. Например, для сложных информационных объектов могут указываться формат(ы) их представления, инструментальные средства, с помощью которых можно работать с экземплярами объекта, приводиться шаблон(ы) и прототипы (образцы) и т.д.

Каждый дескриптор типа объекта строится по определенному плану, в соответствии с определенной метамоделью. Возможно, некоторые типы будут иметь общую метамодель описания. В частности, это возможно в том случае, если библиотека поддерживает иерархии типов и несколько типов объектов принадлежат к одному и тому же «надтипу».

Учитывая вышесказанное, библиотека типов объектов должна быть снабжена средствами

[5] Не исключено, что объект может участвовать в нескольких процессах одновременно.

инвентаризации имеющихся в ней дескрипторов типов объектов, метамоделей (каталоги типов объектов и метамоделей), а также средствами интеллектуальной поддержки пользования библиотекой и ее развития. Для иллюстрации специфики этих средств рассмотрим кратко несколько операций, которые они должны поддерживать (табл. 1).

Таблица 1. Примеры операций над описаниями типов объектов

Операция	Назначение, особенности реализации, ограничения
Создать новый	Для создания описания нового типа нужно иметь метамодель его описания. Поэтому программисту необходимо дать возможность ознакомиться с уже имеющимися метамоделями и, если ни одна из них не подходит, создать новую (возможно, путем редактирования одной из имеющихся)
Удалить	Удалить описание типа можно только при условии, если в системе отсутствуют экземпляры данного типа или ссылки на них. Такие экземпляры и/или ссылки могут быть ассоциированы с хранящимися в системе описаниями (протоколами) процессов выполнения программ деятельности
Редактировать	Операция редактирования описания типа в некотором смысле подобна операции удаления, поскольку может привести к существенным изменениям описания, из-за чего может произойти нарушение целостности описаний процессов выполнения программ деятельности, содержащих экземпляры «устаревшего» типа или ссылающихся на таковые. Поэтому, если по каким-то причинам нецелесообразно вводить новый тип, следует отслеживать порождение и использование версий описания типа

5. Библиотека операций

Наряду с библиотекой типов объектов средства поддержки программирования деятельности для определенной предметной области должны включать и развиваемую библиотеку описаний (дескрипторов) операций. Подобно описанию типа объекта, описание операции также строится в соответствии с некоторой метамоделью. В самом общем виде метамодель описания операции мыслится как кортеж $< I, O, \Delta D >$, где I – множество возможных входных операндов (входов), O – множество возможных выходных операндов (выходов), ΔD – «резерв» аспектов доопределения операции.

Конечно же, нельзя обойтись без определений названия операции, ее назначения и кода, который будет использоваться в программах деятельности для ссылки на операцию. При необходимости из ΔD можно «извлекать» и другие характеристики, например:

требования к возможным входам (выходам), удовлетворение которых переводит возможный вход (выход) в допустимый;

зависимость выходов от входов. Можно задать взаимно однозначное соответствие типа $0 = f(i)$, где $o(i)$ принадлежит $O(I)$. Но всегда ли по конкретному входу получается единственный выход?

ресурсы, необходимые для выполнения операции (вспомогательные объекты, преобразователи);

метрики для количественной и качественной оценки входов, выходов, ресурсов;

время, необходимое для выполнения операции, рассчитываемое, например, как функция от входа и объема ресурсов;

способ выполнения.

Международные стандарты (МС) ISO серии 9000, в особенности МС ISO 9001 [5, 6][6], существенно активизировали процесс принятия во внимание существенных характеристик операции[7] ([7, 8], посвященные разработке метамодели операции). Так, к упомянутым выше характеристикам можно добавить:

типовые объекты-участники процесса выполнения операции, их роли и правила их конфигурирования для инициализации процесса;

метод идентификации процессов осуществления действия;

план (метамодель) описания процесса осуществления действия и метод(ы) получения описательной информации (см. выше);

характеристики, используемые для оценки процессов осуществления действия (например, результативность и эффективность[8] [9, 10]).

Извлекая из ΔD те или иные характеристики, мы лишь в той или другой степени уменьшаем объем этого базового элемента. Можно ли исчерпать его полностью? Заданный вопрос можно считать чисто риторическим. Номенклатура характеристик операции, используемых при ее описании, определяется лишь практическими потребностями дальнейшего использования этого описания.

Библиотека операций должна быть снабжена средствами инвентаризации имеющихся в ней дескрипторов операций, метамоделей (каталоги типов объектов и метамоделей), а также средствами интеллектуальной поддержки пользования библиотекой и ее развития. Специфика этих средств во многом аналогична тому, что было сказано для действий над описаниями типов объектов, и определяется прежде всего тем, что в системе могут присутствовать описания (протоколы) процессов выполнения программ деятельности, в которых использовались те или иные опорные или опорно детализируемые операции.

6. Внешнее и внутреннее представления программ деятельности

Для внешнего представления программ деятельности средства поддержки их создания должны предоставлять в распоряжение программиста как минимум один язык программирования, который позволяет использовать типы объектов и операции, имеющиеся в соответствующих библиотеках, а также задавать определенный порядок выполнения операций[9].

Языки программирования деятельности условно можно разделить на текстовые и графические. Хотя, в принципе, программу деятельности можно представить и в табличном виде.

Обычно предпочтение отдается графическим программам как более наглядным. Тот, кто хочет создавать и использовать графические программы деятельности, уже сейчас имеет в своем распоряжении достаточно богатый арсенал средств, основанных на идее графа с нагрузками на

---

[6] В настоящее время действует уже [11].

[7] С точностью до терминологии, принятой нами (в отличие от МС ISO серии 9000).

[8] Согласно [10]:
результативность – степень реализации запланированной деятельности и достижения запланированных результатов;
эффективность – связь между достигнутым результатом и использованными ресурсами.

[9] В частности, должно обеспечиваться распараллеливание действий. Но можно выдвинуть и другие требования, например, обеспечение возможности идентификации определенных событий, наступление которых активизирует выполнение тех или иных фрагментов программы.

вершинах и/или дугах. Собственно, и классификацию этих средств можно осуществить по признаку преимущественного[10] размещения нагрузки – на вершинах или дугах.

Под нагрузкой на вершинах мы понимаем более или менее богатый набор символов (фигур) – прямоугольник, ромб, овал и т.д. – с возможностью помещения в них надписей. Примерами нотаций с преимущественной нагрузкой на вершинах графа являются обычные блок-схемы алгоритмов и программ [12], IDEF3 [13], EPC [14] и т.д. вплоть до нотации BPMN [15], развиваемой в настоящее время консорциумом OASIS (Organization for the Advancement of Structured Information Standards). Для того, чтобы нарисовать блок-схему, можно воспользоваться, например, MS Visio, поддерживающим множество известных нотаций.

Ярким, если не единственным, примером нотации с преимущественной нагрузкой на дугах является нотация, корни которой уходят в Р-технологию программирования [16], прекрасно зарекомендовавшую себя при создании программных средств различного назначения, в частности, систем символьной обработки. В свое время данная нотация была «застандартизована» [17].

Нагрузки на дугах Р-графа делятся на два вида: «условие» (предикат) и «действие». Если «условие» выполняется, движение по дуге разрешено, при этом должно быть выполнено «действие». Синтаксис и семантика нагрузок определялись используемым базовым языком программирования (РТРАН, Паскаль, С и т.д.). Для поддержки программирования в Р-графах были разработаны и применялись графические редакторы, как автономные, так и входящие в состав Р-технологических комплексов программиста (РТК) [18].

Не стоит пытаться отдавать предпочтение той или другой нотации[11]. На наш взгляд, единственной проблемой, с которой постоянно будет сталкиваться программист, является размещение текстовых операторов и других текстовых элементов на графе. Ведь для того, чтобы программа действительно описывала способ выполнения какой-то операции, в ней должны использоваться операторы, ссылающиеся на заранее определенные типы объектов и операции, а также на создаваемые и используемые экземпляры типов объектов. И здесь, к сожалению, графические нотации имеют вполне определенные ограничения на длину текстов, ассоциированных с графическими элементами. Очевидно, что применение для этих ссылок упоминавшихся нами ранее кодов существенно снижают такие характеристики графических программ, как наглядность и узнаваемость описываемых ими действий.

Наконец, отметим, что любая «графическая» программа должна содержать и сугубо «текстовую» часть, не ассоциированную с графом (название, параметры, объявления объектов/данных).

Наряду с внешним представлением программ важным аспектом системы программирования деятельности является внутреннее (машинное) представление программ. Сразу оговоримся, что это ни в коем случае не обычный загрузочный исполняемый код, поскольку каждый очередной оператор программы в общем случае не выполняется непосредственно, а несет информацию для

[10] Как правило, в любой графической нотации нагружаются и вершины, и дуги. Вопрос лишь в том, каков «удельный вес» этих нагрузок.

[11] Собственно говоря, учитывая макрохарактер программ деятельности, мы считаем, что на данном этапе можно использовать для их представления любую нотацию, обеспечивающую необходимый уровень формализации и определенный набор выразительных средств. Это утверждение ни в коей мере не означает, что мы подвергаем сомнению качества обсуждаемых здесь т.н. графических нотаций.

внесения дополнений в план. Варианты реализации внутреннего представления программ могут быть самые разные, например, XML-документы или данные в специально разработанной для этих целей базе и управляемой с помощью конкретной системы управления базами данных.

Для перевода (трансляции) программы из внешнего представления во внутреннее должен быть разработан соответствующий транслятор. Разработку такого транслятора нельзя считать тривиальной задачей, поскольку очень важно, чтобы он «умел» находить в исходных программах как можно больше имеющихся там ошибок.

7. Особенности разработки и отладка программ деятельности

При разработке программ деятельности существенным фактором является «богатство» предоставляемых системой программирования библиотек типов объектов и операций. При этом следует ожидать, что лишь относительно небольшая часть типов объектов и операций могут иметь «универсальный» характер. Большинство же их будет относиться к весьма узким предметным областям. При этом вряд ли можно надеяться на то, что программист, перед которым стоит конкретная задача, сразу найдет даже в специализированной библиотеке все, что ему нужно. Поэтому постоянной и существенной составляющей программирования деятельности будет «обогащение» библиотек типов объектов/данных и операций как средствами обеспечения узнаваемости имеющегося и нужного, так и средствами добавления новых элементов или корректного развития не вполне подходящего[12].

По-видимому, и процесс отладки программ деятельности будет иметь свою специфику. Как известно, для того, чтобы в ответственных случаях перед практическим использованием программы снизить вероятность возникновения нежелательных ситуаций, нужно либо доказать правильность программы, либо хорошенько ее протестировать. Оставляя в стороне возможности «доказательства», укажем на источники трудностей, с которыми, вероятнее всего, придется столкнуться при тестировании программ деятельности из-за их специфики:

работа с реальными объектами;

принципиальная возможность неоднозначности «правильного» результата.

Последнее, на наш взгляд, вообще заставляет задуматься над философскими основами того, что принято называть отладкой…

8. Выполнение программ деятельности

Для обеспечения выполнения программ деятельности требуется наличие двух взаимодействующих механизмов: распределительного и исполнительного.

Распределительный механизм (РМ) предназначен для своевременного обеспечения процесса выполнения программ деятельности ресурсами. Для выполнения программы деятельности может потребоваться целый спектр разнородных ресурсов, в т.ч. вычислительных, из которых будут формироваться экземпляры операндов и преобразователи. В состав РМ должны входить:

классификатор ресурсов;

база данных доступных ресурсов;

[12] Что, как мы уже упоминали, должно быть поддержано средствами идентификации и управления версиями.

блок, принимающий решение о предоставлении ресурсов, частью которого могут являться специально подготовленные люди.

Исполнительный механизм (ИМ) обеспечивает собственно выполнение программ деятельности. Для ИМ входом является внутреннее представление программы деятельности, выходом – внутреннее представление процесса ее выполнения. Выполнение программы начинается с построения в соответствии с логикой программы начальной части «каркаса» плана процесса до тех пор, пока в программе, например, не встретятся условные переходы. Если выполнение программы еще не завершено, запоминается текущая «точка входа» в программу[13].

Полученный «каркас» плана в общем случае нуждается в доопределении: для того, чтобы работа, включенная в план, могла быть выполнена, ей должны быть предоставлены необходимые ресурсы. Кроме того, для работ плана могут быть установлены и определенные ограничения, например, временные. При этом возможны следующие случаи:

работы, способы выполнения которых представлены обычными компьютерными программами, могут немедленно запускаться на выполнение (задачу предоставления ресурсов решает при этом соответствующая служба операционной системы);

для работ, способы выполнения которых представлены программами деятельности, создается следующий уровень декомпозиции процесса, на котором, в свою очередь, начинает строиться «каркас» плана;

для работ, которые требуют ресурсы, отличные от вычислительных, осуществляются запрос к РМ, его удовлетворение и связывание ресурсов с процессом.

Сигнал о завершении очередной работы плана анализируется ИМ, при этом возможны следующие случаи:

результат работы позволяет осуществить условный переход, то есть выбирается нужная ветвь программы, по которой достраивается «каркас» плана;

после завершенной данной работы в плане следуют одна или несколько других работ, то есть инициируется их выполнение;

после завершенной данной работы в плане других работ нет, но есть незавершенные другие работы плана;

после завершенной данной работы в плане других работ нет, а также нет других незавершенных работ плана. Значит, выполнение программы завершено.

Еще одной функцией ИМ является отслеживание соблюдения временных ограничений на выполнение работ. Это необходимо как для работ, способы выполнения которых представлены компьютерными программами (например, для выхода из ситуаций зависания или зацикливания), так и работ, выполняемых людьми – здесь ИМ является «гарантом» соблюдения плана.

ИМ должен обеспечивать одновременное корректное выполнение практически неограниченного количества программ деятельности.

В сущности, пару «РМ + ИМ» можно рассматривать как некоторую «технологическую опе-

[13] Для «параллельных» программ речь может идти не об одной, а нескольких «точках входа».

рационную систему» (ТОС) [19][14], которая, в отличие от обычных операционных систем, активно использует для разных целей человеческие ресурсы.

9. Выводы

В заключение данной статьи представляется целесообразным подытожить все вышесказанное:

1. Феномен программирования имеет глубинную природу и проявляется лишь у человека, способного порождать и использовать информационные объекты, описывающие (фиксирующие) способы осуществления тех или иных определенных видов деятельности. При этом у человека нет причин каким-либо образом заранее вводить ограничения на природу (виды) объектов, участвующих в программируемой им деятельности (входных, выходных, вспомогательных, преобразователей входов в выходы и т.д.).

2. «Обычные» компьютерные программы составляют обширное, но всего лишь подмножество всех потенциально возможных или реальных программ деятельности, поскольку они сами являются информационными объектами на специфических компьютерных носителях, предназначены для обработки таких же информационных объектов и средством их выполнения является компьютер.

3. Программирование деятельности имеет в настоящее время мощный резерв, заключающийся в создании и использовании специальных средств программирования и исполнения программ. При условии наличия таких средств последовательно осуществляемое программирование деятельности в конкретной предметной области должно приводить к ситуации, когда человеку отводится выполнение лишь тех действий, которые не способен выполнить компьютер (принятие нестандартных решений, творчество и т.д.). При этом в идеале полностью исключается навязывание человеку или группам людей выполнение каких-либо программ. Особенно, если эти программы превосходят некоторый (не очень высокий) «порог сложности».

4. Следует ожидать, что программирование деятельности, поддержанное соответствующими средствами, привнесет новое качество в область, которая в настоящее время обычно называется «моделированием бизнес-процессов»[15]. Как известно, такое моделирование проводится, по крайней мере, либо с целью изучения какой-либо деятельности для ее последующей автоматизации, либо для регламентации какой-либо деятельности. В первом случае «модель» есть вспомогательный, промежуточный продукт, как правило, не имеющий самостоятельной ценности. Во втором случае «модель» есть конечный продукт, по существу, программа, предназначенная для выполнения людьми. Программирование же деятельности в описанном нами смысле в обоих случаях приводит к конечному продукту, который может выполняться с помощью компьютера.

При этом автор не намерен никого лишать права свободно размышлять над способами осуществления той или иной деятельности с карандашом в руке или с компьютером, тем более, налагать ответственность за результаты этих размышлений. Единственное, чего он хочет, это сформировать у как можно большего числа потенциально заинтересованных лиц понимание того,

[14] Вариант описываемой ТОС был разработан в результате проведения цикла работ [20–22] и, в частности, демонстрировался на международной выставке «Технова» (Австрия, г. Грац, 1991 г.).

[15] Читателю настоятельно рекомендуется ознакомиться с критикой данного понятия, изложенной автором данной статьи в [2].

что упомянутые результаты, если они плохие (расплывчатые, противоречивые, некорректные, слишком сложные), то они либо малополезны, либо способны нанести даже вред. Если же они хорошие (четкие, целостные, правильные, прозрачные), то есть принципиальная возможность, при желании или необходимости, применить для их использования нашего старого и доброго друга – компьютер.

СПИСОК ЛИТЕРАТУРЫ

1. Фуксман А.Л. Технологические аспекты создания программных систем / Фуксман А.Л. – М.: Статистика, 1979. – 184 с.
2. Малышев О.В. Резервы программирования деятельности. Терминология / О.В. Малышев // Математичні машини і системи. – 2010. – № 1. – С. 150 – 161.
3. Лазарев А.С. Расшифрованная Библия, или Реквием цивилизации / Лазарев А.С. – К.: А.С.К., 2003. – 1064 с.
4. Малышев О.В. Воплощенное знание / О.В. Малышев // Математичні машини і системи. – 2009. – № 1. – С. 55 – 69.
5. ISO 9001:2000. Quality management systems. Requirements. – International standard. – Third edition. – [Чинний від 2000-12-15]. – 32 p.
6. ISO 9001:2000. Системы менеджмента качества – Требования. – Международный стандарт. – 3-е изд. – [Действующий с 2000-10-15]. – 34 с.
7. Малышев О.В. Реконструкция мета-модели процесса по стандартам ISO серии 9000:2000 / О.В. Малышев // Методы менеджмента качества. – 2004. – № 9. – С. 17 – 20.
8. Малышев О.В. Моделирование процессного подхода в проекте СМК / О.В. Малышев // Корпоративные системы. – 2004. – № 6. – С. 10 – 16.
9. ISO 9000:2005. Quality management systems – Fundamentals and vocabulary. – International standard. – Third edition. – [Чинний від 2005-09-15]. – 38 p.
10. ISO 9000:2005. Системы менеджмента качества. Основные положения и словарь. – Международный стандарт. – 3-е изд. – [Действующий с 2005-09-15]. – 38 с.
11. ISO 9001:2008. Quality management systems. Requirements. – International standard. – Fourth edition. – [Чинний від 2008-11-15]. – P. 36.
12. ГОСТ 19.701-90 ЕСПД ((ИСО 5807-85)). Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Государственный комитет СССР по управлению качеством продукции и стандартам. – 1990. – С. 23.
13. Process Description Capture Method IDEF3 [Електронний ресурс]. – Режим доступу: http://www.idef.com/IDEF3.html.
14. Event-Driven Process Chain [Електронний ресурс]. – Режим доступу: http://www.google.com.ua/search?hl=uk&q=event-driven+process+chain&meta=&aq=f&oq=.
15. Business Process Modeling Notation (BPMN). Version 1.2 OMG Document Number: formal/2009-01-03 Standard document URL [Електронний ресурс]. – Режим доступу: http://www.omg.org/spec/BPMN/1.2.
16. Глушков В.М. Технология программирования и проблемы ее автоматизации / В.М. Глушков, И.В. Вельбицкий // УСиМ. – 1976. – № 6. – С. 75 – 93.
17. ГОСТ 19.005-85 Единая система программной документации. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения. – М.: Государственный комитет СССР по стандартам. – 1985. – С. 13.
18. Вельбицкий И.В. Технологический комплекс производства программ на машинах ЕС ЭВМ и БЭСМ-6 / Вельбицкий И.В., Ходаковский В.Н., Шолмов Л.И. – М.: Статистика, 1980. – 246 с.
19. Малышев О.В. Типовая технология программирования. Ч 3: Технологическая операционная система: учебное пособие / Малышев О.В. – Киев: Международный научный центр технологии программирования «Технософт», 1991. – 63 с.
20. Малышев О.В. Типовая Р-технология программирования. Основные проектные решения / О.В. Малышев, И.Е. Щетинин // УСиМ. – 1989. – № 3. – С. 40 – 44.
21. Малышев О.В. Типовая технология программирования. Ч. 1. Базовая система понятий: учебное пособие / О.В. Малышев, И.Е. Щетинин; под ред. О.В. Малышева. – Киев: Международный научный центр технологии программирования «Технософт», 1990. – 68 с.
22. Типовая технология программирования. Ч. 2. База технологических знаний: учебное пособие / О.В. Малышев, И.Е. Щетинин, В.Н. Борчевский, А.А. Зубатенко; под ред. О.В. Малышева. – Киев: Международный научный центр технологии программирования «Технософт», 1991. – 93 с.

Стаття надійшла до редакції 11.11.2009