УДК 681.3.06

В.И. Гриценко, А.В. Анисимов, Н.Д. Пашковец, О.В. Бабак

Общая технология построения бизнес-правил при реинжиниринге программных *legacy*-систем.

Статья продолжает тему реинжиниринга программных *legacy*-систем. В предыдущих публикациях описаны алгоритмы решения возникающих проблем при выполнении реинжиниринга программных *legacy*-систем. В данной статье изложен ряд проблем, с которыми аналитики сталкиваются на стадии комплексного выполнения работ по реинжинирингу упомянутых систем. Рассмотрены вопросы методического характера, описаны конкретные алгоритмы и предложена общая методология автоматизации построения бизнес-правил, посредством которых осуществляется модификация программных *legacy*-систем.

The article continues the subject of reengineering of software legacy-systems. The algorithms of solving the arising problems at the implementation of reengineering of software legacy-systems are described in the previous articles. A number of problems that the analysts should solve at a stage of the complex performance of the works on reengineering of the mentioned systems are stated in the present article. The problems of the methodical nature are considered, the concrete algorithms are described and the general methodology of constructing the business-rules is suggested by means of which the modification of software legacy-systems is performed.

Стаття продовжує тему реінжиніринга програмних *legacy*-систем. У попередніх публікаціях описано алгоритми вирішення проблем, які виникають при виконанні реінжиніринга програмних *legacy*-систем. В даній статті викладено деякі проблеми, які трапляються на стадії комплексного виконання робіт з реінжинірингу згаданих систем. Розглянуто питання методичного характеру, описано конкретні алгоритми і запропоновано загальну методологію автоматизації побудови бізнес-правил, за якими здійснюється модифікація програмних *legacy*-систем.

Введение. В данной статье авторы продолжают рассматривать вопросы реинжиниринга программных *legacy*-систем, начало чему положено в статьях [1–4], где изложен ряд проблем, с решением которых аналитики сталкиваются в процессе выполнения работ по реинжинирингу программных *legacy*-систем. Рассмотрены отдельные вопросы методического характера, описаны конкретные алгоритмы и предложена методология автоматизации построения бизнес-правил (БП), посредством которых осуществляется модификация программных *legacy*-систем.

Перечисленные вопросы рассматриваются с точки зрения их комплексного технологического взаимодействия. Предложена общая технология построения БП в виде взаимосвязанных средств автоматизации построения БП (САПБП) для выполнения реинжиниринга программных *legacy*-систем. САПБП представляют собой интерактивную систему, составными частями которой являются программные средства, предназначенные для автоматизированной обработки отдельных шагов технологического процесса реинжиниринга, выполнение которых в сочетании с методическими и методологическими правилами и указаниями приводит к эффективной модификации *legacy*-систем. На этом этапе последние рассматриваются как единый комплекс программных модулей в сочетании с потоками данных, посредством которых осуществляется обмен бизнес-объектами между компонентами модифицируемого программного комплекса. В связи с этим рассмотрены вопросы передачи бизнес-объектов между модулями системы через внешние файлы как главной составляющей компоненты при отображении проблемной области в программное представление. Описаны принципы настройки применяемых в САПБП парсеров на грамматики языков программирования, использованные в модулях *legacy*-систем. Это позволяет расширить диапазон применения САПБП и провести стандартизацию и унификацию средств автоматизации для использования их в различных проблемных областях.

Постановка задачи

Предлагаемая задача – разработка методов и средств автоматизации построения БП с учетом их комплексного взаимодействия для мо-

делирования программных *legacy*-систем. Написание БП обусловлено необходимостью представления упомянутых программных систем в виде комбинаций БП и их сочетаний. Цель такого представления – поиск путей и методов модификации программной системы исходя из новых требований к ней и применяемым технологиям. Выполнение такого представления ручным способом – занятие довольно трудоемкое. Поэтому целесообразно создание средств автоматизации процессов, выполняемых на стадии моделирования анализируемых программных *legacy*-систем. Средства автоматизации на этом этапе должны предусматривать прежде всего формирование моделей выходных программных продуктов. В настоящее время можно определить два пути решения данной проблемы, а именно: использование механизма дескрипторов и построение матрицы связей между БП.

Использование механизма дескрипторов. Если каждому БП поставить в соответствие специальный дескриптор, который в предикатной формализации описывает отношения между бизнес-терминами, входящими в БП, то возникает возможность построения конструктора моделей анализируемых модулей из сформированных БП. Конструктор, анализируя бизнес-термины дескрипторов на наследование и базируясь на дереве внутримодульных вызовов, сформированного на стадии *Preprocessing* [2], определяет условия и порядок выполнения БП, создавая модель выходного программного продукта. Если при анализе бизнес-терминов и условий перехода возникает неопределенность, конструктор выдает соответствующее сообщение с указанием шага создания модели, на котором эта неопределенность возникла. Аналитик как лицо, принимающее решение, сообщает конструктору свой выбор, с учетом которого конструктор продолжает построение модели. При этом аналитик имеет возможность проследить, насколько удачно его решение и, при необходимости, осуществить возврат и изменить выбор.

Построение матрицы связей БП. Каждому БП ставится в соответствие строка специальной матрицы связывания БП, куда заносятся ссылки на все БП, с которыми связанно данное БП, а также условия этих связей. Эти связи и их условия определяются из дерева внутримодульных вызовов, сформированного на стадии *Preprocessing* [4]. В этом случае модель выходного программного продукта создается в процессе последовательного обхода матрицы в интерактивном режиме по аналогии со случаем использования дескрипторов.

Описанные подходы позволяют в интерактивном режиме выполнять построение и тестирование модели выходного программного продукта. Возврат при моделировании может быть многоуровневым, т.е. аналитику предоставляется возможность вернуться не только к ближайшему, но и к более раннему диалогу и даже к коррекции БП. Это позволяет проверить различные пути формирования модели выходного программного продукта и выбрать наиболее приемлемый к реализации. Анализ построенных моделей выходных программных продуктов дает возможность определиться с архитектурой создаваемой программной системы, в которой учитываются новые требования к системе и особенности применяемых технологических и информационных платформ.

Информационные связи между рабочими таблицами на стадии *Preprocessing*

На рис. 1 приведена схема информационных связей между рабочими таблицами парсеров САПБП, описанных в [1–4]. Предполагается, что анализируемая система включает в себя n модулей, поэтому строится n таблиц переменных модулей (ТПМ), принимаемых аргументов (ТПА), передаваемых параметров (ТПП) и имен наследования (ТИН). Исходя из n ТИН, строится одна для всей системы таблица межмодульных связей (ТМС) и таблица определения порождений (ТОП), а исходя непосредственно из n ТПМ, строится таблица вызываемых модулей (ТВМ) по алгоритмам, описанным в [1–4]. ТИН для каждого модуля системы строится на базе таблицы переменных этого модуля с использованием таблицы передаваемых параметров и таблицы принимаемых аргументов этого модуля. В таком порядке определяется хронологическая последовательность построения и

использования рабочих таблиц парсерами в САПБП.

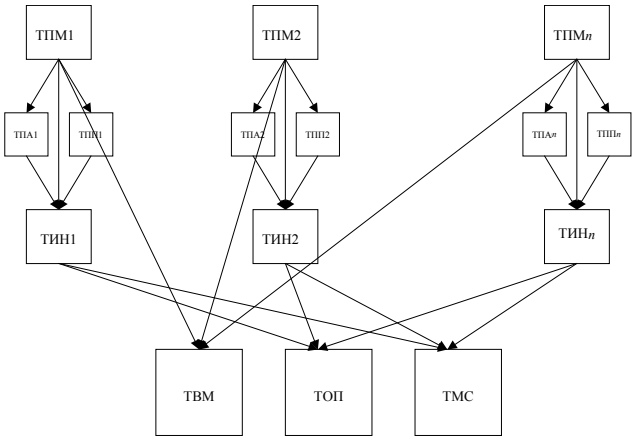

Рис. 1. Схема информационных связей между рабочими таблицами парсеров САПБП при выполнении операций на стадии *Preprocessing*

Определение имен бизнес-объектов, передаваемых между модулями через файлы

Программный модуль шага задания, имя которого указано в операторе *EXEC*, может использовать входные, выходные или промежуточные наборы данных. Последние могут создаваться в этом пункте задания, либо в предыдущих пунктах того же задания, или другими заданиями. Все используемые программным модулем наборы данных должны быть описаны в этом модуле, а в задании описываются операторами *DD*.

Соответствие между набором данных, используемым в программе, и оператором *DD*, описывающим этот набор, устанавливается при помощи имени оператора *DD* (*dd*-имя). В программе на языке *Assembler* это имя указывается в операнде *DDNAME* макрокоманды *DCB*. В программах на языках высокого уровня *dd*-имена наборов данных задаются в статье описания файлов. Имена операторов *DD* в задании соответствуют *dd*-именам наборов данных в программе. Правила этого соответствия устанавливаются соглашениями, связывающими языковые понятия конкретного языка с *dd*-именами наборов данных в языке управления заданиями. Для разных языков программирования эти соглашения различны.

Операторы ввода-вывода языка программирования *FORTRAN* содержат условные номера наборов данных, называемые ссылочными номерами наборов данных. Эти номера включаются в *dd*-имена и записываются в форме *FTxxFyyy*, где *xx* – ссылочный номер набора данных в операторе ввода-вывода, *yyy* – порядковый номер набора данных, используемый для индивидуализации *dd*-имен, имеющих один и тот же ссылочный номер.

В языке *COBOL* в разделе *ENVIRONMENT DIVISION* утверждение *SELECT* содержит имя файла, а утверждение *ASSIGN* – внешнее имя. Эти же имена указываются в задании в качестве *dd*-имен соответствующих им наборов данных.

Имена файлов исходной программы на языке *PL*/1 связываются с *dd*-именами соответствующих им наборов данных при помощи опции *TITLE* утверждения *OPEN*. Если утверждение *OPEN* не содержит этой опции, то компилятор образует *dd*-имя из первых восьми литер имени файла. Такое же *dd*-имя записывается в задании.

Для программ на языке *RPG* наборы данных создаются для всех файлов, определённых в бланке описания файлов. Их *dd*-имена должны совпадать с именами файлов на бланке.

Хотя для разных языков программирования соглашения соответствия имен операторов *DD* в задании и *dd*-имен наборов данных в программе различны, существует общий подход к установлению этого соответствия. Рассмотрим его на примере языка программирования *COBOL*.

В разделе оборудования в параграфе *FILE-CONTROL* утверждение *SELECT* указывает имя файла, обрабатываемого в программе. Каждому файлу, указанному в утверждении *SELECT*, соответствует статья описания файла (*FD*) или статья описания сортируемого файла (*SD*). В этом же параграфе с помощью утверждения *ASSIGN* описывается назначение файлу внешнего носителя, представленного в виде системного имени следующей структуры:

SYSnnn-(класс_устройства)-(шифр_устройства)-(организация_файла)-(имя_файла).

Для рассматриваемой проблемы интерес представляет только имя файла, которое является словом длиной от одной до восьми литер, указывающее внешнее имя, под которым файл известен системе. Отметим, что имя файла в системном имени может отсутствовать.

Имя файла, указанное в утверждении *SELECT* и в статье описания файла (логическое имя файла), совпадает с именем соответствующего ему *DD*-оператора в задании на языке *JCL*, а имя файла из системного имени утверждения *ASSIGN* (физическое имя файла) совпадает со значением параметра *DSNAME* этого *DD*-оператора, представленного в виде простого имени. Значение параметра *DSNAME* (сокр. *DSN*) определяет имя набора данных и называется *ds*-именем (*data set name*) и может быть простым или составным. Простое имя состоит не более чем из восьми букв и цифр и начинается с буквы. Составное – из простых имен, разделенных точками. Оно не может иметь более 44 знаков, включая точки.

При указании раздела библиотеки в *DD*-операторе *ds*-имя имеет следующий вид:

DSNAME = имя библиотеки(имя раздела).

Перед именем временного набора данных (последовательного или библиотечного) записываются знаки && или &. Отсутствие параметра *DSNAME* в *DD*-операторе также означает, что набор данных временный.

В связи с тем, что для решения задач выявления бизнес-терминов, нас не будут интересовать временные (создаваемые и уничтожаемые в пределах задания) наборы данных, а из библиотечных наборов данных – конкретные наборы данных, то получим следующий алгоритм установления соответствия между логическими и физическими именами файлов:

- если в программе приведены утверждения *SELECT* и *ASSIGN*, а также указано имя файла в системном имени в утверждении *ASSIGN*, то физическое имя файла из утверждения *ASSIGN* и логическое имя файла из утверждения *SELECT* заносятся в таблицу имен файлов. Первым в строку этой таблицы записывается физическое имя файла, а далее следуют все логические имена, под которыми этот файл фигурирует в программе;
- если имя файла не указано в системном имени в утверждении *ASSIGN* программного модуля, то физическое имя файла выбирается из параметра *DSNAME* в *DD*-операторе, имя которого соответствует имени файла, указанного в утверждении *SELECT*;
- если в программе отсутствуют утверждения *SELECT* и *ASSIGN*, то логическое имя файла берется из статьи описания файла (*FD*) или из оператора *OPEN*. По этому имени в задании на языке *JCL* отыскивается оператор *DD* с аналогичным именем, в котором физическое имя файла определяется, как значение параметра *DSNAME*.

Если значением параметра *DSNAME* есть ссылка, т.е. представлена в одной из форм:

*.имя-*DD*, *.имя-пункта.имя-*DD*, *.имя-пункта.имя-процедуры.имя-*DD*,

то физическое имя файла используется как значение параметра *DSNAME*, взятое из *DD*-оператора, на который сделана ссылка.

Определим три постулата передачи данных через файлы между модулями, настолько очевидными, что не требуют математического доказательства.

***Постулат* 1**. О передаче данных файла $F1$ от модуля $M1$ к модулю $M2$ можно говорить только в том случае, когда хронологически модуль $M2$ выполняется после модуля $M1$, и при этом файл $F1$ в модуле $M1$ – выходной или обновляемой, а в модуле $M2$ – входной.

***Постулат* 2**. Модули $M1$ и $M2$ должны одновременно принадлежать одному из маршрутов дерева вызова программ, т.е. на дереве вызова должен существовать маршрут выполнения программ, включающий в себя модули $M1$ и $M2$.

***Постулат* 3**. Пересечение множества A выводимых данных в файл $F1$ в модуле $M1$ и множества B вводимых данных из файла $F1$ в модуле $M2$, не должно быть пустым:

$$A \cap B \neq \varnothing.$$

Именно проверку этих условий необходимо выполнить для окончательного установления

факта передачи данных через файлы между двумя модулями.

Подводя итоги, определим следующие этапы и очередность установления соответствия между логическими и физическими именами файлов в анализируемой системе:

В процессе анализа программных модулей системы для каждого модуля строится таблица имен файлов, в которую помещаются все имеющиеся в анализируемом программном модуле сведения о физических и логических именах, используемых в данном модуле. В построенных на этом этапе таблицах имен файлов, как правило, не для всех логических имен файлов будут определены их физические имена. Дополняться таблицы имен файлов физическими их именами будут на этапе анализа заданий на языке *JCL*, соответствующим этим программным модулям, по описанному в [3] алгоритму.

Анализируя информацию из уже заполненных таблиц имен файлов для всех программных модулей *legacy*-системы, определяются имена бизнес-объектов, передаваемых между модулями на уровне файлов.

Настройка САПБП на язык программирования анализируемого модуля

Описанная технологическая схема автоматизации формирования БП предполагает настройку на ряд грамматических конструкций, относящихся к языкам программирования анализируемых модулей, отмеченных при описании операций автоматизации. Значения параметров настройки помещаются в специальный набор настройки на входную грамматику (ННГ), состоящий из разделов, относящихся к конкретным языкам программирования. На сколько языков программирования настроена САПБП, столько и будет разделов в наборе настройки на входную грамматику.

Разделы набора настройки на входную грамматику в свою очередь содержат секции настройки парсеров системы на описанные грамматические конструкции входного языка программирования.

Формирование разделов набора настройки на входную грамматику при настройке САПБП на работу с конкретным языком программирования выполняются в следующей последовательности:

- в первую секцию формируемого раздела набора настройки на входную грамматику записывается ключевое слово *LANG* для указания входного языка программирования, после которого помещается имя входного языка программирования. Например, запись *LANG FORTRAN* обозначает, что данный раздел содержит информацию для настройки системы на входной язык программирования *FORTRAN*. При выполнении начальной настройки САПБП по расширению имени входного модуля определяет название входного языка программирования. После этого управляющая программа САПБП находит в наборе настройки на входную грамматику раздел, содержащий признак *LANG* со стоящим после него именем входного языка программирования. Этот раздел помещается в системную таблицу настройки на грамматику (ТНГ) входного языка программирования, из которой парсеры системы будут извлекать информацию для настройки на грамматические конструкции текущего входного языка программирования модуля;
- в секцию формируемого раздела набора настройки на входную грамматику записывается признак определения лексем обращения к программам в виде *CALL*, после чего перечисляются все ключевые слова, используемые для идентификации операций обращения к программным модулям во входном языке программирования. Например, для языка *COBOL* рассматриваемые параметры настройки будут иметь такой вид: *CALL CALL*, а для языка *Assembler* – соответственно *CALL CALL*, *FETCH*, из чего следует, что в языке *COBOL* парсер построения дерева вызова модулей во входной строке будет искать только лексемы *CALL*, а в языке *Assembler* – лексемы *CALL* и *FETCH*. Следовательно, парсер построения дерева вызова модулей в ТНГ будет искать признак определения лексем обращения к программам, т.е. лексему *CALL*, после чего в свою таблицу настройки (ТН) поместит ключевые слова идентификации операций обращения к программным модулям, взятые из ТНГ;
- в набор настройки на входную грамматику записывается признак определения лексем пе-

редачи данных между модулями в виде *PASS*, после чего перечисляются все ключевые слова, используемые для идентификации операций передачи данных анализируемому модулю. Например, для языка *COBOL* рассматриваемые параметры настройки будут иметь такой вид: *PASS LINKAGE SECTION CURRENT-DATE*, *TIME-OF-DAYCOM-REG*. Следовательно, парсер определения межмодульных связей в ТНГ будет искать признак определения лексем передачи данных, т.е. лексему *PASS*, после чего в свою таблицу настройки поместит ключевые слова идентификации операций передачи данных между модулями, взятые из таблицы настройки на грамматику;

• в набор настройки на входную грамматику записывается признак определения лексем пересылки-присвоения между переменными в модуле в виде *MOVE*, после чего перечисляются все ключевые слова, используемые для идентификации операций пересылки-присвоения. Например, для языка *COBOL* рассматриваемые параметры настройки будут иметь такой вид: *MOVE MOVE*. Значит, парсер отслеживания связей наследования между переменными будет искать в ТН на грамматику признак определения лексем пересылки-присвоения данных, т.е. лексему *MOVE*, после чего в свою ТН поместит ключевые слова идентификации операций пересылки-присвоения данных между переменными, взятые из ТН на грамматику;

• в набор настройки на входную грамматику записывается признак определения лексем, выполняющих функции операторных скобок для внутримодульных императивных программных блоков в модуле в виде слова *BLOCK*, после чего перечисляются все ключевые слова, используемые для идентификации операторных скобок. Например, для языка *COBOL* рассматриваемые параметры настройки будут иметь такой вид: *BLOCK PERFORM*, *EXIT*, *GO TO*, а для языка *FORTRAN* – соответственно *BLOCK SUBROUTINE*, *FUNCTION*, *RETURNSTOP*. Следовательно, парсер выделения программных блоков в ТН на грамматику будет искать признак определения лексем операторных скобок для внутримодульных императивных программных блоков, т.е. лексему *BLOCK*, после чего в свою ТН поместит ключевые слова идентификации операторных скобок, взятые из ТН на грамматику;

• в набор настройки на входную грамматику записывается признак определения завершения вложения *SQL* в виде слова *SQL*, после чего перечисляются все ключевые слова, используемые для идентификации завершения вложения *SQL* в данный язык программирования. Таким образом, парсеры определения межмодульных связей и отслеживания связей наследования в ТН на грамматику будут искать признак определения указателя слова завершения вложения *SQL*, т.е. лексему *SQL*, после чего в свою ТН поместит ключевые слова идентификации завершения вложения *SQL* в данный язык программирования, взятые из таблицы настройки на грамматику;

• в таблицу расширений имен исходных модулей (РИМ), для каждого входного языка программирования записываются все возможные расширения имен, которые встречаются для данного языка программирования.

В табл. 1 приведена структура раздела набора настройки на входную грамматику, перечислены секции раздела и указаны их назначения.

Набор настройки на входную грамматику заполняется системным администратором (системным аналитиком), который вносит для каждого входного языка программирования, с которым предполагается работа САПБП, всю описанную информацию, необходимую для настройки парсеров САПБП.

В табл. 2 приведен пример заполнения секций для раздела языка программирования *COBOL*. Так же заполняются секции для каждого языка программирования, на которые настраивается САПБП.

Начиная работу, САПБП по расширению имени файла, содержащего модуль исходной программы, из таблицы РИМ определяет язык программирования, на котором написан входной модуль. Затем, из набора настройки на входную грамматику читает раздел, соответствующий языку программирования входного модуля, и выполняет настройку всех парсеров на этот язык программирования.

Таблица 1

Имя секции ННГ	Назначение секции	Комментарии
LANG	Указатель входного языка программирования	Помещается имя входного языка программирования
CALL	Указатель лексем обращения к программам	Перечисляются все ключевые слова, используемые для идентификации операций обращения к программным модулям во входном языке программирования
PASS	Указатель лексем передачи данных между модулями	Перечисляются все ключевые слова, используемые для идентификации операций передачи данных между модулями
MOVE	Указатель лексем пересылки-присвоения между переменными	Перечисляются все ключевые слова, используемые для идентификации операций пересылки-присвоения между переменными
BLOCK	Указатель лексем идентификации операторных скобок	Перечисляются все ключевые слова, используемые для идентификации операторных скобок
SQL	Указатель слова завершения вложения *SQL*	Перечисляются все ключевые слова, используемые для идентификации завершения вложения *SQL* в данный язык программирования

Таблица 2

Имя секции ННГ	Ключевые слова					
	1	2	3	4	…	*N*
LANG	*COBOL*					
CALL	*CALL*					
PASS	*LINKAGE SECTION*	*CURRENT-DATE*	*TIME-OF-DAY*	*COM-REG*		
MOVE	*MOVE*					
BLOCK	*PERFORM*	*EXIT*	*GO TO*			
SQL	*END-EXEC*					

Таблица РИМ предварительно заполняется системным администратором (системным аналитиком), который заносит для каждого входного языка программирования все возможные расширения имен, встречающиеся для данного языка программирования. Пример заполнения строки таблицы РИМ для языка *COBOL* и *Assembler* приведен в табл. 3.

Таблица 3

Язык программирования	Расширение файлов			
	1	2	…	*N*
COBOL	*cbl*	*cob*		
Assembler	*asm*			

В случае, когда исходный модуль содержит встроенные фрагменты программ, написанные на другом языке программирования, то их распознавание и пометка происходят на этапе синтаксического анализа. В процессе работы управляющая программа САПБП по расставленным меткам встроенных фрагментов программ выполняет переключение на встроенный язык программирования и извлекает из набора настройки на входную грамматику соответствующий раздел, по данным из которого осуществляет настройку парсеров на обработку встроенного языка программирования. По окончании обработки встроенной части программы САПБП возвращает настройку парсеров системы на охватывающий язык программирования и продолжает обработку модуля исходной программы. При этом все описанные рабочие таблицы являются общими как для охватывающей части обрабатываемого модуля, так и для встроенных частей программы.

На рис. 2 показана описанная схема настройки САПБП на грамматику текущего входного языка программирования.

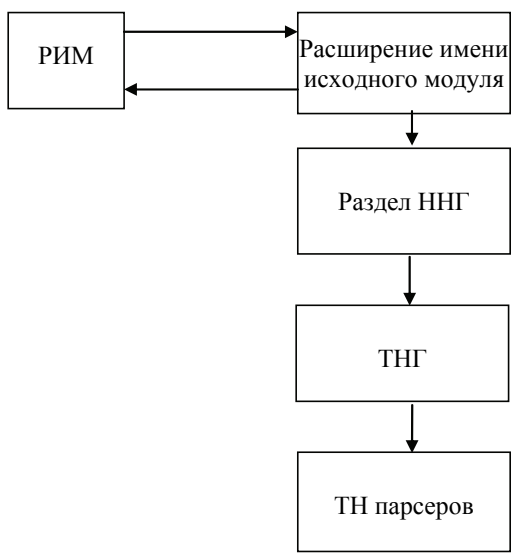

Рис. 2. Схема настройки САПБП на грамматику языка программирования

Результаты

С целью упорядочения и регламентации работы с логическими переменными, используемыми в БП как выражение различных условий, проведена их классификация и выделены сле-

дующие типы используемых логических переменных:

контекстные – логические переменные, не требующие начальной инициализации;

объявляемые – логические переменные, для которых необходима установка начального значения;

первичные – логические переменные начальной активизации БП;

дополнительные – логические переменные с функциями установки дополнительных условий активизации БП;

– для описания всех программных процессов и исходя из выбранной концепции построения БП необходимо выделение трёх типов БП:

статические – для описания линейных и простых процессов ветвления;

полустатические – для описания различного рода вложенных процессов действия и простых циклических процессов;

динамические – для описания вложенных процессов ветвления и вложенных циклических процессов;

– предложен механизм фиксирования и отслеживания состояния логических переменных с целью организации последовательности вызова БП, для чего определены правила и методы построения статических и динамических векторов состояния для текущего отслеживания условий выполнения БП и предложена методика построения матрицы условий запуска БП, используемой с одной стороны, – как карта контроля при написании БП, а с другой – как механизм определения момента активизации БП в процессе функционирования БП;

– определены условия корректного построения комплекса БП и принята методика их проверки;

– разработан механизм отслеживания наследования бизнес-объектов для всей *legacy*-системы в целом с учетом построения цепочек наследования для вложенной формы *SQL* с целью общесистемного расширения таблиц наследования модулей;

– предложена методика разбора *JCL*-заданий, обработка значений параметров, задаваемых в операторах шагов задания и для задания в целом, с целью определения схемы функционирования *legacy*-системы в зависимости от возникающих текущих условий и ситуаций;

– разработаны типовые алгоритмы хронологизации вызова модулей *legacy*-системы, определения межмодульных связей и отслеживания связей наследования между переменными в масштабах шагов задания и для всего задания;

– определены методики и алгоритмы автоматизации выявления паттернов в модулях *legacy*-систем для решения проблемы обобщения и унификации создаваемых БП, а также их классификации в рамках конкретной *legacy*-системы или для ряда схожих систем;

– предложены методики построения специальных структурированных хранилищ БП (репозитории), а также инструментального интерфейса для работы с репозиториями как в автоматизированном, так и в интерактивном режимах;

– разработана общая технология построения БП в виде взаимосвязанных средств автоматизации построения БП (САПБП) для выполнения реинжиниринга программных *legacy*-систем;

– определены принципы настройки применяемых в САПБП парсеров на грамматики языков программирования модулей *legacy*-систем, что позволяет расширить диапазон применения САПБП и провести стандартизацию и унификацию применения средств автоматизации для использования их в различных проблемных областях.

Заключение. Целью авторов было комплексное исследование возможности применения БП определенного вида для описания всех основных программных процессов, а также разработка основных методических концепций и средств автоматизации для написания БП.

Если принять предлагаемую концепцию построения БП, то некоторые затронутые проблемы могут стать предметом отдельного исследования, в результате которого появятся коммерческие приложения, не имеющие прямого отношения к вопросам БП, но решаемые подобными методами. Например, механизм построения динамических векторов как результат воздействия поступающих управляющих сигналов с интерпретацией их как программных пере-

ключателей может быть использован при разработке компактного программного обеспечения для микроконтроллеров или других подобных устройств. Собственно САПБП может стать основой при разработке различного рода программных систем для автоматизации работ по выполнению миграции одних систем программирования в другие или для стыкования программных систем разного технологического уровня.

1. *Реализация* реинжиниринга программных *legacy*-систем / А.В. Анисимов, В.В. Белодед, Н.Д. Пашковец и др. // УСиМ. – № 6. – 2008. – С. 40–48.
2. *Автоматизация* построения бизнес-правил в процессе реинжиниринга программных *legacy*-систем на этапе анализа их функциональных структур / В.И. Гриценко, А.В. Анисимов, В.В. Белодед и др. // УСиМ. – 2009. – № 3. – С. 56–64.
3. *Построение* бизнес-правил для *SQL*-вложений и *JCL*-заданий / В.И. Гриценко, А.В. Анисимов, В.В. Белодед и др. // Там же. – 2009. – № 4. – С. 26–33.
4. *Анализ* программных *legacy*-систем для определения объектов – кандидатов в бизнес-термины / В.И. Гриценко, А.В. Анисимов, Н.Д. Пашковец и др. // Там же. – 2009. – № 5. – С. 69–75.

Поступила 14.08.2009

Тел. для справок: (044) 526-4187 (Киев)

●

Правила подготовки материалов

К рассмотрению принимаются не опубликованные ранее работы по тематике, приведенной на второй странице обложки журнала. Все статьи рецензируются. Решение редколлегии по содержанию каждого номера журнала утверждается ученым советом МНУЦИТиС. Одобренные к печати материалы редактируются. В случае отклонения рукописи один экземпляр и рецензия возвращаются автору. В одном номере журнала публикуется только одна статья автора, в том числе и в соавторстве.

В редакцию необходимо представить:

1. Рукопись (2 экз.), напечатанную через два интервала, объемом не более 16 страниц, на одной стороне листа формата А4 (кегль 12). Один экземпляр должен быть подписан автором(ами).

Страницы оригинала должны быть пронумерованы и иметь поля: левое – 25 мм, правое – 10 мм, верхнее – 20 мм, нижнее – 25 мм.

2. Аннотацию (2 экз.), напечатанную на отдельной странице (до 5 строк) с указанием фамилии автора(ов) и названия статьи на русском, украинском и английском языках; через два интервала.
3. Сопроводительное письмо организации за подписью руководителя.
4. Дискету 3,5″ с текстом статьи, аннотацией и иллюстрациями.
5. Сведения об авторе(ах) – фамилия, имя, отчество, ученая степень, место работы, должность, адрес, телефон, факс, *e-mail*.
6. Копию квитанции о подписке на журнал УСиМ (не менее чем на полгода).

В начале статьи необходимо указать индекс УДК. Используемая литература приводится общим списком в конце статьи в порядке упоминания. Графики, рисунки и таблицы с подписями должны быть распечатаны на отдельных страницах либо выполнены тушью для сканирования.

Для подготовки текста на дискете необходимо использовать редактор *Microsoft Word* любой версии (шрифт *Times New Roman*; кегль 12, интервал двойной; отступ 1 см.), для набора формул – редактор *Microsoft Equation Editor v.* 2.0/3.0 из состава *Microsoft Office*. Иллюстрации могут быть выполнены в любом графическом редакторе.

Материалы можно высылать электронной почтой (по адресу *gor@usm.freenet. kiev.ua*) с обязательным дублированием на бумаге в двух экземплярах или почтой (простое письмо).

В соответствии с постановлением президиума ВАК Украины от 15.01.2003 г. № 7-05/1 «Про підвищення вимог до фахових видань, внесених до переліків ВАК України» статьи, принимаемые к опубликованию, должны состоять из следующих элементов:

- постановка проблемы и ее связь с научными или практическими заданиями;
- анализ последних исследований и публикаций (где начато разрешение данной проблемы), на которые опирается автор;
- выделение неразрешенной части общей проблемы, чему посвящена предлагаемая статья;
- формулировка цели статьи (постановка задачи);
- изложение основного материала исследований с полным обоснованием полученных научных результатов;
- выводы из данного исследования и перспективы дальнейших разработок в данном направлении.

Редакция обращается с просьбой к авторам, желающим опубликовать статью в нашем журнале на украинском или английском языке, прилагать к направляемым материалам русский аналогичный вариант текста.

Редколлегия