Единая универсальная графическая оболочка для записи программ на любых языках. Визуальная технология программирования нового поколения.

Вельбицкий И.В.
Фонд ГЛУШКОВА
Украина
glushkov.org
ivelbit@gmail.com

***Аннотация*— Определяется Визуальная технология программирования нового поколения (ВТР), использующая, так называемые, Р-схемы (ISO/IEC 8631). Р-схемы - это нагруженные по дугам ориентированные графы. Основное отличие ВТР – простота и единая на всем жизненном цикле графическая форма записи программ, которая обеспечивает новые принципы проектирования, отладки, наглядную и компактную (более чем на порядок) запись по сравнению с записью в существующих языках программирования. Ввод графической программы проще и быстрее на порядок. В виду своей простоты доступна широкой массе специалистов, не только программистам. Определяется новая графическая парадигма ВТР, в основе которой единая графическая оболочка для всех языков, состоящая только из одной горизонтальной дуги. Из программирования исключаются все традиционные операторы: goto, if, for, while, и т.д., labels и скобки типа begin-end, {-}. Вводятся новые индустриальные принципы, лингвистическое, параллельное, трехмерное и многомерное графическое программирование, доказательство правильности и самодокументированности графических программ. Широко используется цвет в записи графических программ. Хорошо интегрируется с существующими системами программирования. В настоящее время графическое программирование ВТР реализовано в среде Qt-Criotor C++ (www.glushkov.org). Это позволило вполне обоснованно провести анализ преимуществ ВТР в сравнении с традиционными технологиями, использующими современные языки программирования, UML, ООП, SOA и т.д.**

Ключевые слова - графическое программирование, программирование без традиционных операторов типа goto, if. for, while и т.д., цвет в программировании, метаязыки, сетевые графики, параллельное (трехмерное и многомерное) программирование, самодокументирование программ, доказательство правильности программ.

I. ВВЕДЕНИЕ. ИСТОРИЯ

В конце 60-х годов прошлого столетия нами впервые было введено понятие технологии программирования и ориентация на промышленные принципы разработки бортовых программ для ракетно-космических систем типа Р-36М (САТАНА) бывшего СССР. В результате обобщения этих работ была разработана, так называемая, Р-технология программирования широкого применения [1-3,6]. Эта технология была достаточно широко известна в СССР и странах-членах СЭВ. По вопросам Р–технологии было проведено три всесоюзных и одна международная конференции, десятки специализированных семинаров, опубликовано более 600 работ по разным областям ее применения. Р-технология до распада Союза и СЭВ победила на конкурсе технологий для совместной разработки единой Технологии СЭВ во всех 10 странах Содружества. На нее в 1988 был получен международный стандарт ISO 8631[3].

Сейчас, двадцать лет спустя, работы по Р-технологии возобновились. Проведен ее анализ и сопоставление с тем, что есть в программировании. Пересмотрена концепция с учетом развития техники, новых языков и сред. Осуществлена реализация на современных платформах и обсуждение новой концепции на Международных конференциях [7-9].

В результате был сделан вывод, что новая концепция Р-технологии не только не устарела, но и вписывается в современные тенденции развития программирования, обновляя их, образуя новое поколение визуальной технологии программирования Р-схемами (ВТР), которая на порядок улучшает современные подходы к программированию. Главное в предлагаемой концепции то, что Р-схема – это не новый (еще один) язык, а **универсальная графическая оболочка - единая (!) для** всех известных языков: Fortran, Cobol, C++, Java, Delphi и т.д. В то же время она мощнее и много проще, нагляднее и компактнее этих языков и упрощает существующий процесс и среды разработки программ – IDE.

II. ОПРЕДЕЛЕНИЕ БАЗИСА ВТР

Графическая оболочка нового поколения состоит из схем. Элементарная схема, которая называется *Р-схемой*, состоит из одной горизонтальной дуги между двумя вершинами, Рис.1. Дуга имеет направление только вправо или влево. Из вершины Р-схемы могут выходить любое число дуг вправо и/или влево, рис. 2. Дуги выходящие из вершины просматриваются (читаются, понимаются, анализируются, и выполняются) последовательно сверху вниз и от вершины к вершине по соответствующей стрелке дуги, начиная от первой слева вершины и кончая последней вершиной справа. В Р-схемах вертикальные дуги

являются вспомогательными, они соединяют вершины с горизонтальными дугами, которые являются основными

Рис. 1. Элементарная Р-схема (одна дуга вправо или влево) для записи любой программы на любом языке.

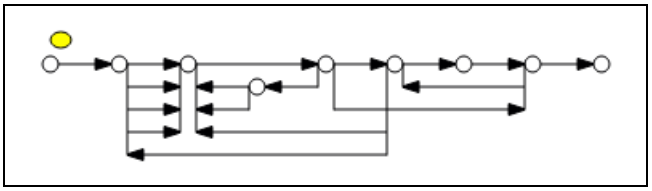

Рис. 2. Запись Р*-схемы 4-х циклов без ограничений числа альтернативных дуг и без деталей реализации (без записи информации на дугах).

и нагружаются информацией. Вершина не имеет имени и задает *состояние* программы или процесса ее разработки. Ввод графа осуществляется только вводом горизонтальных дуг около выделенной одной или двух вершин. Например для ввода Р*-схемы рис.2, состоящей из 13(n) дуг, необходимо 12(n-1) нажатий клавиш мыши или клавиатуры. Первая дуга Р-схемы и надписи на ней (Рис.1) всегда рисуются автоматически.

В Р-схемах на дуге сверху пишется *Условие* прохождения по дуге, а снизу выполняемые при этом *Действия,* Рис.1. Условие и/или Действия могут отсутствовать. На запись Условий и Действий (Рис.1, 3-5) не накладывается никаких ограничений – они могут быть записаны на любом языке: русском, английском, китайском, математическом, языке программирования и т.д. в одну или несколько строк.

Если Условие истинно или отсутствует, то выполняются Действия под этой дугой и происходит переход по стрелке к следующей вершине.

Если Условие ложно, то Действия под дугой не выполняются и рассматривается следующая сверху вниз выходящая из этой вершины дуга, а если ее нет, то дуга, следующая за вершиной, на которую она (дуга с последним ложным условием) указывает.

Если Условие начинается с *ключевого* слова, то это условие является всегда истинной *логической константой Р-схемы* и выполняется особым способом, оговоренном при определении графической оболочки (Рис.4). Как правило, логическая константа *присоединяется* к своему Действию, образуя единый результирующий код, но могут быть более сложные действия (см. Рис.4 вторую колонку справа). Такая возможность обеспечивает эффективное графическое задание описаний данных и метаданных.

Если Условие является произвольным текстом (Рис.5 в пунктирной рамке), то значение условия определяется другой Р-схемой с именем из рамки. По

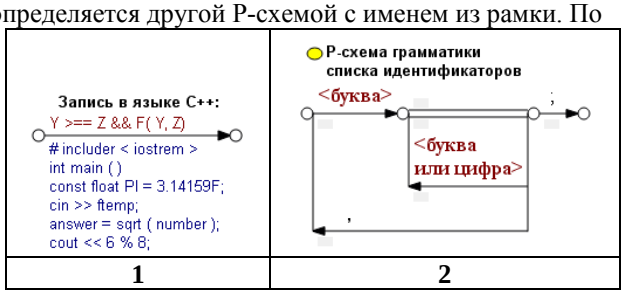

Рис. 3. Р-схема фрагмента программы в языке С++ и грамматики списка идентификаторов.

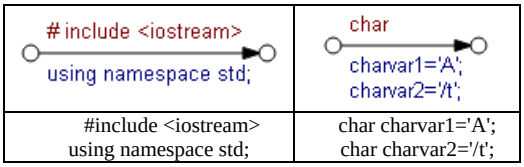

Рис.4. Запись логических констант Р-схем и соответствующих им кодов С++(ниже в колонке)

Рис. 5. Принцип связи произвольных текстов (в пунктирном прямоугольнике) с определением Р-схем.

умолчанию после возврата из другой Р-схемы произвольный текст в пунктирной рамке считается истинным и выполняются действия под ним. Такая возможность обеспечивает эффективную разработку больших программных проектов в едином языке Р-схем.

Любая схема строится из элементарных Р-схем и используется **на всем(!) жизненном цикле** программы. Каждая Р-схема имеет свое имя, которое записывается около желтого эллипса (Рис.5). Новая Р-схема задается двойным кликом левой кнопки мыши на свободном месте Рабочего поля *Графического редактора - GIDE*. Такой граф согласно ISO/IEC 8631 назван Р-схемой. Теоретически Р-схемы эквивалентны машине Тюринга.

III. ПРИМЕРЫ ЗАПИСИ Р-СХЕМ ПРОГРАММ

На рис.6 приведена запись Р-схемы оператора выбора и цикла и соответствующая ей запись в языке С++. Р-схема несравненно нагляднее, в два раза компактнее и вводится в **37** раз быстрее в компьютер (за

только **4** нажатия клавиш). Из записи Р-схемы исключена ½ лишних символов С++ (на Рис.6 выделены красным). Пользователь ВТР не видит правую часть Рис.6, которая приведена лишь как пояснение Р-схемы в традиционных для читателя обозначениях.

Запись в Р-схемах традиционных операторов цикла (рис.7) нагляднее и мощнее хотя бы потому, что дуги для их записи нагружены лишь частично – на некоторых дугах нет условий или нет действий, или – того и другого.

На рис.8 приведено сравнение Р-схем с другими известными графическими методами записи алгоритмов, типа UML, Flow Chart и др. Р-схема единственная из графических методов, которая используется на всем жизненном цикле программ и компактнее всех известных.

Таким образом, ВТР визуализирует прежде всего ту часть существующих традиционных языков программирования, которая служит основным источником запутывания, ошибок и трудностей разработки программ – это так называемые операторы языков. Линейные части записи программ: выражения, функции, операторы присваивания, описания данных, ввода, вывода и др. «безопасные» конструкции языков программирования погружаются в графическую оболочку почти без изменений. Такая запись линейных частей обеспечивает их визуализацию (наглядность, компактность) и преемственность с существующими системами программирования.

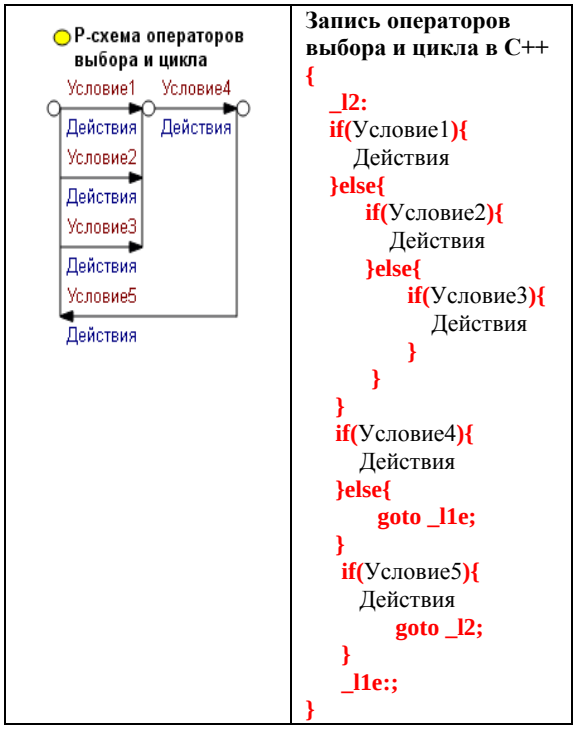

Рис. 6. Запись операторов выбора и цикла в Р-схемах и в С++. Красным выделены лишние символы для Р-схемы.

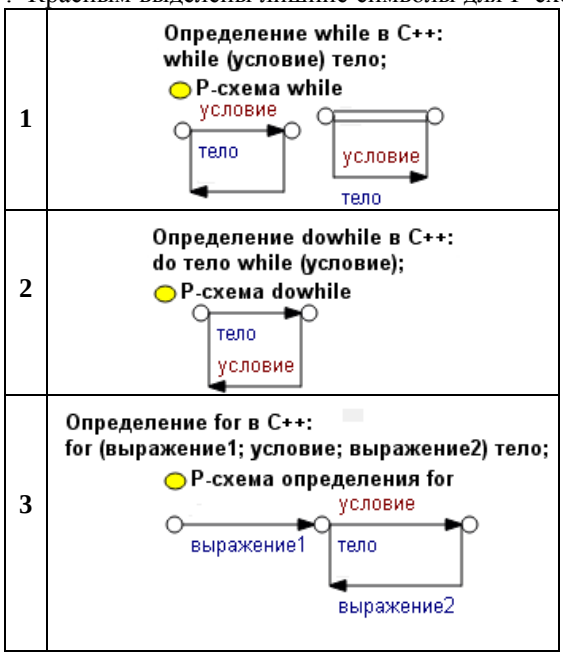

Рис.7. Запись определений цикла в С++ и Р-схемах.

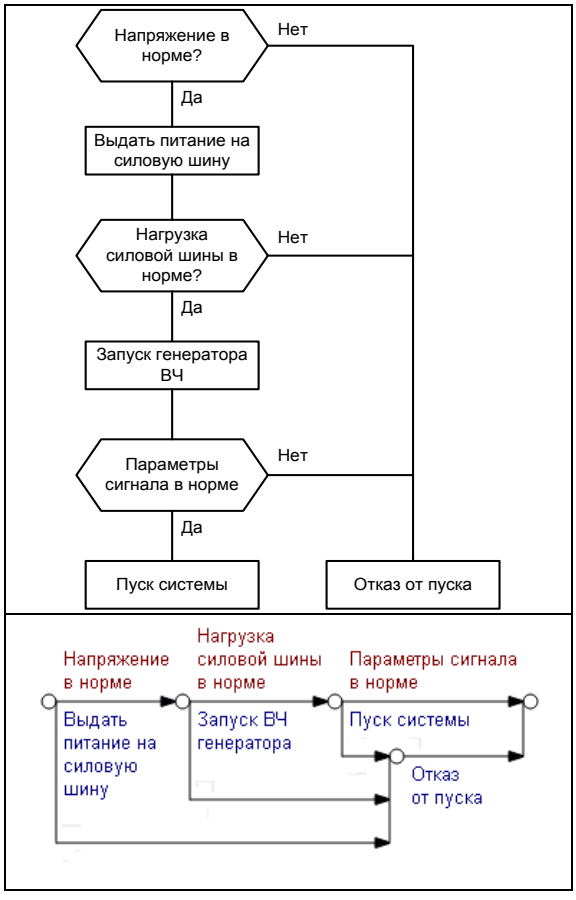

Рис. 8. Фрагмент предстартовой подготовки космического корабля многоразового использования БУРАН, записанный в UML (сверху) и в Р-схемах.

Р-схема (нагруженный по дугам граф) – это универсальный способ задания информации в математике. Исторически такие графы (схемы) давно используются в *Сетевом планировании и управлении*, обеспечивая наиболее наглядное представление о ходе работ, их обеспеченности и завершенности, см. Рис.9. На дуге сверху таких графов обычно записывается название работ: 01, 11, 12 и т.д., а снизу или в скобках (см. первую дугу Р-схемы, Рис.9) – время их выполнения. Использование таких схем в программировании открывает большие перспективы организации планируемой индустрии разработки программных проектов в единой системе обозначений: пожеланий Заказчика, объектов Программы и системы Управления их разработкой.

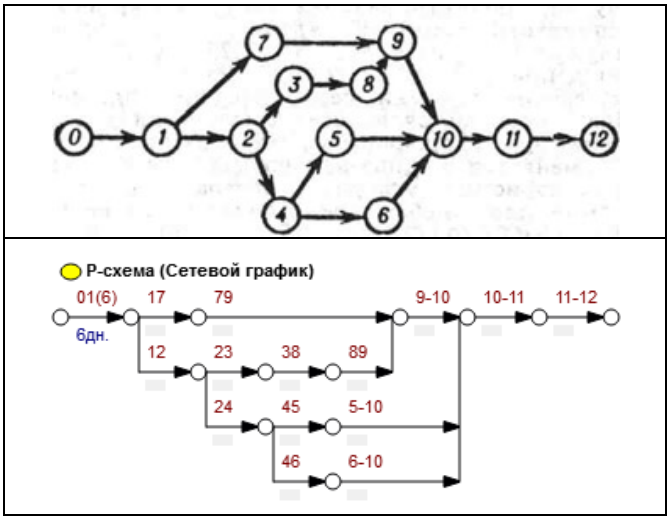

Рис. 9. Запись сетевого графика (сверху) в Р-схемах.

IV. РАСШИРЕНИЕ БАЗИСА

ВТР допускает гибкое развитие изобразительных средств. Например, для изображения графа типа петли используется *специальная* двойная дуга без стрелок (Рис.10), что позволяет более наглядно изобразить оператор типа **while**, рис.7.1 и очень многие ситуации при записи реальных алгоритмов и макроопределений, рис. 3.2.

Вершины Р-схем также могут иметь специальную конфигурацию – квадратик, ромбик, прямоугольник и т.д., Рис.11. Это означает специальное состояние Р-схемы: параллельное выполнение исходящих ветвей дуг, трехмерное и многомерное изображение Р-схем, подключение лингвистического процессора со специальной (в виде грамматики) трактовкой записей на дугах Р-схем см. Рис.3.2 и т.д. Для перехода на специальную конфигурацию вершины или дуги необходимо дважды кликнуть по ним левой кнопкой мыши.

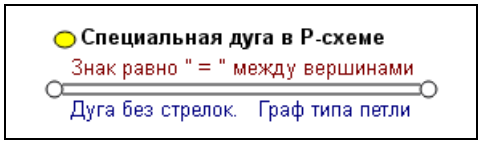

Рис.10. Дуга без стрелок для записи графа типа петли.

Условие Действия Условие Действия	**Квадратик** – безальтернативная запись одиночных дуг, эквивалентна такой последовательности дуг: Условие Действия Условие Действия
condition1 Actions condition2 Actions	**Ромбик** - ожидание исполнения внешнего события указанного в Условиях на дугах исходящих из ромбика;
Условие Действия Условие Действия Условие Действия	**Вертикальный прямоугольник** – параллельное выполнение дуг между прямоугольниками. На Р-схеме параллельное выполнение 3-х дуг.

Рис.11. Примеры специального использования вершин

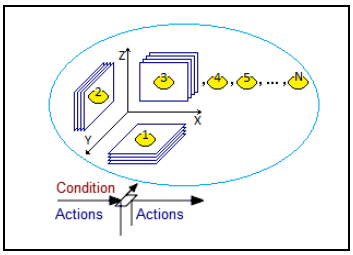

Рис. 12. Принцип организации трехмерных и многомерных вычислений с помощью Р-схем.

Для многих приложений: расчет трехмерных траекторий, параметрическое представление многомерных пространственных поверхностей, линейное программирование, и т.д. удобно перейти на многомерные Р-схемы с вершиной в виде специального параллелограмма, Рис. 12.

В ВТР широко используется цвет для выделения вершин, дуг и записей на дугах, Рис. 13. Обычно *желтым* отмечается эллипс перед названием каждой новой Р-схемы. *Коричневый* цвет используется при записи Условий; *голубой* – при записи Действий. *Красным* на рис.13 отмечен маршрут дуг для автоматической генерации теста; *зеленым* – вершины, где допустимы прерывания работы программы и т.д.

Одной из самых мощных новых возможностей графического программирования, которая отсутствует в

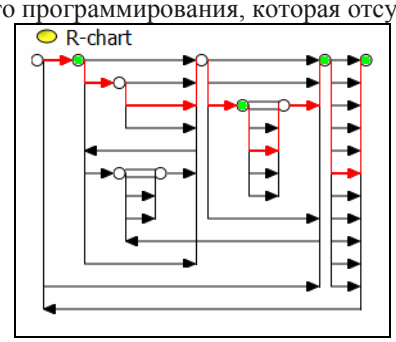

Рис. 13. Использование цвета при записи Р-схемы

традиционном программировании, является использование двух форм графов: Р и Р*. В Р*-схеме удалены все надписи на дугах (детали реализации) из Р-схемы, и тем самым значительно увеличена ее (Р*) компактность см. рис. 2, 13 и 14-15 ниже. Например, Р*-программа на Рис. 13 в **65 раз компактнее** традиционной ее записи в языке С++. Такое написание позволяет просматривать программу, ее логику и структуру, как бы сверху – «с высоты птичьего полета». Это помогает программисту определять стратегию продолжения работ и проводить обсуждение программы без лишних деталей реализации, подключая их (детали реализации) по мере необходимости.

V. РЕАЛЬНЫЙ ПРИМЕР

Ниже на Рис.14, 15 приведены Р, Р*-схемы реальной программы в языке Delphi для обработки некоторого класса специальных файлов [4]. Р-схемы примера демонстрирует самодокументируемость технологии нового поколения. На первой Р-схеме изображена архитектура (оглавление) всей программы и порядок ее определения. В первой колонке написано ее имя и характеристики: **type=class**. Во второй – определены данные (**//поля**) и т.д. Все методы (имена) в программе запоминаются в Графической среде разработки – GIDE и в дальнейшем задают последовательность (сетевой график) их определения. Такого нет в традиционном программировании.

Традиционная запись этой программы в языке Delphi [4 стр.17-19] занимает 2.5 страницы, 121 строку, 2080 символов без комментариев. Графическая программа (Рис.13) содержит 24 (в **5.3** раза меньше) эквивалент текстовых строк в этой статье, 1220 (в **1.7** раза меньше) символов и **37** горизонтальных дуг, для ввода которых потребовалось в **23.2** раза меньше нажатий клавиш на клавиатуру. В результате из традиционной записи этой программы удалено **860** (**41.3%**) символов, которые стали лишними (мусор) для графической записи Р-схем.

В графическом программировании с помощью только одной схемы (дуги) впервые можно изобразить структуру (идею, архитектуру) программного проекта, без деталей реализации, рис.15. Такая запись меньше в **45.1** раза традиционной записи этой программы в

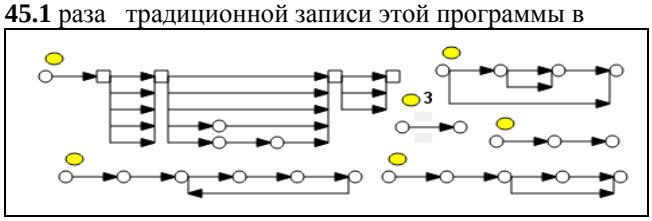

Рис.15. Р*- схема той же программы (Рис.14) без деталей реализации **компактнее в 45.1** раза.

языке Delphi. Чем больше реальная программа, тем больше превосходство компактной графической (Р*) программы в 50, 100 и более раз. Компактная Р*-схема облегчает разработку, обсуждение, защиту и демонстрацию проекта. Она впервые позволяет начать разработку от идеи, архитектуры, эскиза проекта, постепенно наращивая и развивая такую основу (скелет) деталями и возможностями реализации. Работа может проводиться в любой концепции (ООП, КОП и т.д.) и любом языке, из которого используются только простейшие, линейные установившиеся в классической математике объекты и понятия типа функций, процедур, выражений, формул, структур данных и т.д. Ввод такой программы в машину осуществляется быстрее в **15.4** раза. Для ввода программы рис.15 и 13, например, требуется **всего 37** (число горизонтальных дуг) нажатий кнопок мыши или клавиатуры.

VI. ГРАФИЧЕСКАЯ ПАРАДИГМА ВТР

Первая программа была написана немногим более 65 лет тому назад. Это небольшой срок для эволюции формирования фундаментальных принципов программирования.

Существующая стратегия развития традиционного программирования сводится к постоянному *увеличению* (и усложнению) базиса: команда, оператор, функция, модуль, объект, класс, среда и т.д. Каждый новый шаг случаен, уникален и основан на «голом эмпиризме», возможностях финансирования (рекламы) автора или фирмы. Поэтому в этой модели развития постоянно увеличивается «вавилонская башня» языков и стилей программирования, которая разъединяет программистов и не объединяет их достижения.

С самого начала программирование опиралось на фундаментальные принципы математики, имеющие многовековую историю развития. Из математики в программирование были взяты основные принципы записи выражений, формул, функций, описаний данных и т.д. Как правило эти записи обеспечивают понимаемость (наглядность) языков программирования и надежную без ошибок запись программ. Основные трудности и проблемы программирования возникают от использования специфических операторов языков типа **goto, if, for, while** и т.п., которые не имеют прямых аналогов в классической математике.

В свое время Э. Дейкстра предложил структурное конструирование программ без **goto.** По мнению многих экспертов – это до сих пор самое эффективное предложение в технологии программирования.

В технологии нового поколения наоборот: базис *уменьшается* до ядра (до одной дуги), до сути процесса программирования и человеческого мышления (можно сказать, до Бозона Хи́ггса — «частицы бога» в программировании), понятному и начинающему программисту и специалисту, не связанному с программированием, и школьнику, обеспечивая всем реальную стратегию построения второй (компьютерной) грамотности.

В технологии ВТР нового поколения из программирования **исключаются** операторы перехода, выбора и цикла типа **goto, for, while,** а также **метки** и скобки типа **begin-end, {-}** и т.д. Для сегодняшнего дня они слишком 1) ориентированы на компьютер (на задание команд ему) чем на человека (на обеспечение удобства и качества процесса его мышления при реализации своей конкретной задачи); 2) они примитивны по своим возможностям, 3) имеют сложный синтаксис, 4) используют массу лишних знаков-паразитов типа '**;**', **метки**, ключевые слова типа **then**, **else**, **do**, **continue**,**...** скобки типа **begin-end**, **{-}** и т.д. 5) не позволяют документировать мысли, мотивацию действий программиста при разработке своей программы для чего стало необходимо использовать второй, дополнительный язык диаграмм UML и наконец, 6) требуют переформулировки под себя условия задачи, которую надо запрограммировать. Поэтому они усложняют и запутывают в общем-то простой процесс программирования, создают ситуацию, когда «за деревьями леса не видно», не видна идея алгоритма и программы. По этой причине эти символьные операторы **goto**, **if**, **for** и т.д. полностью исключены из технологии Графического программирования нового поколения. Вместо них из классической математики предлагается использовать теорию нагруженных по дугам графов.

В результате программирование приобретает единый (один единственный!) графический эквивалент (ядро, графическую оболочку) для всех языков программирования. **В новом программировании инструмент (графическое ядро) развивается, подстраивается под решаемую задачу и ее исполнителей, а не задача трансформируется под инструмент (операторы языка), как в традиционном программировании. Это значительно упрощает и решает существующие проблемы традиционного программирования, включая ООП.**

VII. СЕМЬ ПРЕИМУЩЕСТВ

1. Простота. В основе (базисе) графической оболочки ВТР всего одна горизонтальная дуга, Но эта дуга мощнее традиционных операторов языков программирования. Для ее задания не требуется никаких ключевых слов, ввод в компьютер быстрее и проще. Описание сути ВТР занимает одну страницу. Графическая оболочка ВТР применима для всех языков и на всем жизненном цикле программ. Так как графическая оболочка очень проста, то ее трансляция весьма эффективна в смысле занимаемой памяти и быстродействия результирующих кодов. Это позволяет также провести эффективную аппаратную реализацию.

2. Компактность. Компактность от 3-х до 100 и более раз по сравнению с Традиционной записью программ. Компактность тем выше, чем больше программный проект. Главное достоинство этой формы – сразу видна схема всего проекта, по которой удобно понимать (рассказывать, обсуждать, защищать, развивать, дополнять, утверждать, пересылать и т.д.) проект, а это оказалось очень важным в профессиональном программировании

3. Наглядность. Программа, ее структура, архитектура, идея алгоритма охватываются с одного взгляда. Впервые язык программирования (Р-схема) обеспечивает *самоопределение* того, что он описывает. Из программирования *исключены* все ключевые слова-паразиты для определения операторов языка, а это 40-50% всех символов программы. Все это значительно облегчает понимание программы и процесса ее разработки, который *автоматически вытекает* из первых шагов определения программы.

4. Определение данных. Р-схемы обеспечивают концептуальное единство, математическую строгость и графическую наглядность систем хранения и обработки данных, повышая их интеллектуальную составляющую. Графическая запись описательной части современных языков программирования компактнее в 1.5-2 раза и несравненно нагляднее. Для больших хранилищ данных обеспечивается возможность наглядного описания в графах их структуры (каталогов, метаданных) с гибким, быстрым механизмом обращения к ним для передачи на обработку в информационные системы, которые имеют те же графические Р-схемы для обработки этих данных.

5. Мощность. Теоретически Р-схема эквивалентна машине Тюринга. Это значит, что приведенной одной дуги Р-схемы достаточно, чтобы записать любой алгоритм. Практически любой язык программирования может иметь указанную простейшую графическую оболочку, единую(!) для всех языков. Это значит все, что может быть записано на этих языках, включая ООП, может быть более эффективно записано в графической их оболочке с подключением к программированию более мощного зрительного аппарата ассоциативного мышления человека.

В новой графической парадигме инструмент (Р-схема) подстраивается, развивается под решаемую

задачу и ее исполнителей, а не задача трансформируется под существующие фиксированные операторы языков (инструмент), как сейчас в традиционном программировании. Это снимает многие проблемы современного программирования и открывает дорогу доказательному программированию и накоплению профессионального опыта программистов. Большое достоинство новой графической оболочки, которого нет в традиционном программировании, в том что она обеспечивает единую знаковую систему для Заказчика, Программиста, Менеджера данных и Менеджера промышленной разработкой программных изделий.

6. Перспективность. ВТР особенно эффективна для решения логически сложных (запутанных) задач и тем совершенствует и развивает современное программирование, включая ООП. В ней просто задается доказательное, трехмерное и многомерное графическое программирование. Оперативно в ходе выполнения программы естественным путем включаются параллельные, лингвистические и другие спецпроцессоры, которые программируются единым языком Р-схем. Упрощаются процессы проектирования и выделения классов объектов. Процесс отладки программ становится доказуемым. Упрощение базиса, упрощает и его надстройку – среду визуального программирования, которая впервые может быть персонализирована под пользователя с запоминанием, накоплением и развитием его интеллекта. Открываются новые перспективы распределенных Интернет-технологий разработки программ.

7. Преемственность. Не надо ничего ломать и делать «по-другому». Всё существующее в программировании остается и добавляется лишь графическая оболочка на то, что есть. Профессиональная разработка графической оболочки «на то, что есть» может быть проведена в течение месяца (в будущем этот процесс автоматизируется). Пользователь имеет право выбора, как работать: используя графическую оболочку или так, как привык. В графической оболочке он будет широко использовать опыт предшествующих поколений программистов. Это значит, что на дугах Р-схем он может использовать *линейные* конструкции языков программирования (процедуры, функции, описания, выражения и т.д.), пришедшие в программирование из классической математики, в привычном для них виде, а потому лучше защищенных от ошибок.

VIII. ЗАКЛЮЧЕНИЕ

Чтобы построить хорошее здание надо начать с его фундамента. То, что предлагается – это новый фундамент программирования, который обеспечивает другую более простую и естественную (человеческую) красоту развития всей надстройки. Мы исходили из того, что нельзя строить хорошее здание, все время компенсируя (нейтрализуя) недостатки его фундамента, как это делается сейчас в программировании в системах IDE, SOA и др. Существующий фундамент практически не менялся с конца 40-х годов, когда были написаны первые программы на Ассемблере и Фортране. Основной его недостаток – машинная, операторная, сложная и примитивная (маломощная) организация. За более чем 65 лет она устарела и не соответствует задачам времени.

Новый фундамент более человеческий (не машинно-ориентированный), простой и мощный. Он просто определяется на рис.1, 2 и текстом на ½ страницы в разделе II, все остальное – иллюстрации преимуществ и сравнение с тем, что есть, известно. Все сделано по А.Эйнштейну: «делай настолько просто, насколько возможно, но не проще этого», потому что проще и эффективнее этого ничего пока нет. В результате резко на порядок увеличивается культура и эффективность программирования. Программирование упростилось и стало доступно детям, школьникам и специалистам не программистам. Мы надеемся, что даже «святая троица [5] как эталон современной профессиональной сложности и красоты (SOAP, WSDL and UDDI)+SOA» от этого только выиграет – она будет проще и понятнее народу и программистам.

Автор благодарен Программистам Антону ХОДАКОВСКОМУ и Александру ГУБОВУ за реализацию первой системы РР*-графического программирования в среде С++.

ЛИТЕРАТУРА

[1] V.M. Glushkov and I.V. Velbitskiy, “Programming technology and problems of its automation”, *USIM*, Kyiv, no. 6, pp. 75-93, 1976.

[2] I.V. Velbitskiy, “Programming technology”, *Technika*, Kyiv, p. 279, 1984.

[3] *Information technology, Program constructs and convention for their Representation*, International standart ISO/IEC 8631, Second edition 1989 Geneva 20, Switzerland, ISO/IEC Copyright Office, p.7, 1989.

[4] A.N. Valvachev etc. “Programming in Delphi. Textbook. Chapter 3 Object-oriented programming”, *INTERNET*, p. 19, 2005.

[5] Леонид Черняк, «SOA – шаг за горизонт. Открытые системы :: Менеджмент ИТ», *Открытые системы #09/2013*, p.12

Article in a journal:

[6] W.K. McHenry. “Technology: A soviet visual programming”, Journal *of Visual Languages and Computing*, vol.1, no. 2, pp. 199-212, 1990.

Article in a conference proceedings:

[7] I.V. Velbitskiy, “Next generation visual programming technology with R-charts”, *Plenary report, MEDIAS-2012. Dedicated to 100 anniversary of Alan Turing (IEEE)*, Cyprus, pp. xiv-xxxiv, 2012.

[8] L.F. Drobushevich, “Common use of UML and R-chart notations in the training process for software system development methods”, *MEDIAS*-2010, Cyprus, pp.73-77, 2010.

[9] I.V. Velbitskiy, “ Graphical Programming and Program Correctness Proof”, *Plenary report, 9th IEEE Computer Science & Information Technologiens Conference*, Yerevan, Armenia, 23-27 Sept.2013, IEEEXplore,CSIT-20

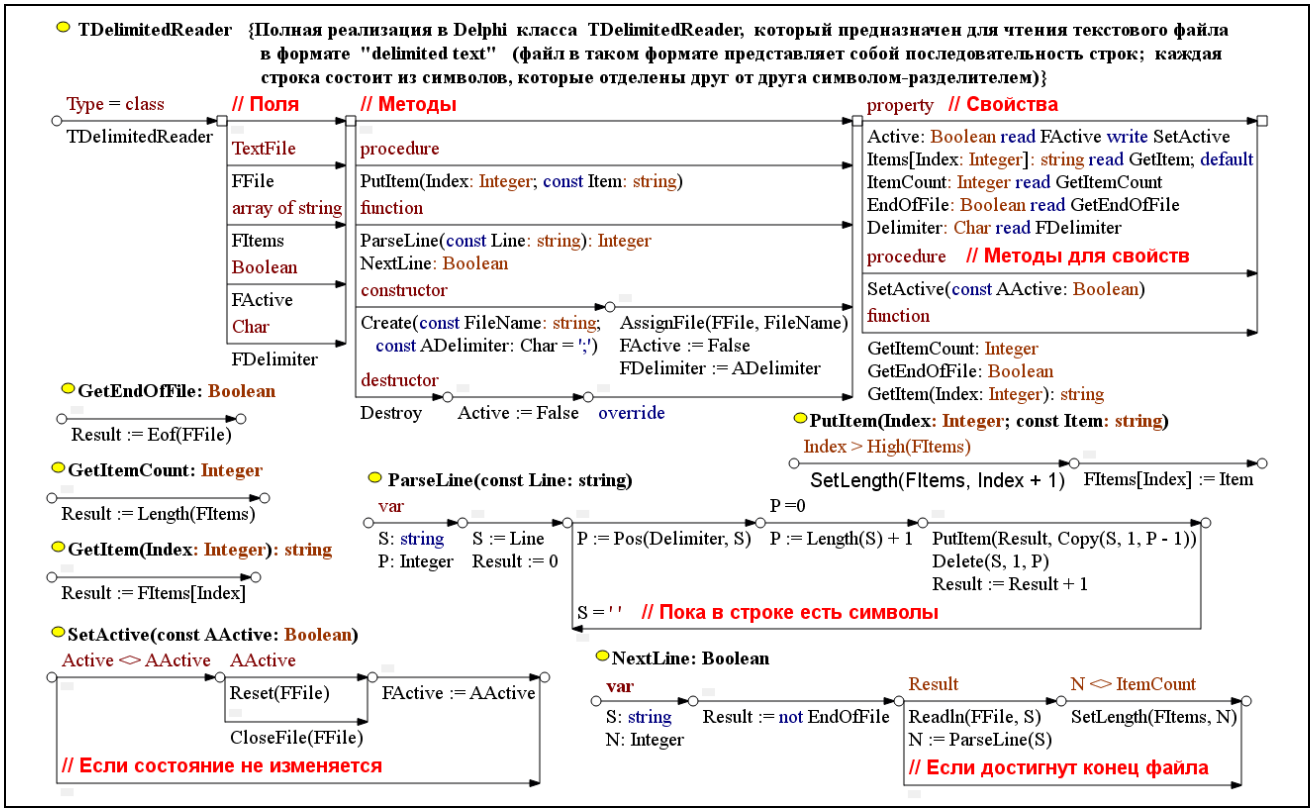

Рис. 14. Графическая P-схема **компактнее в 5.3** раза и несравненно наглядне традиционной записи программы в языке Delphi [4], из которой **удалено 860 (41.3%)** ненужных символов.