Новая Математическая Полиглот-Модель Программирования только в Графах

Игорь Вельбицкий

Фонд Глушкова, http://glushkov.org/
Киев, Украина, ivelbit@gmail.com (моб. 050 3318499)

Аннотация. Впервые после 1947 года предлагается использовать более простую и математически строгую модель программирования графами, нагруженными только по *горизонтальным* дугам. Такой граф является *полиглотом*, имеет международный *сертификат* ISO 8631:1989 и используется *на всем* жизненном цикле процесса программирования и выполнения любых работ на компьютере. По сравнению с традиционной моделью программирования новая модель имеет в СОТНИ(!) раз *лучшие* характеристики по *компактности* записи программ, занимаемой ими *памяти*, скорости *ввода* в компьютер и *скорости* выполнения. Для двух небольших приведенных в статье программ из 24(452) строк C++ эквивалентная графическая программа занимает 4(10) строк. В графической программе исключаются больше половины =142-63%(4364-73%) символов из программы C++: ключевых слов, блочных скобок, типографских знаков (перевод строк, отступы) и т.д. Ввод одного оператора C++ в компьютер занимает в среднем 20(29) клика клавиатуры, а графического оператора (дуги) – 0,8(0,21), что в 25(138) раза меньше (быстрее) и т.д. Существенно *упрощаются, улучшаются* и *ускоряются* процессы *проектирования* алгоритмов и программ, процессы *программирования, доказательства* правильности и *самодокументирования* программных проектов. Чем больше и логически сложнее программный проект, тем больше эффект применения новой полиглот-модели. Новая модель настолько проста, что позволяет программировать ВСЕМ, а не только программистам. В статье приведена история возникновения и проверки новой полиглот-модели, описание ее сути, преимуществ, реализованной графической среды программирования и перспективы ее использования.

***Ключевые слова*: Граф, RR*-схемы, Наглядность, Компактность, Простота, Цвет, Чертеж, Полиглот, Быстрый ввод, Доказательство правильности, Сетевые графики, Блок-схемы, UML, Google Blockly**

I. Введение

Программирование в графах начало формироваться в 70-х при разработке всех систем управления ракетно-космических комплексов бывшего Союза [1] и осознания важности формального документирования процесса их разработки для облегчения быстрого внесения постоянных исправлений и улучшений с одной стороны и работ Дейкстры [2], который впервые показал математическую нестрогость и избыточность традиционной модели программирования. В результате были сделаны первые шаги по использованию так называемых **Р**циональных графических **R-схем** [1,3]. Сейчас, согласно новой модели, **ВСЕ** традиционные машинно-ориентированные операторы (типа: **if**, **else**, **for**, **while**, **goto**, **метки**, скобки типа **begin-end**, **{-}** и т.д.) *исключены* из программирования. **Они устарели.** Для человека их слишком много, они сложны, эмпиричны, маломощны и обеспечивают примитивно-ремесленническую технологию программирования. Для *нейтрализации* их недостатков человек тратит слишком много усилий, создавая огромное число языков, методов и сред программирования, которые «якобы упрощают», а на самом деле разъединяют специалистов и делают программирование слишком сложным, не эволюционным и недоступным для **всех**.

Вместо всего этого предлагается только *одна,* математически строгая, графическая и человеко-ориентированная **сущность** – **R-схема** [3-8]. Для ввода в машину этой сущности требуется в *среднем менее* одного (до e=0,01, в 100 раз меньше) нажатия клавиши мыши и/или клавиатуры. Из традиционной программы исключаются до 70% лишних символов: ключевых слов, знаков препинания, типографских знаков и т.д. В 1989 году на R-схемы получен стандарт ISO [5]. Графическая программа в R-схемах в 100 и более раз *компактнее* и несравненно *нагляднее* традиционной записи программ. В новой концепции графическая запись чертежа проекта совпадает с записью программы и сетевым графиком ее разработки. В любой промышленности до сих пор такого не было – чертеж автомобиля *отличается* от самого автомобиля и сетевого графика его разработки. В результате процесс программирования многократно *упрощается*, *ускоряется* и *улучшается*, имеет *доказательство* правильности, *самодокументирование*, *математическое сопровождение* и автоматическую *генерацию* и *оптимизацию* программ. Обеспечивается преемственность со всем лучшим, что есть в традиционном программировании (ООП, АОП, WEB, CLOUD и др.) и прежде всего с библиотеками программ. Открываются новые перспективы промышленной разработки больших программ. Программирование впервые получает строгую *математическую* основу (культуру) и становится доступным **всем**, а не только программистам.

II. Суть процесса программирования в графах

В концепции программирования графами (ISO 8631:1989 [5]) вместо традиционных операторов типа, **if**, **for**, **goto** и т.д. (**всех** около 10) на протяжении всего жизненного цикла программ предлагается использовать только ***одну*** горизонтальную дугу – R-схему, Рис.1, у которой сверху всегда записано *Условие*, а снизу – *Действия*, которые выполняются, если Условие «истинно». Для их записи в одну или несколько строк могут использоваться любые языки**:** английский, русский, китайский и т.д., язык математики, а также любые языки программирования. Обеспечивается принцип полиглота.

Рис.1. R-схема: одна горизонтальная дуга вправо или влево.

Из вершины может исходить любое число альтернативных дуг вправо и/или влево, Рис.2-11. Эти дуги просматриваются (анализируются) последовательно сверху вниз до появления истинного Условия. После чего выполняются соответствующие Действия и переход по стрелке дуги в новое состояние программы (графа). Если Условие «ложно» и дуга последняя, то осуществляется переход (без выполнения Действий) по стрелке в следующее состояние графа.

На Рис.2 приведен пример определения нового оператора: «3-х (может быть любое число) Условий и Цикла». На Рис.2a приведена его R-схема, на Рис.2b – его R* математическая R-схема без деталей реализации (без записей на дугах), а на Рис.2c – эквивалентная традиционная запись в C++, где красным отмечены *лишние* символы для R-схемы на Рис.2a. Таких лишних символов на Рис.2c – 79 или с отступами (абзацами) и переводами строк – **142** (**63%**). Запись нового оператора Рис.2a в **2**, а оператора на Рис.2b в **6** раз компактнее

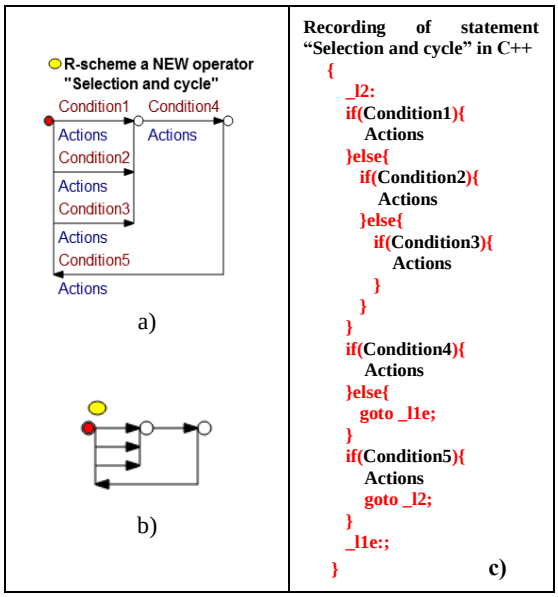

Рис.2. Запись нового Оператора «Выбора и цикла» в RR*- схемах и в C++. Красные символы в C++ являются лишними для R-схем.

записи его в C++ на Рис.2c. Ввод одного оператора C++ (Рис.2c) в компьютер занимает в среднем 20(=142/7) клика клавиатуры, а ввод графического оператора или дуги R-схемы (Рис.2a) занимает 0.8(=4/5) клика, что в **25**(=20/0,8) раза меньше и быстрее. Ввод R-схемы на Рис.9 осуществляется в **138(**=29/0,21**)** раз быстрее, где 29(=**4364**/150) – число символов для ввода одного соответствующего оператора C++, а 0,21(=21/100) – число кликов для ввода одного графического оператора (дуги) на Рис.9. R-схемы типа «петли» и «ромашки» приведены на Рис.3,11. Дуга в виде знака «равно» между вершинами объединяет их в одну соответствующую вершину графа.

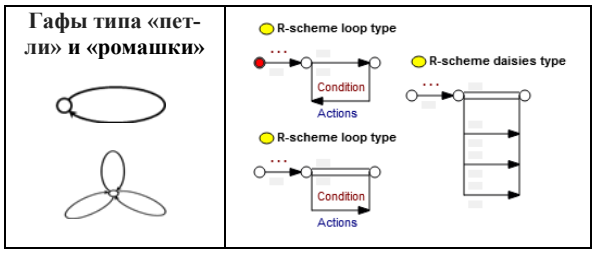

Рис.3 R-схемы графов типа «петли» и «ромашки».

Над дугой могут использоваться *ключевые* слова типа **real**, **procedure**, **function**, **form** и т.д., которые всегда истинны и задают новое понимание и использование всех элементов R-схем. Например, на Рис.4a они задают безальтернативное, то есть обязательное, *параллельное* выполнение **всех** исходящих из вершины дуг в соответствии с ключевым словом на дуге. Семантика такого задания наглядно изображена в общепринятых обозначениях на Рис.4c. Многие описания переменных и соответствующие ключевые слова могут быть исключены из новой концепции, потому что они однозначно вычисляются (понимаются) из структуры записи этих переменных на дугах R-схем, как в математике. Это значительно увеличивает наглядность и компактность записи RR*-схем.

Вершины графа не имеют имени, но могут иметь различную *конфигурацию* и *цвет* для реализации светофоров в программе, определения &-дуг, 3D-программирования и т.д. Например, красная вершина всегда определяет начало проекта, треугольная – повторяемые макроопределения R-схем, на Рис.4b,6,7 используются прямоугольники для записи параллельно выполняемых &-дуг, на Рис.5

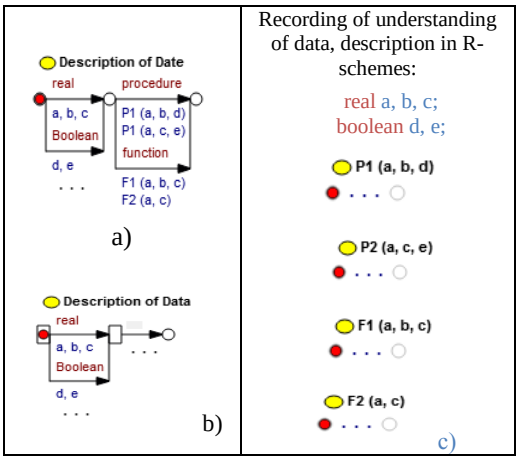

Рис.4. Определение Данных, Процедур и Функций.

принцип 3D-программирования и документирования мотивации принимаемых решений изображен с помощью вершины в виде параллелограмма.

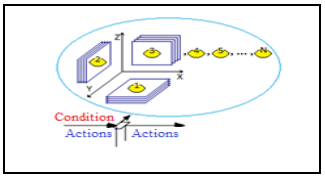

Рис. 5. Организация 3D программирования и мотивации принимаемых решений.

Такой граф имеет имя, которое записывается сверху около желтого эллипса, Рис.1-4,6,8,10,11. Имя может быть с параметрами или без них, Рис.6. Программа задаетя любым числом взаимосвязанных по имени таких графов, Рис.6. В отличие от традиционной записи программ такой граф R (Рис.6) имеет запись R* (Рис.7) математической абстракции программы без записей на дугах (без деталей реализации). Такая запись R* позволяет задавать модели программ и производить над ними математические исследования, тождественные преобразования, оптимизацию по различным параметрам – времени, памяти, а также проводить классификацию для нового типа архивов графических программ и т.д. Она компактнее и существенно упрощает проектирование программ. Например, для Рис.6 компактность R=**7**, а для Рис.7 компактность в R*=**35,7** раз больше по сравнению с записью этой программы традиционным текстом в C++ [9], см. также Рис.2abc и 8abc. Зафиксированная на сегодня максимальная компактность *графических оболочек* для реальных программ равна: R=**19**, R*=**130**. Чем больше

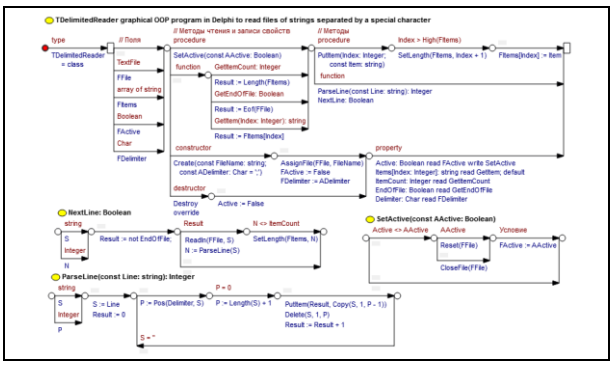

Рис. 6. R-схема ООП программы в **7** раз **более компактна** чем традиционная запись этой программы в Delphi [9].

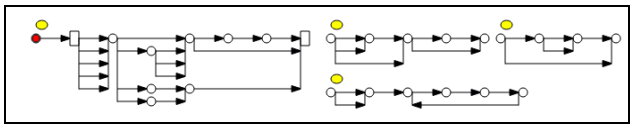

Рис. 7. R*- схема записи программы без деталей реализации в **35.7** раз компактнее записи ее в C++.

и логически сложнее программа, тем больше ее компактность, Рис.2a-2b и 6-7.

На всем жизненном цикле новой концепции чертеж проекта программы совпадает с изделием, его документацией и сетевым графиком разработки, Рис.6-8. Руководитель работ **впервые** может около каждой работы на дуге R-схем записать в особых скобках имя исполнителя и расчетное время ее выполнения и **проконтролировать** качество выполняемых работ.

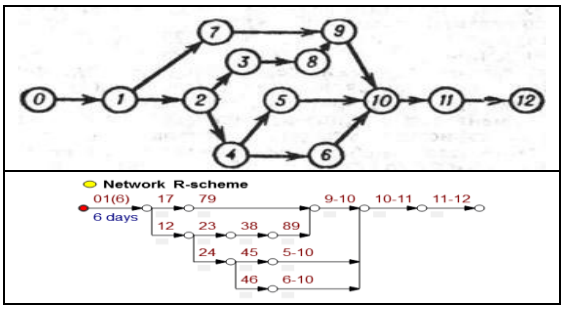

Рис. 8. Запись сетевого графика в R-схемах.

Ввод каждой горизонтальной дуги графа делается одним нажатием мыши или клавиатуры или пальца по сенсорному экрану. Вертикальные дуги и вершины графа не вводятся в компьютер и рисуются автоматически графическим редактором. При одном нажатии может быть введено сразу несколько новых горизонтальных дуг. Например, для задания R-схемы на Рис.2b требуется 4(для 5 дуг) нажатия левой кнопки мыши или клавиатуры, Рис.3a – 3(4), Рис.6 – 25(39), Рис.10c – 2(4). В результате получается, что для задания одной дуги графа требуется в среднем меньше одного (до e=0,01, в 100 раз меньше) нажатия клавиатуры или мыши, Рис.9 – 21(=10x10=100, e=21/100=**0,21**). R-схема (программа) 100x100 дуг имеет e=**0,0302**, R=**133**, R*=**400** и содержит в **578** раз меньше символов, поэтому ввод ее в компьютер быстрее в **578** раз, чем эквивалентной программы в C++ и т.д. Для задания дуг не требуется языка, поэтому новая концепция является полиглотом.

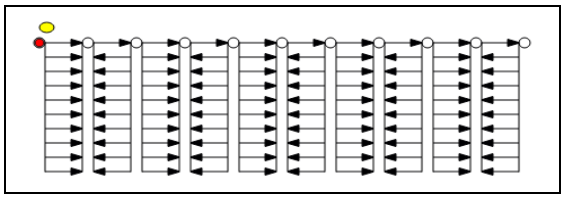

Рис. 9. Ввод R-схемы (программы) из 10x10=100 дуг нажатием **21** клавиши(быстрее в **138** раза) и компактность R=**15**, R*=**45** по сравнению C++.

Человек всегда стремится найти графический образ любым своим действиям. Известно огромное число графических способов записи программ: Flow-chart, UML, Workflow, SDL, ДРАКОН, Google BLOCKLY[9] и др. Как правило – это графы, нагруженные по вершинам, Рис.10a. R-схемы – это графы, нагруженные по дугам, Рис.10bc и 11(внизу), которые имеют заметные количественные и качественные преимущества среди известных, см. Рис.10a и 10bc, Рис.11(вверху) и (внизу). В отличие от традиционных блок-схем, UML и т.д., R-схемы *выполнимы* на компьютере на всем жизненном цикле любых работ, более просты, наглядны и компактны.

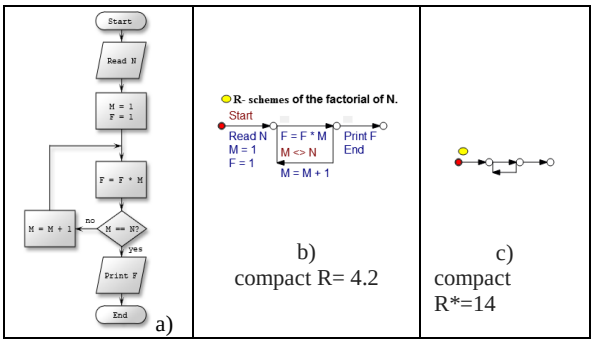

Рис. 10. Пример записи алгоритма вычисления факториала числа N в блок-схемах(a) и в RR*-схемах(b,c)

Таким образом, для всех языков – естественных, математики и программирования предлагается единая графическая оболочка. Запись программ и их проектов в этой оболочке, а также их трансляция и интерпретация становятся проще. Программа (R-схема) впервые является объектом математики и для человека(!), а не только для компьютера. RR*-схемы идеальны для *обратной* трансляции программ на любых языках в новую единую графическую оболочку. В результате программа из «вещи в себе», понятной только автору (не всегда доступному), становится прозрачной для развития и улучшения.

Графическая форма записи имеет резервы по расширению и развитию в будущем. Для записи вершин и дуг может использоваться цвет. Вершины и стрелки дуг могут иметь разную конфигурацию, дуги могут быть двойными, волнистыми, пунктирными и т.д.

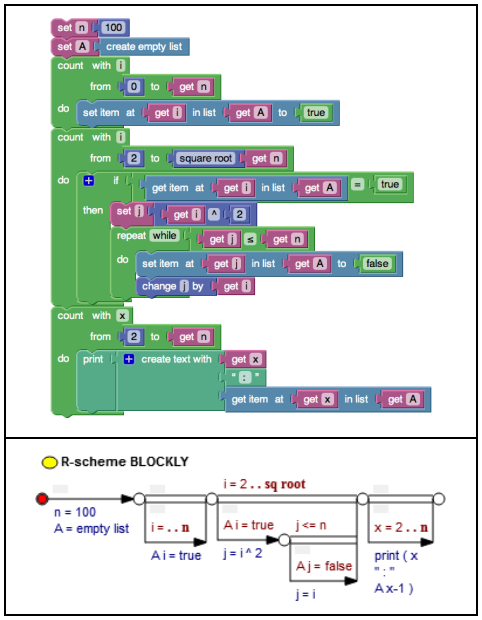

Рис. 11. Пример записи программы в визуальном языке программирования Google Blockly [10] и R-схеме. Компактность R=**3**, R*=**14**

III. Технология Проектирования и Доказательства Правильности Программ

Традиционная концепция проектирования алгоритма (программы) основана на построении вычислительной схемы решения исходной задачи с помощью около десяти фиксированных, текстовых, машинно-ориентированных операторов. Для этого исходная задача преобразуется под эти операторы. Это всем известный сложный, много-

ступенчатый процесс. В результате этого процесса исходная задача преобразуется «до неузнаваемости», теряется связь и мотивация принимаемых решений в процессе преобразования исходной постановки дадачи, и понимание «как все это работает». Для упрощения этого процесса придумываются многочисленные методы, что еще больше запутывает и усложняет процесс программирования.

Новая концепция проектирования основана на построении логической схемы исходной задачи. Метод «step by step from logic» используется для записи условия задачи в единой графической оболочке R-схем и в тех же словах, что и формулировка исходной задачи. В этом процессе выявляются и уточняются все основные неточности в постановке задачи. В результате Логическая R-схема задачи становится одинаково понятной исполнителю и заказчику и утверждается ими. После этого построение проекта алгоритма и программы осуществляется формальными методами математического вывода, который одновременно будет доказательством правильности программы и процесса ее разработки. Получаемая в результате документация отличается от традиционной, так как сохраняет мотивацию принимаемых решений, более компактна и самодокументируема. Графическая среда следит и направляет человека таким образом, чтобы его формальные преобразования исключали из исходной логической схемы все неоднозначности понимания и превращали ее в язык естественный для человека (как правило в графический язык простейшей математики) и далее – в язык компьютера. Язык естественный для человека постоянно совершенствуется (эволюционирует) и приближается к профессиональному языку соответствующего коллектива разработчиков программ. Важно, что язык R-схем такого описания **один (!)** и для компьютера, и для заказчика, и для всех исполнителей проекта и его пользователей на всем его жизненном цикле.

IV. Реализация графической среды программирования

В настоящее время *реализована* (лабораторный вариант, программисты: **А.Ходаковский**, **А.Губов**) графическая среда разработки программ, которая включает в себя: систему ввода R-схем и любых текстов на их дугах на любом языке; формирование и запоминание дерева проекта; графический редактор; преобразователь R-схем в R* и обратно; компилятор RR* в C++ и т.д. Графическая среда разработана на C++ как плагин REditor к *Qt Creator*. Она состоит из 5 областей, которые делят и формируют экран монитора. Основную (б'ольшую, 90%) центральную часть монитора занимает 1) Рабочее поле, на котором осуществляется разработка всех R-схем Проекта. Сверху в трех строках располагается 2) меню формирования архитектуры среды, 3) Панель инструментов (из 14 графических иконок) и 4) Панель открытых (неограниченное число) имен Рабочих полей R-схем. 5) Последняя область занимает слева узкую (5%) полосу экрана монитора для хранения *Дерева* R-схем. Таким образом, реализованная Графическая среда обобщила существующий опыт программирования графами, и является достаточно полной (по нашим оценкам на 80%) для построения на ее основе коммерческой версии.

Один из *независимых* пользователей сказал следующее о новой графической среде**:** «После двух недель рисования схем в редакторе я решил задачу, которую без R-схем не решил бы никогда**…** Я воспринимаю R-схемы как полезный и красивый инструмент не только в программировании, но и везде, где нужно понять логику взаимодействия частей любой проблемы или работы. По сравнению с известными Блок-схемами, UML и др. они имеют заметные преимущества. R-схемы выполнимы на компьютере на всем жизненном цикле любых работ, более наглядны, компактны и не загромождены лишними деталями (фигурами). Подходят для визуализации нелинейных алгоритмов (например, классов в C++) или структур данных. Поэтому мне совершенно не понятно, почему этот инструмент не пользуется ***бешеной*** популярностью» В.Бушкевич, 04.12.2015.

V. Заключение

Таким образом, предлагается некоторая новая математическая концепция (культура) программирования интересная своими преимуществами, простотой, человечностью и возможностью перехода на нее **для всех**, а не только для программистов. Эта культура позволяет включить в профессиональное программирование доказательный стиль, развитые и отработанные веками математические методы анализа и синтеза, а также новую технологию промышленной разработки больших программ. Новая концепция позволила в 4-х страницах (в 450 строках) текста данной статьи в качестве иллюстрации и пояснений описать КОМПАКТНО 7 примеров графических программ (R-схем), которые в традиционной концепции занимают суммарно 41000 строк C++ и Delphi.

Новые предложения из-за своей простоты и наглядности особенно эффективны для начального обучения программированию. Поэтому они могут быть включены в систему обязательного образования любых специалистов. Программировать должны уметь **ВСЕ**, оно должно стать элементом всеобщей грамотности и культуры общества. Новая культура начинает программирование нового XXI века и открывает большие перспективы в будущем для построения человеко-компьютерного общества и его *совместной* с человеком *эволюции*. Вживленный в человека компьютер новой архитектуры и новой концепцией программирования будет наряду с мозгом управлять физиологией конкретного человека, компенсируя ее недостатки от рождения и проводя лечение по жизни без традиционных лекарств. Управлять сильнее и эффективнее, чем известные «цилиндры Фараона».

Известно, что в мозгу человека главную роль играют ***связи*** между нейронами и областями мозга. Аналогичную роль в **R-схемах** играют *дуги* между вершинами. Число дуг, их направление, 3D-ориентация между различными вершинами и R-схемами не имеют ограничений. Вершина может запоминать число обращений к ней и иметь порог чувствительности в своей работе. Ничего подобного нет в современном программировании. Все сказанное и огромный современный потенциал микроэлектроники и устройств памяти заставляет *сегодня* проанализировать главный, начиная с 1947г., недостаток наших компьютеров – сложная и примитивная организация процесса программирования. Настало время провести анализ, обобщение опыта, выбрать все лучшее и создать в новом тысячелетии СОФТ и КОМПЬЮТЕР принципиально нового **мозгоподобного** поколения. Трудоемкость создания такого проекта мы оцениваем в ½ года и еще ½ года на реализацию и маркетинг, а может и меньше, учитывая существующую версию реализации R-схем, накопленный мировой опыт разработки программ и эволюционную, поэтапную реализацию.

Литература

[1] "СЕРГЕЕВ В.Г. – Главный конструктор систем управления ракет и космических комплексов", Украина, Харьков, 448 с., 2014.

[2] E.Dijkstra, «Letters to the editor: go to statement considered harmful», *Communications of the ACM*, pp. 147–148, 1968.

[3] В.М.Глушков, И.В.Вельбицкий, «Технология программирования и проблемы ее автоматизации», УСИМ, №6, с. 75-93, 1976.

[4] И.В.Вельбицкий, «Технология программирования». Техника, Украина,279с, 1984.

[5] «Information technology, Programme constructs and convention for their Representation», International standard ISO/IEC 8631, 1989.

[6] W.K.McHenry, «Technology: A soviet visual programming». *Journal of Visual Languages and Computing*, v.1, № 2, 1990.

[7] I.V.Velbitskiy, «Graphical Programming and Program Correctness Proof», IEEE DOI: 10.1109/ CSIT-13.6710368 pp.85-89, 2013.

[8] I.V.Velbitskiy, «Graphical programming of new generation. Single universal graphical shell for all languages» Global IT, Las Vegas, IEEE DOI 10.1109/CSITechnol.7358261 pp.111-115, 2015.

[9] A.N.Valvachev. http://www.rsdn.ru/article/Delphi/

[10] The visual Google Blockly, *habrahabr.ru*