ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ №3(30), 2016, с. 45–70

УДК 004.514:004.924:004.75

В. П. Фраленко

Универсальный графический интерфейс визуального проектирования параллельных и параллельно-конвейерных приложений

Аннотация. В статье описывается многофункциональный графический интерфейс, предназначенный для визуального проектирования вычислительных процессов и технологических цепочек в различных прикладных областях. Пользователь формулирует прикладную задачу с помощью набора готовых функциональных блоков и редактора связей, устанавливающего каналы передачи данных, запускает составленные схемы задач в параллельно-конвейерном режиме и отслеживает выполнение встроенными инструментами.

Сравнение с существующими аналогами выявило особенности интерфейса, расширяющие возможности разработчиков программного обеспечения, помогающие программисту увидеть схему задачи в целом, ускоряющие процесс разработки, снижающие количество ошибок и повышающие оперативность принятия решений.

Ключевые слова и фразы: графический интерфейс, поддержка вычислений, параллелизм, конвейер, схема, задача.

Введение

Проблема инструментальной поддержки разработчиков программного обеспечения в настоящее время привлекает внимание как крупных компаний — «Microsoft», «Intel», «IBM», «Nvidia» и «AMD», так и инициативных исследователей и разработчиков на интернет-площадках «GitHub» и «Sourceforge». В сфере организации высокопроизводительных вычислений основное внимание в последнее десятилетие отдается следующим направлениям:

Работа выполнена в рамках СЧ НИР шифр «Мониторинг–СГ–1.2.5.1» по Программе Союзного государства «Разработка космических и наземных средств обеспечения потребителей России и Беларуси информацией дистанционного зондирования Земли» и при частичной финансовой поддержке Российского фонда фундаментальных исследований (проект №16–29–12839–офи_м).

- программным решениям, направленным на повышение эффективности использования доступного аппаратного обеспечения, в том числе компиляторам и сопутствующим им средам разработки; эти решения способствуют устранению «узких мест» в коде или предлагают разработчику «подсказки» для повышения скорости работы программы;
- новым технологиям программирования, в том числе в рамках «OpenMP», «MPI», «CUDA», «OpenCL», «OpenAAC»; перечисленные технологии расширяют возможности проектирования параллельных прикладных систем;
- универсальным и специализированным программным библиотекам, содержащим функционально ориентированные наборы возможностей; к ним, например, можно отнести библиотеки «Boost», «OpenCV» и «ArrayFire».

В то же время, анализ существующих решений показывает, что относительно малое внимание обращается на такое важное направление инструментальной поддержки программистов, как workflow-системы, обеспечивающие координацию и выполнение сложных комбинированных задач, составляющих в целом технологические цепочки и процессы. Здесь речь идет не только о бизнес-процессах, но и вообще о любых процессах, требующих управления и координации. В рамках настоящей работы предлагаются решения, направленные не только на организацию таких вычислительных процессов, но и на поддержку программистов с целью лучшей организации разработки модульного программного обеспечения. Все эти решения реализуются в виде нового универсального графического интерфейса, фактически являющегося эффективным инструментальным средством разработчика. Workflow-системы, снабженные такими интерфейсами, способны решать следующие глобальные задачи:

- организация совместной работы разнородных программных компонентов и управление логикой решения сложных многокомпонентных задач;
- обеспечение максимально эффективного использования разнородного аппаратного обеспечения (процессоров общего назначения, графических ускорителей, вентильных матриц и пр.);
- ускорение разработки новых решений за счет использования ранее созданного программного кода, в том числе без его перекомпиляции.

В качестве примеров известных workflow-систем можно указать «CODE» [1], «HeNCE» [2], «GRADE» [3], «EDPEPPS» [4], «Microsoft Workflow Foundation» [5], «Triana» [6], «ANSYS EKM» [7], «Kepler» [8] и «НСКиД» [9], из наиболее современных нельзя не упомянуть системы «SAASFEE» [10] и «Orange4WS» [11]. Проведя соответствующий анализ причин малопопулярности таких систем, следует сказать, что их освоение и успешное использование на начальных этапах весьма трудозатратно, так как требует обучения персонала, мышления в рамках некоторой строго заданной модели, использования ограниченного набора классов и интерфейсов, зачастую базирующихся на устаревших программных библиотеках. Самый основной их недостаток — отсутствие универсальности, привязка пользователя к конкретной реализации интерфейса.

Предлагаемый интерфейс наследует принципы построения универсальных интерфейсов, заложенные в работах [12–16] и отчетных материалах ИПС им. А.К. Айламазяна РАН, которые были ориентированы на создание приложений для авиакосмической отрасли. В то же время он служит развитием идей универсальности, которым следовали эти разработки.

1. Главное окно и средства разработки интерфейса

Результат работы созданного графического интерфейса — описания задач (тексты в формате XML), визуализация самих схем, которые в дальнейшем могут быть исполнены сторонними высокопроизводительными вычислительными средствами и системами, например, [9, 17, 18] и др., в том числе с поддержкой параллельного и параллельно-конвейерного режима работы исполняемых модулей (функций). Интерфейс позволяет не только формировать тексты задач, но и инициировать их исполнение (запуск), используя для этого специальный файл с параметрами настройки внешней вычислительной среды; позволяет соединять отдельные модули каналами передачи данных, устанавливать в диалоговых окнах их настройки и пр., содержит средства отслеживания состояния запущенных вычислительных задач. Главное окно интерфейса позволяет формировать схемы решения задач и выбирать необходимое для решения конкретной задачи количество аппаратных ресурсов, включая центральные процессорные устройства (ЦПУ) и/или графические процессорные устройства (ГПУ), что отражено на рис. 1.

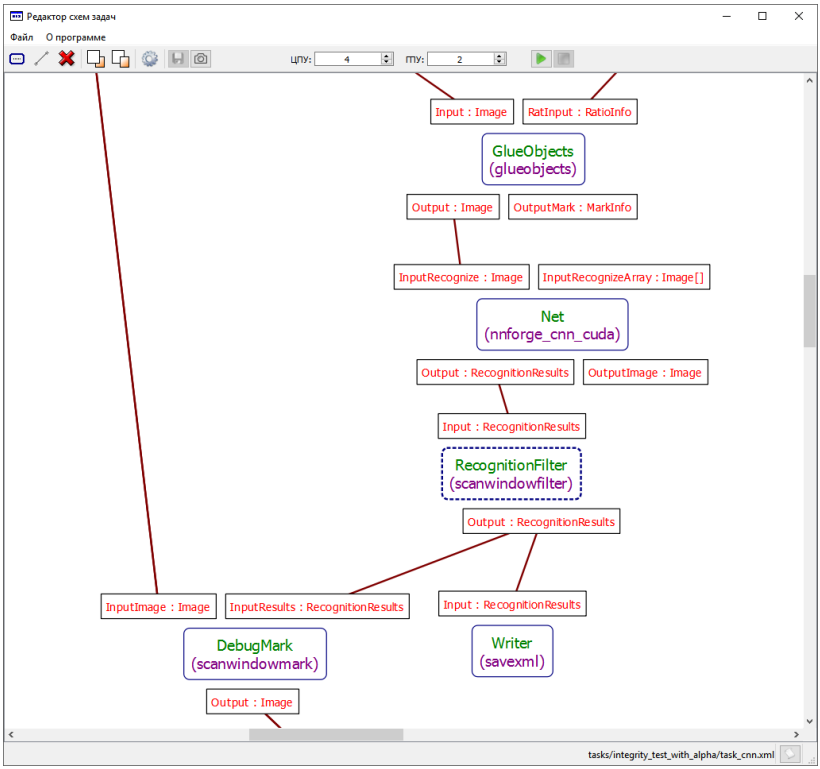


Рис. 1. Главное окно графического интерфейса

В качестве средства разработки интерфейса было использовано открытое программное обеспечение — кроссплатформенный инструментарий Qt на языке программирования C++ [19]. Qt позволяет разрабатывать открытое программное обеспечение, следующее правилам, закрепленным в соглашении GNU [20]. На рис. 2 показан пример интеграции инструментария Qt со средой разработки «Microsoft Visual Studio» в рамках файл-проекта интерфейса.

На рис. 3 приведено диалоговое окно редактора «Qt Designer» с одной из дополнительных форм интерфейса. Рисунок иллюстрирует возможности по добавлению элементов управления с последующей их настройкой под нужды разработчика.

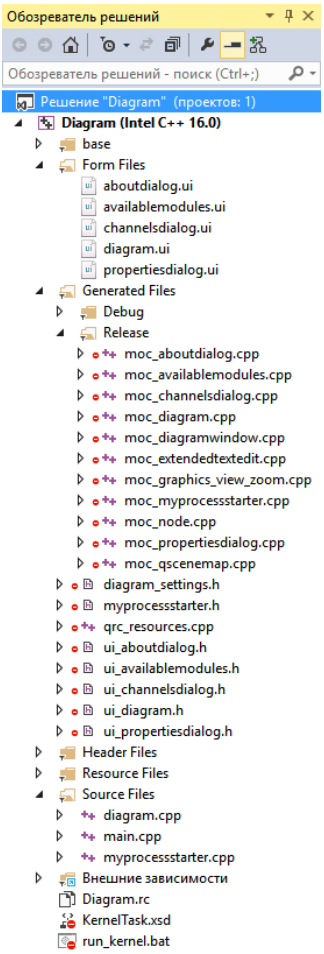


Рис. 2. Обозреватель решений в «Microsoft Visual Studio 2013» с Qt-проектом графического интерфейса

2. Требования к построению графического интерфейса

Рассмотрим основные принципы и характерные особенности, заложенные при создании универсального графического интерфейса.

(1) Кроссплатформенность и единство внешнего вида на различных операционных системах (Windows, Linux и пр.). Характеристика

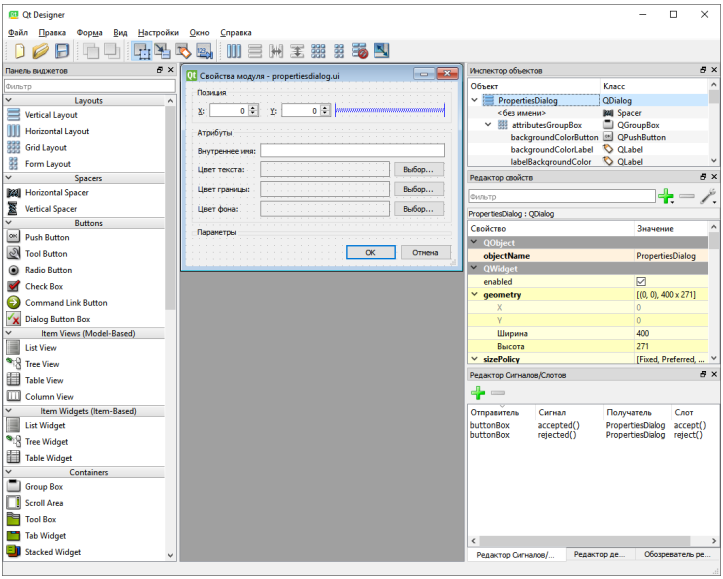


Рис. 3. Диалоговое окно редактора «Qt Designer»

обеспечивается выбранным инструментарием разработки Qt.

(2) Поддержка многопользовательского (облачного) режима функционирования сторонней системы организации вычислений, обеспечивающая минимизацию конфликтов работы нескольких пользователей.

(3) Визуально-блочный редактор решаемых задач, являющийся единым рабочим пространством без вспомогательных текстовых и других инструментов для настройки схем задач, что понижает сложность обучения пользователей системы и повышает их производительность.

(4) Управление списком отображаемых данных, возможность изменения стилевых и визуальных составляющих. В том числе должны быть предусмотрены функции, позволяющие скрывать или показывать сведения о типизации каналов связи модулей, информацию о типе подключенного модуля, о его «псевдониме» в схеме; менять цвет окраски модулей и каналов связи на схеме; масштабировать как сами блоки, так и шрифты надписей.

(5) Сохранение описания задачи (схемы) в виде текста в формате XML, что обеспечивает как необходимый уровень гибкости при

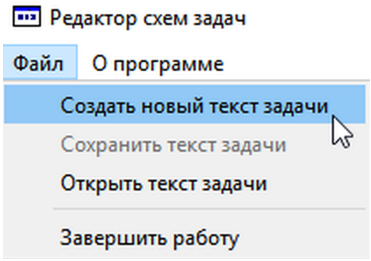


Рис. 4. Меню «Файл» и «О программе»

проектировании и эксплуатации подсистемы загрузки описаний задач в графический интерфейс, так и позволяет использовать полученные схемы в различных сторонних вычислительных системах.

(6) Обеспечение стандартных функций интерфейса, таких как перемещение по рабочему пространству; уменьшение/увеличение масштаба; горизонтальные и вертикальные полосы прокрутки; клавиатурный ввод и пр.

(7) Наличие функции сохранения описания задачи в виде графического файла (в точечном и/или векторном формате), что облегчает формирование отчетов.

(8) Возможность настройки не только схемы решения задачи, но и отдельных модулей путем изменения значения их параметров. Вспомогательный интерфейс со списком параметров отдельных модулей должен по умолчанию быть скрыт, что позволяет снизить информационную нагруженность основного окна интерфейса.

(9) Поддержка специальных окон редакторов параметров. Вид редактора должен определяться соответствующими типами параметров модулей.

(10) Поддержка интерактивного режима работы пользователя с основными модулями системы.

(11) Возможность подключения модулей для отображения специальных когнитивных образов с применением современных средств визуализации OpenGL [21] и/или Vulkan [22].

3. Порядок создания схемы решаемой задачи

Схемы задач создаются пользователем в интерактивном режиме на основе программно реализованных модулей. Главное окно интер-

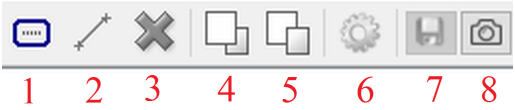


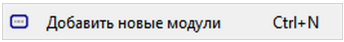
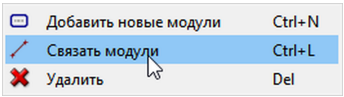
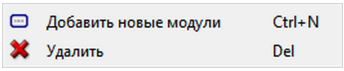
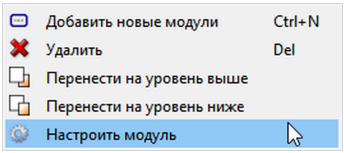
Рис. 5. Динамическая панель инструментов

фейса содержит меню «Файл» и «О программе» (см. рис. 1 и 4). С помощью меню «Файл» пользователь может осуществлять операции по созданию нового текста задачи; сохранять текст задачи, подготовленный в интерфейсе (в формате XML); открывать ранее сохраненный текст задачи и завершать работу графического интерфейса. Обращение к меню «О программе» вызывает появление типового окна с информацией о разработчиках программы.

Рассмотрим действия программиста по формированию схемы задачи на конкретном примере. Добавление в блок-схему новых модулей может быть осуществлено одним из следующих способов: нажатием на соответствующую кнопку-иконку (1) на панели инструментов (см. рис. 5); сочетанием клавиш «Ctrl + N»; вызовом контекстного меню (см. таблицу 1). Наличие нескольких вариантов позволяет интерфейсу быть удобным для пользователей с разными предпочтениями. Кнопки-иконки (4) и (5) являются вспомогательными, они позволяют управлять Z-буфером сцены. Графическая подсистема интерфейса обеспечивает полное отсутствие геометрических пересечений модулей между собой. Пересекаться с блоками/накладываться друг на друга могут лишь визуализируемые каналы связи (см. рис. 1). Кнопка-иконка (8) на панели инструментов позволяет сохранить схему задачи как высококачественный графический файл (см. рис. 5).

В результате выполненных действий появится окно «Добавление новых модулей» (см. рис. 6). Пользователь может добавить сразу все необходимые для решения задачи модули, указав нужное количество модулей каждого типа и подтвердив решение по каждому из них с помощью специальной иконки (чекбокса). Например, из рис. 6 следует, что при нажатии на кнопку «OK» будет добавлено три модуля — «dat_reader» (модуль чтения файлов телеметрии космического аппарата в формате «dat»); модуль «dataread» (модуль загрузки настроек искусственной нейронной сети) и модуль «annsave» (для сохранения настроек нейронной сети). В качестве подсказки отображается: информация о каналах модулей (в частности, на рис. 6 визуализируются сведения о наличии канала «Output» у модуля «dat_reader»); данные

Таблица 1. Динамически формируемое контекстное меню

№	Условие формирования	Вид контекстного меню
1	в начале работы, когда нет ранее добавленных модулей, либо если пользователь запросил вызов меню в точке сцены без объектов	
2	выделена пара модулей (для того, чтобы связать их каналом связи или удалить)	
3	выделен канал/каналы связи или более двух объектов на сцене	
4	выбран один модуль	

о поддержке модулем внутреннего параллелизма. Так, например, на рис. 6 имеются отметки, показывающие, что модули типа «anna», «dat_encryption» и «dat_extractor» имеют параллельную внутреннюю реализацию.

Эти и другие сведения содержатся в специальных XML-файлах, описывающих модули. Все описания модулей хранятся в системной папке «Plugins». В качестве примера можно привести листинги 1 и 2, содержащие полные сведения о модулях «dat_reader» и «annsave».

Файлы описаний модулей создаются программистом, готовящем непосредственно программные реализации. Метаданные в виде атрибутов и секций используются как графическим интерфейсом, так и внешней вычислительной средой. Следует отметить атрибут «parallel» (флаг, принимающий значения «no» и «yes», в первом случае модуль не обладает внутренним параллелизмом (реализуемым программистом), во втором случае — обладает). Атрибуты «nodecountmin» и «nodecountmax» определяют минимальное количество вычисли-

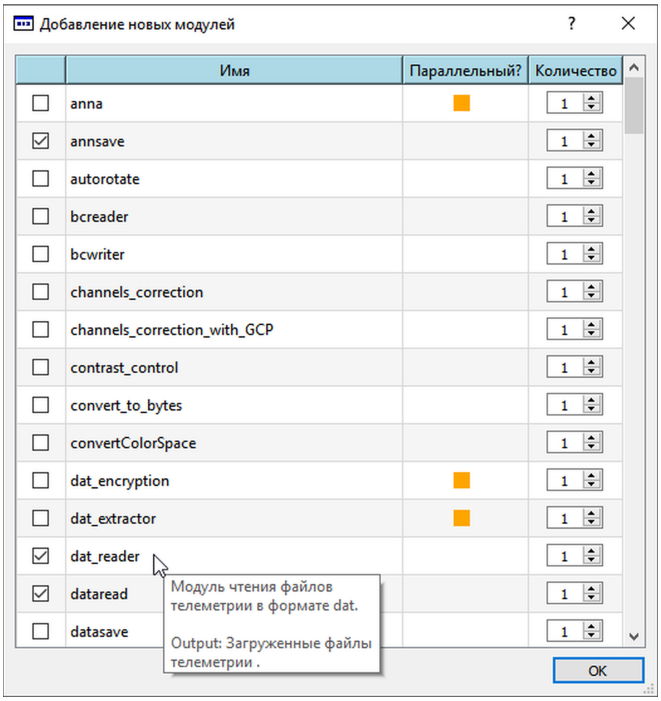


Рис. 6. Форма для добавления новых модулей

тельных устройств, необходимое для работы модуля, и наоборот — максимальное число задействуемых вычислительных устройств. Атрибут «desc» секции «moduledesc» задает описание модуля в целом, его функции и назначение. В секции «init» приводится информация о параметрах модуля, в том числе в каждом случае имя («name») и тип переменной («type»), значение параметра по умолчанию. Секция «designer» может содержать информацию с описанием параметров, в частности, для этого используются подсекции «vardesc». В секции «channels» описываются имеющиеся в схеме каналы данных. Каждый канал («channel») в свою очередь описывается типом («type», принимает значение «out» или «in»), именем («name») и типом передаваемых данных («datatype»). Тип передаваемых данных не является обязательным атрибутом. В случае отсутствия явной типизации в реализации сторонней вычислительной системы значения атрибутов могут выступать подсказкой для пользователя интерфейса.

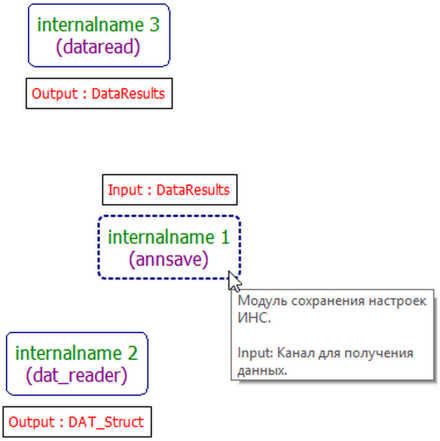


Рис. 7. Результат использования формы для добавления новых модулей

Листинг 1. Метаданные модуля «dat_reader»

```
<?xml version="1.0" encoding="utf-8"?>
<module parallel="no" nodecountmin="1" nodecountmax="1" xmlns="http://
    tempuri.org/ModuleSchema.xsd">
  <designer>
    <moduledesc desc="Модуль чтения файлов телеметрии в формате dat." />
    <vardesc name="path" desc="Путь к xml-файлу с описанием входных
        файлов телеметрии (*.dat)." />
    <channeldesc name="Output" desc="Загруженные файлы телеметрии." />
  </designer>
  <init>
    <var name="path" type="file" />
    <var name="loop" type="int">1</var>
    <var name="part_size" type="int">2147483647</var>
  </init>
  <channels>
    <channel type="out" name="Output" datatype="DAT_Struct" />
  </channels>
</module>
```

В результате произведенных действий получаем схему задачи, в которую добавлены три модуля без установленных каналов передачи данных (см. рис. 7). Дадим комментарий к рис. 7. Расположенному в центре модулю типа «annsave» автоматически было присвоено внутрен-

Рис. 8. Динамическая панель инструментов в состоянии выбора пары модулей

нее уникальное имя «internalname 1» (первичный ключ), необходимое для того, чтобы можно было добавить в схему неограниченное количество модулей и в дальнейшем корректно их связывать. Черными прямоугольниками с красным текстом обозначены точки подключения каналов связи (текст в таких прямоугольниках приводится в формате «Имя канала< : Тип данных>»). Если модуль/модули выделен(ы) с помощью манипулятора типа «мышь», то граница синего прямоугольника, ограничивающего сведения о типе модуля и его «internalname», становится пунктирной. Информация о типе модуля визуализируется текстом фиолетового цвета. Интерфейс обеспечивает точечный высококачественный рендеринг схемы с управляемым коэффициентом масштабирования, при этом обеспечивается поддержка технологии сглаживания (anti-aliasing) и поддержка средств визуализации с высоким разрешением (например, retina-систем).

Листинг 2. Метаданные модуля «annsave»

```
<?xml version="1.0" encoding="utf-8"?>
<module parallel="no" nodecountmin="1" nodecountmax="1" xmlns="http://
    tempuri.org/ModuleSchema.xsd">
  <designer>
    <moduledesc desc="Модуль сохранения настроек ИНС." />
    <vardesc name="path" desc="Полный или относительный путь к
        создаваемому xml-файлу." />
    <channeldesc name="Input" desc="Канал для получения данных." />
  </designer>
  <init>
    <var name="path" type="file" />
  </init>
  <channels>
    <channel type="in" name="Input" datatype="DataResults" />
  </channels>
</module>
```

Для примера сформируем канал связи между модулями с внутренними именами «internalname 3» и «internalname 1». Первый об-

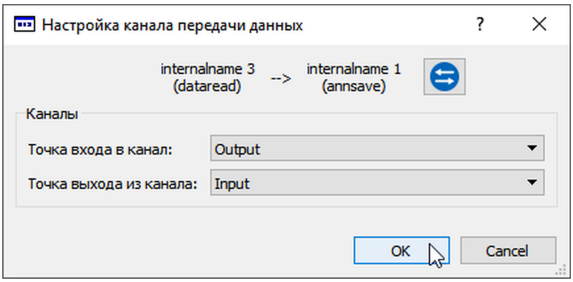


Рис. 9. Форма для добавления нового канала связи

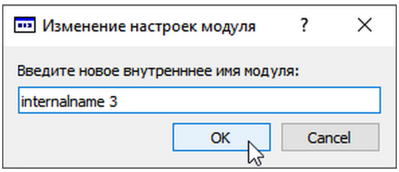


Рис. 10. Форма изменения внутреннего имени модуля

ладает каналом «Output», второй — каналом «Input», при этом типы передаваемых данных совпадают (передаются данные в формате «DataResults») (см. рис. 7). Приведение одних форматов передаваемых данных к другим осуществляется сторонней вычислительной системой. При выборе пары модулей автоматически становятся доступными контекстное меню (см. в таблице 1 строку №2) и кнопки-иконки (2) и (3) на панели инструментов (см. рис. 5). Соответственно, могут быть выполнены функции добавления канала связи и удаления выделенных модулей (см. рис. 8).

Для добавления нового канала связи следует выполнить одно из действий: нажать на соответствующую кнопку-иконку (2) на панели инструментов (см. рис. 5 и 8); воспользоваться сочетанием клавиш «Ctrl + L»; вызвать контекстное меню и нажать на необходимую кнопку (см. в таблице 1 строку №2). В результате появится диалоговое окно, представленное на рис. 9.

По умолчанию принимаем, что верхний модуль является первым, а нижний вторым. Если такой порядок не устраивает пользователя, то он может поменять способ их связи в цепочке, для этого предусмотрена кнопка-иконка с двумя стрелками на фоне синего круга (см. рис. 9). Списки каналов в форме формируются автоматически. Для точки

входа в канал формируется список вариантов от первого модуля, для точки выхода — соответствующий список от второго модуля. Уже добавленный канал передачи данных можно удалить в главном окне интерфейса, используя для этого кнопку-иконку (3) на панели инструментов (см. рис. 10), клавишу «Del» или контекстное меню (см. в таблице 1 строку №3).

Листинг 3. Текущая схема решения задачи

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<task xmlns="http://tempuri.org/KernelTask.xsd">
  <modules>
    <module internalname="internalname 3" name="dataread">
      <var name="path"></var>
    </module>
    <module internalname="internalname 2" name="dat_reader">
      <var name="path"></var>
      <var name="loop">1</var>
      <var name="part_size">2147483647</var>
    </module>
    <module internalname="internalname 1" name="annsave">
      <var name="path"></var>
    </module>
  </modules>
  <channels>
    <channel>
      <out>internalname 3.Output</out>
      <in>internalname 1.Input</in>
    </channel>
  </channels>
</task>
```

Кнопка-иконка (7) на панели инструментов (см. рис. 5) позволяет сохранить в XML-файлах текущую схему решения задачи и одновременно с этим информацию о геометрическом положении модулей на сцене (см. листинги 3 и 4).

Листинг 4. Информация о текущем геометрическом положении модулей на сцене

```
<?xml version="1.0"?>
<locations xmlns="http://tempuri.org/Locations.xsd">
  <location nodeInternalname="internalname 3" X="1846" Y="2911" />
  <location nodeInternalname="internalname 2" X="2034" Y="2995" />
  <location nodeInternalname="internalname 1" X="1847.5" Y="3076.86" />
</locations>
```

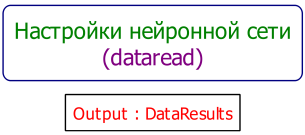


Рис. 11. Обновленный вид модуля в схеме

В листинге 3 следует отметить две основные секции. В секции «modules» приводится информация об используемых в схеме модулях, в том числе текущие значения переменных. Например, у модуля «internalname 2» переменная «loop» по умолчанию установлена в значение «1». В секции «channels» приводится информация о подключенных каналах связи: в подсекции «out» — сведения об источнике данных в формате «[Внутреннее имя модуля].[Имя out-канала]», в подсекции «in» — сведения о точке назначения данных в формате «[Внутреннее имя модуля].[Имя in-канала]». Некоторые модули могут быть не подключены к другим каналами связи, что вполне допускается, так как модуль может работать независимо от других. В листинге 4 для каждого модуля в схеме приводится его «internalname» и, соответственно, координаты на сцене.

По умолчанию добавленным модулям присваиваются внутренние имена в формате «internalname X». По своей сути «internalname» является таким же параметром, как и остальные. Для того, чтобы его изменить, можно пойти двумя путями. Первый путь предполагает выполнение действия типа «двойной клик» на области, очерченной синим прямоугольником со скругленными краями (см. рис. 1 и 7). В результате появится диалоговое окно, представленное на рис. 10. Если задать новое имя — «Настройки нейронной сети», то в схеме модуль визуализируется соответствующим образом (см. рис. 11).

Опишем другой путь настройки модуля. Некоторые параметры остались без инициализации, что отражено в листинге 3, например, параметр «path». Для редактирования параметров (в том числе и «internalname») нужно вызвать специальное окно управления параметрами (см. рис. 12). Это можно сделать как с помощью кнопки-иконки (6) на панели инструментов (см. рис. 5), так и с помощью контекстного меню (см. в таблице 1 строку №4).

Форма допускает изменение позиции модуля; внутреннего имени; цвета текста «internalname»; цвета границы прямоугольной зоны с информацией о модуле и его типе; цвета фона. Доступно управление параметрами; для параметров с типом «file» (путь к файлу) или «dir»

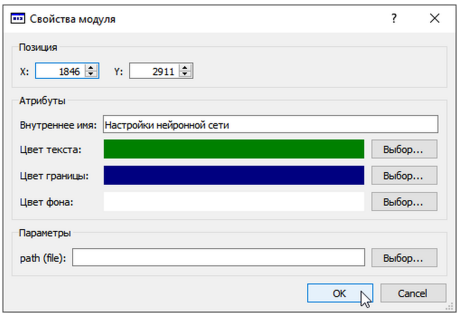


Рис. 12. Основная форма редактирования параметров модуля

(путь к директории) могут быть вызваны соответствующие диалоговые окна.

4. Запуск подготовленной программы на выполнение

Перед запуском программы на выполнение необходимо сохранить текст задачи. Если пользователь забудет сохранить схему, то перед запуском программы на выполнение ему будет автоматически отправлен соответствующий запрос на сохранение (такой же запрос будет отправлен при завершении работы графического интерфейса в случае, если схема не была сохранена).

Листинг 5. Параметризация запуска сторонней вычислительной системы

```
mpiexec.exe -machinefile .\hosts_local -channel nemesis -np %1 .\tester.exe %3
```

Параметризация запуска сторонней вычислительной системы выполняется с помощью специального файла «run_kernel.bat» (см. листинг 5). Экспериментальная проверка проводилась с подключением системы организации вычислений «Nervetic 1.0» [17], в которой в качестве средства поддержки запущенных процессов применяется библиотека MPI (в частности, MPICH [23]). Параметром «machinefile» задается файл со списком доступных вычислительных узлов; с помощью параметра «channel» выбирается транспорт передачи сообщений; параметр «np» (число запускаемых вычислительных процессов) определяется передаваемым от графического интерфейса числом «%1» (количество ЦПУ), второй параметр от интерфейса — «%2» (количество ГПУ) — в экспериментах не использовался, так как имеющаяся реализация вычислительной системы не позволяет ограничивать количество используемых ГПУ (используются все доступные); третий

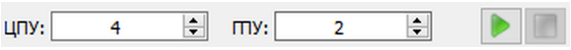


Рис. 13. Панель инструментов для запуска подготовленной программы на выполнение

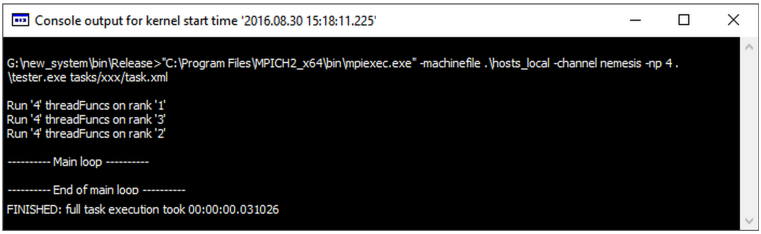


Рис. 14. Вспомогательное окно для отслеживания процесса решения поставленной задачи

параметр от интерфейса — «%3» (путь к файлу с текстом задачи) — передается всем запущенным процессам «tester.exe».

Для выполнения схемы нужно определиться с количеством и характером используемых вычислительных устройств. Интерфейс позволяет задать количество исполнительных ресурсов (см рис. 13). После нажатия кнопки начала выполнения задачи (с иконкой в виде зеленой треугольной стрелки, см. рис. 13) появится окно для отслеживания процесса решения поставленной задачи (см. рис. 14). Окно не закрывается автоматически после завершения счета задачи, что позволяет изучить лог сообщений от сторонней вычислительной системы. После успешного завершения счета появляется соответствующее окно-уведомление (см. рис. 15). Дополнительно информация о проведенных запусках сторонних вычислительных систем сохраняется в специальном логе сообщений в главном окне интерфейса, который по умолчанию скрыт, но может быть в любой момент развернут (см. рис. 16). Пользователь может в любой момент остановить счет, воспользовавшись кнопкой с иконкой в виде прямоугольника (см. самую последнюю кнопку-иконку на рис. 13).

5. Сравнение разработанного интерфейса с существующими аналогами

Далее более детального сравнения рассмотрим особенности интерфейсов, встроенных в классические системы [1–9].

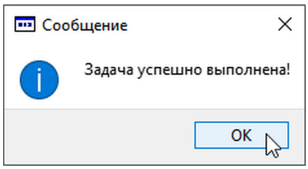


Рис. 15. Успешное завершение выполнения задачи

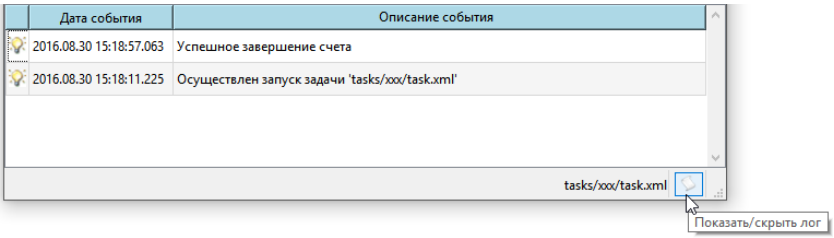


Рис. 16. Лог сообщений о последних запусках программ на выполнение

«CODE» (Computationally-Oriented Display Environment) — графическая среда разработки параллельных программ, предназначенная для задач, имеющих крупнозернистый параллелизм. Задача в целом представляется графом, содержащим последовательные участки (узлы — подпрограммы, реализованные на языках высокого уровня) и определяющим связи между этими подпрограммами. Функционирует на спектре UNIX-платформ, в качестве аппаратной платформы могут использоваться как компьютеры с SMP-архитектурой, так и системы с распределенной памятью. Система «CODE» реализована на языке Tcl и является, по своей сути, системой кодогенерации, позволяющей на основании заданного графа и сегментов программ, написанных на C-совместимых языках, построить программу, функционирующую в параллельном режиме. Требует от разработчика понимания принципов параллельного программирования [1].

«HeNCE» (HEterogeneous Network Computing Environment) — графическая система проектирования параллельных программ, работающих на базе сети рабочих станций, представляет задачу в виде ориентированного ациклического графа и дуг-зависимостей, основана на библиотеке PVM, работает на платформе UNIX. Включает в себя набор утилит для создания, компиляции, выполнения и анализа разработанных программ. Обеспечивает высокий уровень абстрак-

ции для спецификации параллелизма. В том числе имеются средства графического конфигурирования гетерогенного PVM-кластера для запуска программы. Предназначена для разработчиков, желающих эффективно использовать имеющуюся сеть рабочих станций, при этом не вдаваясь в подробности параллельного программирования [2].

«GRADE» — набор средств программирования для разработки приложений с использованием модели передачи сообщений. Представляет собой графический редактор параллельных программ, совмещенный с препроцессором для языка C, создающий параллельные программы на основе PVM/MPI-библиотек. Способен функционировать на гетерогенных кластерах. Включает в себя средства мониторинга и отладки параллельных программ. Функционирует на UNIX-платформе. Состоит из шести средств: «GRAPNEL» — графический язык параллельного программирования; «GPED» — графический редактор для построения «GRAPNEL»-программ; «GRP2C» — препроцессор для генерации программ на PVM/C; «Tape/PVM» — средство генерации трасс; «DDBG» — распределенный отладчик; «PROVE» — средство визуализации трасс [3].

«EDPEPPS» — интегрированная среда для проектирования переносимых параллельных программ. Функционирует на основе PVM. Включает в себя утилиты графического конфигурирования, моделирования параллельных процессов и средства визуализации результатов. Интегрированная среда для обеспечения быстрого, эффективного и гибкого проектирования переносимых параллельных программ. В «EDPEPPS» входят три основных средства:

- графическое средство конфигурирования (configuration tool) для быстрого прототипирования;
- средство моделирования (simulation tool), основанное на виртуальной машине, исполняющей «скелетные» параллельные программы, — генерирует данные о поведении и предполагаемой производительности программы;
- средство визуализации (visualisation tool) для отображения интересующих пользователя характеристик с привлечением для этого данных, полученных при моделировании [4].

Комплекс решений для проектирования и поддержки исполнения «Microsoft Workflow Foundation» включает полнофункциональный визуальный редактор, позволяющий использовать принципы визуально-блочного программирования при разработке различных прикладных workflow-систем. Разработку прикладных решений можно вести на

любом из языков платформы .Net. Позволяет использовать созданные решения как в виде независимого программного продукта, так и в виде web-сервисов. Для организации высокопроизводительных вычислений на многопроцессорной вычислительной системе от прикладного программиста требуется понимание принципов параллельного программирования [5].

Workflow-система «Triana» используется для проектирования и решения прикладных задач в различных областях. Обладает кроссплатформенностью в силу использования технологии Java как основы для реализации. Основным назначением является проектирование прикладных систем без понимания принципов программирования. Поддерживает возможность использования многопроцессорных вычислительных систем для решения прикладных задач [6].

Workflow-система «ANSYS EKM» ориентирована на поддержку бизнес-процессов, позволяет определить общую схему процесса и поддерживает его исполнение. В качестве базовых элементов (модулей схемы) выступают различные web-сервисы, которые могут быть интегрированы для решения прикладной задачи. Значительное внимание уделено управлению доступом к данным и нотификации участников процесса [7].

Система «Kepler» ориентирована на проектирование и поддержку исполнения workflow-процессов для решения широкого круга прикладных задач. Обладает модульной структурой, поддерживает библиотеку специализированных модулей обработки данных. Реализована на основе платформы Java, что обеспечивает кроссплатформенность программного продукта. Использует принцип визуального программирования на этапе проектирования прикладных систем. В состав комплекса входит ряд модулей, позволяющих интегрировать систему с существующими batch-системами многопроцессорных вычислительных систем, что дает использовать ресурсы мультипроцессорных установок для решения прикладных задач. Поддерживается принцип конвейеризаций вычислений. Для реализации прикладных систем, функционирующих в конвейерно-параллельном режиме, требуется понимание принципов параллельного программирования [8].

Программная система «НСКиД» позволяет эффективно решать как вычислительные задачи, допускающие внутренний (потоковый) параллелизм в рамках одного процесса, так и задачи, требующие распределения по данным между несколькими серверами. Гибкость системы обеспечивается использованием двух механизмов: каналы — у

каждого конкретного модуля есть набор входов и выходов, модуль не знает о системе в целом (или цепочке обработки данных, в которой он использован), он лишь получает данные на обработку и отсылает обработанные результаты; схема описания задачи — позволяет задать произвольный набор модулей и связей между ними в виде декларативного описания, которое может быть построено с использованием графического интерфейса. Используется двухуровневый контроль действий пользователя: графический интерфейс на этапе формирования текста прикладной задачи информирует пользователя об ошибках в ее описании; вычислительное ядро в процессе анализа и решения задачи взаимодействует с графическим интерфейсом. Тот, в свою очередь, анализирует получаемые сообщения и уведомляет пользователя о критических событиях [9].

Основываясь на выполненном анализе, можно сделать некоторые обобщающие выводы. Графические интерфейсы визуального проектирования, интегрированные в системы [1–8], фактически являются атрибутами систем организации вычислений или библиотек поддержки параллелизма и не являются отчуждаемыми от них. В программной системе «НСКиД» [9] интерфейс по сути является частично отчуждаемым, поскольку он отслеживает сообщения, отсылаемые вычислительным ядром в стандартный поток вывода, полученная информация используются им для отслеживания текущего состояния запущенной вычислительной задачи.

Дополнительно можно сказать следующее. Разработанный графический интерфейс является кроссплатформенным. При этом он не имеет ряда проблем совместимости с Linux-системами из-за особенностей, характерных для Linux-версии C# (Mono). Единое рабочее пространство с одним основным инструментарием управления оказалось весьма удобным, что показали многочисленные опросы пользователей, использующих подобные системы (к примеру, графический интерфейс системы «НСКиД» имеет фактически три взаимодополняющих инструмента для работы со схемой задачи; опросы выполнены автором настоящего исследования). Только интерфейс системы «НСКиД» и новый интерфейс имеют средства для полноценной настройки визуального представления схемы задачи. Схема задачи в графических интерфейсах всегда визуализируется в виде графа, однако не всегда отдельные элементы схемы можно свободно перемещать по рабочему пространству (например, положение блоков в системах «CODE», «HeNCE» и «GRADE» строго зафиксировано).

Лишь в системе «НСКиД» форма с параметрами модулей всегда отображается в основном окне (без возможности скрыть эту форму), разработанный интерфейс удовлетворяет общепринятым требованиям и отображает эту форму лишь по запросу пользователя. Недостаток текущей реализации интерфейса — отсутствие средств для пошагового отката только что сделанных изменений. Из рассмотренных систем такой возможностью обладают лишь «Microsoft Workflow Foundation» и «НСКиД». Отличительной особенностью практически всех интерфейсов является отсутствие специализированных встроенных средств визуализации результатов выполнения вычислительной схемы. Только графические интерфейсы систем «ANSYS EKM» и «НСКиД» обладают встроенными средствами для визуализации результатов. Новый интерфейс стал более универсальным и ориентируется на разработанные пользователями специальные модули визуализации, включаемые в схему задачи на общих правах с основными модулями (то есть эти модули визуализации не являются частью интерфейса).

Заключение

В настоящей работе приведены основные особенности и детали реализации многофункционального графического интерфейса, позволяющего строить визуальные схемы различных вычислительных процессов, функционирующих в конвейерно-параллельном режиме, проводить эксперименты на реальных вычислительных устройствах и решать необходимые прикладные задачи.

Удобство интерфейса при проектировании вычислительных схем высокой сложности подтверждено выполненным сравнением с аналогами и опытом использования в ИПС им. А.К. Айламазяна РАН при выполнении работ космической тематики.

Предполагается дальнейшее совершенствование интерфейса: расширение подсистемы настроек, реализация функции сохранения схемы задачи в виде векторного изображения и проработка системы подключаемых модулей для визуализации результатов вычислений.

Список литературы

[1] *CODE Visual Parallel Programming System*, URL: http://www.cs.utexas.edu/users/code/ ↑ 47,61,62,65

[2] *HeNCE (Heterogeneous Network Computing Environment)*, URL: http://www.netlib.org/hence/ ↑ 47,61,63,65

[3] *GRADE — Graphical Environment for Parallel Programming*, URL: http://www.ercim.eu/publication/Ercim_News/enw36/kacsuk.html ↑ 47,61,63,65

[4] *An Environment for the Design and Performance Evaluation of Portable Parallel Software*, Final EDPEPPS Simulator: Report / Center for Parallel Computing University of Westminster. EDPEPPS/41. London, July 1997. ↑ 47,61,63,65

[5] Д. Шукла, Б. Шмидт. *Основы Windows Workflow Foundation*, ДМК Пресс, М., 2008, 352 с. ↑ 47,61,64,65

[6] *Triana — Open Source Problem Solving Software*, URL: http://www.trianacode.org/ ↑ 47,61,64,65

[7] *ANSYS — Simulation Driven Product Development*, URL: http://www.ansys.com/ ↑ 47,61,64,65

[8] *The Kepler Project — Kepler*,URL: https://kepler-project.org/ ↑ 47,61,64,65

[9] В. М. Хачумов, И. П. Тищенко, А. А. Талалаев, К. А. Константинов, В. П. Фраленко, Ю. Г. Емельянова. *Нейросетевая система контроля телеметрической информации, диагностики подсистем космических аппаратов, обработки космических снимков (ПС НСКиД)*, Свидетельство о государственной регистрации программы для ЭВМ № 2012613261, дата приоритета: 18.11.2011, дата регистрации: 06.04.2012. ↑ 47,61,65

[10] Marc Bux, Jörgen Brandt, Carsten Lipka, Kamal Hakimzadeh, Jim Dowling, Ulf Leser. *Proceedings of the VLDB Endowment*, т. **8 12**, 2015, URL: https://www2.informatik.hu-berlin.de/~buxmarcn/publications/bux_saasfee_vldb_2015.pdf ↑ 47

[11] Anže Starič, Janez Demšar, Blaž Zupan. «Concurrent software architectures for exploratory data analysis», *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **5**:4 (2015), с. 165–180, URL: http://eprints.fri.uni-lj.si/3191/1/2015-Staric-WIREs-DMKD.pdf ↑ 47

[12] В. П. Фраленко. «Графический интерфейс программной системы распознавания образов, основанный на моделях искусственных нейронных сетей», XI ежегодная научно-практическая конференция УГП им. А.К. Айламазяна «Программные системы: теория и приложения» (2007), с. 97–103, URL: https://docs.google.com/uc?export=download&id=0B-Qay3kEFxqfdWdncFA5Q1VNeTg ↑ 47

[13] А. Н. Виноградов, Ф. В. Калугин, М. Д. Недев, С. В. Погодин, А. А. Талалаев, И. П. Тищенко, В. П. Фраленко, В. М. Хачумов. «Выделение и распознавание локальных объектов на аэрокосмических снимках», *Авиакосмическое приборостроение*, 2007, №9, с. 39–45. ↑ 47

[14] Ю. Г. Емельянова, А. А. Талалаев. «Сетевые модели функционирования прикладных параллельных систем обработки потоков данных», *Авиакосмическое приборостроение*, 2012, №5, с. 10–19. ↑ 47

[15] И. П. Тищенко, Д. Н. Степанов, В. П. Фраленко. «Разработка системы моделирования автономного полета беспилотного летательного аппарата», *Программные системы: теория и приложения*, 2012, №3, с. 3–21, URL: http://psta.psiras.ru/read/psta2012_3_3-21.pdf ↑ 47

[16] А. А. Талалаев, В. П. Фраленко. «Комплекс инструментальных средств для проектирования нейросетевых прикладных систем», *Научно-технический вестник Поволжья*, 2013, №4, с. 237–243. ↑ 47

[17] А. А. Талалаев, В. П. Фраленко. *Визуально-блочное проектирование нейросетевых прикладных систем (ПК «Nervetic 1.0»)*, Свидетельство о государственной регистрации программы для ЭВМ № 2014611688, дата приоритета: 11.12.2013, дата регистрации: 07.02.2014. ↑ 47,60

[18] А. Ю. Агроник, В. П. Фраленко. «Библиотека алгоритмов высокопроизводительной обработки данных от системы технического зрения беспилотного летательного аппарата», *Программные системы: теория и приложения*, 2016, №2, с. 61–71, URL: http://psta.psiras.ru/read/psta2016_2_61-71.pdf ↑ 47

[19] *Qt — Download Open Source (RU)*, URL: http://www.qt.io/ru/download-open-source/ ↑ 48

[20] *Лицензии — Проект GNU — Фонд свободного программного обеспечения*, URL: http://www.gnu.org/licenses/ ↑ 48

[21] *OpenGL Overview*, URL: https://www.opengl.org/about/ ↑ 51

[22] *Vulkan — Industry Forged*, URL: https://www.khronos.org/vulkan/ ↑ 51

[23] *MPICH | High-Performance Portable MPI*,URL: https://www.mpich.org/ ↑ 60

Рекомендовал к публикации *д.т.н. В. М. Хачумов*

Пример ссылки на эту публикацию:

В. П. Фраленко. «Универсальный графический интерфейс визуального проектирования параллельных и параллельно-конвейерных приложений», *Программные системы: теория и приложения*, 2016, **7**:3(30), с. 45–70.

URL: http://psta.psiras.ru/read/psta2016_3_45-70.pdf

Об авторе:

Виталий Петрович Фраленко

К.т.н., старший научный сотрудник Исследовательского центра мультипроцессорных систем ИПС им. А. К. Айламазяна РАН, автор более 70 публикаций. Область научных интересов: распознавание образов, искусственный интеллект и принятие решений, параллельные алгоритмы, графические интерфейсы.

e-mail: alarmod@pereslavl.ru

Vitaly Fralenko. *Universal graphical user interface for visual design of parallel and parallel-pipelined applications.*

Abstract. The article describes a multipurpose graphical user interface designed for the visual design of computational processes and process chains in a variety of application areas. User formulates an applied problem through a set of ready-made function blocks and link editor for establishing data transfer channels, launches tasks schemes in parallel-pipeline mode and tracks performance in a built-in tools.

A comparison with existing analogues revealed a interface features that extends the capabilities of software developers, helps the programmer to see the problem as a whole scheme, accelerates the development process, reduces the number of errors, and increases the efficiency of decision-making. (*In Russian*).

Key words and phrases: graphical user interface, support for computing, parallelism, conveyor, scheme, task.

References

[1] *CODE Visual Parallel Programming System*, URL: http://www.cs.utexas.edu/users/code/

[2] *HeNCE (Heterogeneous Network Computing Environment)*, URL: http://www.netlib.org/hence/

[3] *GRADE — Graphical Environment for Parallel Programming*, URL: http://www.ercim.eu/publication/Ercim_News/enw36/kacsuk.html

[4] *An Environment for the Design and Performance Evaluation of Portable Parallel Software*, Final EDPEPPS Simulator: Report / Center for Parallel Computing University of Westminster. EDPEPPS/41. London, July 1997.

[5] D. Shukla, B. Schmidt. *Essential Windows Workflow Foundation*, DMK Press, M., 2008 (in Russian), 352 p.

[6] *Triana — Open Source Problem Solving Software*, URL: http://www.trianacode.org/

[7] *ANSYS — Simulation Driven Product Development*, URL: http://www.ansys.com/

[8] *The Kepler Project — Kepler*, URL: https://kepler-project.org/

[9] V. M. Khachumov, I. P. Tischenko, A. A. Talalaev, K. A. Konstantinov, V. P. Fralenko, Ju. G. Emeljanova. "Neural system for telemetry data control, spacecraft subsystems diagnostics and satellite images processing (PS NSKiD) // Svidetelstvo o gosudarstvennoj registracii programmy dlja JeVM No 2012613261, data prioriteta: 18.11.2011, data registracii: 06.04.2012." (in Russian).

[10] Marc Bux, Jörgen Brandt, Carsten Lipka, Kamal Hakimzadeh, Jim Dowling, Ulf Leser. *Proceedings of the VLDB Endowment*, vol. **8** **12**, 2015, URL: https://www2.informatik.hu-berlin.de/~buxmarcn/publications/bux_saasfee_vldb_2015.pdf

[11] Anže Starič, Janez Demšar, Blaž Zupan. "Concurrent software architectures for exploratory data analysis", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **5**:4 (2015), pp. 165–180, URL: http://eprints.fri.uni-lj.si/3191/1/2015-Staric-WIREs-DMKD.pdf

[12] V. P. Fralenko. “The graphic interface of image recognition program system on basis models of artificial neural networks”, XI ezhegodnaja nauchno-prakticheskaja konferencija UGP im. A.K. Ajlamazjana “Program Systems: Theory and Applications”, 2007, pp. 97–103 (in Russian), URL: https://docs.google.com/uc?export=download&id=0B-Qay3kEFxqfdWdncFA5Q1VNeTg

[13] A. N. Vinogradov, F. V. Kalugin, M. D. Nedev, S. V. Pogodin, A. A. Talalaev, I. P. Tischenko, V. P. Fralenko, V. M. Khachumov. “Isolation and Identification of Local Facilities on Aerospace Images”, *Artificial Intelligence and Decision-making*, 2007, no.9, pp. 39–45 (in Russian).

[14] Ju. G. Emeljanova, A. A. Talalaev. “The Network Models of Applied Parallel Systems Functioning for Data Flow Processing”, *Aerospace Instrument-making*, 2012, no.5, pp. 10–19 (in Russian).

[15] I. P. Tischenko, D. N. Stepanov, V. P. Fralenko. “The Development of Modeling System of Unmanned Aerial Vehicle Autonomous Flight”, *Program Systems: Theory and Applications*, 2012, no.3, pp. 3–21 (in Russian), URL: http://psta.psiras.ru/read/psta2012_3_3-21.pdf

[16] A. A. Talalaev, V. P. Fralenko. “Complex of Tools for the Neural Network Application Systems Designing”, *Scientific and Technical Volga region Bulletin*, 2013, no.4, pp. 237–243 (in Russian).

[17] A. A. Talalaev, V. P. Fralenko. “Visually-block design of neural network application systems (PC “Nervetic 1.0”) // Svidetelstvo o gosudarstvennoj registracii programmy dlja JeVM No 2014611688, data prioriteta: 11.12.2013, data registracii: 07.02.2014.” (in Russian).

[18] A. Ju. Agronik, V. P. Fralenko. “Library of High-performance Algorithms for Processing of Data from Unmanned Aerial Vehicle Vision System”, *Program Systems: Theory and Applications*, 2016, no.2, pp. 61–71 (in Russian), URL: http://psta.psiras.ru/read/psta2016_2_61-71.pdf

[19] *Qt — Download Open Source (RU)* (in Russian), URL: http://www.qt.io/ru/download-open-source/

[20] *Licenses — GNU Project — Free Software Foundation* (in Russian), URL: http://www.gnu.org/licenses/

[21] *OpenGL Overview*, URL: https://www.opengl.org/about/

[22] *Vulkan — Industry Forged*, URL: https://www.khronos.org/vulkan/

[23] *MPICH — High-Performance Portable MPI*, URL: https://www.mpich.org/

Sample citation of this publication:

Vitaly Fralenko. “Universal graphical user interface for visual design of parallel and parallel-pipelined applications”, *Program systems: theory and applications*, 2016, **7**:3(30), pp. 45–70. (*In Russian*).

URL: http://psta.psiras.ru/read/psta2016_3_45-70.pdf