УДК: 004.434

Название: Языковые средства организации вычислений в области биоинформатики

Автор(ы):

Грехов Г.А. (ООО «НЦИТ Унипро»),

Скопин И.Н. (Институт вычислительной математики и математической геофизики СО РАН, Новосибирский государственный университет)

Аннотация: В статье описан подход к созданию вычислительных платформ на примере UGENE Workflow Designer. Эта платформа, основанная на внутреннем языке программирования UWL, демонстрирует, как, следуя принципам внутренней модели платформы и синтаксиса языка программирования, сохранять систему целостной и удобной для развития и поддержки.

Система UGENE Workflow Designer предназначена для решения задач биоинформатики, а язык UWL является DSL языком этой области. В статье описаны и обоснованы возможности использования этой вычислительной платформы в качестве основы переиспользования ее средств в некоторых других прикладных областях.

Ключевые слова: языки программирования, DSL, конвейерные вычисления, *UGENE*, биоинформатика

1. Введение. Современная прикладная наука богата различного рода вычислительными задачами, и ученым ежедневно требуется вести те или иные расчеты. Кроме того, работа во многих областях знаний подразумевает не просто ведение сложных расчетов, но и многообразие решаемых вычислительных задач. Примером такой области является биоинформатика.

Исследователю этой области в своей работе требуется использовать множество программных инструментов, каждый из которых нацелен на решение конкретной задачи. То есть нужно заниматься поиском этих инструментов и осваивать их интерфейс. Зачастую найденная программа может работать в операционной системе, отличной от системы пользователя, что создает дополнительные трудности для исследователя прикладной области. Во многих случаях возникает необходимость решать последовательность вычислительных задач, решение каждой из которых использует результаты, полученные в ходе работы предыдущих. Другими словами, требуется создавать вычислительные конвейеры для серий сложных однотипных задач над различными входными данными.

Решением указанных проблем могут стать системы, представляющие собой вычислительные платформы, которые предлагают необходимый набор инструментов с единым интерфейсом, а также допускают адаптивное расширение самим пользователем.

В данной статье обсуждается такая система, предназначенная для исследователей в области биоинформатики, называемая UGENE Workflow Designer [1]. Она является частью большей системы Unipro UGENE [6], свободного программного обеспечения для работы молекулярного биолога. Workflow Designer является подсистемой, отвечающей именно за вычислительную часть вопроса, в то время как обработку результатов, полученных в ходе вычислений, можно совершать при помощи средств визуализации и редактирования UGENE. Вся система разрабатывается на языке C++ с использованием фреймворка Qt [7], что позволяет ей функционировать в операционных системах семейства Windows, UNIX и Mac OS.

Workflow Designer развивается во многом по мере поступления запросов от пользователей, и изначально эта система подвергалась экстенсивному развитию, то есть простому наращиванию функциональности в ходе удовлетворения этих запросов. Вычислительные схемы (конвейеры) сохранялись в бинарных форматах файлов, а задача разработки средств управления потоками данных в конвейерах, просмотра временных результатов не ставилась. Впоследствии стало очевидным, что без решения этой задачи система превратилась бы в чрезмерно тяжелую как для использования, так для сопровождения и дальнейшего развития.

Было принято решение рассматривать UGENE Workflow Designer как среду разработки на dataflow-языке программирования [4], а вычислительные схемы — как программы на этом языке. Предусматривается, что такое развитие должно удовлетворять следующему требованию: прежде чем вносить какое-либо улучшение в Workflow Designer, нужно проанализировать, как это улучшение будет отражено в языке, не нарушая его особенностей и избегая дублирования средств. Выполняется и другое требование, повышающее гибкость системы: построенные схемы сохраняются в файлы не в бинарном, а в текстовом виде. Текст схемы — это тоже программа на предлагаемом языке программирования, но написанная с помощью его нового текстового синтаксиса. Таким образом, появились два эквивалентных конкретных синтаксиса (графический и текстовый) одного языка программирования, который стал называться UGENE Workflow Language или, сокращенно, UWL.

UWL является DSL [5] языком. Область его использования определяется требованием решения задач биоинформатики и молекулярной биологии. Специфика этих областей выражается в основном в наборе вычислительных инструментов (стандартной библиотеке), который предоставлен пользователю. Другой особенностью языка, пришедшей из предметной области, является отсутствие циклов в вычислительных схемах, т.к. в решаемых

задачах они не требуются. С учетом сказанного выше платформу языка UWL можно переиспользовать и в некоторых других областях.

2. Вычислительная модель и абстрактный синтаксис. Язык UWL, как и любой другой язык программирования, имеет абстрактный синтаксис, который описывает его вычислительные возможности, т.е. модель вычислений, и конкретный синтаксис, определяющий, как эта модель представляется пользователям. Как уже упоминалось, UWL предлагает два варианта конкретных синтаксисов: текстовый, обеспечивающий традиционные возможности записи программ, и графический, конструкции которого изображаются как элементы схем.

Абстрактный синтаксис [5] предназначен для внутреннего представления программ и характеризуется, в основном, вычислительной моделью языка программирования. Конкретные синтаксисы [5] предназначены для программистов, то есть пользователей UGENE Workflow Designer, и непосредственно позволяют конструировать (программировать) конвейеры вычислений.

2.1. Вычислительные элементы и их порты. Описываемый язык UWL является dataflow-языком [4], поэтому основную часть его вычислительной модели составляют вычислительные элементы и соединяющие их каналы связей. Вычислительный элемент (worker или actor) — это некоторая многократно исполняющаяся программа. Это может быть реализация какого-либо биологического алгоритма, или программа, считывающая или записывающая данные на жесткий диск или удаленную базу данных, или какой-либо служебный элемент, управляющий потоками данных.

Используются вычислительные элементы трех типов:

1) чистые генераторы данных (Data readers);
2) вычислители или генераторы данных в контексте других данных (Computers);
3) писатели данных (Data writers);
4) служебные (Dataflow elements).

В UWL наиболее распространенными элементами, естественно, являются вычислители, так как они предоставляют всю вычислительную функциональность.

Каждый вычислительный элемент имеет один или более портов. Порт (Port) — это средство, с помощью которого вычислительный элемент обменивается данными с другими элементами. Соответственно, не имея портов, вычислительный элемент не имеет и смысла, так как его невозможно включить в какой-либо конвейер вычислений. Порты бывают входные, в которые приходят данные из других вычислительных элементов; и выходные, куда отправляет данные вычислительный элемент, хозяин этого порта.

Чистые генераторы данных (Data readers) могут иметь только выходные порты. Эти элементы генерируют данные (например, считывают их из файлов) и отправляют их в выходные порты остальным элементам.

Вычислители (Computers) имеют и входные, и выходные порты. Эти элементы получают данные из входных портов и на основе них (в их контексте) производят новые данные, которые отправляют в свои выходные порты. Поэтому второе название вычислителей — это генераторы данных в контексте других данных. Примером вычислителя может являться элемент поиска каких-либо интересных исследователю участков в биологических последовательностях, например, элемент поиска открытых рамок считывания (или ORF). Он принимает биологическую последовательность из входного порта, ищет в ней ORF, представляющие собой обычные биологические аннотации, и отправляет найденные аннотации в выходной порт. Эти аннотации произведены и существуют в контексте исходной биологической последовательности.

Писатели данных (Data writers) имеют только входные порты. Эти элементы получают данные из входных портов и записывают их, например, в файл, получая конечный продукт вычислительного конвейера.

Служебные элементы (Dataflow elements), как и вычислители, имеют и входные, и выходные порты, но они не имеют вычислительной семантики. Они используются для управления потоками данных. Эти элементы отвечают за такие процессы, как маркировка и фильтрация данных, группировка (merging), мультиплексирование двух потоков данных в один.

Соединяя между собой порты вычислительных элементов, пользователь строит вычислительную схему (конвейер, workflow). Схема представляет собой ориентированный граф, вершинами которого являются вычислительные элементы, а ребрами — соединения портов. Этот граф также называется графом соединения портов или графом потоков данных.

Исходя из специфики dataflow [4], вычислительные элементы могут исполняться только тогда, когда в их входных портах появились все необходимые входные данные. Соответственно, первыми в схеме исполняются чистые генераторы данных, которые не имеют входных портов. Чистые генераторы данных необходимы для работы схемы, иначе ни один из вычислительных элементов ни разу не сможет исполниться, так как не будет иметь входных данных.

2.2. Параметры. Помимо портов, вычислительные элементы могут иметь параметры. Они необходимы для настройки работы схемы. Например, у читателей данных есть такой параметр, как входные файлы, а у писателей — выходные файлы. Различные

вычислительные алгоритмы могут иметь множество параметров, таких как пороги тех или иных величин, процент точности поиска, выбор реализаций алгоритмов и многое другое.

Параметры бывают обязательными и опциональными. Первые необходимо указывать для управления работой элемента. Например, нужно обязательно задать имя входного файла для считывания данных. Если пользователь не выставил какой-либо обязательный параметр в схеме, то такая схема не пройдет валидацию и не запустится.

Опциональные параметры могут быть не заданы, и в этом случае во время запуска схемы будут использованы их значения по умолчанию. Выбор того, какие из параметров назначить обязательными, а какие опциональными, остается за программистом.

2.3. Шина данных. Соединяя между собой выходной и входной порты каких-либо двух вычислительных элементов, пользователь создает шину данных [4]. Шины данных (Bus) служит для передачи данных между вычислительными элементами. Они также образуют ребра графа потоков данных.

Выходной порт может иметь одну или более шин данных (т.е. можно передавать данные из одного вычислительного элемента нескольким другим). А входной порт может иметь ровно одну шину (т.е. его вычислительный элемент принимает данные ровно от одного элемента). Последнее ограничение служит для упрощения вычислительной модели. Отказ от него ведет к внедрению мультиплексирования потоков данных внутри одного порта, а это сложно как для программирования, так и для понимания пользователями. Вместо этого следует использовать служебный элемент мультиплексор, который соединит два потока данных (две шины) в один, и затем по новой шине направить этот поток в порт нужного элемента.

2.4. Сообщения. Данные в шине передаются в сообщениях. Сообщение — это форма упаковки данных для передачи по шине. Одно сообщение может содержать несколько единиц данных, и каждая единица имеет свой идентификатор. Другими словами, сообщение — это структура, представляющая собой ассоциативный массив, в котором ключом является идентификатор, а значением — единица передаваемых данных.

В описываемой вычислительной модели сообщения, входящие в порт некоторого вычислителя, автоматически передаются на выходные порты этого элемента и ждут отправления. Когда вычислитель отправляет новые данные в свой выходной порт, то эти новые данные и старое входное сообщение (автоматически помещенное на выход и ждущее отправления) соединяются в новое сообщение, которое отправляется на все выходные шины. Таким образом, данные, произведенные первым генератором данных, в конечном итоге достигнут последнего писателя данных.

Описанный механизм говорит о том, что данные передаются в шинах в контексте своего существования. Вычислительная модель поддерживает некоторые способы генерации новых контекстов (используя служебные элементы), но, в любом случае, во время работы схемы всегда выполняются следующие свойства:

1. Данные в шине передаются всегда или в полном контексте своего существования в схеме, или вообще без своего контекста.
2. Данные в шине нельзя модифицировать. Можно лишь воспроизвести новые данные в контексте других.

Эти два пункта являются основным требованием к расширению системы, на котором базируется вся вычислительная модель, и который следует учитывать при развитии языка и разработке новых элементов.

2.5. Слоты. Из сказанного выше понятно, что сообщение может содержать несколько единиц данных. Это достигается тем, что данные из сообщений не удаляются, а только добавляются. И, в конце концов, в порт какого-либо вычислителя может прийти сообщение, содержащее различные единицы данных, предназначенные для разных вычислителей. Некоторых из них передаются для этого вычислителя, а другие нужно просто передать в выходной порт. Для разграничения таких единиц данных предназначены слоты.

Слот — это параметр порта, описывающий данные, приходящие во входной порт или выходящие из выходного порта. Таким образом, порт — это не просто сущность, принимающая или отправляющая сообщения в шину, а набор слотов, описывающих данные, которые этот порт принимает или отдает. Соответственно, как и порты, слоты бывают входные и выходные.

Выходные слоты предназначены лишь для того, чтобы идентифицировать отправляемые в порт данные сообщения. Они задают тот самый ключ для ассоциативного массива сообщений.

Входные слоты позволяют задавать, какие из единиц данных пришедшего сообщения предназначены для этого порта. Таким образом, решается проблема разграничения данных в сообщениях.

Пользователь при конструировании схемы, помимо соединения портов, настраивает, как соединены и слоты. Это значит, что пользователь указывает, какие именно данные в шинах предназначены для того или иного вычислительного элемента. Соединение портов создает так называемые соседние вычислительные элементы в графе потоков данных. И между этими соседними элементами создается шина передачи данных. Соединение слотов

описывает передачу данных не только между соседними элементами, а между двумя произвольными элементами, между которыми есть путь в ориентированном графе соединения портов.

2.6. Идентификация данных. Каждый вычислительный элемент, порт и слот имеют свои идентификаторы: element_id, port_id и slot_id, соответственно. Причем каждый из них уникален в своем контексте. Например, идентификатор слота уникален в контексте своего порта.

Каждой единице данных в сообщении можно присвоить идентификатор вида

element_id.port_id.slot_id

Так как в одном сообщении может содержаться не более одной единицы данных с каждого слота, то такой идентификатор является уникальным в контексте одного сообщения, а значит, это позволяет полноценно функционировать всем описанным механизмам.

3. Модель данных.

3.1. Типизация. В описываемой вычислительной модели поддерживается строгая статическая типизация. Все данные, передаваемые по шинам, имеют конкретный тип. И каждый слот описывается типом передаваемых/принимаемых данных. Соединять два слота можно только в том случае, если их типы совпадают. Таким образом, единица данных, помещенная в какое-либо сообщение, идентифицируется типизированным слотом, а значит, может быть передана только слоту с таким же типом.

Соответствие типов проверяется статически, т.е. перед стартом вычислений на стадии валидации. При несовпадении типов пользователь будет об этом извещен, а вычисления не будут начаты.

В настоящее время UGENE Workflow Designer позволяет оперировать следующими типами данных:

1) Text — любые текстовые данные;

2) Sequence — биологическая последовательность. Содержит символы, описывающие нуклеотиды ДНК, РНК или белковых последовательностей, и некоторые метаданные;

3) MSA — множественное выравнивание. Содержит список последовательностей и информацию о расположении их друг относительно друга;

4) Assembly — данные ассемблирования. Содержит результаты работы ассемблеров;

5) Variation track — набор вариаций. Содержит данные о вариациях: SNP, инсерции, делеции;

6) Annotation table — набор аннотаций. Описывает участки в последовательностях, выравниваниях и данных ассемблирования дополнительными сведениями.

3.2. Интерфейс базы данных. Архитектура всей системы UGENE (не только Workflow Designer) предоставляет способ взаимодействия с базой данных. В ней описан интерфейс работы с базой данных (database interface, dbi), реализуя который, программист может быстрым и легким способом начать использовать ту или иную СУБД.

UGENE Workflow Designer предназначен прежде всего для вычислений над данными очень большого объема, которые не помещаются в оперативной памяти. Поэтому использование базы как временного хранилища данных, является очень эффективным средством для достижения этой цели.

В настоящее время происходит постепенное внедрение dbi в работу вычислительных схем. Цель этого внедрения — сделать так, чтобы в шинах передавались не сами данные, а их идентификаторы во временной базе данных. Применение этого механизма уже произведено для самых ресурсоемких типов данных в UGENE Workflow Designer: sequence, msa, assembly, variation track.

Использование описанного механизма очень полезно для вычислений над данными большого объема, ведь он позволяет, например, не держать в оперативной памяти все 3.2 Гб генома человека, а доставать из базы данных только нужную в данный момент его часть.

В настоящее время UGENE предоставляет возможность работы с довольно небольшой и быстрой базой SQLite [3].

4. Конкретный синтаксис. Язык программирования UWL имеет два конкретных синтаксиса: графический и текстовый.

Графический конкретный синтаксис с помощью визуальных средств UGENE Workflow Designer позволяет конструировать вычислительную схему. Пользователю предоставляется графическая сцена и набор вычислительных элементов (палитра), которые он может с помощью мыши добавлять на сцену, перемещать для удобного визуального расположения и соединять порты элементов стрелками, создавая, таким образом, шины данных. Также, существуют графические элементы для регулирования параметров и соединения слотов.

Текстовый конкретный синтаксис предоставляет пользователю все возможности конструирования схем, используя текстовый редактор. Он будет подробно описан ниже.

Графический и текстовый синтаксисы являются эквивалентными. То есть любая вычислительная схема, сконструированная с помощью одного синтаксиса, может быть сконструирована и с помощью другого синтаксиса. Более того, можно создать какую-либо схему с помощью графического синтаксиса, сохранить ее в файл, и этот файл будет семантическим эквивалентом графической схемы в текстовом синтаксисе. Изменив

текстовый файл, можно переоткрыть его в UGENE Workflow Designer и наблюдать изменения.

5. Текстовый синтаксис. В данной статье проведен поверхностный обзор основных конструкций текстового синтаксиса UWL. Формальное описание этого синтаксиса доступно в документации UGENE Workflow Designer [1].

Программа, написанная с помощью текстового синтаксиса, состоит из нескольких частей:

1. Заголовок файла.
2. Описание схемы:
    1) описание вычислительных элементов;
    2) описание соединения портов;
    3) описание соединения слотов;
    4) метаданные.

Заголовок файла схемы на языке UWL состоит из первой обязательной строки и нескольких строк-комментариев, описывающих предназначение схемы.

Пример заголовка:

```
#@UGENE_WORKFLOW
# Описание предназначения
# схемы
```

Описание схемы — это один блок, содержащий в себе все внутренние описания. Блок озаглавлен ключевым словом workflow и опционально может содержать имя схемы.

Пример блока схемы:

```
workflow “Имя схемы” {
}
```

Описание каждого вычислительного элемента схемы — это отдельный блок, содержащий значения параметров. Заголовок блока — это уникальный идентификатор элемента, о котором было сказано в описании вычислительной модели. Обязательными параметрами каждого элемента являются имя и тип элемента. Тип идентифицирует функциональное предназначение элемента (что это за элемент).

Пример вычислительного элемента:

```
read-file {
   type : read-text;
   name : “Прочитать текстовые данные”;
   url-in : D:/file.txt;
}
```

Как видно из примера, вычислительный элемент имеет тип read-text, т.е. этот элемент считывает входной файл в строку (а именно, D:/file.txt, указанный в параметре url-in) и отправляет считанные данные в выходной порт.

Описание соединения портов также содержится в отдельном блоке, который называется служебным словом «.actor-bindings». Идентификаторы вычислительных элементов не могут содержать символ '.', поэтому никаких конфликтов имен не может быть. Внутренний синтаксис этого блока — это перечисление соединения портов вида:

element-id-1.port-id-1 -> element-id-2.port-id-2

Граф потоков данных не должен содержать циклов (это проверяется при открытии схемы).

Описание соединения слотов не содержится ни в каких блоках, а состоит из простого перечисления вида:

element-id-1.port-id-1.slot-id-1 -> element-id-2.port-id-2.slot-id-2

Очень важно, чтобы левая часть соединения портов/слотов описывала именно идентификатор выходного порта/слота, а правая — входного. Это проверяется при открытии схемы.

Пример описания соединений:

```
 .actor-bindings {
   read-text.out-text->write-text.in-text
}
read-text.text->write-text.in-text.text
read-text.url->write-text.in-text.url
```

Последний блок — это метаданные (.meta), который описывает визуальное изображение схемы (позиции элементов на сцене, угол наклона стрелок соединения портов, цвет, шрифт и прочее) и короткие названия параметров. В UGENE разработан механизм запуска любой схемы из командной строки. Для того чтобы передавать какие-либо параметры схемы во время запуска, в метаданных схемы можно назначить каждому параметру уникальное короткое имя, которое и будет именем параметра. Также можно задать описание этого параметра. Например:

```
.meta {
   parameter-aliases {
      write-text.url-out {
         alias : out;
         help : «Путь до выходного файла»
      }
   }
   visual {
      # описание визуальной информации вида
      # ключ : значение;
   }
}
```

Если схема пользователя содержит этот фрагмент кода, и если он работает с интерфейсом командной строки UGENE для запуска выполнения схем, то он может регулировать путь до выходного файла во время запуска с помощью созданного параметра «out»:

```
./ugene --task=schema.uwl --out=/tmp/file.txt
```

Здесь --task=schema.uwl — это параметр, говорящий о том, что нужно запустить workflow-схему из файла schema.uwl.

Блок visual при создании схемы, используя текстовый синтаксис, обычно описывать не требуется, так как проще и удобнее сделать это, используя графический интерфейс UGENE Workflow Designer.

6. Создание пользовательских вычислительных элементов. Пользователю предоставлена возможность создания собственных вычислительных элементов, а не только использование стандартной библиотеки UGENE. Есть два способа добиться этого:

1) интеграция стороннего инструмента с интерфейсом командной строки;
2) написание скрипта на языке ECMAScript [2].

В UGENE встроен механизм подключения сторонних инструментов. Для того чтобы инструмент был совместим для работы в Workflow Designer, должны быть выполнены условия:

1. Инструмент имеет интерфейс командной строки.
2. Он оперирует входными и выходными файлами, форматы которых поддерживает UGENE.
3. Данные в файлах этих форматов имеют один из типов, которые поддерживает UGENE Workflow Designer.

Если требуемый инструмент удовлетворяет этим условиям, то пользователь может запустить специальный визард (мастер), в котором, пройдя шаг за шагом, он интегрирует инструмент со всей системой. Этот визард предложит настроить параметры, входные и выходные порты и слоты нового элемента. Последний шаг настройки — указание того, как запускать этот инструмент через интерфейс командной строки. Здесь можно использовать только что созданные имена параметров и слотов.

В результате работы визарда на упомянутой выше палитре появится новый полноценный вычислительный элемент, который можно использовать в любой схеме. Связь между сторонним инструментом и UGENE во время исполнения схемы происходит с помощью временных файлов.

Другой способ создания новых вычислительных элементов — это использование ECMAScript [2]. UGENE разрабатывается на языке C++ с использованием фреймворка Qt [7],

в который встроен интерпретатор языка ECMAScript. Поэтому не составило большого труда интегрировать в Workflow Designer способ написания элементов на ECMAScript. Элемент на ECMAScript — это всегда вычислитель, так как в этом языке нет поддержки ввода и вывода информации.

Пользователь может запустить диалог создания нового элемента, в котором нужно указать параметры и входные и выходные слоты портов вычислителя. Затем нужно создать скрипт нового элемента. Описанные параметры и слоты можно использовать как переменные внутри этого скрипта. Из переменных входных слотов можно считывать пришедшие в порт данные, а в переменные выходных слотов — помещать данные для отправления.

Пользователю предоставлен набор утилитных функций для работы с основными типами данных UGENE Workflow Designer.

7. Планы развития. В настоящее время разработчикам и руководителям проекта UGENE известны несколько направлений, в которых нужно развивать UWL и Workflow Designer в целом. Первое из них — это развитие интеграции с ECMAScript. Есть определенная цель — создать язык с полноценной стандартной библиотекой прототипов и утилит и платформу для вычислений в области биоинформатики. В настоящее время биоинформатикам и молекулярным биологам известны несколько языков, таких как BioPython или BioPerl, которые позволяют программировать на неком подобии DSL. Создание в UGENE нового языка на основе ECMAScript даст серьезное развитие этой области, так как помимо написания обычных скриптов вся система в целом предоставляет мощный механизм конвейеров вычислений.

В рамках развития интеграции с ECMAScript планируется создать объектную модель биоинформатики, то есть создать прототипы для всех типов данных Workflow Designer (Sequence, MSA и т.д.) и снабдить эти прототипы основными методами, актуальными для этих типов данных.

Другим направлением развития UWL и Workflow Designer в целом является разработка возможности отладки выполнения схем. В настоящее время уже ведется работа над возможностью временной остановки (паузы) и возобновления схемы, изменению параметров схемы «на лету», удобного просмотра содержимого шин данных во время паузы и прочих инструментов. Также должна быть возможность отладки и работы скриптов элементов.

Немаловажным и ресурсоемким направлением развития в области отладки схем является перезапуск вычислений с незначительными изменениями параметров. В этом случае должны повторяться заново не все вычисления, а лишь те, которых касаются совершенные изменения параметров.

Третье направление развития относится к распределенным вычислениям. В UGENE Workflow Designer уже есть некоторые наработки по удаленному запуску схем и мониторингу их исполнения. Планируется развить эту инфраструктуру с целью организации не просто облачных, а распределенных вычислений, то есть использовать несколько компьютеров в процессе выполнения схемы. Первый шаг к этому уже совершается — перевод всех типов данных Worflow Designer для работы с интерфейсом базы данных. Когда вся система будет работать через единый интерфейс, то нетрудно будет изменить его реализацию для работы с какой-либо другой сетевой СУБД, какой как MySQL или PostgreSQL, необходимой для распределенных вычислений.

8. Заключение и выводы. UGENE Workflow Designer является вычислительной платформой, которая может быть расширена в различных направлениях: как в наборе вычислительных элементов, так и в функционально-семантическом плане. Последнее достигается именно за счет поддержки внутреннего языка программирования UWL. Язык программирования всегда имеет какие-либо особенности, договоренности, законы, и следуя им, получается создавать гибкую, целостную, легко поддерживаемую систему.

UGENE предназначен для решения задач биоинформатики. Но Workflow Designer может быть довольно просто переделан для решения задач из некоторых других областей. Язык UWL никак не связан с биоинформатикой, кроме как типами данных и стандартной библиотекой вычислительных элементов.

Система распространяется под свободной лицензией, потому если есть необходимость использовать ее в новой области знаний, то это вполне достижимая цель. Для этого необходимо:

1. Выяснить, какими данными оперирует эта область знаний, и выявить необходимые новые типы.
2. Интегрировать новые типы данных и их вычислительные элементы: читатели и писатели нужных форматов файлов.
3. Доработать служебные элементы с учетом новых типов.
4. Разработать набор вычислителей, необходимых в этой области знаний. Для экономии ресурсов приветствуется использование уже готовых сторонних инструментов и механизм их интеграции с Workflow Designer.
5. При необходимости использования ECMAScript нужно разработать утилитные функции и/или прототипы для новых типов данных.

Все перечисленные шаги имеют подготовленную основу в UGENE, поэтому действия, необходимые для совершения каждого шага, в некоторой степени минимизированы.

Список литературы

1. Документация о UGENE Workflow Designer // UGENE v.1.11.4 documentation. 2012. [Электронный ресурс]. URL: http://ugene.unipro.ru/documentation/wd_manual/index.html (дата обращения: 14.01.2013).
2. Спецификация языка ECMAScript-262 // ECMAScript Language Specification. 2011. [Электронный ресурс]. URL: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (дата обращения: 14.01.2013).
3. Haldar S. Inside sqlite. O'Reilly, 2007. 76 p.: ISBN: 9780596550066.
4. Johnston W.M.,Hanna J.R.P., Millar R.J. Advances in dataflow programming languages. // ACM Computing Surveys (CSUR). 2004. №36 (1). P. 1-34.
5. Kleppe A. Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison-Wesley Professional, 2008. 207 p.: ISBN:0321553454 9780321553454.
6. Okonechnikov K., Golosova, O., Fursov, M., the UGENE team. Unipro UGENE: a unified bioinformatics toolkit // Bioinformatics. 2012. №28 (8). P. 1166-1167.
7. Qt // Qt [Электронный ресурс]. URL: http://qt.digia.com/ (дата обращения: 14.01.2013).

UDK: 004.434

Title: Programming language approach to organization of computations in bioinformatics

Author(s):

German Andreevich Grekhov (LLC "NCIT Unipro"),

Igor Nikolaevich Skopin (Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk State University)

Abstract: This paper presents the approach to creation of computational platforms in the basics of UGENE Workflow Designer. This platform is based on the internal programming language. It demonstrates a method of keeping the system consistent and handy to maintain and develop if you follow the concepts of the platform internal model and the concepts of the programming language syntax.

UGENE Workflow Designer is designed to solve the bioinformatics tasks; and the UWL language is the DSL of bioinformatics. The paper describes the feature of how to reuse the basics of the platform in some other application areas.

Keywords: programming languages, DSL, computational pipelines, workflow, UGENE, bioinformatics