Реализация службы управления сценариями в распределенной вычислительной среде

О. В. Сухорослов, И. В. Лазарев

1. Введение

Система распределенных вычислений IARnet [1] ориентирована на интеграцию высокоуровневых ресурсов в рамках современной концепции Grid-вычислений. В основе системы лежит понятие *информационно-алгоритмического ресурса* (*ИАР*), который представляет собой абстрактный ресурс с четко описанной функциональностью и программным интерфейсом удаленного доступа к этой функциональности, скрывающим за собой один или несколько первичных ресурсов. Абстракция ИАР позволяет описать общую модель доступа к ресурсам распределенной вычислительной среды (РВС) и использовать РВС для решения широкого класса прикладных задач.

В частности, концепция типизированных ИАР позволяет осуществлять *композицию ресурсов* РВС с целью создания новых, более сложных составных ресурсов. Подобная композиция может быть проведена путем описания набора уже имеющихся ресурсов заданных типов и схемы их взаимодействия. Новый ресурс-агрегат может динамически подключать к себе необходимые ресурсы во время его инициализации, причем каждый раз это могут быть различные экземпляры ресурсов одного типа.

Это открывает возможность для переноса в РВС широкого класса задач, для решения которых требуется набор типовых ресурсов с заданной функциональностью, взаимодействующих по установленному сценарию. Подобные сценарии координированного использования множества ИАР для решения определенной задачи мы будем называть *прикладными вычислительными сценариями* или просто *сценариями*.

Более точно прикладной вычислительный сценарий можно определить как формальное представление процесса решения некоторой задачи в РВС, включающее в себя:

- описание элементарных *подзадач*, из которых состоит решение исходной задачи;
- описание *ресурсов*, которые решают указанные подзадачи;
- описание зависимостей между подзадачами, а именно — *потоков управления*, которые определяют последовательность выполнения подзадач и синхронизацию между ними, и *потоков данных*, которые определяют передачу информации между подзадачами;
- описание *событий*, которые могут влиять на ход выполнения процесса, и правил их обработки.

Приведенное определение соответствует понятию *сценария* (*workflow*), рассматриваемого в рамках так называемой workflow-методологии. Достаточно заменить слова «процесс решения задачи в РВС» на «производственный процесс» или «бизнес-процесс», «подзадача» — на «операция», а «ресурс» — на «исполнитель». Workflow-методология направлена на «автоматизацию, полностью или частично, бизнес-процесса, при которой документы, информация или задания передаются для выполнения необходимых действий от одного участника к другому в соответствии с набором процедурных правил» [5]. Данное определение, несмотря на привязку к бизнес-процессам, хорошо отражает суть workflow-методологии как некоторого подхода к автоматизации, вообще говоря, различных процессов.

В рамках workflow-методологии также вводится понятие системы управления сценариями. *Системой управления сценариями* (workflow management system) называется система, позволяющая создавать сценарии, запускать и управлять их выполнением. Система управления сценариями состоит из набора программных компонентов, предназначенных для хранения и интерпретации описаний процессов (сценариев), создания и управления экземплярами запущенных процессов, а также организации их взаимодействия с участниками процесса и внешними приложениями. Программное приложение, непосредственно выполняющее интерпретацию и запуск сценария, а также управляющее экземплярами запущенных процессов, называется *средой выполнения сценариев*.

В [4] показано, что workflow-методология может быть успешно применена не только в сфере бизнес-процессов, но и при организации сложных вычислений, охватывающих набор распределенных ресурсов. Приход workflow-методологии в распределенные вычисления способен вывести их на качественно новый уровень так же, как это произошло при эволюции программных приложений. Однако при этом необходимо учитывать специфику распределенных вычислительных процессов. Приведем основные отличия распределенных вычислительных процессов от бизнес-процессов, на которые ориентируются традиционные системы управления сценариями:

- Использование сложных вычислительных ресурсов.
- Количество ресурсов, которые потребуются для решения вычислительной задачи, может быть неизвестно априори, так как для некоторых классов задач трудно оценить предстоящий объем вычислений.
- Необходимость работы в динамичной распределенной среде, в которой ресурсы не известны априори и могут быть подвержены отказам.
- Работа с большими объемами данных.
- Необходимость выполнять большое количество идентичных заданий с переменными параметрами.
- Для многих вычислительных задач характерны иерархии подзадач (подсценариев), создаваемых и уничтожаемых по необходимости.

Указанные особенности распределенных вычислительных процессов определяют требования, которым должна удовлетворять система управле-

ния сценариями, подходящая для описания и выполнения данных процессов в виде сценариев. Традиционные системы управления сценариями, рассчитанные на работу с бизнес-процессами, в подавляющем большинстве не подходят для решения научных вычислительных задач. Поэтому, требуется разработка новых систем управления сценариями, с одной стороны опирающихся на сформировавшуюся workflow-методологию, а с другой стороны — специально рассчитанных на требования вычислительных задач.

Задачу реализации сценариев в распределенной вычислительной среде можно разбить на две главных подзадачи:

- Создание или адаптация существующих языков и средств описания сценариев.
- Разработка аналога системы управления сценариями — компонента РВС, реализующего развертывание и выполнение сценариев в среде.

В настоящей статье рассматриваются общие принципы реализации службы управления сценариями (СУС) в системе IARnet, производится выбор формализма для описания сценариев и программных средств для реализации СУС, а также описывается создание действующего прототипа СУС и его тестирование на примере решения прикладной задачи.

2. Принципы реализации службы управления сценариями в системе IARnet

В рамках работ по созданию системы IARnet планируется использовать методологию сценариев для описания распределенных вычислительных сценариев и разработки *службы управления сценариями*, позволяющей осуществлять выполнение подобных сценариев. Рассмотрим основные принципы реализации данной службы безотносительно того, на каком формальном языке задаются сами сценарии.

2.1. Развертывание сценария

Как было отмечено выше, распределенный вычислительный сценарий может быть представлен благодаря композиции ресурсов в виде нового информационно-алгоритмического ресурса. Подобный подход позволяет сохранить общность архитектуры IARnet и использовать стандартные механизмы доступа к ресурсам для работы с прикладными сценариями из любой точки среды. Это может быть особенно полезно в случае, если создатель сценария хочет сделать его доступным на постоянной основе для пользователей среды. Кроме того, в случае если выполняющийся сценарий представлен в среде в виде ресурса, участвующие в сценарии ресурсы могут использовать стандартные механизмы для взаимодействия со сценарием.

Под развертыванием сценария в СУС будем понимать процесс развертывания в среде ИАР, который представляет данный сценарий. При развертывании сценария службе должно быть передано описание сценария. Служба может провести проверку описания сценария на корректность

и реализуемость, после чего она должна обеспечить развертывание соответствующего агента доступа, который мы будем называть агентом сценария. В обязанности агента сценария входит обслуживание запросов на запуск сценария и передача запросов запущенным экземплярам сценария. Данные вопросы рассматриваются далее.

Обратной развертыванию операцией является удаление сценария из службы, при котором происходит остановка всех запущенных экземпляров сценария и удаление агента сценария, а также — всех связанных со сценарием артефактов.

2.2. Запуск сценария на выполнение

При запуске сценария, развернутого в СУС, происходит создание нового экземпляра сценария (также представляемого в виде ИАР), с которым связано состояние процесса выполнения сценария. Агенту сценария должны быть переданы все начальные (входные) данные, необходимые для запуска сценария. Выполнение экземпляра сценария осуществляется в рамках определенной среды выполнения сценариев, которая осуществляет интерпретацию описания сценария, производит указанные в сценарии действия, следит за состоянием выполнения сценария и обрабатывает поступающие от участников сценария вызовы. Среда выполнения, а следовательно, и служба, может поддерживать одновременный запуск сразу нескольких экземпляров сценария.

Для осуществления внешних взаимодействий с ресурсами и службами среды, требуемых для выполнения входящих в сценарий действий, среда выполнения сценариев использует прикладной API системы IARnet.

В свою очередь, участники сценария используют для обратного вызова экземпляра сценария ИАР данного экземпляра, который передает приходящие вызовы среде выполнения сценариев. Обратные вызовы необходимы для того, чтобы участвующие в сценарии ресурсы могли уведомить экземпляр сценария о наступлении определенных событий.

2.3. Получение результатов и контроль состояния выполнения сценария

В случае если для выполнения сценария не требуется длительного времени, результат его выполнения может быть получен непосредственно от ИАР экземпляра сценария как ответ на вызов операции запуска сценария.

Однако ожидается, что некоторые сценарии могут потребовать для своего выполнения нескольких часов, дней и т. д. В подобных случаях, при запуске сценария на выполнение ИАР экземпляра сценария должен возвращать клиенту уникальный идентификатор экземпляра сценария, по которому в дальнейшем клиент может запрашивать информацию о состоянии выполнения сценария и его результат. В этом случае на агента сценария возлагаются дополнительные обязанности по обработке данных запросов клиента.

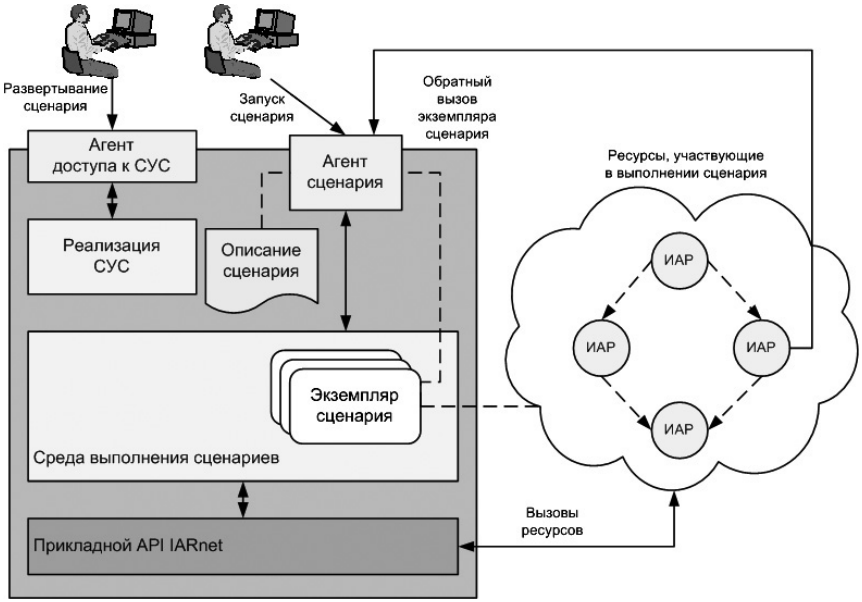

Рис. 1. Архитектура службы управления сценариями IARnet

2.4. Архитектура службы управления заданиями

На рис. 1 изображена архитектура службы управления сценариями (СУС) IARnet и ее взаимодействие с пользователями и ресурсами среды. Основными элементами СУС являются:

- Агент доступа к СУС.
- Реализация СУС.
- Агенты сценариев.
- Среда выполнения сценариев.
- Прикладной API IARnet.

3. Выбор формализма для описания распределенных вычислительных сценариев

Поскольку предполагается, что составлением сценариев будут заниматься непосредственно пользователи среды, то к средствам описания сценариев предъявляются особые требования. В первую очередь, данные средства должны быть простыми в изучении и поддерживать наглядное представление сценария. Кроме того, данные средства должны обладать достаточной выразительностью, т. е. позволять описывать все основные элементы сценария и управляющие конструкции.

В рамках workflow-методологии существует несколько подходов к описанию сценариев, рассмотренных подробно в [4], а именно:

- Использование скриптовых языков.
- Использование графов:
  - Ориентированные ациклические графы.
  - Сети Петри.
- Смешанные подходы, сочетающие в себе элементы скриптовых языков и графового представления.

Из проведенного в [4] анализа был сделан вывод о том, что наиболее подходящим способом представления сценариев в IARnet являются сети Петри, как подход, основанный на строгом математическом формализме, поддерживающий представление состояния процесса и нейтральный по отношению к технологиям распределенных вычислений.

Поскольку в настоящее время существует множество разновидностей сетей Петри, встает вопрос о выборе того или иного конкретного формализма. Возникшие первыми простые сети Петри, подробное изложение теории которых можно найти, например, в [2], оперируют неотличимыми друг от друга черными фишками, которые не несут в себе никакой информации. Данный формализм не является достаточно выразительным для использования в IARnet, поскольку не позволяет моделировать передаваемые между участниками сценария сложно структурированные данные при помощи фишек.

Этого недостатка лишены так называемые *цветные сети Петри* (Coloured Petri Nets, CPN) [8], являющиеся расширением классических сетей Петри (цветные сети Петри вместе с предикатными сетями относят к высокоуровневым сетям Петри). Перечислим характерные особенности цветных сетей Петри:

- Фишки вместо простого обозначения содержимого места преобразуются в объект, который может содержать в себе один или более параметров, каждый из которых может принимать дискретный набор значений. В соответствии с этим фишки различаются по типам параметров (переменных). Чтобы отличать фишки различных типов их можно окрашивать в различные цвета.
- К местам добавляется информация о типах фишек, которые могут находиться в данном месте.
- К дугам, исходящим из мест, добавляется информация о типах фишек, которые могут участвовать в возбуждении переходов, инцидентных этим дугам.
- К переходам может быть добавлено условие возбуждения перехода, в зависимости от значений переменных, содержащихся во входных фишках.
- К дугам, исходящим из переходов, добавляется информация о типах фишек, исходящих из перехода, и о преобразовании переменных.
- К начальной маркировке сети добавляется информация о значениях переменных, содержащихся в фишках.

Таким образом, в цветных сетях фишки становятся носителями сложной структурированной информации, что позволяет использовать такие сети для описания более сложных, в том числе вычислительных, процессов.

Для описания распределенных вычислительных сценариев в системе IARnet больше всего подошла бы такая объектная модификация цветных сетей, в которой фишками могли бы быть произвольные объекты языка Java. Это позволило бы обеспечить лучшую интероперабельность между текущей реализацией IARnet и СУС.

4. Выбор ПО для работы с сетями Петри

При реализации СУС IARnet было решено не создавать программное средство для работы с сетями Петри с нуля, а использовать как основу для создания СУС одну из систем, доступных в открытых исходных кодах в Internet. К подобным системам были предъявлены следующие требования:

- Наличие в составе пакета редактора сетей, интерпретатора (среды выполнения) и средств отладки.
- Реализация на языке Java.
- Открытость и расширяемость, свободная доступность исходных кодов.
- Поддержка цветных (высокоуровневых) сетей Петри. Желательно, чтобы в качестве фишек использовались произвольные Java-объекты.
- Возможность вызова внешних компонентов при срабатывании переходов сети.
- Возможность обратных вызовов сценария внешними компонентами.
- Поддержка многопоточности (одновременного выполнения переходов).
- Легкость освоения. Наличие подробной документации пользователя, и документации для разработчика.
- Зрелость продукта. Наличие поддержки и регулярных обновлений.

На сайте Petri Nets Tool Database [10], где представлена наиболее полная в Интернете база данных по программному обеспечению, так или иначе связанному с сетями Петри, предусмотрен довольно удобный и гибкий механизм поиска по этой базе данных. Задав на этом сайте в форме поиска главные из требований, предъявленных выше (поддержка высокоуровневых сетей Петри, реализация на Java, наличие в пакете редактора сетей и свободная доступность пакета), был произведен предварительный отбор возможных кандидатов, список которых после отбора свелся к пяти пакетам:

- Income Suite.
- jFern.
- Petri Net Kernel.
- Renew.
- YAWL.

Эти пакеты подробно рассмотрены далее.

4.1. Income Suite

Пакет Income Suite [6], разработанный немецкой фирмой Get Process, был сразу исключен из рассмотрения, несмотря на то, что это достаточно зрелое и развитое средство. Дело в том, что хотя его разработчики и предоставляют свой продукт бесплатно для академических организаций, исходных кодов этой системы нет в свободном доступе, т. е. невозможно вносить в нее необходимые изменения, а значит и использовать ее в качестве одной из составных частей системы IARnet.

4.2. jFern

Rakiura jFern [7] — компактная среда для работы с сетями Петри, созданная специально для Java-разработчиков. По словам авторов это «средство моделирования сетей Петри, полностью интегрированное с языком Java». В последнюю версию продукта входят инструменты для создания и редактирования сетей в наглядном графическом отображении.

Основные характеристики jFern (версия 3.0.0):

- Небольшой объем (ядро продукта без учета графического интерфейса состоит из 41 класса и занимает объем 130 килобайт).
- Возможность включать в описание сети произвольный Java-код при задании условий срабатывания переходов и наборов фишек.
- Эффективное представление сетей Петри, позволяющее с большим быстродействием запускать сети в моделирующей программе (как утверждают разработчики — настолько, насколько это позволяет Java).
- Шесть встроенных шаблонов для разработки сетей.

Графический вид и маркировка сети в jFern не привязаны к ее описанию, что позволяет легко запускать сеть с разными начальными маркировками и лучше справляться с промежуточными состояниями сети, а также предоставляет лучшие возможности для исправления ошибок.

В систему интегрирован графический пакет для редактирования, графического отображения моделирования сетей и их отладки.

Фишка в jFern может быть любым Java-объектом (java.lang.Object), а значит типы данных, которые она может «нести в себе», могут быть практически любыми. Как следствие, в jFern были введены «бесцветные» места, которые могут содержать фишки с произвольными Java-объектами в качестве значений. Это означает, что разработчик сети может, при необходимости, помещать различные логические типы фишек на одно и то же место.

Условия срабатывания переходов в jFern реализованы с помощью т. н. guards — небольших блоков Java-кода, которые приписываются дугам, предшествующим переходам и возвращают булево значение. Действия (actions), которые совершаются сетью при срабатывании какого-либо перехода, также задаются в виде Java-кода, приписанного переходу.

Таким образом, сеть Петри, созданная с помощью jFern, представляет собой полноправный Java-объект, что может помочь обеспечить интероперабельность при взаимодействии действующего сценария с другими ресурсами IARnet.

Однако в настоящей реализации системы jFern графические средства еще недостаточно развиты, существующие средства отладки слишком просты, а многопоточные сети не поддерживаются напрямую. Кроме того, на данный момент пакет поддерживается нерегулярно, так как разрабатывается одним человеком. Это означает, что в нынешнем виде jFern не отвечает предъявленным требованиям и требует внесения существенных изменений.

Основные достоинства jFern:

- Система создана специально для Java-разработчиков.
- Исходные коды находятся в свободном доступе.
- Цветные сети поддерживаются.
- В качестве фишек могут использоваться Java-объекты.
- При срабатывании переходов возможен вызов любого Java-кода.
- Компактность реализации.

Основные недостатки:

- Несовершенные графические средства.
- Слабо развитые средства отладки.
- Многопоточность не поддерживается.
- Продукт поддерживается нерегулярно (дата последнего обновления: 6 мая 2002 г.).

4.3. Petri Net Kernel

Petri Net Kernel [9] — система, созданная в Университете Гумбольдта, г. Берлин. Она предоставляет инструментарий для разработки программ, основанных на сетях Петри. По словам авторов, основная идея проекта заключалась в том, чтобы сделать структуру данных сетей Петри доступной для разработчиков ПО. Эта система предназначена для создания, анализа, моделирования или тестирования сетей Петри, и для интегрирования готовых сетей в программные средства. Программист может произвольно сочетать различные компоненты PNK и совмещать их в одном программном средстве. Существуют версии PNK на языках программирования Java и Python. В настоящее время обновления пакета выпускаются только на Java.

В PNK нет привязок к тому или иному классу сетей Петри. Поддерживаются как простые, так и цветные сети Петри. Кроме того, типы сетей могут задаваться самим программистом. В качестве фишек могут использоваться переменные Java. Готовые сети хранятся в общепринятом стандартном формате PNML, основанном на XML.

Входящая в пакет моделирующая программа поддерживает многопоточность. Программист может вставлять в текст описания сети произвольный Java-код, запускаемый при срабатывании переходов. Однако из графического редактора это сделать невозможно.

Несмотря на все упомянутые достоинства, система достаточно трудна в освоении. По сути PNK представляет собой конструктор из отдельных классов — элементов сетей Петри, что делает систему неподходящей для неподготовленных пользователей.

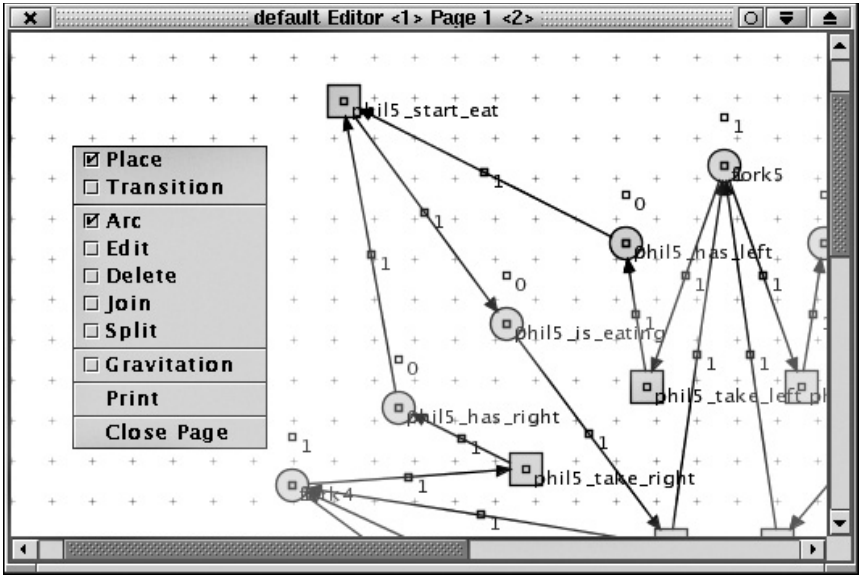

Рис. 2. Интерфейс Petri Net Kernel

Кроме того, в пакет PNK входит очень простой и несовершенный графический редактор сетей. Впрочем, разработчики предусмотрели возможность замены этого редактора на другой, при условии что новый будет согласован с универсальным интерфейсом PNK.

В целом, пакет представляет собой достаточно развитое и функциональное средство для работы с сетями Петри. Однако повышенная сложность освоения пакета, несовершенство встроенного графического редактора и ориентированность в основном на создание новых программных средств (а не на создание и моделирование сетей) делают PNK не вполне подходящим выбором. Таким образом, использование пакета в системе IARnet возможно лишь после внесения в него существенных изменений.

Основные достоинства PNK:

- Язык реализации — Java.
- Исходные коды находятся в свободном доступе.
- Цветные сети поддерживаются.
- В качестве фишек могут использоваться Java-объекты.

Основные недостатки:

- Система трудна в освоении.
- Несовершенство встроенного редактора сетей.
- Ориентированность на создание программных средств со структурой данных сетей Петри, а не самих сетей.
- Дата последнего обновления: 4 апреля 2002 г.

4.4. Renew

Разработанная в университете города Гамбурга система Renew [11] представляет собой редактор и моделирующую программу для объектной модификации цветных сетей Петри, названной разработчиками Reference Nets. В пакте достигнуто бесшовное соединение формализма сетей Петри и языка Java. В Renew сеть Петри может реализовывать методы Java, и восприниматься извне как набор обычных Java-объектов. С другой стороны, обычный Java-код может быть легко использован при задании сетей. Например, в описании сети возможны вызовы любого Java-класса, что позволяет нам широко использовать возможности этого языка при описании сценариев.

Моделирующая часть системы Renew поддерживает двунаправленные синхронные каналы (это означает, что допускается одновременное срабатывание переходов в разных ветвях сетей) и позволяет запускать многочисленные экземпляры одной сети. Renew хорошо приспособлена для быстрого создания прототипов и исполняемых моделей процессов.

Как уже было упомянуто, используемый в Renew тип сетей — это т. н. Reference Nets, класс сетей Петри, введенный самими разработчиками и представляющий собой цветные сети с объектными расширениями. Помимо обычных дуг, здесь введены также:

- Тестовые дуги, изображаемые без стрелок, которые позволяют переходам срабатывать, оставляя фишки на своих местах. Эти дуги удобны, например, при создании «глобальных» переменных в сети — выделяя под такие переменные одно или несколько мест и соединяя их тестовыми дугами с теми местами, где используются «глобальные» переменные, мы добиваемся существенного упрощения вида сети.
- Резервные дуги, изображаемые со стрелками с обеих сторон. Такие дуги представляют собой входную и выходную дуги перехода, соединенные воедино. При срабатывании перехода, к которому ведет такая дуга, фишка снова возвращается на входное (и, в данном случае, выходное) место перехода.
- Гибкие дуги, изображаемые со сдвоенными стрелками. Такие дуги предназначены для работы с массивами. Гибкие дуги могут быть эффективно использованы при моделировании распределенных алгоритмов [12].
- А также несколько добавочных типов дуг, предназначенных только для работы в последовательном режиме Renew (без поддержки многопоточности).

Кроме того, были введены виртуальные места, позволяющие делать «ярлыки» для того или иного места, т. е. создавать виртуальную копию какого-либо уже существующего места в другой части сети, что часто заметно упрощает внешний вид сети.

Как и в пакете jFern, условия срабатывания и действия, выполняемые при срабатывании того или иного перехода, в Renew реализованы с помощью задаваемых на языке Java выражений (т. н. guards и actions), только в данном случае и те и другие приписаны самому переходу.

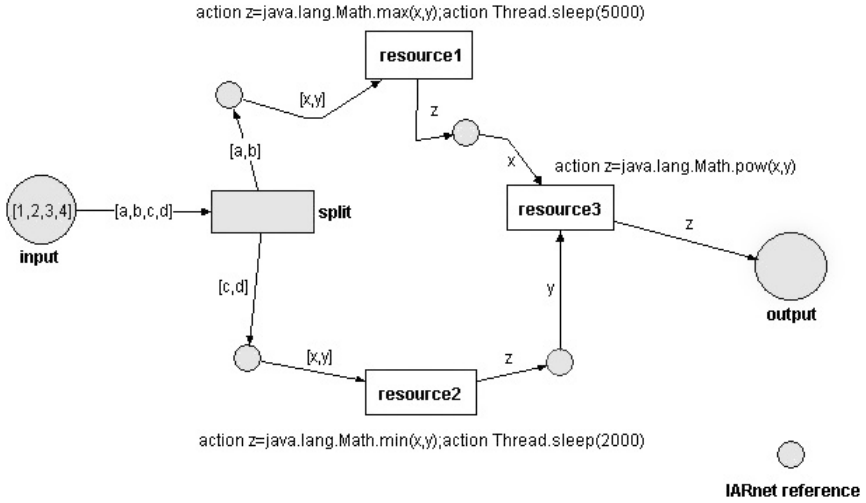

Рис. 3. Пример простой сети Петри, смоделированной в Renew

Также возможно введение в сеть понятия времени. Так, та или иная фишка может стать доступной, или какой-либо переход может начать срабатывать только с некоторого момента времени.

В Renew уже включены средства удаленного доступа к работающим экземплярам сетей через графический интерфейс, что позволяет в удаленном режиме отслеживать состояние и контролировать процесс выполнения сети.

Система также включает в себя (на данный момент — в тестовом варианте) средства для работы на многопроцессорных системах.

Благодаря системе дополнительных модулей (плагинов, plug-ins), в Renew достаточно легко интегрируются новые функции и компоненты, что позволяет применять Renew даже в тех областях, в которых это изначально не планировалось разработчиками.

Система обладает достаточно развитыми средствами отладки, такими как контрольные точки, интерпретация «шаг за шагом», срабатывание переходов «по клику» в графическом интерфейсе и т. п.

Renew сохраняет сети в собственном формате RNW, однако предусмотрена возможность сохранения как в стандартном для сетей Петри формате PNML, так и в графическом формате PostScript.

Основные достоинства Renew:

- Язык реализации — Java.
- Исходные коды — в свободном доступе.
- Цветные сети поддерживаются.
- Поддержка многопоточности.
- В качестве фишек могут использоваться Java-объекты.

- При срабатывании переходов возможен вызов произвольного Java-кода.
- Удобный графический редактор, позволяющий легко создавать сети и при необходимости задавать дополнительные параметры в виде Java-кода.
- Есть встроенные средства удаленного доступа.
- Есть средства отладки.
- Достаточно регулярные обновления на протяжении нескольких лет (дата последнего обновления: 5 мая 2006 г.).

Основные недостатки:

- Существующая реализация системы включает в себя только простые средства анализа сетей, такие как проверка сети на корректность. Предполагается, что в основном анализ сетей будет осуществляться динамически, при их интерпретации.
- Renew — система, разработанная в академических кругах, и поэтому поддерживается настолько, насколько это позволяет занятость разработчиков в других проектах.

4.5. YAWL

Пакет YAWL [13] (Yet Another Workflow Language) был разработан в Квинслендском Технологическом Университете Австралии. Он представляет собой набор средств для работы с описаниями сценариев, созданными на собственном одноименном языке YAWL.

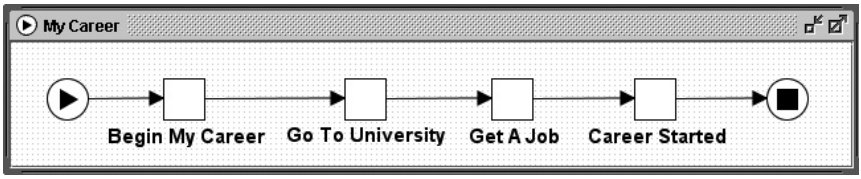

Рис. 4. Интерфейс и пример описания сценария в YAWL

Произведя подробный анализ всех существующих языков описания сценариев, разработчики пакета предложили свой собственный язык, взяв сети Петри за основу. Он сочетает в себе элементы как сетей Петри, так и других языков описания сценариев, при этом подчиняясь определенным формальным законам. Но в строгом смысле этот язык уже не принадлежит к сетям Петри (к примеру, как видно на рис. 4, описания сценариев уже не содержат мест и переходов), а следовательно, к нему не применимы стандартные методы анализа сетей Петри и он не входит в область рассмотрения данной статьи.

4.6. Выводы

Таким образом, из предварительно отобранных пакетов по разным причинам только три могут быть использованы для создания СУС IARnet —

это jFern, PNK и Renew. Однако именно Renew является наиболее зрелым продуктом, с одной стороны достаточно удобным и простым в использовании, и в то же время обладающим большими возможностями для создания и интерпретации вычислительных сценариев. Стоит отметить, что этот пакет на настоящий момент поддерживается наиболее активно. Renew также требует наименьших внесенных изменений, чтобы быть полностью пригодным для поставленных целей. Кроме того, благодаря системе дополнительных модулей Renew, внести такие изменения будет проще всего.

5. Реализация прототипа СУС IARnet

На рис. 5 изображена схема практической реализации прототипа СУС IARnet. При ее создании были использованы как уже готовые встроенные средства Renew, так и компоненты, написанные с нуля (на рисунке они выделены цветом). К числу первых относятся:

- Графический интерфейс Renew (Renew GUI на рисунке), через который происходит взаимодействие пользователя с СУС.
- Интерпретирующие компоненты Renew (Renew Simulation Server), посредством которых осуществляется интерпретация сценария.

К числу вторых принадлежат:

- Дополнительный модуль Renew WfMS Plugin, добавленный в графический интерфейс Renew.
- Реализация СУС IARnet Renew-WfMS — часть IARnet, предназначенная для взаимодействия с Renew и реализованная в виде ИАР.
- ИАР сценария, прикрепленный к Renew Simulation Server для того, чтобы сценарий мог принимать и отправлять вызовы через IARnet.

Жизненный цикл сценария здесь выглядит так: сначала пользователь открывает описание сценария (Сеть на рисунке) в графическом интерфейсе Renew. Затем, посредством Renew Wfms Plugin, встроенного в этот интерфейс, он

(0) соединяется по сети (connect) с удаленной службой IARnet Renew WfMS и

(1.0) отправляет ей описание сценария для дальнейшего развертывания (deploy).

После этого служба IARnet Renew WfMS

(1.1) создает на серверной стороне новый процесс,

(1.2) запускает сервер-интерпретатор (Renew Simulation Server),

(1.3) создает прикрепленный к нему ИАР (ИАР сценария),

(1.4) передает интерпретатору экземпляр сети,

(1.5) отправляет на пользовательскую сторону ссылку на ИАР сценария, которая затем

(1.6) помещается в специальное место экземпляра сети с именем “IARnet reference”.

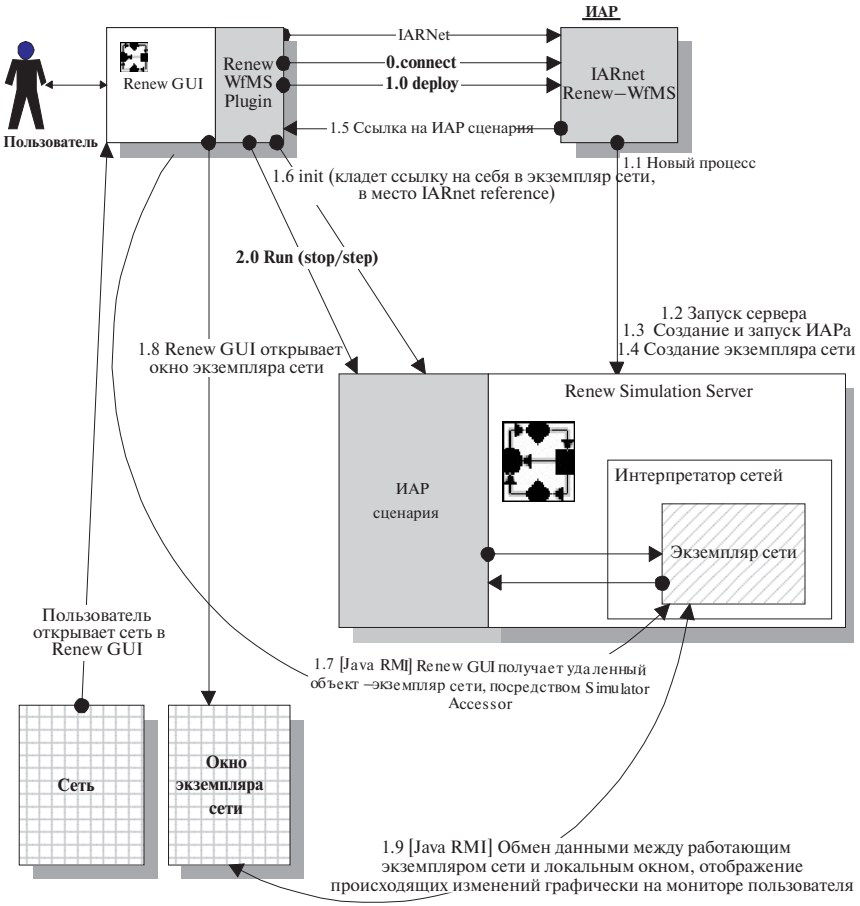

Рис. 5. Взаимодействие модуля Renew WfMS Plugin, службы IARnet Renew WfMS и экземпляра сценария

Теперь сценарий может передать эту ссылку ресурсам-участникам и далее принимать от них обратные вызовы через IARnet.

Во время выполнения сценария пользователь может наблюдать за ходом процесса через графический интерфейс Renew GUI. Это реализовано с помощью встроенных средств Renew следующим образом:

(1.7) После развертывания сценария Renew GUI получает от СУС ссылку на удаленный объект — экземпляр сети Renew, созданный сервером-интерпретатором и доступный через Java RMI.

(1.8) Renew GUI открывает на пользовательской стороне отдельное окно экземпляра сети.

(1.9) Все дальнейшие обновления содержимого этого окна происходят автоматически через Java RMI.

Итак, к этому моменту все подготовительные действия завершены, и пользователь может передать ИАР сценария

(2.0) команду Run на запуск сценария. После этого аналогичным образом сценарий может быть остановлен командой Stop, или может быть осуществлено пошаговое выполнение сценария, по команде Step. После завершения работы пользователь останавливает удаленный процесс с помощью команды Terminate.

На рис. 6 показано взаимодействие работающего сценария и ресурсов-участников. Как видно из рисунка, это взаимодействие полностью происходит средствами прикладного API IARnet. Остановимся подробнее на обратных вызовах работающего сценария ресурсами. Эти вызовы могут выполнять различные роли — от передачи сценарию свежих данных до сообщения о том, что ресурс закончил свою работу. Такой вызов в настоящей реализации СУС состоит из двух параметров: имени определенного

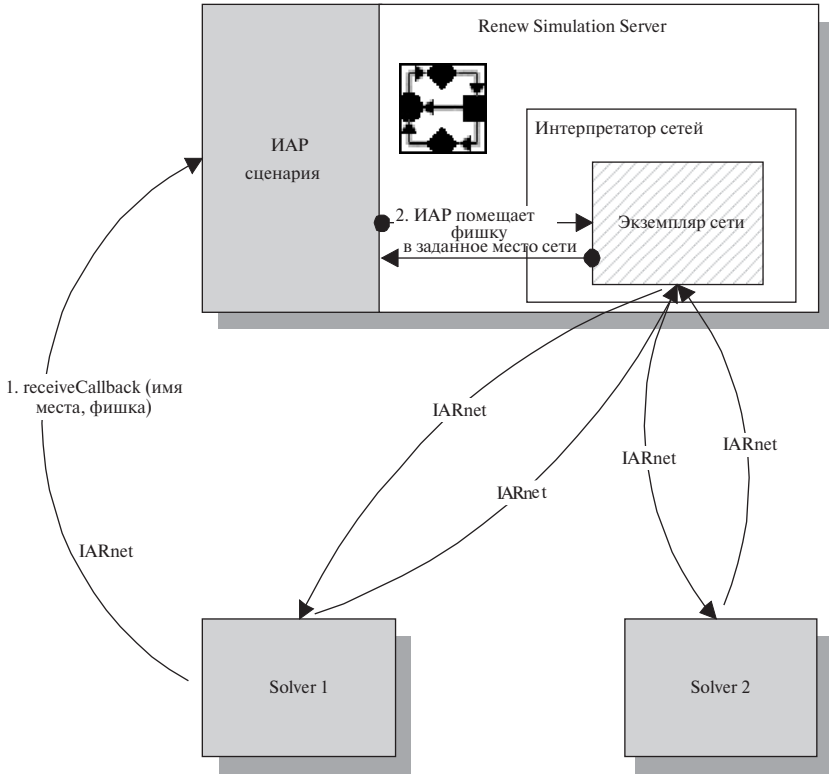

Рис. 6. Взаимодействие работающего сценария и ресурсов-участников

места сети и содержимого фишки. Таким образом, ресурс, осуществляя обратный вызов сценария, передает его ИАР необходимые данные в виде содержимого фишки и сообщает, в какое место сети их следует положить. ИАР сценария, в свою очередь, принимает эти данные и дает команду интерпретатору поместить их в качестве фишки в указанное место сети.

6. Тестирование прототипа СУС IARnet

6.1. Тестовая задача и алгоритм решения

В качестве тестовой задачи был взят распределенный алгоритм решения системы дифференциальных уравнений, описанный в [3]. Рассмотрим систему двух векторных обыкновенных дифференциальных уравнений относительно векторных переменных $x_1 \in \mathbb{R}^{n_1}$ и $x_2 \in \mathbb{R}^{n_2}$:

$$\dot{x}_1 = f_1(x_1, x_2); \quad \dot{x}_2 = f_2(x_1, x_2);$$
$$t \in [0, T], \quad x_1(0) = X_1, \quad x_2(0) = X_2.$$

Пусть интегрирование каждого из этих уравнений производится отдельным ресурсом (типичный пример ИАР), который представляет собой «обычную» (не распределенную) реализацию какого-нибудь метода решения задачи Коши (например, метода Эйлера). Для краткости, в дальнейшем будем обозначать эти ресурсы R1 и R2.

Перед началом вычислений эти два ресурса обмениваются «прогнозными» (экстраполирующими) траекториями «своих» переменных, аппроксимирующими неизвестную пока траекторию $x(t) = (x_1(t), x_2(t))$. Пусть прогнозом изменения первой переменной является функция $p_1(t)$, а второй — $p_2(t)$. Располагая этими функциями, каждый из вычислительных ресурсов независимо от другого может решить приближенное уравнение со «старыми» начальными условиями, отличающееся от точного тем, что в правой части вместо точных значений «чужих» переменных стоят их прогнозные значения. Результатом будут две аппроксимирующие траектории $\xi_1(t)$ и $\xi_2(t)$.

Ясно, что при этом возникнет ошибка, которая будет возрастать с ростом времени t. Каждый ресурс следит за отклонением своей компоненты аппроксимирующей траектории от прогноза, который был отослан другому ресурсу. Для определенности, пусть это отклонение превысило пороговое значение у ресурса R1. И пусть это произошло в момент времени t_1. Тогда ресурс R1 должен:

- оповестить ресурс R2, что вычисления следует приостановить, пока не будут получены обновленные значения прогноза;
- уточнить прогноз своих переменных $p_1(t)$ на промежутке $[t_1, T]$;
- послать ресурсу R2 новый прогноз.

Получив обновленные данные, ресурс R2 сформирует и начнет решать на промежутке $[t_1, T]$ новую задачу Коши, заменив в своем аппроксимирующем уравнении старый «чужой» прогноз на новый. Начальные условия

в новой задаче соответствуют непрерывному продолжению ранее полученного решения.

Важно заметить, что в момент времени t_1 пороговое ограничение у ресурса R2, вообще говоря, не нарушается, и необходимость в построении им нового прогноза $p_2(t)$ и пересылки его ресурсу R1 в этот момент отсутствует. Поэтому первый ресурс, не останавливаясь, продолжает интегрировать свое «текущее» аппроксимирующее уравнение.

Если первым будет превышено пороговое значение погрешности прогноза у второго ресурса, он должен будет вести себя аналогично. Далее все повторится.

В конечном итоге, в процессе работы алгоритма каждый ресурс сформирует «свою» компоненту кусочно-гладкой аппроксимирующей функции $\xi(t) = (\xi_1(t), \xi_2(t))$, приближающей точное решение $x(t) = (x_1(t), x_2(t))$.

Алгоритм очевидным образом обобщается на случай более двух групп уравнений, решаемых совместно.

6.2. Описание сценария решения тестовой задачи

На рис. 7 и 8 изображена сеть Петри, описывающая сценарий решения тестовой задачи в соответствии с приведенным выше алгоритмом. Как

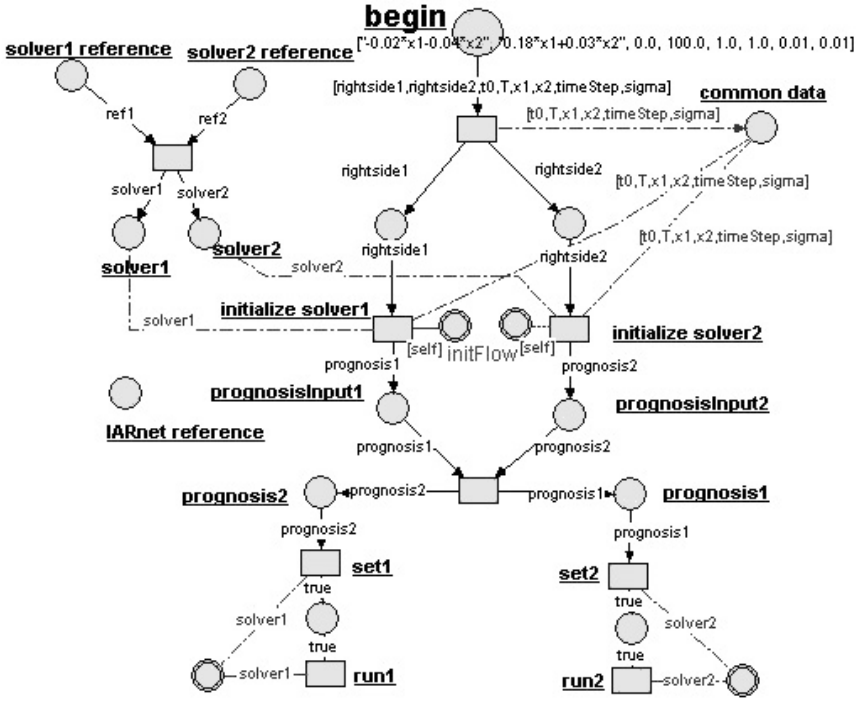

Рис. 7. Построенная сеть, часть 1

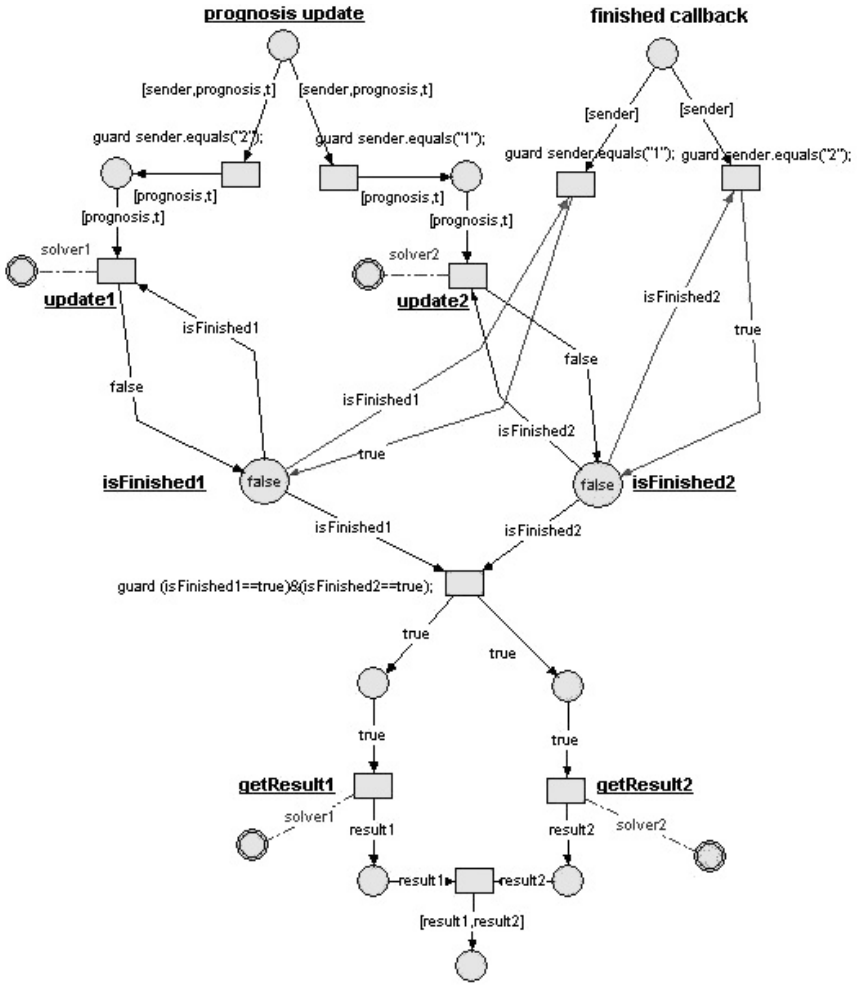

Рис. 8. Построенная сеть, часть 2

видно из рисунков, эта сеть состоит из двух частей. Первая из них отвечает за инициализацию решателей, передачу им начальных прогнозов и запуск решателей на выполнение. Вторая — за прием от решателей обратных вызовов об обновлении прогноза или о завершении работы и за действия, которые совершаются сценарием при приеме того или иного обратного вызова.

Рассмотрим сначала первую часть сети (рис. 7). В место *begin* помещены начальные данные, нужные для решения системы уравнений — их правые части *rightside1* и *rightside2*, значения начального и конечного

момента времени t_0 и T, начальные условия x_1 и x_2, шаг по времени *timeStep* и пороговая точность *sigma*. В места *solver1 reference* и *solver2 reference* помещены ссылки на решатели, которые в дальнейшем используются для вызова решателей сценарием. В место *IARnet reference* при запуске сценария помещается ссылка на ИАР сценария, которая передается решателям при их инициализации и в дальнейшем используется ими для осуществления обратных вызовов.

Когда сценарий запускается, правые части уравнений передаются соответственно первому и второму решателям, а прочие начальные данные — и тому, и другому. Это происходит при срабатывании переходов *initialize solver1* и *initialize solver2*, ответственных за инициализацию решателей. После инициализации каждый из решателей отправляет сценарию свой первоначальный прогноз, который сценарий в свою очередь передает другому решателю. Передача прогнозов решателям осуществляется при срабатывании переходов *set1* и *set2*. После получения начального прогноза решателям могут быть переданы команды на запуск, что и происходит при срабатывании переходов *run1* и *run2*.

Теперь перейдем ко второй части сети (рис. 8). Через некоторое время после запуска каждый из решателей начинает посылать сценарию обратные вызовы, содержащие обновление прогноза, свое текущее значение времени, а также свой уникальный идентификатор, который сценарий использует для опознания отправителя. При получении такого обратного вызова в место *prognosis update* помещается фишка, которую сценарий передает одному из решателей, в зависимости от отправителя. Так, прогноз от первого решателя передается второму, и наоборот. Это происходит при срабатывании переходов *solver1 update prognosis* или *solver2 update prognosis* соответственно. При срабатывании одного из этих переходов (т.е. при обновлении прогноза одного из решателей) соответственно в одно из мест *Finished1* или *Finished2* помещается фишка с логическим значением false. Это означает, что соответствующий решатель продолжает свою работу. Даже если этот решатель к этому моменту уже завершил вычисления, при получении свежего прогноза (это означает, что работа завершена преждевременно) он «откатывается» назад и продолжает работу с указанного момента времени.

Второй вид обратных вызовов, которые может принимать сценарий, — это сообщения о завершении работы одним или другим решателем. При этом в место *finished callback* помещается фишка, содержащая идентификатор отправителя. Получив такую фишку, сценарий помещает логическое значение true в одно из мест *Finished1* или *Finished2* соответственно.

Как только значения фишек в местах *Finished1* и *Finished2* становятся равными true одновременно, срабатывает следующий за ними переход, который, в свою очередь вызывает срабатывание переходов *getResult1* и *getResult2*, каждый из которых запрашивает у соответствующего решателя результаты его вычислений. При получении результатов от обоих решателей сценарий записывает полученные результаты в текстовый файл, с помощью вызова соответствующего Java-кода.

6.3. Результаты вычислительных экспериментов

Решаемая система дифференциальных уравнений задавалась в виде:

```
rightSide1 = -0,02*x1 - 0,04*x2;
rightSide2 = 0,18*x1 + 0,03*x2;
x1(0)=1; x2(0)=1;
t0=0; T=100.
```

Значения шага по времени *timeStep* и пороговой точности *sigma* были приняты равными 0,01.

В ходе вычислительных экспериментов выяснилось, что новые прогнозы, приходящие с большой частотой обрабатываются неудовлетворительно: из-за простоты выбранной тестовой задачи время работы решателей между моментами обновлений их прогнозов было существенно меньше времени, требуемого для осуществления передачи нового прогноза другому решателю по сети. Поэтому передаваемые сценарию обновления прогнозов скапливались в соответствующих местах сети, ожидая передачи соответствующим решателям. В результате этого решатели могли получать уже «устаревшие» прогнозы и работать неэффективно — один из решателей мог «убегать» по времени далеко вперед от другого, что вызывало ряд последующих неэффективных «откатов» обоих решателей.

В связи с этим было принято решение ввести искусственную задержку на каждом шаге итерации решателей с тем, чтобы уменьшить частоту приходящих от них обратных вызовов (прогнозов). Это приблизило рассматриваемую модель к реальной ситуации, с которой приходится иметь дело при распределенных вычислениях: время локальных вычислений намного превышает время обмена данными по сети. В противном случае распределенное решение задачи является неэффективным.

Искусственная задержка была сделана случайно-переменной по величине в интервале от 0,1 до 0,2 секунды. Это помогло достичь желаемого результата, хотя и за счет увеличения времени решения тестовой задачи. Процессы решения обоих решателей стали практически синхронными, т. е. в каждый момент времени они находились на близких шагах итерации. При этом передаваемые сценарию обновления прогнозов больше не скапливались внутри сценария.

В ходе дальнейших вычислительных экспериментов было проверено, что полученное в результате запуска тестового сценария решение качественно, а с некоторой степенью точности и количественно, повторяет точное аналитическое решение рассматриваемой системы уравнений. Подробное изучение характеристик данного распределенного алгоритма выходило за рамки тестирования прототипа СУС IARnet и может быть проведено в рамках дальнейших исследований.

7. Заключение

В ходе работ, описанных в настоящей статье, была создана архитектура службы управления сценариями системы IARnet и выбран формализм для описания распределенных вычислительных сценариев. На основе этого

был создан рабочий прототип службы управления сценариями IARnet, испытания которого продемонстрировали принципиальную применимость методологии сценариев при решении задач в распределенной вычислительной среде.

Литература

1. *Емельянов С. В.*, *Афанасьев А. П.*, *Волошинов В. В.*, *Гринберг Я. Р.*, *Кривцов В. Е.*, *Сухорослов О. В.* Реализация Grid-вычислений в среде IARnet // Информационные технологии и вычислительные системы. 2005. № 2. С. 61–75.
2. *Котов В. Е.* Сети Петри. М.: Наука, 1984.
3. *Кривцов В. Е.*, *Попков Е. Ю.* Схема распределенного решения системы дифференциальных уравнений // Проблемы вычислений в распределенной среде: прикладные задачи / Труды института системного анализа Российской академии наук (ИСА РАН). М.: УРСС, 2004.
4. *Лазарев И. В.*, *Сухорослов О. В.* Использование workflow-методологии для описания процесса распределенных вычислений // Проблемы вычислений в распределенной среде: Модели обработки и представления данных. Динамические системы / Труды ИСА РАН. М.: КомКнига/URSS, 2005. Т. 14. С. 26–70.
5. *Hollingsworth D.* The Workflow Reference Model. Workflow Management Coalition, 1995.
6. Income Suite (www.get-process.com/en/home.php).
7. JFern (http://sourceforge.net/projects/jfern).
8. *Jensen K.* An introduction to the theoretical aspects of Coloured Petri Nets. In J. de Bakker, W.-P. de Roever, G. Rozenberg, editors, A Decade of Concurrency. Vol. 803 of Lecture Notes in Computer Science. P. 230–272. Springer-Verlag, 1994.
9. Petri Net Kernel (www.informatik.hu-berlin.de/top/pnk/index.html).
10. Petri Nets Tools Database (www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html).
11. Renew (www.renew.de).
12. Wolfgang Reisig. Elements of Distributed Algorithms: Modelling and Analysis with Petri Nets. Springer-Verlag, 1998.
13. YAWL (www.yawl.fit.qut.edu.au).