УДК 681.5:622.276; 622.279

ПРИМЕНЕНИЕ СЕТЕЙ ПЕТРИ К ПОСТРОЕНИЮ АДАПТИРУЕМОГО РАСПРЕДЕЛЕННОГО ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Д.Г. Леонов
(*РГУ нефти и газа (НИУ) имени И.М. Губкина*)

Процесс непрерывного совершенствования современных систем диспетчерского управления является сложной наукоемкой задачей, носящей междисциплинарный характер и базирующейся на интеграции знаний и опыта производственных, академических, научно-исследовательских, учебных институтов и учреждений в области автоматизации, управления и информационных систем [5]. Используемые в системах диспетчерского управления транспортом нефти и газа программно-вычислительные комплексы (ПВК) являются сложными системами, обеспечивающими проведение режимных расчетов, решение задач многовариантного планирования и поиска рациональных режимов с различными уровнями детализации.

Ключевым компонентом подобных комплексов является подсистема моделирования, эффективная реализация которой требует решения ряда научно-инженерных задач. На первых этапах построения ПВК разработка, как правило, ведется небольшими рабочими группами, причем работа нередко носит инициативный характер. Для этих этапов характерно отсутствие должного внимания к документированию и формализации проектирования, поскольку основные усилия разработчиков направлены на реализацию ключевых функциональных возможностей.

После перехода к промышленному использованию перед разработчиками встает ряд проблем как организационного, так и технологического характера, связанных с необходимостью технической поддержки и сопровождения комплексов, наращивания функциональности, а также интеграции в существующие системы. Бурное развитие современных информационных технологий (смена поколений процессоров и операционных систем, появление новых стандартов и подходов к организации распределенных систем, широкое распространение мобильных устройств с постоянно меняющимися возможностями) приводит к тому, что инфраструктура вычислительных комплексов устаревает значительно быстрее их прикладной составляющей и требует постоянной адаптации.

Решаемые проблемы зачастую вступают в противоречие между собой: адаптация к современным технологиям и необходимость обеспечения обратной совместимости с существующим парком вычислительной техники; быстрое прототипирование с использованием средств быстрой разработки (RAD) и необходимость обеспечения высокой вычислительной производительности; увеличение числа задач, востребованных пользователями, и борьба с перегруженностью пользовательского интерфейса. Возрастает роль инструментальных средств, обеспечивающих совместную разработку и контроль версий, автоматизированное тестирование и документирование, создание виртуальных полигонов, соответствующих пользовательским конфигурациям, автоматизированную обработку кода в целях обеспечения локализации и кроссплатформенности. Кроме того, сложность решаемых задач обусловливает необходимость привлечения математического аппарата, обеспечивающего обобщенное формализованное представление о системе в целом.

Развитие архитектуры ПВК "Веста" и подходов к ее формализации

Эволюционное развитие архитектуры разрабатываемого в РГУ нефти и газа (НИУ) имени И.М. Губкина ПВК "Веста" [8, 9] создало предпосылки для преобразования его транспортной подсистемы в основу открытой интеграционной платформы (ОИП), позволяющей объединить разнородные программные комплексы, а также обеспечить применение облачных технологий [10, 11]. ОИП стандартизирует формат и протоколы обмена данными на основе открытого программного интерфейса (Application Programming Interface – API) и представляет собой распределенную

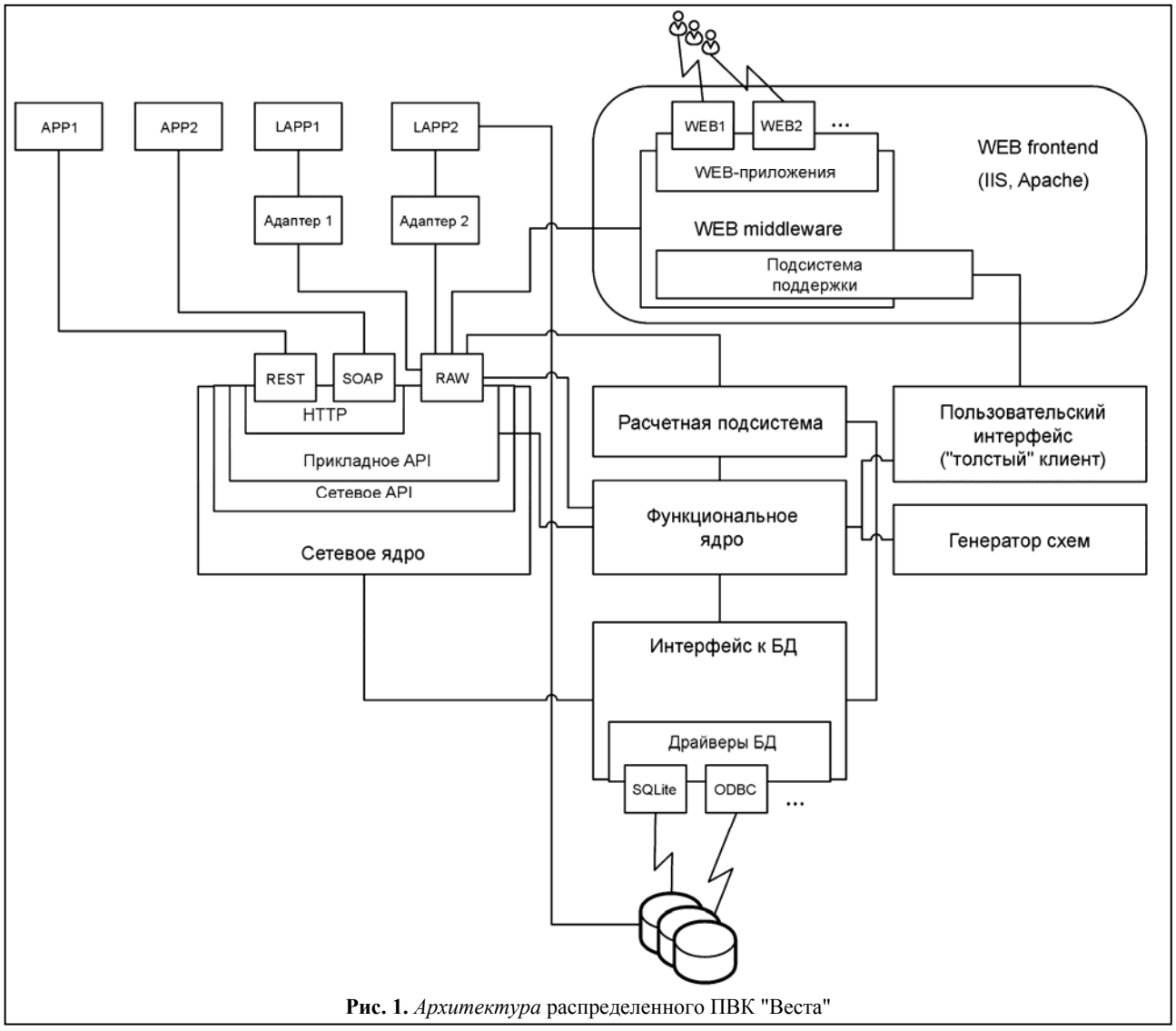

Рис. 1. *Архитектура* распределенного ПВК "Веста"

гетерогенную компьютерную среду, обеспечивающую общий открытый интерфейс передачи данных в унифицированном формате (рис. 1).

Изменение характера решаемых задач потребовало смены подхода к формализации представления о проектируемой системе. В ПВК "Веста" первоначально для этих целей использовался аппарат теории конечных автоматов [8] в сочетании со средствами автоматизированной генерации исходного кода, который хорошо зарекомендовал себя при проектировании так называемых реактивных систем [12, 13]. Однако при переходе к полностью асинхронным распределенным системам модели конечных автоматов становятся плохо применимыми из-за необходимости выявления "глобальных" состояний системы в целом.

В настоящее время наиболее распространенным подходом к формализации параллельных систем с асинхронным взаимодействием является аппарат сетей Петри, позволяющий свести исследование механизмов взаимодействия к изучению логической структуры сети [4, 6, 7]. Для решения поставленной задачи было выбрано подмножество иерархических раскрашенных сетей Петри, для которых характерно наличие составных переходов, срабатывание которых может быть разложено на составные действия, а фишкам позиций (мест) соответствуют передаваемые от перехода к переходу объекты, имеющие дополнительные атрибуты, что хорошо соответствует объектно-ориентированной архитектуре рассматриваемого комплекса.

Для визуализации и моделирования использовался программный пакет CPN Tools 4.0.1 (AIS group, Технический университет Эйндховена) [1].

Модель транспортной подсистемы

Транспортная подсистема ПВК "Веста" относится к системам класса middleware (таким, как ZeroMQ, RabbitMQ и т. п.), традиционным решением для которых является использование механизма передачи сообщений в качестве основы обработки и передачи данных. При отправке сообщения ставятся в очередь, из которой последовательно выбираются для последующей обработки и доставки клиентам (рис. 2).

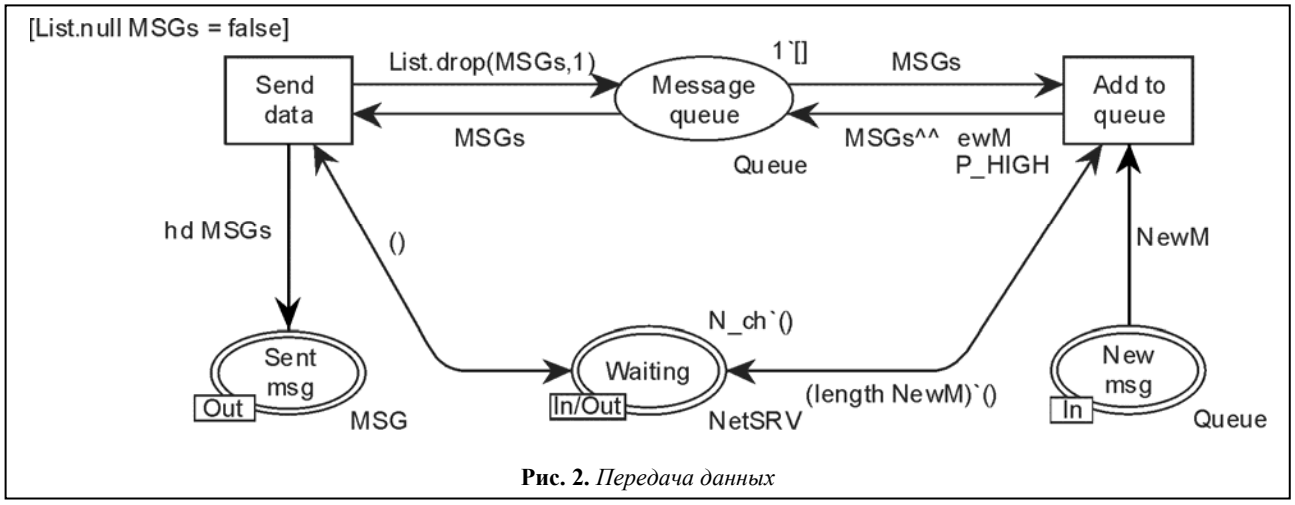

Рис. 2. *Передача данных*

Сообщения состоят из заголовка со служебной информацией и собственно данных:

```
colset ID = INT;
colset ClientInfo = product ID* Subs;
var ClInfo: ClientInfo;
colset MSG = product ClientInfo *STRING;
var msg: MSG;
```

Основными компонентами системы являются сетевой менеджер, клиенты, расчетные модули и менеджер схем. Сетевой менеджер отвечает за регистрацию разнородных клиентов и передачу данных. Менеджер схем обеспечивает запуск расчетных модулей по запросу клиентов.

Для маршрутизации потоков сообщений в сетевом менеджере предусмотрены отдельные очереди для каждого типа клиентов:

```
colset MSG = record to:INT*from:INT*t:ClType*data:STRING;
colset ClType = with Calc|API|NetSrv|CalcMgr;
```

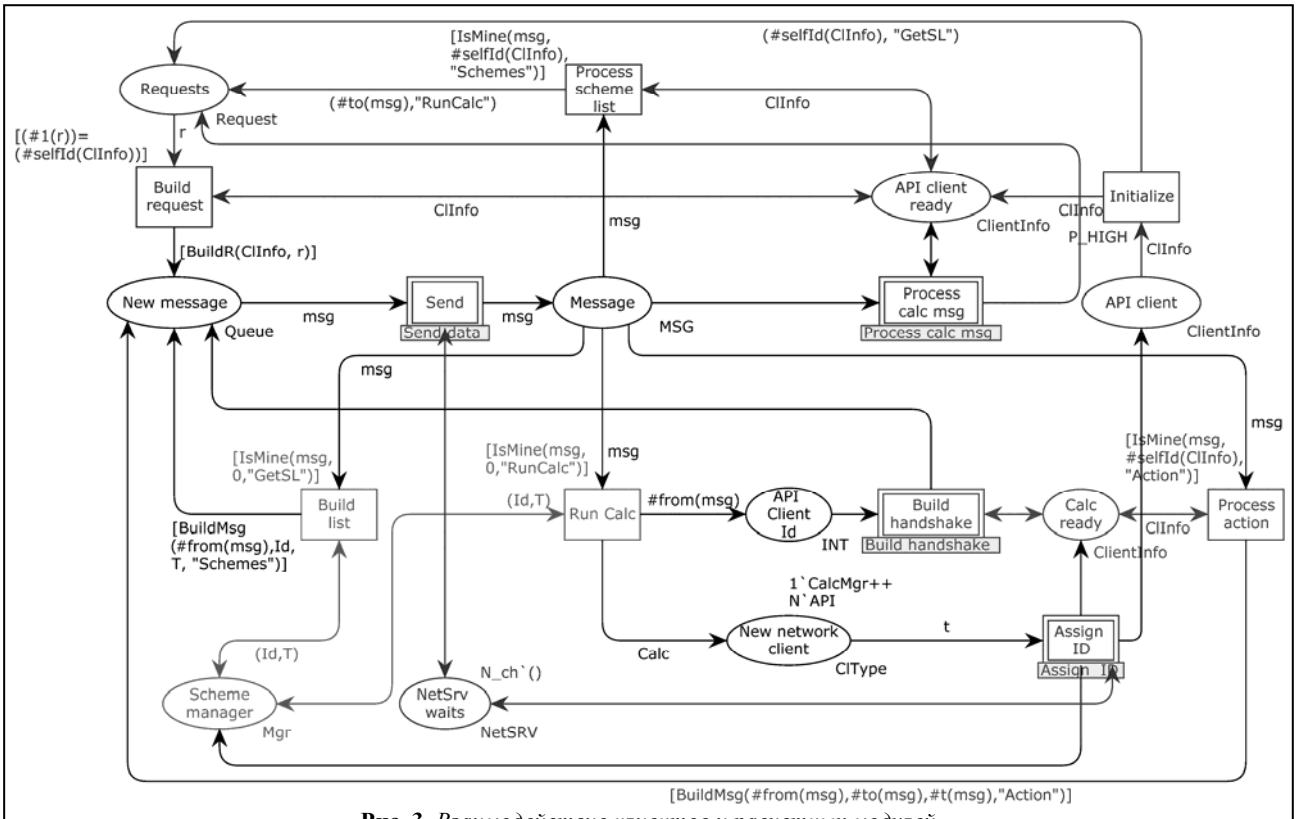

Рис. 3. *Взаимодействие клиентов и расчетных модулей*

В целом работа каждой новой пары "клиент–расчетный модуль" разбивается на несколько шагов (рис. 3):

1. Регистрация клиента в сети;
2. Запрос у менеджера схем списка доступных схем;
3. Запрос к менеджеру схем на запуск расчетного модуля с конкретной схемой;
4. Запуск расчетного модуля менеджером схем;
5. Регистрация расчетного модуля в сети;
6. Отправка идентификационных данных расчетного модуля клиенту;
7. Взаимодействие клиента и расчетного модуля.

Автоматизация формирования программного кода

Следующим этапом исследований стал переход от ручного программирования результатов моделирования к автоматизированному формированию программного кода на основе построенных сетей Петри. Автоматизированное формирование каркаса распределенного приложения, во-первых, позволяет минимизировать усилия прикладных программистов при реализации смоделированного взаимодействия, во-вторых, обеспечивает решение задачи перехода к другой реализации транспортной инфраструктуры или другой язык программирования. Таким образом, программное обеспечение (ПО) моделирования в сочетании с генератором программного кода можно рассматривать как средство быстрой разработки (RAD), обеспечивающее быстрое прототипирование распределенной системы.

Преобразование в программный код моделей, основанных на переходах между состояниями, может быть реализовано двумя способами. В первом случае в приложение добавляется компонент, максимально точно имитирующий передачу сообщений и связанные с их обработкой переходы между состояниями. Данный подход хорошо соответствует задаче перехода от моделей, основанных на конечных автоматах, к локальным приложениям [8, 12, 13]. Но поскольку при построении распределенных систем транспортная подсистема уже, как правило, использует ту или иную реализацию очереди сообщений, целесообразной является трансляция терминов (состояний, переходов и сообщений) модели в термины выбранного транспортного решения.

Подобный подход, обеспечивающий использование модели, основанной на сетях Петри, в качестве основы для прикладных задач, развивается в библиотеке PNML Framework [3], реализующей стандарт ISO/IEC-15909 [2]. Данный стандарт определяет формат PNML (Petri Net Markup Language), обеспечивающий переносимое представление различных видов сетей Петри в виде XML-файла. PNML Framework обеспечивает считывание PNML-файлов с возможностью встраивания обработчиков для конвертации в произвольный формат.

К сожалению, PNML не обеспечивает поддержки иерархических раскрашенных сетей Петри, наиболее подходящих для моделирования распределенных систем, обменивающихся сложными наборами данных, поэтому формат данных пакета CPN Tools несовместим с PNML Framework. При этом формат данных CPN Tools также основан на языке XML, что делает возможным его независимую обработку сторонним транслятором. Ключевые для моделей сущности сохраняются в виде тегов <place> (позиции), <trans> (переходы), <arc> (дуги), <color> (цвета). Связь между ними задается с помощью атрибутов id, transend, placeend. Тег <page> отвечает за хранение в одном файле сетей, соответствующих разным уровням иерархии. Возможность работы со строковыми переменными обеспечивает включение в модель метаинформации, например, отвечающей за привязку отдельных ее фрагментов к различным компонентам распределенной системы.

Листинг 1. **Фрагмент исходного CPN-файла**

```
<arc id="ID1424929876" orientation="PtoT" order="1">
  <transend idref="ID1424929862"/>
  <placeend idref="ID1424929873"/>
</arc>
<place id="ID1424929873">
  <text>API client ready</text>
</place>
<trans id="ID1424929862" explicit="false">
  <text>Process calc msg</text>
</trans>
```

В листинге 1 представлен очищенный от вспомогательных тегов фрагмент CPN-файла, описывающий дугу из позиции "API client ready" к переходу "Process calc msg".

Прототип транслятора PNML/CPP был реализован с помощью языка преобразования XML-документов XSLT. В листинге 2 представлен один из шаблонов, отвечающих за преобразование исходной логической структуры CPN-файла в код C++ (листинг 3), использующий вызовы сетевого API ПВК "Веста".

Листинг 2. **XSLT-шаблон**

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" encoding="utf8"/>
<xsl:template match="workspaceElements">
  <xsl:apply-templates select="cpnet/page"/>
</xsl:template>
<xsl:template match="cpnet/page">
  <xsl:apply-templates select="arc"/>
</xsl:template>
<xsl:template match="arc">
  <xsl:variable name="TId"  select="transend/@idref"/>
  <xsl:variable name="PId"  select="placeend/@idref"/>
<xsl:choose><xsl:when test="@orientation='TtoP'">
<xsl:text>
// шаблон обработчика</xsl:text>
<xsl:value-of select="../trans[@id=$TId]/@id"/>
<xsl:text>
void OnAM</xsl:text><xsl:value-of
    select="translate(../trans[@id=$TId]/text,
    '&#x20;&#x9;&#xD;&#xA;', '')"/>
<xsl:text>(AMessage* pMessage)
{
//  вызов прикладной функции
</xsl:text><xsl:value-of
    select="translate(../place[@id=$PId]/text,
    '&#x20;&#x9;&#xD;&#xA;', '')"/>
<xsl:text>(pMessage);
}
</xsl:text>
</xsl:when>
<xsl:otherwise>
<xsl:text>
// шаблон прикладной функции </xsl:text>
<xsl:value-of select="../place[@id=$PId]/@id"/>
<xsl:text>
void </xsl:text>
<xsl:value-of
    select="translate(../place[@id=$PId]/text,
    '&#x20;&#x9;&#xD;&#xA;', '')"/>
<xsl:text>(AMessage* pMessage)
{
    AsgardRead((LPARAM)pMessage, [&amp;](CArchive&amp; ar)
    {
          // считывание данных из сообщения: ar &gt;&gt; var
    });
//  прикладной код
//  ...
//  отправка сообщения
    AsgardWriteAndPost(ACT_VESTA, AR_CLIENTS,
          AM_</xsl:text><xsl:value-of
    select="translate(../trans[@id=$TId]/text,
    '&#x20;&#x9;&#xD;&#xA;', '')"/>
<xsl:text>, [&amp;](CArchive&amp; ar)
    {
          // формирование данных для сообщения: ar &lt;&lt; var
    });
}
</xsl:text>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

Листинг 3. **Фрагмент сгенерированного кода**

```
// шаблон прикладной функции ID1424929873
void APIclientready(AMessage* pMessage)
{
    AsgardRead((LPARAM)pMessage, [&](CArchive& ar)
    {
          // считывание данных из сообщения: ar >> var
    });
//  прикладной код
//  ...
//  отправка сообщения
    AsgardWriteAndPost(ACT_VESTA, AR_CLIENTS,
          AM_Processcalcmsg, [&](CArchive& ar)
    {
          // формирование данных для сообщения: ar << var
    });
}
// шаблон обработчика ID1424929862
void OnAMProcesscalcmsg(AMessage* pMessage)
{
//  вызов прикладной функции
    APIclientready(pMessage);
}
```

Использование XSLT-шаблонов обеспечивает быструю реализацию транслятора, но имеет ряд недостатков, затрудняющих применение в условиях реальной разработки.

Во-первых, как видно из приведенных листингов 2 и 3, фрагменты генерируемого кода включаются непосредственно в шаблон, при этом специфика обработки XSLT требует чередования в шаблоне фрагментов кода на C++ со специфичными XSLT-тегами и замены ряда стандартных символов на так называемые entities ("&" на "&" и т. п.). То есть подготовка транслятора для другой платформы потребует полной переделки шаблона.

Во-вторых, XSLT не имеет встроенных средств, позволяющих исключить дублирование полученных фрагментов. В случае применения приведенного в качестве примера шаблона к сети, содержащей позиции либо переходы, к которым подходят несколько дуг, результатом работы станет создание нескольких одноименных функций, исключение которых потребует ручной работы.

Наконец, результатом работы транслятора является сплошной поток данных (одиночный файл либо отдельная страница в браузере), разбиение которого по компонентам распределенной системы необходимо проводить вручную.

Поэтому дальнейшим направлением работы является создание специализированного транслятора, обеспечивающего разбор XML-дерева с его последующей обработкой и включением внешних шаблонных файлов, отделенных от кода транслятора, а также с использованием метаинформации из различных фрагментов модели для формирования программного кода, привязанного к различным компонентам распределенной системы.

Заключение

Переход в формализации системы на иерархические раскрашенные сети Петри позволил провести моделирование взаимодействия существующих компонентов, а также эксперименты по подключению компонентов, находящихся на стадии проектирования. Трансляция XML-представления полученных сетей Петри в автоматически генерируемый программный код обеспечивает быстрое прототипирование распределенной системы и ее перенос на другие платформы.

ЛИТЕРАТУРА

1. CPN Tools [Электронный ресурс]. – URL: http://cpntools.org/.

2. The Petri Network Markup Language [Электронный ресурс]. – URL: http://www.pnml.org/.

3. PNML Framework [Электронный ресурс]. – URL: http://pnml.lip6.fr/.

4. Анисимов Н.А., Голенков Е.А., Харитонов Д.И. Композициональный подход к разработке параллельных и распределенных систем на основе сетей Петри // Программирование. – 2001. – № 6. – С. 30.

5. Григорьев Л.И., Костогрызов А.И. Актуальность и основы инновационного пути развития АСДУ // Автоматизация, телемеханизация и связь в нефтяной промышленности. – М.: ОАО "ВНИИОЭНГ", 2016. – № 3.

6. Котов В.Е. Сети Петри. – М.: Наука, 1984.

7. Коротиков С.В., Воевода А.А. Применение сетей Петри в разработке программного обеспечения центров дистанционного управления и контроля // Научный вестник Новосибирского государственного технического университета. – 2007. – № 4. – С. 15–32.

8. Леонов Д.Г. Объектно-ориентированная технология разработки систем поддержки принятия диспетчерских решений в транспорте газа // Автоматизация, телемеханизация и связь в нефтяной промышленности. – М.: ОАО "ВНИИОЭНГ", 2000. – № 4–5.
9. Леонов Д.Г., Васильев А.В. Построение многоуровневой системы поддержки принятия диспетчерских решений, основанное на развитии распределенной архитектуры программно-вычислительного комплекса "Веста" // Автоматизация, телемеханизация и связь в нефтяной промышленности. – М.: ОАО "ВНИИОЭНГ", 2014. – № 6.
10. Папилина Т.М. Платформа разработки прикладных web-инструментов для диспетчерского персонала нефтегазовой отрасли // Автоматизация, телемеханизация и связь в нефтяной промышленности. – М.: ОАО "ВНИИОЭНГ", 2015. – № 11.
11. Папилина Т.М., Леонов Д.Г., Степин Ю.П. Моделирование и оценка эффективности функционирования системы облачных вычислений в АСДУ // Автоматизация, телемеханизация и связь в нефтяной промышленности. – М.: ОАО "ВНИИОЭНГ", 2016. – № 7.
12. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб.: Питер, 2009.
13. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. – СПб.: Наука, 1998.

LITERATURA

1. CPN Tools [Elektronnyy resurs]. – URL: http://cpntools.org/.
2. The Petri Network Markup Language [Elektronnyy resurs]. – URL: http://www.pnml.org/.
3. PNML Framework [Elektronnyy resurs]. – URL: http://pnml.lip6.fr/.
4. Anisimov N.A., Golenkov E.A., Kharitonov D.I. Kompozitsional'nyy podkhod k razrabotke parallel'nykh i raspredelennykh sistem na osnove setey Petri // Programmirovanie. – 2001. – № 6. – S. 30.
5. Grigor'ev L.I., Kostogryzov A.I. Aktual'nost' i osnovy innovatsionnogo puti razvitiya ASDU // Avtomatizatsiya, telemekhanizatsiya i svyaz' v neftyanoy promyshlennosti. – M.: OAO "VNIIOENG", 2016. – № 3.
6. Kotov V.E. Seti Petri. – M.: Nauka, 1984.
7. Korotikov S.V., Voevoda A.A. Primenenie setey Petri v razrabotke programmnogo obespecheniya tsentrov distantsionnogo upravleniya i kontrolya // Nauchnyy vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta. – 2007. – № 4. – S. 15–32.
8. Leonov D.G. Ob"ektno-orientirovannaya tekhnologiya razrabotki sistem podderzhki prinyatiya dispetcherskikh resheniy v transporte gaza // Avtomatizatsiya, telemekhanizatsiya i svyaz' v neftyanoy promyshlennosti. – M.: OAO "VNIIOENG", 2000. – № 4–5.
9. Leonov D.G., Vasil'ev A.V. Postroenie mnogourovnevoy sistemy podderzhki prinyatiya dispetcherskikh resheniy, osnovannoe na razvitii raspredelennoy arkhitektury programmno-vychislitel'nogo kompleksa "Vesta" // Avtomatizatsiya, telemekhanizatsiya i svyaz' v neftyanoy promyshlennosti. – M.: OAO "VNIIOENG", 2014. – № 6.
10. Papilina T.M. Platforma razrabotki prikladnykh web-instrumentov dlya dispetcherskogo personala neftegazovoy otrasli // Avtomatizatsiya, telemekhanizatsiya i svyaz' v neftyanoy promyshlennosti. – M.: OAO "VNIIOENG", 2015. – № 11.
11. Papilina T.M., Leonov D.G., Stepin Yu.P. Modelirovanie i otsenka effektivnosti funktsionirovaniya sistemy oblachnykh vychisleniy v ASDU // Avtomatizatsiya, telemekhanizatsiya i svyaz' v neftyanoy promyshlennosti. – M.: OAO "VNIIOENG", 2016. – № 7.
12. Polikarpova N.I., Shalyto A.A. Avtomatnoe programmirovanie. – SPb.: Piter, 2009.
13. Shalyto A.A. Switch-tekhnologiya. Algoritmizatsiya i programmirovanie zadach logicheskogo upravleniya. – SPb.: Nauka, 1998.