Р13-2013-91

К. М. Саламатин [1,*]

DiCME — РАСПРЕДЕЛЕННАЯ СРЕДА ВЗАИМОДЕЙСТВИЯ КОМПОНЕНТОВ СИСТЕМЫ АВТОМАТИЗАЦИИ ЭКСПЕРИМЕНТОВ ДЛЯ ФИЗИКИ НИЗКИХ ЭНЕРГИЙ

Направлено в журнал «Информатика и ее применения»

[1] Международный университет «Дубна», Дубна
[*] E-mail: del@tmpk.ru

Саламатин К. М. Р13-2013-91

DiCME — распределенная среда взаимодействия
компонентов системы автоматизации экспериментов
для физики низких энергий

Сложные современные системы автоматизации экспериментов (САЭ) являются распределенными компонентными системами. Функциональность и информация одних компонентов должна быть доступна другим. Использование алгоритмов RPC вводит связанность компонентов и выполняется синхронно по схеме «один с одним», что затрудняет реализацию систем реального времени. Разработанная среда обеспечения взаимодействия процессов DiCME использует новый метод динамического связывания и является мощным инструментом интеграции компонентов, обеспечивающим автоматическую компоновку САЭ и возможность использования кода компонентов в различных экспериментах без редактирования остальных составляющих системы.

Работа выполнена в соответствии с протоколом о совместных работах Лаборатории нейтронной физики им. И. М. Франка ОИЯИ и университета «Дубна».

Препринт Объединенного института ядерных исследований. Дубна, 2013

Salamatin K. M. P13-2013-91

DiCME — Distributed Components Messaging Environment
for Low Energy Physics Experiments Automation

Complicated modern experiments automation systems (EAS) are component-based distributed systems. Functionality and data of some components should be available to others. RPC algorithms introduce hard components coupling and are executed synchronously in a «one-to-one» mode, which complicates the implementation of real-time systems. The designed layer for inter-process communication DiCME uses a new method of dynamic binding and is a powerful tool for automatic integration of components that provides the use of the EAS components code in different experiments without editing the other parts of the system.

The investigation has been performed in accordance with the protocol on the collaboration between the Frank Laboratory of Neutron Physics, JINR, and University «Dubna» on the development of the EAS programming methods.

Preprint of the Joint Institute for Nuclear Research. Dubna, 2013

Современные системы автоматизации экспериментов (САЭ) реализуются в виде распределенных приложений, в которых необходима организация взаимодействия между процессами (IPC — InterProcess Communication), выполняемыми на разных ЭВМ (возможно, и на разнородных процессорах). Взаимодействие сводится к использованию функциональности одного процесса другим и передаче информации. Важно обеспечить каждому процессу прозрачный способ использования средств IPC. Многие из современных приложений используют для целей IPC технологии RMI, CORBA, DCOM и др. Базовые реализации этих технологий являются в некотором роде расширением RPC (Remote Procedure Call), так как позволяют вызвать метод удаленного объекта, который, по сравнению с классическим RPC в конфигурации клиент–сервер, обеспечивает лучшую прозрачность. Однако, как и в RPC, взаимодействие процессов выполняется по схеме «один с одним», в некоторых технологиях — синхронно. Для САЭ это является ограничением, так как в системах реального времени события возникают в случайные моменты времени и синхронность взаимодействия приводит к потерям времени на ожидание. Другое и более важное свойство — способ связывания компонентов. Как правило, компонентные технологии предоставляют возможность статического и динамического связывания. Динамическое связывание компонентов САЭ является критическим свойством, от которого зависит возможность повторного использования кода. Однако способ динамического связывания, используемый в популярных технологиях (CORBA, DCOM и др.), для САЭ является избыточно сложным и может быть существенно упрощен, если учесть специфику САЭ. Эти (и другие) соображения стимулировали выполнение разработки специальной среды взаимодействия DiCME (Distributed Components Messaging Environment) для использования в САЭ.

Область интересов автора — автоматизация спектрометрии нейтронов на реакторе ИБР-2 [1] и ускорителе ИРЕН [2], поэтому процесс разработки DiCME, проектные решения проиллюстрированы на материалах систем автоматизации спектрометрических экспериментов. Данная разработка выполнена в общем виде для систем с сервисориентированной архитектурой (SOA) [3] и пригодна для использования в системах автоматизации для физики низких энергий и в других проблемных областях.

1. ОСОБЕННОСТИ САЭ, ВЛИЯЮЩИЕ НА АЛГОРИТМЫ СРЕДЫ ВЗАИМОДЕЙСТВИЯ

На основании анализа состава функций, используемых в различных САЭ, в работе [4] сформулированы две основные группы компонентов САЭ по принадлежности к логике приложения:

1) базовые компоненты с детерминированным характером взаимодействия, реализующие основную задачу эксперимента (базовую логику) — получение экспериментальных данных; это интерфейсы пользователя, программа управления экспериментом, программы управления условиями эксперимента и подсистемы регистрации данных (DAQ — ввод, преобразование, архивирование потока данных);

2) компоненты (и некоторые методы базовых компонентов), реализующие вспомогательную логику — сервисные функции, обработку нештатных ситуаций и др., не влияющие (в штатных условиях) на выполнение основной задачи.

Характерные особенности способа взаимодействия группы базовых компонентов следующие:

- схема взаимодействия определена базовой логикой эксперимента и фиксирована;
- способ взаимодействия «один к одному»;
- в процессе взаимодействия клиент запрашивает выполнение метода и нужна гарантированная доставка сообщений — запроса клиента и ответа сервиса;
- в качестве ответа (результата вызова метода) клиенту нужен только сигнал о завершении выполнения запроса;
- нет существенных оснований к тому, чтобы в тело кода клиента вводить информацию о конкретном связанном сервисе (например, адрес), и наоборот; мы можем иметь дело со слабо связанными компонентами.

Группа компонентов, реализующих вспомогательную логику, может взаимодействовать по другим правилам:

- схема взаимодействия вспомогательных компонентов не фиксируется;
- способ взаимодействия «один к одному»;
- в составе группы два типа компонентов — источники и потребители информации; в процессе взаимодействия компоненты-источники публикуют информацию, а не вызывают методы потребителей;
- источник может публиковать информацию различного типа (адреса экспериментальных данных, информацию о состоянии узлов установки, диагностические сообщения и др.); потребителю требуется информация определенного типа, потребителей информации любого типа может быть несколько либо ни одного — факт невостребованности этой информации не влияет на выполнение базовой логики эксперимента.

Средства обеспечения взаимодействия должны обслуживать все процессы и все варианты их активности в прикладной системе реального времени. К средствам обеспечения взаимодействия процессов выдвигаются следующие требования:

1) автоматический поиск и динамическое связывание компонентов; автоматический поиск существенно улучшает эксплуатационные свойства системы;

2) асинхронный механизм вызова методов, так как при синхронном вызове снижается скорость выполнения логики приложения (и пропускная способность), а также усложняется программирование системы реального времени, где процессы выполняются одновременно, и события, требующие обработки, возникают асинхронно;

3) возможность передать информацию нескольким другим процессам;

4) одна и та же среда взаимодействия должна обрабатывать все обмены в рамках САЭ, так как гомогенную систему намного легче программировать и поддерживать;

5) прозрачность взаимодействий в распределенной системе; где бы ни исполнялся процесс, он должен иметь возможность взаимодействовать с любым процессом в системе, используя единый механизм, который не зависит от размещения процессов;

6) возможность перемещения процесса с одной машины на другую; это облегчит устранение аварийных ситуаций, приводящих к выходу из строя ЭВМ;

7) интерфейс компонента не должен зависеть от границ ЭВМ — код должен быть одинаков при обращении к процессу на той же машине или на другой, в том числе иного типа;

8) потеря процесса, ЭВМ или разрыв сетевой связи не должны приводить к разрушению остальной части системы;

9) автоматическая адаптация к используемой конфигурации.

Реализация среды взаимодействия, отвечающей перечисленным требованиям, значительно упростит разработку, облегчит эксплуатацию и развитие САЭ, улучшит эффективность и надежность прикладной системы.

2. ПРОЕКТНЫЕ РЕШЕНИЯ

2.1. Известные решения.

DIM. Приведенному составу требований в некоторой степени соответствует среда взаимодействия DIM (Distributed Information Management) [5]. Однако эта среда, предназначенная для физики высоких энергий, используется в крупных системах (например, в [6, 7] — сотни ЭВМ, тысячи компонентов) и имеет следующие особенности:

• введены двухуровневые списки серверов, что является избыточным для задач физики низких энергий;

• заполнение списков серверов (конфигурирование) выполняется разработчиками, в связи с чем требуется персонал сопровождения, возникает источник ошибок оператора;

• заполнение списков серверов разработчиками вносит статическую связанность, в результате реализация среды взаимодействия теряет универсальность и ухудшаются условия повторного использования кода компонентов; для уникальных систем [6, 7 и др.] потеря возможности повторного использования кода не имеет значения, но противоречит постановке задачи в данной работе;

• не учитывается специфика САЭ для физики низких энергий, соответственно, нет разделения компонентов по принадлежности к базовой и вспомогательной логике, и это усложняет развитие прикладной системы, а также уменьшает возможность повторного использования кода компонентов.

CORBA. Одна из важнейших характеристик среды взаимодействия — способ связывания компонентов, обеспечивающий нормальное функционирование приложения. Для систем автоматизации экспериментов многократное изменение методики эксперимента свойственно самому процессу экспериментальных исследований. Поэтому для САЭ нужен механизм, обеспечивающий динамическое связывание компонентов в соответствии с изменениями базовой логики эксперимента. Наиболее популярная компонентная технология CORBA [8] реализует динамическое связывание через специальный набор средств: DII (Dynamic Invocation Interface) на стороне клиента и DSI (Dynamic Skeleton Interface) на стороне сервера. Через эти интерфейсы для настройки каждого взаимодействия осуществляется ряд вызовов, во время которых выясняются имя метода сервера, типы и значения его аргументов, тип результата, выполняется вызов метода и получение результата. В итоге при использовании динамического связывания по технологии CORBA осуществляется следующее:

• для поддержки сценария взаимодействия клиентский и серверный компоненты дополняются избыточным кодом, не связанным с реализацией логики приложения;

• среда обслуживания коммуникаций и дополнительный код в клиентском и серверном компонентах привязываются к данной технологии;

• существенно усложняется программирование реализации как среды взаимодействия, так и компонентов. Использование динамического связывания требует дополнительно несколько сотен операторов;

• процесс взаимодействия занимает около 130 мс, что в ~ 40 раз медленнее эквивалентного вызова при статическом связывании [8].

Все это ведет к ухудшению условий повторного использования кода, и это является решающим доводом против связывания по технологии CORBA

в САЭ. Отрицательная оценка своего опыта применения CORBA при разработке САЭ дана также в работе [9].

В связи с этим было принято решение о выполнении разработки распределенной среды обеспечения взаимодействия компонентов DiCME, учитывающей специфику задач автоматизации экспериментальных исследований.

Рассмотрим возможности организации связи компонентов, которые возникают в среде, предназначенной для выполнения эксперимента.

2.2. Интерфейсы и логика взаимодействия компонентов САЭ. Для двух групп компонентов (базовых и вспомогательных) можно определить (и фиксировать их состав) варианты интерфейсов. *Для базовых компонентов* достаточны интерфейсы, обеспечивающие:

- выполнение метода сервиса — передачу запроса на выполнение действия;
- передачу сигнала о завершении выполнения действия;
- публикацию информации, адресованной вспомогательной логике.

Для удаленного выполнения метода можно использовать компоненты несколькими различными способами. В CORBA используется вызов метода. Для этого необходимо произвести ряд обращений к библиотеке интерфейсов и несколько обращений к диску, и это выливается во множество машинных циклов. В данной работе вместо вызовов методов используется обмен сообщениями с последующей интерпретацией сообщения в теле сервиса. Это дает преимущество в виде относительной независимости запрашивающего и отвечающего сервисов. Web-сервисы, работа которых основана на обмене сообщениями, слабо связаны между собой и могут на основании содержания сообщений предпринимать то или иное действие (либо в результате интерпретации сообщения не предпринимать никакого).

В рамках базового состава компонентов присутствуют компоненты разных типов. Информация о типе компонента (о его функциональном назначении) может варьировать способ поиска компонента и использования интерфейса, а информация о типе сообщения — способ его интерпретации. Таким образом, взаимодействие базовых компонентов может обслуживаться одним интерфейсом, обеспечивающим передачу текстового сообщения, содержащего идентификатор адресата, информацию о типе компонента и типе сообщения. В случае удаленного вызова метода сообщение дополняется списком параметров, после завершения выполнения метода клиенту возвращается только сигнал синхронизации, разрешающий продолжение развития базовой логики. Конкретная информация о результатах выполнения метода (например, список зарегистрированных файлов и т. п.) адресуется вспомогательной логике.

В составе *вспомогательных компонентов* выделены источники информации и ее потребители. Источники вырабатывают, например, следующую информацию:

• сигнал отказа (NACK) при приеме сообщения и детализирующую информацию;

• сигнал о возникновении ошибки при выполнении метода и детализирующую информацию;

• информацию о зарегистрированных экспериментальных данных;

• периодически передаваемую информацию о состоянии контролируемых объектов;

• информацию о событии, изменении состояния;

• данные мониторинга объектов и др.

Нет необходимости, чтобы потребители информации имели явного клиента, а источники — конкретный сервис. Источники и потребители должны быть связаны по типу информации. Для вспомогательных компонентов также достаточно одного интерфейса, обеспечивающего публикацию информации с указанием ее типа, но без ожидания подтверждения ее получения или использования, однако потребитель может инициировать дополнительные действия, например, для организации цепочки обратной связи.

2.3. Внешнее связывание базовых компонентов. В САЭ логику эксперимента (базовую логику) в автоматическом режиме работы выполняет программа управления экспериментом. Модель этой программы проста, она передает управление для выполнения работы двум процессам, один из которых обеспечивает формирование состояния установки (условий эксперимента), а другой — регистрацию экспериментальных данных, и осуществляет их чередование.

Таким образом, программе управления требуются: описание нужного состояния установки и от каждого процесса — сигнал завершения работы, который будет инициировать следующие действия: устанавливать нужные состояния экспериментальной установки и выполнять регистрацию данных в этих условиях, — и именно ей нужно обеспечить доступ к компонентам, которые выполнят эти действия.

САЭ является конечным автоматом, список состояний которого описан в задании на эксперимент [4]. В описании состояния присутствуют идентификаторы и названия компонентов, а также списки параметров. Такая строка несет информацию о каждом используемом сервисе, достаточную для его поиска и связывания с ним всегда одного и того же компонента — программы управления экспериментом; список параметров может быть передан сервису для интерпретации и выполнения действия. *Назовем этот метод связывания внешним*, поскольку необходимая для связывания и параметризации действия информация предоставляется средствами, внешними по отношению к программе управления экспериментом и среде обеспечения взаимодействия DiCME.

2.4. Связывание компонентов вспомогательной логики. Наиболее существенным отличием способа реализации вспомогательной логики от базовой

является необходимость передать информацию нескольким процессам, состав которых, вообще говоря, источнику информации неизвестен. Можно решить эту задачу несколькими способами, при которых:

1) каждый потребитель периодически опрашивает источник информации и получает ее по готовности (поллинг);

2) источник информации использует широковещательное сообщение, потребитель опознает нужное сообщение и инициирует взаимодействие;

3) потребитель однократно декларирует интерес к информации определенного типа, после чего специальный сервис обслуживает всех «подписавшихся» потребителей при появлении этой информации.

Очевидно, что поллинг не выгоден по причине неоправданного увеличения трафика. Второй и третий механизмы, работающие с использованием прерываний, по крайней мере в два раза уменьшат количество сообщений по сети. Помимо этого обеспечивается возможность параллельной работы, так как клиенту не нужно ждать ответа сервера, и несколько клиентов получают обновления одновременно. Широковещательные сообщения с использованием протокола UDP — достаточно простой способ решить задачу, однако UDP не гарантирует доставку. Потеря сообщения вспомогательным компонентом не приводит к нарушению выполнения логики эксперимента, тем не менее в данной работе выбран третий вариант, обеспечивающий минимальный трафик, гарантированную доставку сообщения и прозрачный алгоритм взаимодействия.

При таком способе связывания компонентов вспомогательной логики:

• возникает возможность динамически включать в САЭ вспомогательные компоненты на любом этапе выполнения эксперимента;

• обеспечивается полная свобода в развитии вспомогательной логики.

Возникновение источников информации нового типа (и соответствующих потребителей этой информации) не приводит к изменению среды обеспечения взаимодействия.

3. АЛГОРИТМЫ СРЕДЫ ОБСЛУЖИВАНИЯ ВЗАИМОДЕЙСТВИЯ КОМПОНЕНТОВ

Среда обслуживания взаимодействия процессов (компонент DiCME и библиотека функций) должна решать следующие задачи:

1) занесение в память (кэширование) списка активных компонентов и обновление его с заданной периодичностью;

2) автоматический поиск и подключение компонентов к системе;

3) динамическое связывание компонентов;

4) обеспечение асинхронного выполнения методов сервиса;

5) передачу синхронизирующих сообщений о завершении выполнения вызванного метода;

6) публикацию информации различного типа: о зарегистрированных экспериментальных данных, возникших ошибках, отказах и др.;

7) регистрацию «подписчиков» на предоставление публикуемой информации определенного типа;

8) передачу подписчикам информации требуемого типа по мере ее появления.

3.1. Динамическое подключение компонентов к САЭ. Обнаружение компонентов для динамического объединения в САЭ выполняется при помощи блока, реализующего протокол SLP (Service Location Protocol). Данный блок содержит маяк, который по запросу компонента рассылает multicast-сообщения, содержащие идентификатор (GUID), тип компонента (например, DAQ, CONDITION и др.) и другую информацию (опционально). Данная информация кэшируется соответствующей функцией блока SLP. Компоненту предоставлен интерфейс, при помощи которого он может получить список типов и идентификаторов сервисов, доступных в данный момент.

Блок SLP периодически опрашивает локальную сеть при помощи multicast-сообщений с целью обновления кэша.

В зависимости от требуемого режима работы, выполняется настройка блока SLP с помощью ряда параметров. Наиболее важные из них:

- интервал обновления кэша компонентов;
- частота включения маяка;
- время, в течение которого компонент считается активным (online);
- включение режима форсированного поиска: если запрошенный компонент не числится в реестре активных, будет выполнен поиск его по сети и др.

Как правило, поиск нужного компонента идет либо по заранее известному ID (как в случае с поиском компонентов, перечисленных в задании на эксперимент), либо по его типу (например, поиск программой управления экспериментом подсистемы регистрации данных).

3.2. Механизм обслуживания базовой логики. Для управления экспериментом оператору предоставлены команды «Start», «Pause», «Continue», «End». Команда «Start» от интерфейса пользователя поступает в компонент управления экспериментом, который реализует базовую логику, используя файл задания. Структура файла показана на рис. 1. Задание на эксперимент содержит описания всех нужных состояний установки, при которых будет выполнена регистрация данных, и обеспечивает информационную поддержку динамического связывания компонентов (программы управления экспериментом с компонентами управления окружением образца и DAQ).

Список состояний (рис. 1) разбирает программа управления экспериментом. Алгоритм работы программы при реализации базовой логики показан на рис. 2.

Программа управления экспериментом выбирает описание очередного состояния установки, расчленяет его на описания отдельных условий и передает их DiCME. После ответа всех компонентов, управляющих условиями, о завершении работы программа управления экспериментом запускает экспозицию

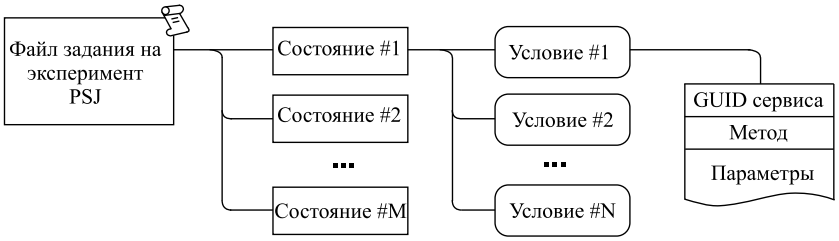

Рис. 1. Структура файла задания на эксперимент

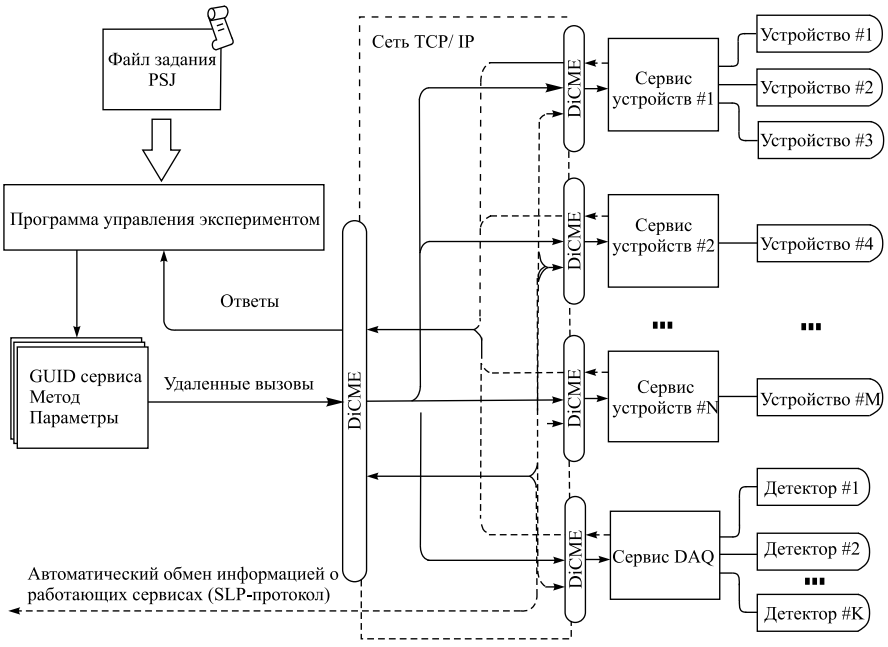

Рис. 2. Использование компонента DiCME при реализации базовой логики

данных и ожидает сигнал о завершении работы подсистемы DAQ. Эти действия выполняются для всех состояний, перечисленных в файле задания.

Программа управления и DiCME прозрачны для списка параметров. Данный метод связывания не ограничивает развитие методики эксперимента, представленной в виде описания конечного автомата, и при любых ее изменениях не затрагивает среду взаимодействия DiCME.

В случае, когда ответ исполняющего компонента содержит NACK или ERROR, среда сообщений дублирует ответ (публикует) для компонентов вспомогательной логики, сообщая детализирующие данные.

3.3. Механизм обслуживания вспомогательной логики. Любой компонент может «подписаться» на получение информации определенного типа, выполнив обращение к диспетчеру событий (через специальный метод компонента DiCME) с сообщением типа нужной информации. Потребитель может подписаться на произвольное количество типов сообщений. Его идентификатор регистрируется диспетчером событий в списках по типам.

Публикуемая информация также регистрируется диспетчером событий. Программа для каждого поступившего типа сообщения разыскивает соответствующий список (если отсутствует – заводит новый) и сохраняет сообщение. Алгоритм работы компонента DiCME при реализации вспомогательной логики показан на рис. 3. Использованный во вспомогательной логике механизм связывания назван «мягким», так как отсутствует жесткое требование найти потребителя публикуемой информации.

Диспетчер событий реализует передачу поступившего сообщения потребителям, подписавшимся на сообщения данного типа. При появлении нового сообщения оно передается всем подписавшимся потребителям. При появлении нового потребителя ему передается весь список сообщений интересующего его типа. Списки сообщений типа NACK и ERROR сохраняются в файле протокола при поступлении каждого нового сообщения.

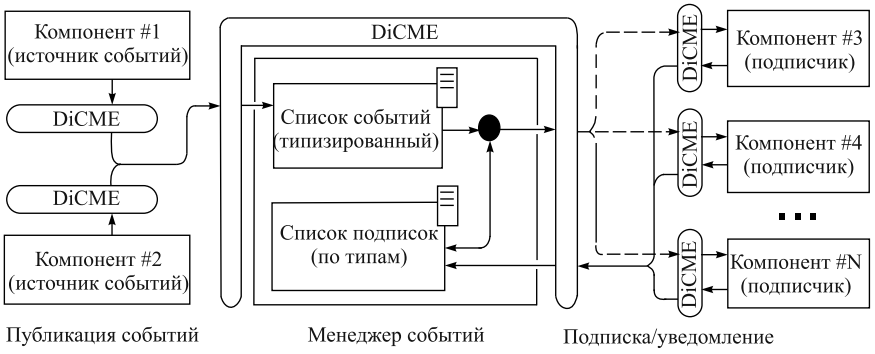

Рис. 3. Алгоритм «мягкого» связывания компонентов вспомогательной логики

4. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ DiCME

Ниже приведены тексты двух примеров подключения к DiCME и использования ее возможностей клиентом и сервисом.

```
// 1. Код клиента:
Const
    // = ID
    ClientGUID: string = 'aba97325-0312-42a5-8ef5-e9b007f367f3';
    ClientName : string = ' CliName';       // = Имя клиента
    ClientType : string = ' CliType';       // = Тип клиента
    ParametersList : string;                // = Список параметров
// ========================================
// Подключение и использование DiCME — 4 оператора:
    dicme := TDiCME.Create(ClientGUID, ClientName, ClientType);
    // Подключение обработчиков событий:
    dicme.OnError := @ClientOnError;
    dicme.OnResponse := @ClientOnResponse;
    //- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    - - - - - - - - -
    // Использование списка параметров из файла задания:
    ParametersList:='{device:"polarizer", param:"angle", value:20}';
    // JSON - формат
    // Передача сообщения:
    dicme.Execute(ServiceGUID,'SetParam', ParametersList, False, nil);
// ========================================
// Обработчики событий, принимающие ответы:
procedureClientOnResponse(constSeed : Pointer; Response : String);
begin
    // somecode  ...
end;
procedureClientOnError(constSeed : Pointer; Response : String);
begin
    // somecode  ...
end;
// ========================================
// 2. Коды сервиса:
Const
    ServiceGUID: string = 'db727048-d3db-4f27-8d5c-0eeb036f6fad';
    //  = ID сервиса
    ControllerName : string = ' Kolhida - Motors';
    //  = Имя контроллера
    ServiceType : string = ' DevicesControl';
```

```
    // = Тип компонента
// ==========================================
// Подключение DiCME – 5 операторов:
    dicme := TDiCME.Create(ServiceGUID, ControllerName, ServiceType);
    // Подключение обработчиков событий:
    dicme.OnError := @ControllerOnError;
    dicme.OnMethod := @ControllerMethod;
    // Регистрация метода передачи сообщений:
    dicme.RegisterRPCMethod('SetParam');
    // Включение маяка:
    dicme.ChangeBeaconState(True);
// ==========================================
// Обработчики событий, принимающие и обрабатывающие сообщения:
procedureControllerMethod(constMethodName : string; const
ParametersList : string;
                  isNotify : Boolean; varResult : string);
begin
// При возникновении данного события параметр ParametersList в данном
// примере будет иметь значение
{device : "polarizer", param : "angle", value : 20}
    // somecode  ...
Result :=' {status : "ok"}';        //JSON − формат
end;
procedureControllerOnError(constSeed : Pointer; Response : String);
begin
    // somecode  ...
end;
// ==========================================
```

Любой компонент может подключаться к DiCME и выступать как в роли клиента, так и в роли сервиса, используя соответствующие команды. Клиенту нет необходимости пересылать поступившие сообщения типа NACK и ERROR во вспомогательную логику: DiCME автоматически отлавливает эти сообщения и вместе с детализирующей информацией высылает их диспетчеру событий.

ЗАКЛЮЧЕНИЕ

Данная среда обслуживания взаимодействия процессов является связующим слоем между компонентами, который облегчает кодирование, компоновку и модернизацию системы в целом, повышает ее надежность, облегчает *условия повторного использования кода компонентов*.

Отличительные особенности данной работы:

• для систем, у которых базовой логикой приложения управляет один компонент (например, для САЭ), разработан и применен *новый метод динамического связывания компонентов — внешнее связывание*. Благодаря внешнему связыванию не требуется изменение среды взаимодействия или взаимодействующих компонентов при изменении базовой и вспомогательной логики приложения;

• введена *классификация компонентов* по принадлежности к базовой логике эксперимента и вспомогательной логике;

• использованы *различные механизмы обслуживания* базовой и вспомогательной логики: «жесткий» для базовой и «мягкий» для вспомогательной;

• введены универсальные интерфейсы взаимодействия компонентов, и фиксирован их состав;

• механизм RPC реализуется посредством передачи сообщения с последующей его интерпретацией в теле компонента-севера; благодаря этому *минимизирована связанность компонентов*, реализован *асинхронный механизм выполнения действий, существенно упрощено программирование компонентов и реализация DiCME*;

• поиск *сетевого хранилища данных* (NAS — Network Archive Service) реализован с помощью тех же механизмов, что и поиск других компонентов;

• разработанная среда взаимодействия позволяет *динамически изменять состав и адреса* базовых и вспомогательных компонентов, что облегчает восстановление работоспособности системы при отказах, улучшает условия эксплуатации и управления;

• сняты ограничения на развитие логики эксперимента и вспомогательной логики *без дополнительного программирования;*

• разработанная среда коммуникаций инвариантна относительно:

— изменения базовой логики приложения;

— появления источников сообщений нового типа;

— появления новых обработчиков событий.

Компонент DiCME и функции среды обслуживания взаимодействия процессов написаны в общем виде, не связаны с конкретной областью приложения и могут быть использованы в различных системах автоматизации экспериментов и в других областях применения.

Работа выполнена в соответствии с протоколом о совместных работах Лаборатории нейтронной физики им. И. М. Франка ОИЯИ и университета «Дубна».

ЛИТЕРАТУРА

1. *Драгунов Ю. Г. и др.* Модернизация импульсного исследовательского реактора ИБР-2 // АЭ. 2012. Т. 113, вып. 1. С. 29–34.
2. *Belikov O. V. et al.* Physical Startup of the First Stage of IREN Facility // Proc. of Intern. Seminar ISINN-17 (Dubna, May 27–29, 2009). Dubna, 2010. P. 10–16.
3. *Дубова Н.* На пути к SOA // Директор информационной службы. 2005. № 8. С. 12.
4. *Саламатин И. М., Саламатин К. М.* Разработка компонентной САЭ для физики низких энергий на основе использования сетевых технологий. Препринт ОИЯИ Р13-2013-74. Дубна, 2013.
5. *Gaspar C., Dönszelmann M.* DIM — A Distributed Information Management System for the DELPHI Experiment at CERN // Proc. of the 8th Conference on Real-Time Computer Applications in Nuclear, Particle and Plasma Physics, Vancouver, Canada, June, 1993.
6. *Gaspar C., Schwarz J. J.* The DELPHI Experiment Control System // Proc. of the First IEEE Intern. Conf. on Engineering of Complex Computer Control Systems, Florida, November, 1995.
7. *Franek B.* On-Line Experiment Control System for the BaBar Detector at PEP-II at SLAC // Proc. of the Intern. Conf. on Computing in High Energy Physics, Chicago, 1998.
8. http://www.rsdn.ru/article/corba/vsCORBA.xml;

   http://kunegin.com/ref3/corba4/7.htm.
9. *Flemming S. A. et al.* The Open Inspire Architecture for Control, Data Reduction and Analysis // NOBUGS-2008. 2008. Paper 134.

Получено 23 августа 2013 г.

Редактор *А. И. Петровская*

Подписано в печать 16.01.2014.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 1,06. Уч.-изд. л. 1,26. Тираж 245 экз. Заказ № 58159.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@jinr.ru
www.jinr.ru/publish/