ОБЪЕДИНЕННЫЙ ИНСТИТУТ ЯДЕРНЫХ ИСЛЕДОВАНИЙ

Саламатин Кирилл Маркович

Методы построения программных систем для автоматизации экспериментов в области спектрометрии нейтронов с использованием сетевых технологий

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата физико-математических наук

Научный руководитель:
к. ф.-м. н., доцент Сеннер Александр Евгеньевич

Научный консультант:
к. ф.-м. н., начальник отдела Зрелов Петр Валентинович

Дубна – 2015

Оглавление

Введение

Системы автоматизированного управления относятся к приоритетным направлениям науки и техники России. Технологии информационных и управляющих систем включены в Перечень критических технологий Российской Федерации, утвержденный указом Президента РФ от 7 июля 2011 г. N 899, и развитие методов построения этих систем актуально.

Область интересов автора – разработка методов построения программного обеспечения (ПО) управляющих систем автоматизации экспериментов (САЭ) в области спектрометрии нейтронов.

Спектрометрия нейтронов является интенсивно развивающейся областью физики. Нейтронные исследовательские центры работают практически во всех развитых странах мира. Размеры этих центров различаются, количество нейтронных каналов варьируется от ~10 до нескольких десятков, примерно столько же используется различных спектрометров [1,2]. Исследования ведутся в области ядерной физики, физики конденсированных сред, кристаллографии, медицины и др.

Программное обеспечение САЭ для исследований в этой научной области разрабатываются не для одиночных уникальных экспериментов, а для серии исследовательских работ. Анализ регистрируемых в эксперименте спектров позволяет получить информацию о характере взаимодействия нейтронов с ядрами или исследуемыми структурами материалов в зависимости от условий регистрации – температуры, толщины мишени, наличия или отсутствия поляризации нейтронов и ядер и др. Состав условий и способ регистрации экспериментальных данных составляют методику конкретного эксперимента. Исследовательская работа, рано или поздно, подразумевает изменение методики дальнейших работ, обусловленное полученными результатами, и после завершения анализа полученных экспериментальных данных, как правило, возникает план новых исследований, что приводит к неоднократным изменениям методики экспериментов за время жизни САЭ. Если возможность

формирования условий регистрации данных для новой методики эксперимента не была предусмотрена заранее в конструкции экспериментальной установки, то возникает необходимость модификации состава аппаратного и программного обеспечения САЭ. При этом для исследовательских организаций чрезвычайно актуальной проблемой становятся сроки разработки.

Эффективность процесса исследования существенно зависит от уровня автоматизации, обеспечиваемого используемыми средствами получения экспериментальных данных, а также от сроков их создания и модификации. В настоящее время программное обеспечение систем автоматизации экспериментов представляет собой сложную систему, работающую на сети ЭВМ. Для разработки ПО таких САЭ созданы и продолжают развиваться мощные системы программирования, специальные средства построения систем управления промышленным и экспериментальным оборудованием [3], системы SCADA и др. Однако сроки разработки программного обеспечения САЭ в ряде случаев не соответствуют современному уровню развития средств вычислительной техники и технологии программирования. В работе [4] была дана характеристика состояния дел с разработкой средств автоматизации экспериментальных исследований в СССР по материалам опроса 40 исследовательских организаций СССР. Оказалось, что в 200 случаях системы автоматизации создавалось практически независимо, причем на разработку каждой зачастую тратилось до двух-трех лет. За прошедшие более 25 лет со времени публикации работы [4] ситуация мало изменилась. В работе [2] указано, что для уже работающих спектрометров при переходе к использованию ОС LINUX на модификацию программного обеспечения 7 систем с использованием ранее освоенной разработчиками системы EPICS [3] израсходовано ~6 месяцев на систему. Такие сроки приводят к простою дорогостоящего оборудования, задержкам и снижению эффективность работы исследователей и не адекватны современному уровню развития инструментальных средств и технологии программирования.

Диссертация посвящена разработке методов построения ПО САЭ для спектрометрии нейтронов, которые существенного снижают затраты на создание и модификацию такого ПО, повышают надежность и эффективность работы САЭ при выполнении экспериментальных исследований. Практическая работа автора связана с построением программных систем для автоматизации спектрометрии нейтронов на исследовательской ядерной установке ИБР-2 и ускорителе ИРЕН, поэтому разработка методов и ПО САЭ иллюстрируются примерами из области спектрометрии. Однако решения носят общий характер и могут использоваться в других проблемных областях.

Современные спектрометры включают десятки программно-управляемых специализированных устройств, предназначенных для формирования условий, в которых будут регистрироваться данные. На рис. 1 показан пример такой системы.

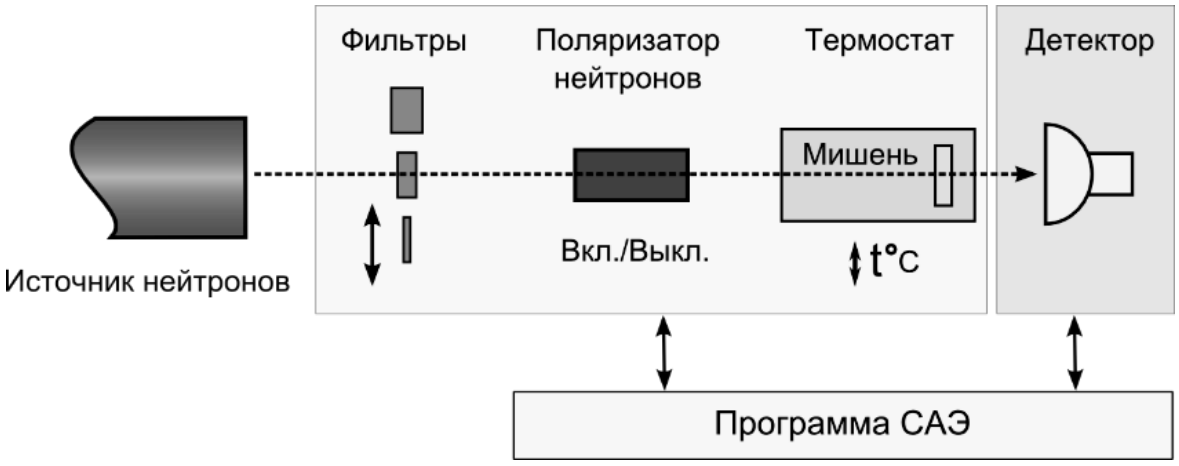

Рис. 1. Условная схема спектрометра

Использование большого количества специального оборудования (десятки…сотни), проблемы согласования интерфейсов этого оборудования с обновляющимися средствами вычислительной техники, различные платформенные зависимости компонентов системы, а также усложнение современных экспериментов привело к тому, что САЭ разрабатывается в виде распределенных сетевых систем, в которых отдельные ЭВМ выполняют роль интеллектуальных контроллеров специального оборудования, как это показано на рис. 2. В сложных случаях в системе присутствуют десятки контроллеров.

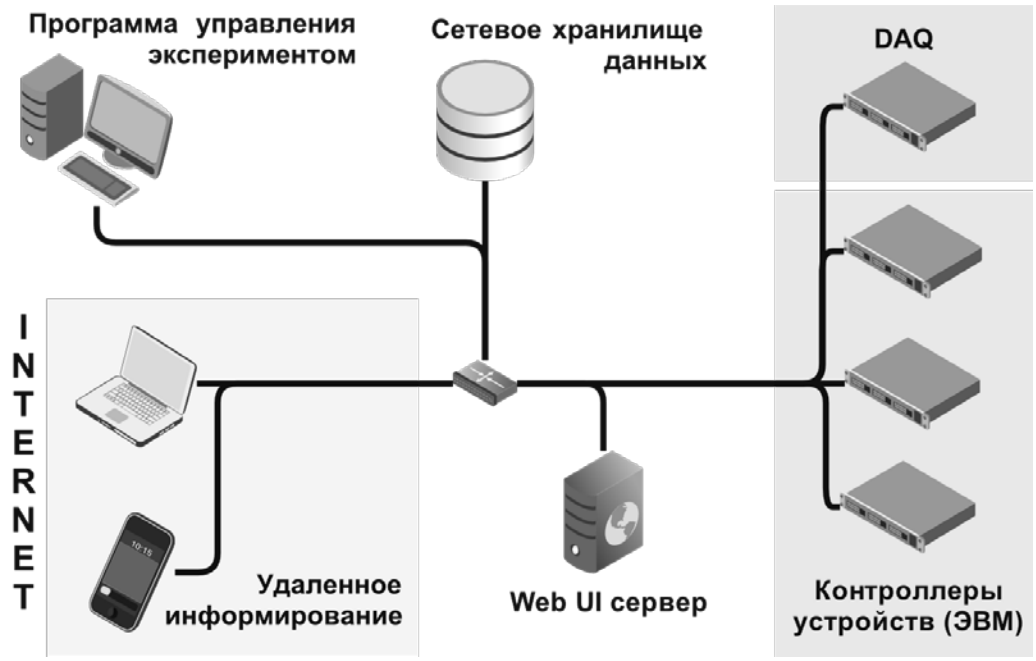

Рис. 2. Пример схемы распределенной САЭ

В составе ПО таких САЭ присутствуют компоненты, выполняющие основные операции (формирование условий, регистрация и сохранение данных) и вспомогательные (визуализация данных, их предварительная обработка, организация обратной связи и др.).

Основным критерием оценки затрат на разработку и модификацию ПО САЭ в исследовательской организации являются сроки. Известный идеолог фирмы Code Gear Дэвид Интерсаймон, как и авторы работы [4], в число существующих сегодня проблем включил слабое взаимодействие между командами разработчиков, а также отсутствие стратегии повторного использования программного обеспечения [5] за пределами одного проекта – в других экспериментах и разных САЭ. Такое повторное использование программ является основной методологией, которая применяется для сокращения затрат труда при разработке сложных систем. В работе [6] рассмотрены различные варианты методологии повторного использования – использование исходного текста программ, программных компонентов, разработка моделей, шаблонов и др. Цель применения этой методологии – разработка линейки продуктов (product line engineering – PLE). PLE – это определение функциональных возможностей программной системы, которые будут меняться от одной системы к другой, и использование этих

возможностей в конкретных вариантах каждой, так, чтобы в конечном итоге иметь один набор программных компонентов для нескольких вариантов систем. Использование программных компонентов без изменения за пределами одного проекта чрезвычайно затруднительно, если нет разработанной структуры ПО САЭ и правил построения программной системы, а также программного обеспечения, обеспечивающего разработку и объединение компонентов в систему, и для распределенного ПО САЭ эта задача не решена. Диссертантом поставлена для спектрометрии нейтронов и решена с использованием сетевых технологий задача построения распределенного ПО САЭ из программных компонентов, которые могут использоваться в различных экспериментах и системах без изменения остальных программных составляющих САЭ. Далее в тексте такие компоненты будут называться унифицированными.

В процессе работы автором выявлены два основных источника, затрудняющие разработку унифицированных компонентов – связанность компонентов и изменение методики исследования. Рассмотрим причины потери возможности унификации компонентов для использования их в разных экспериментах и ПО САЭ без изменения.

1. Связанность определяет состав и объем информационных потоков между компонентами. Связывание компонентов может быть статическим и динамическим. При статическом связывании необходимая для связывания информация (например, сетевые адреса взаимодействующих компонентов, информация об интерфейсах) тем или иным способом фиксируется до начала работы САЭ. Статическое связывание обеспечивает наибольшую скорость выполнения взаимодействия, но при этом связь между компонентами системы становится более тесной. В этом случае сбой одного из компонентов в процессе эксплуатации САЭ с большей вероятностью приводит к потере работоспособности всей системы, и даже небольшие изменения могут потребовать привлечения программистов для модификации нескольких компонентов. При динамическом связывании необходимая для обеспечения взаимодействия компонентов информация вырабатывается в процессе работы

САЭ, и именно этот вариант необходимо использовать при построении ПО САЭ – систем с часто изменяемой логикой работы.

2. Изменение методики эксперимента. Если новая методика исследования требует использовать оборудование, не предусмотренное заранее в архитектуре экспериментальной установки и программах, это приведет также к изменению состава используемых компонентов и необходимости отредактировать часть ранее использовавшихся. При этом редактируется и компонент, управляющий последовательностью выполнения операций в эксперименте, т.е. методикой эксперимента. Если объединение компонентов в систему выполнялось транслятором, то эту операцию приходится выполнять заново. Чаще для упрощения процесса модификации ПО САЭ используется специальный язык, интерпретатор языка включается в состав ПО САЭ, состав и последовательность выполнения операций в эксперименте (скрипт) описывается на этом языке списком вызываемых процедур и значений параметров, а адреса процедур, компонентов и другая информация помещается в конфигурационные файлы или передается через списки параметров. Оба эти варианта вводят статическую связанность компонентов, поэтому автором разработан альтернативный вариант способа представления методики исследования и управления ее выполнением, устраняющий эту проблему.

Анализ тенденций развития ПО САЭ по опубликованным материалам позволил сформулировать концепцию ПО САЭ, которая включает следующие положения [7]:

- распределенное ПО САЭ собирается из взаимодействующих унифицированных, функционально законченных компонентов. Компоненты представляются в исполняемом формате, доступ к процедурам, реализующим функциональность компонента, определяется их внешним интерфейсом;
- каждый компонент ПО САЭ инвариантен относительно изменений методики исследования;

- методика эксперимента (и задание на эксперимент) описывается с помощью унифицированной диалоговой подсистемы описания методики. Подсистема не изменяется при изменении конфигурации спектрометра;
- компоненты автоматически объединяются в ПО САЭ в соответствии заданием на эксперимент;
- перенос программных компонентов на другие ЭВМ в пределах локальной сети не разрушает систему, не приводит к изменению задания на эксперимент или перекомпиляции программных составляющих ПО САЭ;
- процессом выполнения основных операций во время эксперимента управляет унифицированная управляющая программа, алгоритм которой инвариантен относительно изменений методики исследования;
- эксперимент выполняется в автоматическом режиме с возможностью перейти в диалоговый;
- возможна передача функций сопровождения ПО САЭ пользователям.

Указанные выше проблемы, обусловленные связанностью компонентов и изменчивостью методики эксперимента, взаимосвязаны, и для поиска путей устранения их влияния диссертантом выполнен совместный анализ известных сетевых технологий построения распределенных программных систем с изменяемой логикой и особенностей работы компонентов в ПО САЭ. В первых двух главах диссертации приведены результаты этого анализа.

ЦЕЛЬ ДИССЕРТАЦИИ состоит в сокращении сроков создания и модификации ПО САЭ, повышении эффективности процессов регистрации и обработки экспериментальных данных, что в итоге способствует повышению эффективности работы исследователей.

ОБЪЕКТ И ПРЕДМЕТ ИССЛЕДОВАНИЯ. Объектом исследования являются распределенные программные системы автоматизации экспериментов в области спектрометрии нейтронов. Предметом исследования является: сетевые технологии построения таких систем из унифицированных компонентов; методы унификации программных компонентов САЭ; сетевые технологии обеспечения взаимодействия компонентов в распределенных

системах; методы управления работой ПО САЭ, являющегося специализированным распределенным пакетом прикладных программ.

НАПРАВЛЕНИЯ ИССЛЕДОВАНИЙ:

- анализ приемов и средств унификации программ и программных комплексов с целью сокращения сроков их разработки и повышения надежности функционирования;
- анализ известных способов объединения компонентов в распределенную сетевую систему и их динамического связывания с целью выбора оптимальной архитектуры ПО САЭ
- исследование способов описания методики получения экспериментальных данных с целью унификации программ управления работой устройств, формирующих условия регистрации данных;
- анализ популярных сетевых технологий разработки распределенных программных систем с целью выбора вариантов для использования в ПО САЭ;
- анализ и классификация функционального состава и способов взаимодействия компонентов ПО САЭ с целью разработки методов их унификации, использования в различных экспериментах без изменения и оптимизации схемы их взаимодействия в ПО САЭ.

НА ЗАЩИТУ ВЫНОСЯТСЯ:

- Выводы и структура ПО САЭ, полученные на основании анализа и классификации функционального состава и способа взаимодействия компонентов в ПО различных САЭ [7]. Эти результаты существенно упростили схему и алгоритмы взаимодействия компонентов и обеспечили дополнительные удобства в работе пользователей.
- Метод автоматической компоновки распределенного ПО САЭ. Автоматизация компоновки позволила сократить сроки настройки ПО САЭ и выполнять эту работу без привлечения программистов [10].
- Метод динамического связывания, основанный на использовании идентификаторов компонентов, учете особенностей процесса управления

экспериментом и использовании открытых сетевых технологий [10]. Разработанный метод существенно упростил структуру и алгоритмы ПО САЭ и предоставил свободу в расширении состава компонентов, выполняющих основные и вспомогательные операции, без изменения существующих.

- Метод управления процессом получения экспериментальных данных, основанный на использовании описания методики эксперимента, формируемого специальной унифицированной подсистемой. Использование этой подсистемы не требует от экспериментатора навыков программирования [11, 12, 13].

НАУЧНАЯ НОВИЗНА работы заключается в следующем:

1. Предложена новая структура ПО САЭ, включающая разные дисциплины выполнения основных и вспомогательных операций, основанная на результатах классификации по назначению и способу взаимодействия компонентов в ПО САЭ. Выводы, полученные на основании этой классификации [7,13,15], следующие:

- динамическое связывание компонентов для удаленного вызова процедур необходимо только для выполнения основных функций ПО САЭ;
- результат удаленного вызова процедур должен содержать: 1) обязательный сигнал завершения работы, адресуемый вызывающей программе, и 2) детализирующую информацию, адресуемую вспомогательным функциям;
- реализация вспомогательных функций ПО САЭ требует специальной дисциплины динамического связывания компонентов, учитывающей спонтанный характер возникновения запросов на такие операции и независимость основных функций (в штатных условиях) от результатов выполнения вспомогательных операций.

Эти выводы и предложенная структура ПО САЭ позволили существенно упростить алгоритмы и унифицировать средства межкомпонентного взаимодействия и прикладные компоненты.

2. Предложен метод автоматической компоновки распределенного ПО САЭ в соответствии с заданием на эксперимент в условиях изменения задания при переходе от одного эксперимента к другому. Метод основан на использовании сетевого протокола поиска компонентов SLP и адресации на основе идентификаторов, вместо традиционного использования конфигурационных файлов с сетевыми адресами компонентов [10,15]. Автоматизация компоновки позволила существенно сократить сроки настройки ПО САЭ и выполнять эту работу без привлечения программистов.

3. Предложен метод управления составом основных операций в эксперименте программой в соответствии с описанием методики получения экспериментальных данных списком условий их регистрации вместо традиционно используемого списка вызовов процедур [11,13]. Этот метод позволил унифицировать программу управления экспериментом и средства межкомпонентного взаимодействия.

4. Предложен метод динамического связывания компонентов в распределенном сетевом ПО САЭ, использующий разные дисциплины связывания при выполнении основных и вспомогательных операций, и унифицированные средства обслуживания межкомпонентного взаимодействия [10,15,16], что предоставило свободу в развитии состава основных и вспомогательных операций ПО САЭ без изменения других компонентов и существенно упростило алгоритмы взаимодействия компонентов. В отличие от методов динамического связывания компонентов, используемых в технологиях DCOM, CORBA, Ice и др., в разработанном методе осуществлено следующее:

- вместо сетевого адреса компонента используется его уникальный идентификатор, что минимизирует связанность компонентов;
- параметры удаленного выполнения процедуры интерпретируются в самом компоненте, а результат ее выполнения представляется двумя сообщениями: 1) сигналом завершения работы, возвращаемым вызывающему компоненту, и 2) необязательной детализирующей информацией для вспомогательных компонентов.

Благодаря этому устранен диалог между компонентами, используемый в известных технологиях для настройки удаленного вызова процедуры.

Разработанный метод динамического связывания предоставил свободу в развитии состава основных и вспомогательных операций ПО САЭ без изменения других компонентов ПО САЭ и существенно упростил алгоритмы взаимодействия компонентов. Предложенный метод исключил диалог между компонентами, необходимый в известных технологиях для настройки взаимодействия компонентов.

Отличие разработанных в диссертации методов объединения компонентов в систему и динамического связывания компонентов для удаленного выполнения процедур от методов, реализованных в интерпретирующей системе ИС-2 [17], заключается в следующем:

- динамическое связывание и объединение компонентов в систему развиты для использования в распределенном ПО САЭ;
- реализован асинхронный режим исполнения процедур;
- используется динамически составляемый реестр активных компонентов вместо фиксированной таблицы характеристик, используемой в ИС-2;
- результат выполнения процедуры представляют два сообщения: 1) обязательное сообщение о завершении работы процедуры, адресуемое вызывающей программе, и 2) необязательное сообщение, адресуемое вспомогательным программам.

ПРАКТИЧЕСКАЯ ЦЕННОСТЬ. Основные положения, выводы и рекомендации доведены до практической реализации и выдержали проверку в эксплуатации. Результаты данной работы нашли практическое применение при разработке ПО нескольких систем автоматизации экспериментов на ускорителе ИРЕН [18-21] и исследовательской ядерной установке ИБР-2 [22, 23]. Разработанные системы используются в исследованиях, проводимых в ЛНФ ОИЯИ (для установок КОЛХИДА, АУРА, для снятия характеристик источников нейтронов ИБР-2 и ИРЕН), ЛЯП ОИЯИ (система для настройки

детектора для эксперимента Mu2e в Фермилаб), ИЯИ, Троицк (система для исследования несохранения четности при дифракции нейтронов). Имеются заключения о внедрении. С использованием этих систем получены важные физические результаты (например, [22, 23] и др.).

Разработанные алгоритмы могут иметь более широкое применение: подсистема подготовки задания, программа управления экспериментом, средства обеспечения взаимодействия компонентов, Web-интерфейс пользователя, подсистема экспресс анализа и др. применимы в других проблемных областях (например, технологические процессы). Результаты данной работы могут также быть использованы в учебных курсах университета «Дубна».

Работа выполнялась в соответствии с проблемно-тематическими планами ЛНФ ОИЯИ и протоколом о выполнении совместной научно-исследовательской работы ЛНФ ОИЯИ и университетом “Дубна” [24].

РЕАЛИЗАЦИЯ РЕЗУЛЬТАТОВ. Разработанные диссертантом методы реализованы в программах и системах. Прогнозированные характеристики подтверждены численными расчетами и проверкой в экспериментах на источниках нейтронов ИБР-2 и ИРЕН в ОИЯИ [18-23].

Разработанные системы используются в исследованиях, проводимых в: ЛНФ ОИЯИ, ЛЯП ОИЯИ, ИЯИ г. Троицк.

АПРОБАЦИЯ РАБОТЫ. Отдельные разделы работы были представлены на Международной конференции “Ядро 2007” (Воронеж) [12]; международной конференции “IEФ’2011” (Украина, Ужгород) [20]; “ICANS XXI” (в Мито, Ибараки, Япония) [13]; международных семинарах в Дубне “ISINN-17” [20], “ISINN-19” [21], “ISINN-21” [16], “ISINN-22” [13]. Основные результаты работы докладывались на научных семинарах ЛНФ ОИЯИ, ЛИТ ОИЯИ, ИППИ РАН, университета “Дубна”.

ПУБЛИКАЦИИ И ЛИЧНЫЙ ВКЛАД АВТОРА. По теме диссертации опубликовано 11 печатных работ, в том числе в рецензируемых научных журналах представлены 4 статьи [7,10,13,19]. Без соавторов опубликовано 5

работ [10,11,14-16]. В [19-21] диссертанту принадлежит разработка алгоритмов и их программная реализация.

Вклад соискателя является определяющим в работах, положенных в основу диссертации. Результаты, выносимые на защиту, получены лично автором.

СТРУКТУРА И ОБЪЕМ ДИССЕРТАЦИИ. Диссертационная работа состоит из введения, 5 глав и заключения. Работа изложена на 126 страницах машинописного текста, содержит 17 рисунков, 4 таблицы, 3 приложения. Список литературы включает 82 наименования.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении сформулирована постановка задачи и обосновывается ее актуальность. Показаны источники и причины, препятствующие унификации компонентов ПО САЭ: 1) связанность компонентов, которая при изменении одного компонента приводит к необходимости менять другие части ПО САЭ и 2) изменение методики исследования – при реализации новой методики может потребоваться изменить состав оборудования экспериментальной установки, состав компонентов ПО САЭ, программу управления экспериментом и др. Сформулирована концепция ПО САЭ, реализация которой отвечает поставленной в диссертации цели.

Методы построения распределенных программных систем наиболее полно представлены в сетевых технологиях. В главе 1 выполнен анализ сетевых технологий с целью выбора вариантов для использования при разработке ПО САЭ.

В первой главе приведен обзор и выполнен анализ популярных сетевых технологий разработки распределенных систем. На основании анализа сделан выбор (с учетом специфики задач автоматизации экспериментов) структуры ПО и вариантов технологий для использования в ПО САЭ [14]. Рассмотрены методы удаленного вызова процедур RPC, технологии RMI, CORBA, DCOM, Ice, средства разметки сообщений XML, JSON, SOAP, технологии поиска

компонентов, концепция SOA и др. Показано, что популярные технологии построения распределенных программных систем (например, DCOM, CORBA и др.) при реализации динамического связывания компонентов используют алгоритмы, которые являются избыточно сложными для использования в САЭ.

Выводы, сделанные в главе 1 на основании проведенного анализа, следующие:

- необходимо выполнить разработку метода динамического связывания компонентов, учитывающего особенности решаемых задач;
- при разработке механизма динамического связывания компонентов и компоновки ПО САЭ могут быть использованы открытые сетевые технологии форматирования и сериализации сообщений (JSON), поиска компонентов (SLP) и др.;
- архитектуру ПО САЭ следует принять в соответствии с принципами SOA, включая механизм передачи сообщений, функции формирования реестра компонентов, брокер сообщений, прикладные компоненты и некоторые специальные службы.

С целью получить дополнительную информацию для оптимизации разработки метода динамического связывания компонентов и для решения проблем унификации компонентов, связанных с изменением методики исследования, в следующей главе выполнен анализ особенностей работы компонентов в ПО САЭ.

В главе 2. выполнены анализ и классификация функционального состава и способов взаимодействия компонентов ПО САЭ. В проанализированных опубликованных работах методика исследования определяется разными способами – фиксированными фрагментами программ системы или интерпретируемыми скриптами с перечислением названий и параметров вызываемых внешних процедур. Эти варианты вносят связанность компонентов, а попытка реализовать не учтенную ранее методику исследования приводит к изменениям готовых программ, что затрудняет или исключает возможность унификации компонентов. В литературе не были найдены способ

управления экспериментом и средства описания новой (не учтенной при разработке ПО САЭ) методики, не требующие изменения компонентов, скриптов или других составных частей ПО САЭ. Чтобы получить информацию для разработки нового метода динамического связывания компонентов, способа управления последовательностью операций в эксперименте и подсистемы описания методики исследования с нужными свойствами, был выполнен анализ состава и способа взаимодействия разных функциональных компонентов ПО САЭ.

Основные выводы, сделанные на основании этого анализа, следующие [7]:

- динамическое связывание компонентов для удаленного вызова процедур необходимо только в основной логике ПО САЭ;
- результат удаленного вызова процедур должен содержать два типа информации: 1) обязательный сигнал завершения работы, адресуемый вызывающей программе (программе управления экспериментом или интерфейсу пользователя) и 2) детализирующую информацию (список файлов зарегистрированных данных, описание состояния управляемого объекта, диагностическое сообщение и др.), адресуемую вспомогательным функциям;
- реализация вспомогательной логики требует специальной дисциплины связывания компонентов, учитывающей спонтанный характер возникновения запросов на такие операции и независимость основной логики (в штатных условиях) от результатов выполнения вспомогательных операций.

Эти выводы положены в основу разработки специального метода динамического связывания компонентов и новой структуры ПО САЭ, в которой использованы разные механизмы взаимодействия компонентов для выполнения основных и вспомогательных функций.

Исследования, выполненные в гл.1 и 2, показывают, что для достижения поставленной в диссертации цели ключевыми являются решения следующих задач:

- разработка метода управления экспериментом;
- разработка универсальной подсистемы описания методики исследования и составления задания на эксперимент;
- разработка специальных средств для выполнения поиска компонентов, объединения их в систему и их динамического связывания для удаленного выполнения процедур.

В главе 3 рассмотрены известные способы управления экспериментом [11,13]. Таких способов два:

- представление программы управления на языке программирования фиксированной группой процедур, каждая из которых реализует определенную методику исследования;
- использование в составе ПО САЭ интерпретатора, на языке которого составляется программа управления экспериментом (скрипт) в виде списка вызовов внешних процедур и значений параметров.

В обоих способах, по сути, в программу управления экспериментом (программу реализации методики эксперимента) включается описание методики списком вызовов процедур. Недостатком обоих методов является жесткая связанность компонентов. В диссертации предложен альтернативный метод управления экспериментом:

- программа управления экспериментом выполняет последовательно два процесса – формирование условий эксперимента и регистрацию данных в этих условиях;
- конкретизация функционального наполнения этих процессов (состава используемых компонентов) выполняется динамически с использованием файла задания на эксперимент, создаваемого подсистемой описания методики исследования [13].

Основные выводы в главе 3:

1. Использование разработанной подсистемы описания методики эксперимента и метода управления составом и последовательностью операций в эксперименте существенно сократило время настройки ПО САЭ для нового эксперимента и дало возможность выполнять зту работу без привлечения программистов.
2. Использование этой подсистемы [11,13] совместно с разработанными средствами межкомпонентного взаимодействия [10] обеспечило возможность автоматически компоновать ПО САЭ в соответствии с заданием на эксперимент.
3. Разработанные способ описания методики эксперимента и метод управления экспериментом исключили жесткую связанность компонентов, благодаря этому подсистема и программа управления экспериментом пригодны для использования без изменений в ПО различных САЭ.

В главе 4 описана сетевая технология межкомпонентного взаимодействия, реализуемая разработанным компонентом DiCME (Distributed Components Messaging Environment) [10,15]. Компонент DiCME обеспечивает базовую технологическую поддержку работы ПО САЭ. Для этого в состав DiCME, помимо механизма передачи сообщений, включены функции формирования реестра компонентов, брокер сообщений и некоторые другие службы.

Одна из решающих характеристик средств межкомпонентного взаимодействия – способ связывания компонентов. Связывание необходимо программе управления экспериментом для удаленного выполнения процедур управления условиями эксперимента и регистрацией данных. Для динамического связывания информацию дает файл задания, в который программа подготовки задания автоматически заносит идентификатор и тип компонента, определенные в паспорте компонента. Процедура передачи

сообщений фиксирована, эта процедура обеспечивает передачу всех сообщений в рамках ПО САЭ.

Разработанный метод динамического связывания использует для адресации компонентов их идентификаторы, вместо сетевых адресов, используемых в других технологиях. Необходимая для связывания и параметризации действия информация вырабатывается средствами, внешними по отношению к программе управления экспериментом, компонентам и средствам обеспечения межкомпонентного взаимодействия DiCME. В отличие от технологии CORBA и др., использование данного метода динамического связывания полностью устранило необходимость подготовительного диалога между компонентами для настройки удаленного выполнения процедур. Данный метод связывания не ограничивает развитие методики исследования, и ее изменения не затрагивают программу управления экспериментом и средства межкомпонентного взаимодействия DiCME, т.к. программа управления и DiCME прозрачны для списка параметров.

Для связывания вспомогательных компонентов выбран вариант алгоритма “подписки”, при котором компонент-потребитель однократно декларирует интерес к информации определенного типа, после чего специальный компонент обслуживает всех “подписавшихся” потребителей при появлении этой информации.

Основные выводы в главе 4. Предложен метод динамического связывания компонентов для удаленного выполнения процедур в распределенном ПО САЭ и унифицированные средства обслуживания межкомпонентного взаимодействия. В отличие от методов динамического связывания компонентов, используемых в технологиях DCOM, CORBA, Ice и др., предложено ввести в структуру ПО САЭ одновременно две дисциплины связывания: связывание компонентов с обязательной доставкой сообщений (вызов процедуры и сигнал завершения ее работы) при выполнении основных функций и связывание по “подписке” – для вспомогательных функций, когда допустимо отсутствие компонента-получателя. Такая структура более полно (и

не избыточно) соответствует способу использования системы в эксперименте. Разработанный метод динамического связывания и предложенная структура предоставили свободу в развитии состава основных и вспомогательных операций ПО САЭ без изменения других компонентов и существенно упростили алгоритмы взаимодействия компонентов благодаря исключению диалога между компонентами, используемого в технологиях CORBA и др. для настройки удаленного вызова процедур.

Использование в структуре ПО САЭ разных дисциплин выполнения основных и вспомогательных операций и разработанных средств обеспечения взаимодействия компонентов:

- снимает ограничения на расширение состава выполняемых во время эксперимента операций без дополнительного программирования;
- позволяет динамически изменять состав и размещение в сети основных и вспомогательных компонентов, что облегчает восстановление работоспособности системы при отказах.

Численные оценки, выполненные на модели ПО САЭ с использованием DiCME, показывают: 1) в синхронном и асинхронном режимах управления устройствами сетевые задержки по крайней мере на порядок меньше времени работы исполняющих устройств; 2) загрузка процессора обслуживанием межкомпонентного взаимодействия незначительна, и трафик, вызванный взаимодействием компонентов, не ухудшает пропускную способность сети.

Разработанные средства обслуживания взаимодействия компонентов инвариантны относительно

- изменения состава основных операций ПО САЭ;
- добавления к ПО САЭ источников сообщений нового типа;
- добавления к ПО САЭ новых обработчиков событий.

В главе 5 представлено краткое описание структуры и алгоритмов Web-интерфейса пользователя, программы управления выполнением эксперимента, структуры компонентов управления окружением образца. Каждый компонент ПО САЭ содержит:

- группу команд (4–6 операторов) подключения к средствам обеспечения межкомпонентного взаимодействия DiCME при запуске компонента;
- интерпретатор описания условия, если компонент содержит несколько процедур, используемых другими компонентами;
- процедуры, реализующие предусмотренную функциональность.

В главе 5 получены следующие результаты:

Предложен алгоритм универсального Web-интерфейса для управления выполнением основной и вспомогательной логики ПО САЭ и средства быстрого удаленного информирования пользователя о критических событиях в работе системы.

Разработаны алгоритмы программы управления экспериментом, обеспечивающие использование ее в прецизионных экспериментах и в отладочных работах с различными заданиями и в разных системах.

Программистам предоставлена свобода выбора оптимального варианта декларации процедур в компонентах управления условиями регистрации данных. Эти декларации определяют протокол вызова процедур. Подсистема описания методики исследования обеспечивает использование конкретного варианта вызова процедуры в соответствии с документацией из БД, а программа управления экспериментом и среда обслуживания взаимодействия компонентов прозрачны для параметров вызова процедуры.

В заключении перечислены основные результаты.

Исследования методов разработки САЭ, обеспечивающих гибкость программ, возможность использования компонентов ПО САЭ в других системах без изменения, сокращающих сроки разработки и облегчающих модификацию системы пользователями, были начаты автором ряд лет назад [12]. Диссертантом выполнены необходимые промежуточные исследования и проверка решений частных проблем, и, в конечном счете, сформулированы изложенная выше постановка задачи и способ ее решения. В следующих главах описывается проделанная работа.

Глава 1. Анализ сетевых технологий и выбор для использования при построении системы автоматизации экспериментов в области спектрометрии нейтронов

1.1. Цель данного обзора

Выбор архитектуры ПО САЭ, способа ее компоновки и построения среды взаимодействия являются ответственейшими этапами разработки распределенной системы. Развитие процессов интеграции прошло ряд этапов – использование специализированных или пользовательских интерфейсов, данных и др. На сегодняшний день наиболее эффективными являются 1) интеграция на уровне приложений (EAI – Enterprise Application Integration) и 2) интеграция на основе использования Web-сервисов [25].

Технология EAI основана на совместном использовании исполняемого кода, а не данных. Эта технология устраняет необходимость в разработке громоздких приложений. Вместо этого программы разделяются на компоненты, которые затем объединяются, используя определенные программные интерфейсы, специальные технологии и связующее программное обеспечение. В состав средств технологии EAI входят API и библиотеки, созданные для объединения программ на разных платформах, например, RPC (Remote Procedure Call) и ORB (Object Request Broker). Суть этих технологий сводится к тому, что функции одного приложении могут использоваться другими, независимо от используемой платформы [25].

Технология Web-сервисов тоже обеспечивает интеграцию на уровне приложений, но на современном уровне. В этом подходе данные и программные компоненты заключаются в некую "оболочку", чтобы к ним могли получать доступ другие приложения. В отличие от EAI, Web-сервисы используют стандартные способы выполнения интеграции, а EAI-технологии всегда разрабатывались для конкретных продуктов. Помимо этого, технология Web-сервисов с самого начала разрабатывались для использования в

распределенных средах, и Web-сервисы используют стандарты, поддерживаемые консорциумом W3C (World Wide Web). Использование Интернет, возникновение стандартной платформы взаимодействия J2EE, распространение не зависящих от платформы способов разметки сообщений (XML, JSON, …) стимулирует интерес к рассмотрению такого подхода с целью использования его для решения задач автоматизации экспериментов. Благодаря использованию сетевых технологий и соблюдению стандартов возникает реальная (подтвержденная практикой) возможность разработки программных компонентов со свойствами программно-аппаратной независимости и простоты объединения. Помимо этого, появляется возможность уменьшить неоправданные затраты на разработку собственной среды обслуживания взаимодействия [25].

Целью данной главы является анализ известных технологий удаленного выполнения процедур, динамического связывания компонентов, интеграции компонентов в систему, автоматизации настройки сетевого взаимодействия в распределенной системе и выбор (с учетом специфики задач автоматизации экспериментов) оптимальных вариантов для использования в ПО САЭ. В главе сформулирована технологическая основа для разработки программных систем автоматизации спектрометрии нейтронов с использованием сетевых технологий.

1.2. Специфика ПО систем автоматизации экспериментов

Выбор архитектуры и способа интеграции в значительной мере определяется особенностями разрабатываемого приложения. В работе [7] выполнен анализ состава и способа взаимодействия функциональных компонентов, которые могут быть использованы в ПО САЭ. На основании этого анализа сделаны выводы, из которых для данного рассмотрения существенны следующие:

- Функциональная часть ПО САЭ может быть ограничена локальной сетью. “Выход” во внешний мир (Интернет, если это разрешено

административно) требуется только для использования Web-интерфейса пользователя для удаленного управления экспериментом и публикации данных для общего доступа.

- Количество программных компонентов в ПО САЭ невелико (в пределах нескольких десятков). САЭ можно отнести к классу малых…средних информационных систем.
- Любой компонент ПО САЭ может выступать как в роли клиента, так и в роли сервера. Полный состав компонентов в САЭ можно разделить на две группы по признаку использования их в логике приложения: 1) группа основных и 2) вспомогательных компонентов.
- В составе основных присутствуют компоненты, вызывающие удаленные процедуры, и компоненты, исполняющие заказанные действия. Функциональное назначение основных компонентов ПО САЭ определено, последовательность их выполнения в автоматическом режиме (основная логика эксперимента) фиксирована и реализуется программой управления экспериментом. “Вызывающая” программа шлет команду, подлежащую исполнению (а также параметры), “исполняющая” для синхронизации работы возвращает подтверждение завершения действия и информацию для вспомогательных программ.
- 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.
- Наиболее существенной характеристикой ПО САЭ является многократное изменение логики приложения (методики исследования) в течение ее срока жизни. Изначально программа управления экспериментом (и др. компоненты) не имеют информации о составе компонентов управления условиями регистрации данных, используемых процедурах и значениях управляемых параметров. Иначе говоря, в ПО САЭ отсутствует информация о методике предстоящей работы. Поскольку ПО САЭ (в основном режиме) работает автоматически, должен быть введен

специальный механизм, с помощью которого определяется методика работы и динамически связываются компоненты, реализующие нужную функциональность – выполнение нужной последовательности процедур с конкретными значениями параметров.

Предварительные представления о структуре ПО САЭ и необходимых для ее реализации алгоритмах позволяют сформулировать общие требования к средствам коммуникации компонентов:

- соответствие международным стандартам;
- компактность, открытость, простота реализации, надежность;
- платформенная и языковая независимость;
- автоматический поиск и динамическое связывание компонентов;
- возможность удаленного выполнения процедур в синхронном и асинхронном режимах;
- возможность передачи информации одновременно нескольким потребителям; не предполагается передавать большие комплексные структуры данных;
- обеспечение безопасности.

Для выбора структуры ПО САЭ, способа интеграции и средств коммуникации компонентов рассмотрим часто используемые технологии.

1.3. Удаленные вызовы процедур

Для решения проблемы обеспечения мобильности систем на основе архитектуры клиент-сервер, существуют программные пакеты, использующие протоколы удаленного вызова процедур RPC. В случае использования этих средств обращение к процедуре компонента при удаленном вызове выглядит так же, как и обычный вызов. Средства обслуживания RPC, в которых содержится вся информация об особенностях аппаратуры локальной сети и протоколов, переводят вызов в совокупность сетевых взаимодействий [26].

Существует несколько технологий распределенных объектов, а также протоколы, обеспечивающие RPC, например: Java RMI (или RMI) [27], DCOM

[28], CORBA [29], Ice [30], SOAP (спецификация RFC-4227), XML-RPC (спецификация RFC-3529), JSON-RPC (спецификация RFC-4627) и др.

Для того, чтобы компоненты системы взаимодействовали правильно, они должны использовать протоколы и интерфейсы, в которых определены детали процесса их взаимодействия. Степень реализации компонентов не зависит от состояния программ в других частях системы. Замена отдельной функциональной части приложения не требует полной перестройки всей системы. При наличии рациональной технологии программирования возможна параллельная работа нескольких коллективов над разными частями разрабатываемого приложения или системы. В числе достоинств этих технологий [26]:

- сокращение времени разработки приложения;
- уменьшение количества ошибок;
- возможность использования программных компонентов в ПО разных САЭ;
- облегчение будущего изменения системы;
- возможность быстро создавать многофункциональные приложения, используя уже ранее созданные компоненты, что заметно снижает затраты на разработку новой системы.

Удаленные вызовы процедур используются давно. Запрос на удаленное выполнение процедуры и параметры представляется в виде документа в выбранном формате и посредством транспортного протокола передается по сети на другой компьютер, где выполняющая заказанное действие программа извлекает из документа имя процедуры, параметры и другую нужную информацию. После завершения работы процедуры сервер формирует ответ (например, возвращаемые данные), упаковывает и возвращает компьютеру, пославшему запрос. Второй идеей RPC является выполнение автоматического преобразования форматов данных для обеспечения взаимодействия процессов, которые выполняются на разнородных компьютерах. Все эти действия

выполняются средствами обслуживания взаимодействия, а не прикладной программой.

В случае выполнения RPC средствами технологий DCOM, CORBA формат обмена данными остается двоичным, работать с ним сложнее, тем более, если работа распределенной системы выполняется в такой сети, где между отдельными участками сети стоят прокси-серверы и firewall. Помимо этого, в условиях систематического изменения методики эксперимента потребность в отладке сохраняется, несмотря на принимаемые меры обеспечения надежности, поэтому из множества вариантов протоколов RPC рассмотрим технологии с текстовым форматом обмена данными XML-RPC, JSON-RPC и SOAP [31,32].

1.3.1. Протокол XML-RPC. Чтобы реализация распределенной системы не зависела от конфигурации сети и различий в платформах, компания UserLand Software Inc. разработала технологию XML-RPC. Транспортным протоколом (основным) в этой технологии является HTTP, форматом данных – XML. Это позволило снять ограничения, связанные с конфигурацией сети и маршрутом передачи пакетов, т.к. вызовы XML-RPC свободно могут проходить шлюзы, если допускается обработка HTTP-трафика.

Применение XML для форматирования данных позволило упростить реализацию программные средства разработки распределенных приложений, упростились требования к клиенту и серверу. Программы разбора текста в формате XML сейчас имеются практически во всех операционных системах и на всех алгоритмических языках программирования.

1.3.2. Протокол JSON-RPC. JSON-RPC является достаточно простым протоколом, похожим на XML-RPC. В нем определяется только набор типов данных и команд. Протокол JSON-RPC тоже работает посредством передачи запроса серверу, который расшифровывает и выполняет этот запрос. Клиентом является программа, которая вызывает удаленную процедуру. Процедуре может быть передан ряд параметров, в свою очередь процедура может вернуть клиенту данные (результат ее работы). Для удаленного вызова процедуры

может быть использован как протокол TCP/IP, так и HTTP, сериализация выполняется, используя JSON [33]. Вызов должен содержать:

- method – строку с названием вызываемой процедуры;
- params – параметры (массив объектов), которые должны быть переданы вызываемой процедуре;
- id – идентификатор любого типа; используется для установления соответствия ответа определенному запросу.

Получатель запроса должен вернуть корректно составленные ответы на все поступившие запросы. В ответе должны присутствовать:

- result – данные, возвращаемые вызванной процедурой. При возникновении ошибки возвращается null;
- error – возвращается код ошибки или значение null;
- id – идентификатор запроса, которому соответствует данный ответ.

Для случая, когда ответ не нужен, введен специальный тип запроса – notification. Для таких запросов идентификатор id опускается или ответ содержит значение null.

1.3.3. Протокол SOAP. Протокол SOAP был разработан для обеспечения взаимодействия между сервисами и Интернет-приложениями [34]. Это совместная разработка нескольких крупных компаний (Microsoft, IBM, DevelopMentor, UserLand Software, Lotus Development). Целью данной разработки было упрощение средств, с помощью которых можно построить приложение из группы компонентов, написанных на разных языках. Протокол SOAP обеспечивает возможность объединения сервисов в Интернете единым способом даже в тех случаях, когда различаются операционные системы и языки программирования.

В работе [34] дано следующее описание способа взаимодействия между клиентом и сервером по протоколу SOAP:

«

1. Приложение-клиент создает экземпляр объекта SOAPClient.

2. SOAPClient читает файлы описания процедур Web-сервиса (WSDL и Web Services Meta Language – WSML).
3. Клиентское приложение, используя возможности позднего связывания процедур объекта SOAPClient, вызывает процедуру сервиса (формирует пакет запроса и отправляет на сервер). Может быть использован любой транспортный протокол, но обычно используется HTTP.
4. Пакет принимается серверным приложением Listener (может представлять собой ISAPI приложение или ASP страницу), создает объект SOAPServer и передает ему пакет запроса.
5. SOAPServer читает описание Web-сервиса, загружает описание и пакет запроса в XML DOM деревья.
6. SOAPServer вызывает процедуру объекта (приложения), реализующего сервис.
7. Ответ сервера помещается объектом SOAPServer в пакет ответа и отправляется клиенту.
8. Объект SOAPClient разбирает принятый пакет и возвращает приложению-клиенту результаты работы сервиса или описание возникшей ошибки.

WSDL файл – это документ в формате XML, который автоматически создает SOAP Toolkit. В этом файле описываются процедуры, предоставляемые сервисом, их параметры, типы и названия, а также местонахождение Listener`a. Пакет SOAP Envelope – это XML документ, который содержит в себе запрос/ответ на выполнение процедуры. Он является почтовым конвертом, в который вложена информация. Тэг Envelope должен быть корневым элементом этого пакета. Элемент Header может отсутствовать, а Body должен быть прямым потомком элемента Envelope. В случае ошибки выполнения процедуры сервер формирует пакет с подробным описанием ошибки.
» [34].

Приведенная информация из [34] показывает, что SOAP – это несколько протоколов и стандартов. Однако, несмотря на громоздкость этого протокола, разработку можно выполнить быстро, если для выполнения разработки с

использованием SOAP существуют инструменты под обеими платформами, а главное – если можно извлечь пользу из сложностей SOAP.

SOAP может использоваться с разными протоколами прикладного уровня, например: SMTP, FTP, HTTP, HTTPS и др. Однако должны быть определены особенности его взаимодействия с каждым из используемых протоколов.

Ни одна из приведенных технологий удаленного вызова процедур не претендует на полноту. На основании сравнения технологий SOAP и XML-RPC в работе [31] сделаны следующие выводы:

- если у разрабатываемой системы сложная логика, передаются большие структуры данных, нужна детальная информация о клиенте, если требуется, чтобы за стандартом стояли крупные программные фирмы (например, Sun, Microsoft, …) – то следует использовать SOAP;
- в случае, когда данные достаточно простые и в логике системы не используются сложные команды, но приложения должны работать на разных платформах и на разных языках, – целесообразно использовать XML-RPC.

Рассмотрим наиболее распространенные технологии распределенных объектов, обеспечивающих RPC – RMI, CORBA, DCOM и Ice.

1.4. Технологии разработки распределенных систем

1.4.1. RMI. RMI является самым быстрым и простым способом разработки распределенных систем. Это хороший выбор для создания RAD-компонентов и небольших приложений на языке Java. Однако следует исключить RMI из рассмотрения по двум причинам:

- отсутствие языковой независимости;
- связанность объектов, обусловленная данной технологией.

1.4.2. Технология CORBA. Технология CORBA разрабатывается OMG (Object Managment Group, более 800 членов) с 1990-го года. CORBA является стандартом и определяет структуру взаимодействия компонентов (объектов) на уровне приложений модели OSI. Эта модель позволяет рассматривать все включаемые в разрабатываемую распределенную систему приложения, как объекты. При использовании CORBA имеется возможность на основе двухуровневой и трехуровневой архитектуры [35] строить более гибкие системы, чем клиент-сервер. Основным достоинством CORBA является межъязыковая и межплатформенная поддержка. Данная технология позволяет вызывать функции объектов, находящихся в любой точке сети так, как если бы все они были локальными объектами. На рис. 1.1 показана основная структура CORBA 2.0 ORB.

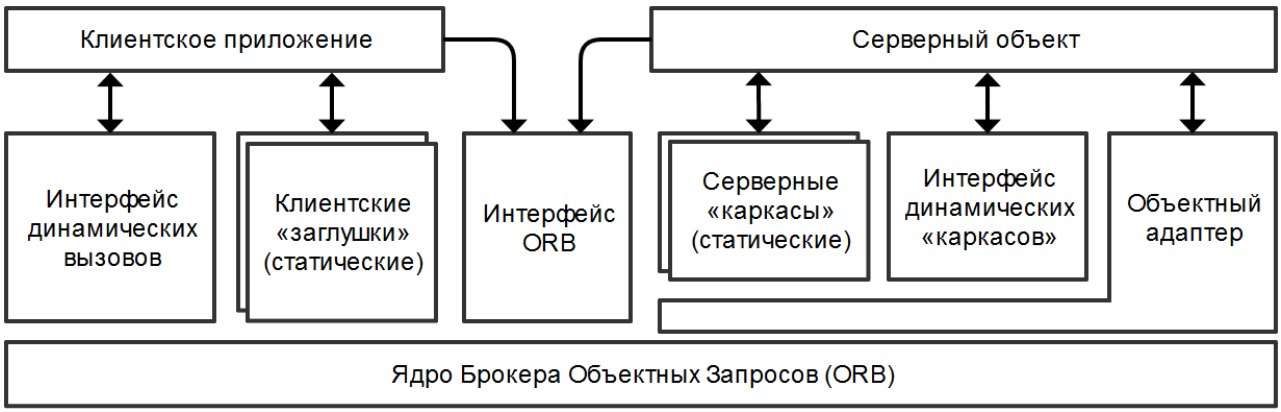

Рис. 1.1. Структура интерфейсов вызовов объектов в CORBA

Клиентские заглушки процедур (IDL Stubs) определяют способы вызова серверов. Интерфейс динамических каркасов (Dynamic Invocation Interface, DII) позволяет клиенту находить серверы и вызывать их процедуры во время работы системы. Серверные каркасы (IDL Skeletons) обеспечивают статические интерфейсы для вызова объектов определенного типа, а Интерфейс динамических каркасов (Dynamic Skeleton Inerface) – интерфейсы для объектов любого типа, которые не были заранее определены в IDL Skeleton. Интерфейс ORB – общий как для клиента, так и для сервера. Объектный адаптер осуществляет коммуникации между объектом и ORB.

Объекты, используя программную шину коммуникаций ORB, напрямую обмениваются запросами с другими объектами, расположенными как локально (на одном компьютере, но в разных процессах), так и удаленно [36].

1.4.3. Технология DCOM. DCOM является расширением архитектуры COM на случай сетевых приложений (см. рис. 1.2). Эта технология была разработана компанией Microsoft в 1996-м году и сейчас остается главным конкурентом CORBA, хотя теперь уже контролируется не фирмой Microsoft, а группой TOG (The Open Group).

Объект COM мы можем рассматривать, как достаточно простой частный случай объекта CORBA. Как и в CORBA, мы видим элементы статической связи компонентов – заглушки.

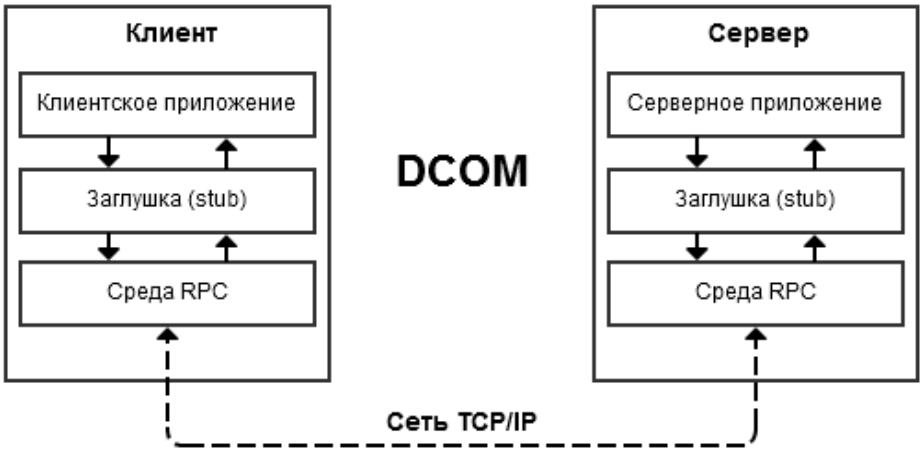

Рис. 1.2. Архитектура DCOM

1.4.4. Сравнение DCOM и CORBA. Прежде чем сравнивать эти две технологии, следует отметить, что DCOM является моделью, а CORBA архитектурой. В связи с этим некоторые пункты сравнения являются условными. В работе [37] выполнено сравнение их по ряду критериев:

1. Уровень абстракции в обеих технологиях примерно одинаковый и достаточно высокий.
2. CORBA более строго подходит к способам обмена и передачи данных. В результате CORBA предоставляет больше возможностей. Эти возможности обеспечивают совместную работу различных средств,

разработанных разными фирмами-производителями программного обеспечения.

3. Поддержку технологии COM выполняют многие фирмы, но в конечном случае только Microsoft решает, как и в каком виде эта технологии попадает от разработчиков Microsoft в открытые спецификации. CORBA – это результат совместных усилий многих фирм. Можно сказать, что CORBA является стандартной инфраструктурой создания распределенных программных продуктов, и подавляющее большинство производителей программного обеспечения, которое должно работать на различных платформах и под управлением разных операционных систем, используют CORBA.
4. CORBA предоставляет разработчикам больше возможностей, чем COM, в виде сервисов и вспомогательных средств.
5. CORBA предлагает несопоставимо большие возможности для управления объектами, и это важно для обеспечения надежности и масштабируемости приложений.
6. Обе технологии примерно одинаковы в плане способов взаимодействия. CORBA имеет определенные преимущества в реализации динамических вызовов за счет того, что средства получения информации о серверах более развиты (имеется репозиторий интерфейсов).
7. И COM, и CORBA имеют примерно одинаковую и высокую производительность.

Таким образом, из упомянутых технологий распределенных объектов CORBA представляет больший интерес. Решение вопроса о целесообразности использовать CORBA в системах автоматизации экспериментов существенно зависит от свойств механизма динамического связывания компонентов в данной технологии. Рассмотрим коротко назначение элементов технологии CORBA, используемых для организации связывания, и свойства реализованных механизмов связывания [36].

1.4.5. Алгоритмы связывания компонентов в CORBA. Для описания интерфейсов связываемых объектов введен язык OMG IDL (Interface Definition Language), который предоставляет синтаксис, не зависящий от технологии. При описании программных архитектур язык OMG IDL используется в качестве универсального способа описания границ объекта, определяющих способ его взаимодействия с другими компонентам системы. Этот язык позволяет описывать интерфейсы с процедурами и атрибутами. Спецификации на IDL могут быть откомпилированы в заголовочные файлы и специальные шаблоны серверов, которые могут непосредственно использоваться программистом. Определенные на IDL процедуры могут быть использованы на любом языке, для которого реализовано отображение из IDL, а затем выполнены. К таким языкам относятся C, Java, SmallTalk, Ada. Средствами IDL можно описать и атрибуты компонента, и наследуемые родительские классы, и вызываемые исключения, и, наконец, интерфейсы, причем с описанием параметров.

CORBA дает возможность реализовать статическое и динамическое связывание. При статическом связывании все необходимые действия для организация удаленных вызовов подготавливаются компилятором idl2 на этапе обработки IDL-файлов. Это эффективный и удобный способ, и единственное, что для него требуется – во время разработки программы подготовить IDL-объявления. Код упаковки процедуры и ее аргументов помещается в файл заглушки (stub), а имя класса совпадает с именем IDL-интерфейса.

Для задач с фиксированным составом компонентов статического связывания вполне достаточно. Но для САЭ характерно многократное изменение методики эксперимента и, соответственно, состава программных компонентов. В этом случае необходим более гибкий подход – динамическое связывание компонентов. Набор средств, обеспечивающих в CORBA выполнение всех необходимых действий для динамического связывания на стороне клиента, называется DII (Dynamic Invocation Interface), на стороне сервера – DSI (Dynamic Skeleton Interface). При этом серверу не известно, какой способ (статический или динамический) использовался на стороне клиента для

формирования запроса, а клиенту не известно, как именно сервер (точнее, компоненты CORBA) будет обеспечивать вызов реальной процедуры. При организации вызова процедуры, естественно, нужно знать, какой интерфейс реализует серверный компонент, название процедуры, количество аргументов и их типы. Чтобы это выяснить, необходимо использовать DII и выполнить несколько вызовов для определения имени процедуры, типов и значений ее аргументов, типа результата, после этого вызвать процедуру и, наконец, получить результат. Помимо этого, компонент-сервер должен содержать код, интерпретирующий все эти запросы. В случае использования технологии CORBA цена динамического связывания следующая:

- Клиентский и серверный компоненты дополняются кодом, который не связан с реализацией логики приложения.
- Среда обслуживания коммуникаций и дополнительные фрагменты программы клиентского и серверного компонентов привязываются к данной технологии.
- Усложняется программирование реализации, как среды взаимодействия, так и компонентов.
- Реализация динамического связывания требует несколько сотен дополнительных операторов, в итоге примерно в 40 раз замедляется процесс взаимодействия по сравнению со статическим связыванием [36,37].

Выигрыш – практически полная свобода при модификации приложения, однако для такой работы потребуется квалифицированный специалист.

CORBA-спецификации затрагивают IDL-отображение в другие языки, API для взаимодействия с ORB и сервисы, предоставляемые шиной ORB. В итоге состав среды обеспечения взаимодействия меняется при изменении приложения. Следует заметить, что, как и классический RPC, CORBA реализует схему взаимодействие один-с-одним, и лишь в последних версиях CORBA возможен асинхронный вызов [36,38].

1.4.6. Технология Ice. Продолжительное (с 1990 г.) использование CORBA дало обширную информацию о ее недостатках [39]. В связи с этим компания ZeroC под руководством одного из разработчиков CORBA (Мичи Хеннинг) выполнила реализацию продукта Ice (The Internet Communications Engine), промежуточного программного обеспечения нового поколения [40]. Это новый объектно-ориентированный набор инструментов, который позволяет разработчикам строить клиент-серверные приложения с минимальными усилиями.

По аналогии с CORBA, Ice вводит объектные модели, однако более простые и более мощные благодаря освобождению от неэффективности прошлой реализации. Введены новые свойства, такие как поддержка протокола UDP, асинхронная диспетчеризация, встроенная защита, агрегация интерфейсов. Устранена необходимость заботиться о ряде деталей – открытии сетевых соединений, сериализации/десериализации данных для передачи по сети, восстановлении разорванных соединений.

Ice поддерживает асинхронное удаленное выполнение процедуры: клиент может включить операцию асинхронно, т.е. использовать proxy (эквивалент CORBA-стаба) обычным способом, однако в дополнение к обычным параметрам должен передать callback-объект. После завершения операции серверная сторона запускает процедуру в ранее переданном callback-объекте, передавая результат или информацию об ошибке.

Однако Ice не содержит принципиально новых (по сравнению с CORBA) решений задачи динамического связывания компонентов. Более того, для асинхронного RPC Ice существенно усложняет схему взаимодействия и еще больше нагружает программу клиента дополнительным расширением.

1.5. “Сервис-ориентированная” архитектура

“Сервис-ориентированная” архитектура SOA (Service-Oriented Architecture) не имеет строгого определения, не содержит существенных новых идей, а является обобщением лучших решений в области создания

распределенных программных систем и способов выполнения их компоновки. SOA предлагает результативный подход к решению одной из самых сложных и актуальных проблем – проблемы интеграции компонентов в прикладную систему с возможностью изменения их состава в соответствии с эпизодическим изменением логики приложения [41, 42].

Отсутствие стандартного способа компоновки распределенных систем, который удовлетворил бы приложения разных типов, приводит к необходимости программирования специальных интеграционных интерфейсов. В случае, когда в компоненты вводятся средства поддержки метода интеграции, возникает серьезная проблема, которая приводит к затруднениям при попытке использования их в других проектах. Для решения подобных проблем недостаточно создать новый набор технологий или выработать единый стандарт интеграции. В настоящее время имеется потребность в концепции такой архитектуры программной среды, в которой возможна адекватная динамике логики приложения динамика в способе построения приложений. Основная идея SOA как раз и заключается в создании такой платформы, которая обеспечит быстрое изменение состава и объединение распределенных программных компонентов для обеспечения нужной логики приложения.

Для SOA характерны следующие основные принципы:

- **Публикация интерфейсов доступа.** Сервисы являются компонентами системы, которые публикуют свои интерфейсы (контракты). Эти контракты не зависят от операционной системы, языка программирования, платформы и других технических особенностей реализации. Компоненты взаимодействуют друг с другом и со вспомогательными службами посредством открытых стандартов.
- **Разделение кода.** Включаемый в информационную систему компонент реализует отдельную функцию, которая является логически обособленной задачей, используемой в соответствии с логикой приложения.

- **Крупноблочная структура** (coarse-grained) сервисов. Сервисы в SOA представляют собой компоненты логики приложения достаточно высокого уровня. Благодаря этому количество сообщений между ними, определенных этой логикой, существенно сокращается по сравнению с множеством низкоуровневых вызовов, используемых, например, в CORBA. В SOA используются интерфейсы, не зависящие от особенностей реализации сервиса. Способ интеграции не требует знания деталей реализации процедуры, которую предоставляет сервис, и способов коммуникации, но обеспечивает средства для декларации возможностей сервиса всем компонентам по сети, если это требуется.
- **Слабая связанность** (loose coupling). Сервисы в системах, построенных на концепции SOA, могут быть реализованы независимо от других служб системы. Для этого необходимо только знание интерфейса используемых сервисов. Изменения в реализации сервиса не влияют на клиента, который этот сервис использует, и наоборот. Благодаря слабой связанности упрощается адаптация системы к изменениям в структуре и составе сервисов.

Каждый из этих пунктов не является специфической особенностью SOA, многие технологии соответствуют этим принципам. Отличительной особенностью построенной на SOA системы является одновременное следование всем указанным принципам. Эти принципы позволяют снять наиболее сложные проблемы интеграции приложений и реагировать на изменения в конфигурации и логике приложения динамично и без сложных преобразований на уровне интеграции компонентов.

Одно из основных преимуществ SOA перед многими традиционными программными моделями состоит в том, что SOA ориентирована на поддержку не программы, а процесса. Сервисы объединяются таким способом, чтобы их выполнение происходило в определенной последовательности и реализовало нужную логику приложения. Каждый из компонентов-сервисов может участвовать в приложениях с различной логикой. Преимущества, которые

обещает комплексный сервис-ориентированный подход, – гибкость программной платформы по отношению к изменениям в логике приложения, а также возможность интегрировать в разрабатываемую систему разные решения сторонних разработчиков.

На данном этапе развития технологий интерфейсы и протоколы взаимодействия компонентов в SOA – это стандарты Web-сервисов [43]. Однако концепция SOA шире идеи Web-сервисов. Она не дает точного описания способа взаимодействия сервисов, но предлагает способ, как добиться, чтобы они понимали друг друга и могли быть объединены. Эта концепция не связана с какой-то определённой технологией. Ее можно реализовать с использованием разных технологий, включая такие технологии, как RPC, DCOM, CORBA, REST или Web-сервисы и одновременно, например, дополнительно использовать механизм файловой системы для обмена данными. Возможную технологическую поддержку SOA можно представить, например, следующим составом компонентов:

- **Шина сообщений** – важнейший уровень программного обеспечения, который совместно с используемой сетью обеспечивает гарантированную отправку и прием сообщений.
- **SOA-реестр** – электронный каталог, в котором хранится информацией о каждом активном компоненте системы. Реестр предоставляет клиентам информацию о компонентах-сервисах, важную для сервисного брокера.
- **Сервисный брокер** – служба, получающая необходимую информацию от SOA-реестра и соединяющая различные компоненты.
- **Программа управления** – компонент, управляющий потоком работ в соответствии с имеющейся моделью. При этом обработка данных на отдельных этапах осуществляется в независимых друг от друга приложениях (компонентах).
- **Группа служебных сервисов**, задача которых – отслеживание нештатных ситуаций, обеспечение жизнеспособности системы и др.

1.6. Автоматизация создания сети

Для создания IP-сети используются специальные сервисы (например, DHCP, DNS), или настройка выполняется вручную. В последнем случае требуется наличие у пользователя определенной квалификации. Упростить эту работу позволяет набор технологий, которые автоматически создают IP-сеть без описания конфигурации или специальных серверов. Набор технологий Zeroconf (Zero Configuration IP Networking) позволяет автоматически соединять компьютеры и подключенные к ним устройства в сеть. Zeroconf решает три проблемы [44]:

1. Выбор сетевого адреса для компонента сети. Согласно RFC 3927 (стандарт для автоматического выбора IP адресов сетевыми устройствами), используется диапазон адресов 169.254.*.* (link local). IPv4 и IPv6 содержат описание способа выбора адреса. Microsoft обозначает это символами APIPA (Automatic Private IP Addressing) или IPAC (Internet Protocol Automatic Configuration).

2. Нахождение компьютера по имени. Для разрешения имени компьютера разные фирмы используют различные протоколы, например: MDNS (Multicast DNS – фирма Apple Computer), LLMNR (Link-Local Multicast Name Resolution – фирма Microsoft). Протоколы имеют мало отличий. Текущая версия LLMNR позволяет выбрать любое доменное имя, что снижает безопасность. Помимо этого, LLMNR не совместим с DNS-SD (DNS Service Discovery).

3. Обнаружение компонентов. Поиск компонента является важной функцией в решении поставленной в работе задачи. Существует ряд стандартов на технологии автоматического поиска и еще больше протоколов, которые тем или иными способом обеспечивают поиск компонентов. Некоторые из них являются взаимодополняющими (например, DHCP и DNS), а другие – конкурирующими (такие как SLP и UPnP). В работе [45] приведен ряд примеров технологий поиска, в числе которых ZeroConf – набор технологий,

обеспечивающих автоматическое создание IP-сетей, и ряд протоколов обнаружения компонентов (SSDP, Salutation, SLP и др.). Реализация Zeroconf идет в двух направлениях: создание отдельных демонов или модификация существующих DHCP клиентов. Около 10 вариантов реализации известны для основных ОС (Windows, Apple, Linux и др.), поддержка Zeroconf встроена в ряд языков (Java, Python, …). Однако единственный протокол для обнаружения компонентов, получивший статус RFC, это SLP [46]. Рассмотрим его более подробно.

1.6.1. Организация SLP. SLP описывает три типа агентов:

- UA – пользовательский агент,
- SA – агент сервиса,
- DA – дополнительный агент каталога.

Приложение может работать и как UA, и как SA, то есть и предоставлять, и запрашивать услугу. UA используется приложением, нуждающимся в конкретной услуге в сети. UA, чтобы идентифицировать услугу, взаимодействует либо непосредственно с SA, либо с DA (если он доступен). SA используется приложением, предоставляющим услугу в сети. Если в сети присутствует DA, то SA передает ему информацию о расположении компонента. В противном случае SA передает ответы на запрос UA непосредственно UA. Роль DA в SLP дополнительная – он служит в качестве центральной точки запросов и ответов. Если присутствует DA, ответы компонентов, посылаемые SA, будут кэшироваться в DA. Затем, когда UA запрашивают компоненты, то отвечать будет непосредственно DA. Наличие DA сокращает трафик и экономит время работы, т.к. для каждого провайдера устраняется необходимости анонсировать свой компонент. Один агент регистрирует компоненты от имени других агентов, минимизируя потери.

Если существует DA, то методы, которыми SLP-агенты соединяются, могут быть разными. Способы, как SA и UA определяют наличие DA, подробно описаны в [46]:

«

Согласно SLP, обнаружение DA в сети осуществляется двумя способами:

- Первый, называемый Активный Поиск DA (Active DA Discovery), использует сам SLP для нахождения DA. UA посылает групповой запрос для service:directory-agent. Каждый DA, получивший этот запрос, передает UA свой адрес.
- Второй метод поиска DA называется Пассивным Поиском DA (Passive DA Discovery). Согласно этому методу, DA периодически посылает запросы пользователям, чтобы UA и SA знали о существовании DA. В случае, если потерян ответ при Активном Поиске DA, проблему решают периодическим анонсированием DA при пассивном поиске.

Когда DA используется в сети, UA и SA соединяются с ним напрямую для запроса и предоставления компонентов. DA обнаруживается Пассивным или Активным поиском DA. Результат поиска передается в сообщении DAAdvert от DA. Эта архитектура дает SLP некоторую гибкость, однако отсутствие DA может затруднить работу SA.

В децентрализованной топологии DA не присутствует в сети; следовательно, UA передают свои запросы SA косвенно. Поскольку DA отсутствует в сети, SA не способны кэшировать свои сервисы и должны реагировать непосредственно на запросы UA. UA посылает запрос на идентификацию необходимой услуги (посылаются многоадресные запросы). Каждый SA предоставляет одноадресный ответ на запрос UA. Ответом идентифицируется компонент и его расположение (адрес и порт).

»

1.6.2. Реализация SLP: OpenSLP. OpenSLP – это реализация SLP с открытым исходным кодом. OpenSLP включает:

- библиотеку API, с помощью которой можно создать SA и UA;
- DA, называемый slpd; настройки его содержатся в файле /etc/slp.conf.

В OpenSLP API использована архитектура callback. Программист может вызвать API-функцию, и в ее контексте callback-функция вернет ответ на запрос.

SLP использует несколько сообщений. Дополнительно к сообщениям регистрации и обнаружения предусмотрены и сообщения для отмены регистрации компонентов, которые уже не доступны, и поиск компонентов по типу, при этом запрашиваются атрибуты конкретного компонента. В таблице 1.1, взятой из работы [46], перечислены API-функции OpenSLP и их назначение.

Дополнительно к библиотеке API, OpenSLP предоставляет утилиту slptool, которую можно использовать для интерактивного выполнения большинства функций SLP. Эта утилита может быть полезной для регистрации или поиска компонента.

Таблица 1.1. Первичные функции библиотеки OpenSLP

Функция	Описание
SLPOpen	Открывает экземпляр OpenSLP API
SLPClose	Закрывает экземпляр OpenSLP API
SLPReg	Регистрирует компонент и URL
SLPDereg	Отменяет регистрацию URL компонента
SLPFindSrvs	Находит зарегистрированный компонент заданного типа
SLPFindSrvTypes	Находит все типы компонентов, зарегистрированных в области видимости

1.7. Система Open Inspire

В недавних работах [47.48] опубликована система Open Inspire, цели которой близки к поставленным в диссертации. Open Inspire реализована полностью на языке Java и основана на следующей идее. В CORBA (и других технологиях) при настройке удаленного вызова процедуры между клиентом и сервером выполняется диалог, во время которого определяются интерфейс, состав параметров и некоторые другие необходимые данные. Инициатором

диалога является программа-клиент. В Open Inspire предложено передать инициативу серверу (названо “инверсия управления” – inversion of control), причем сервер должен занести (dependency injection). нужные данные в специальный буфер, доступный клиенту. При реализации этой идеи авторы Open Inspire отказались [см. 47] не только от использования CORBA, но и от использования практически всех Web-технологий, предложив вместо этого свои сервисы на Java. Авторы [47,48] не смогли исключить использование скриптов, более того, они ввели использование специального языка и диалоговых средств для описания схемы связывания компонентов и конфигурации ПО САЭ. Это требует определенной квалификации, вносит жесткую связанность и необходимость отладки и не является оптимальным для систем с часто изменяемой методикой работы, какими являются САЭ. Поэтому идеология [47,48] не может быть использована в качестве прототипа.

1.8. Выводы

1. Технология CORBA наиболее полно отвечает требованиям для разработки промышленных, открытых, распределенных систем. Однако нас интересует технология разработки систем, неотъемлемым свойством которых является систематическое изменение логики приложения (а как следствие – изменение конфигурации оборудования и состава компонентов).

Механизм организации динамического связывания в CORBA для задач САЭ избыточно сложный, требует включения специальных элементов в среду обеспечения взаимодействия компонентов и в компоненты. Все это затрудняет возможность использования компонентов и среды обслуживания их взаимодействия в разных проектах. Поскольку в ПО САЭ управляющий компонент ожидает от исполняющего только синхронизирующий сигнал завершения работы (см. параграф 1.2), диалог между компонентами для выяснения параметров взаимодействия может быть устранен, что существенно упростит схему взаимодействия. Наконец, свойственная рассмотренным технологиям синхронность взаимодействия компонентов и схема “один с

одним” является существенным ограничением для интересующих нас систем реального времени. В связи с этим механизм динамического связывания CORBA не может быть рекомендован для применения в САЭ и целесообразно выполнить специальную разработку средств обеспечения взаимодействия компонентов, адаптированную к специфике систем автоматизации экспериментов.

2. Для разработки структуры ПО САЭ может быть принята концепция, которая в литературе известна под названием “Сервис-ориентированная архитектура” [43]. Основные принципы этой концепции: функциональная завершенность компонентов, их крупноблочная структура, публикация (или фиксация) интерфейсов, слабая связанность компонентов. Оптимальность выбора данной концепции обусловлена тем, что в ней обязательным является следование одновременно всем перечисленным принципам, в отличие от рассмотренных технологий, использующих только часть из них.

Структура ПО САЭ, в составе которой присутствуют программа управления потоком работ, шина сообщений, реестр компонентов, брокер и другие элементы технологии, а также технология компоновки ПО САЭ и методы, обеспечивающие возможность использовать компоненты без изменений в разных проектах, разработаны в последующих разделах диссертации.

3. Для выбора предпочтительной реализации формата передаваемых сообщений в таблице 1.2 приведены некоторые характеристики протоколов JSON-RPC, XML-RPC и SOAP. В нашем случае (для локальной сети, см. параграф 1.2) расширения, введенные в SOAP по сравнению с XML-RPC, не являются принципиальными, но могут привести к существенному усложнению реализации [49]. В то ж время, протокол XML-RPC достаточно простой, но имеет некоторые неудобства по сравнению с JSON-RPC.

JSON-RPC позволяет иметь ряд вариантов параметров, а использовать только часть из них. Полезен также доступ к параметрам по имени/ключу, что невозможно в большинстве других механизмов RPC. Это упрощает расширение

состава параметров без разрушения действующей системы (и без необходимости создания различных версий системы), а также дает возможность строить расширенные вызовы (например, чтобы увидеть, что произошло), и делает безразличным порядок перечисления параметров.

В плане безопасности, удобства отладки и исправления ошибок варианты примерно эквивалентны [49]. Выбор сделан в пользу JSON-RPC ввиду его достаточности и простоты использования.

Таблица 1.2. Сравнение способов удаленного вызова процедур

Функция	JSON-RPC	XML-RPC	SOAP
Прямая поддержка Unicode	Да	Нет	Да
Простота, компактность	Да	Нет	Нет++
Транспортный протокол	Любой	HTTP	HTTP
Простота создания объекта на сервере	Нет	Да	Нет
Простота обработки данных на стороне клиента	Да	Нет	Нет
Прямая поддержка Null/None	Да	Нет	Да
Имена и ключи к параметрам	Да	Нет	Да
Уведомления	Да	Нет	Да
Сопоставление запрос/ответ	Да	Нет	Да
Ограничение сериализации	Нет	Спецсимволы	Нет

4. Необходимая поддержка динамической компоновки САЭ может обслуживаться протоколом SLP.

5. Выбранные технологии в сочетании с результатами анализа в работе [7] специфики ПО САЭ являются достаточной базой для разработки алгоритмов и реализации унифицированных компонентов ПО САЭ, которые могут быть использованы в других проектах без изменения и в других проблемных областях. Развитие темы построения архитектуры САЭ и средств взаимодействия компонентов в распределенной системе может послужить основой для дальнейшей консолидации усилий разработчиков.

Глава 2. Анализ функционального состава и способа взаимодействия компонентов ПО САЭ

2.1. Цели анализа функций и способа взаимодействия компонентов

Возможность построения по новой технологии ПО САЭ, компоненты которого могут использоваться без изменения в различных экспериментах, зависит от ряда факторов, в числе которых организационные, функциональные и технологические. Функциональные осложнения рассмотрены в параграфах, посвященных разработке соответствующих компонентов. К организационным можно отнести следующие:

- совпадение направлений исследований и технической политики в области автоматизации в разных подразделениях организации;
- отсутствие стремления оставить все «как есть»;
- наличие финансовых ресурсов для достаточно быстрого внедрения нового решения в подразделениях организации, чтобы избежать морального устаревания новых технических решений;
- практическая направленность и реальность – возможность реализации предлагаемых решений на уже имеющихся компьютерных и телекоммуникационных ресурсах.

Основным элементом тиражируемого решения может и должен стать комплекс унифицированных средств программного обеспечения (желательно – и технического, как, например, в ЛИЯФ [50]), а также соответствующая документация и методическое обеспечение ввода новых систем и расширения (изменения) функциональных возможностей действующих.

Разработка технологических средств и методов унификации компонентов программного обеспечения САЭ является наиболее сложной частью выполненных в диссертации исследований. Цели анализа, выполненного в данной главе, следующие:

1. Выявить компоненты, для которых критичны характеристики средств обеспечения взаимодействия – в первую очередь скорость обмена сообщениями и данными.
2. Получить дополнительную информацию для разработки алгоритмов динамического связывания компонентов с учетом особенностей их работы в ПО САЭ. Вывод о необходимости такой разработки сделан в главе 1.
3. Получить информацию для разработки методов, устраняющих влияние изменения методики исследования на возможность реализации унифицированных компонентов, т.е. компонентов, которые можно будет использовать в различных проектах без изменения.

Ниже рассмотрены влияние изменения методики исследования на возможность унификации компонентов ПО САЭ, характеристики компонентов и способы взаимодействия компонентов в ПО САЭ, требования к скоростным характеристикам средств обеспечения взаимодействия процессов, и сформулированы ключевые задачи, решение которых необходимо для реализации распределенного ПО САЭ, концепция которого предложена во введении.

2.2. Влияние изменения методики исследования на возможность унификации компонентов

ПО САЭ должно решать следующие задачи:

1. управление условиями регистрации данных в соответствии с методикой исследования;
2. регистрацию, представление в формате, удобном для обработки, и сохранение потока данных от детекторов;
3. контроль корректности работы экспериментальной установки и достоверности регистрируемых данных.

Автоматизация этих задач существенна для результативного выполнения исследований. Для их автоматизации могут быть использованы компоненты, функциональное назначение которых, например, следующее:

- интерфейс пользователя;
- управление выполнением эксперимента;
- подсистема регистрации данных DAQ (ввод в ЭВМ, преобразование, архивирование потока данных);
- настройка подсистемы DAQ;
- управление оборудованием, определяющим условия регистрации данных;
- средства мониторирования состояния установки;
- средства передачи команд и сообщений;
- обработка отказов, визуализация, предварительная обработка данных.

Такой состав выбран на основании рассмотрения опыта других нейтронных центров [51-53], опыта разработки и эксплуатации САЭ в ЛНФ ОИЯИ [54, 55] и пожеланий экспериментаторов.

Данный список не является полным и не в каждой системе используются все перечисленные функции, но он соответствует в основном текущему состоянию представлений о ПО САЭ в ряде организаций и позволит сделать выводы, необходимые для решения поставленной в работе задачи. Основной состав компонентов включает интерфейс пользователя, программу управления экспериментом, подсистему регистрации данных DAQ, компоненты управления условиями регистрации данных и средства обеспечения межкомпонентного взаимодействия.

Оборудование подсистемы регистрации данных, как правило, является постоянной составляющей экспериментальной установки. Программы настройки программной подсистемы регистрации данных зависят от конкретной реализации контроллера ввода данных и технического задания.

Состав устройств управления условиями регистрации данных определяются конструкцией спектрометра. Состав устройств, используемых в

конкретном эксперименте, определяется методикой исследования. Список программных компонентов, управляющих условиями регистрации данных, и значений управляемых параметров в конкретном эксперименте определяется заданием на эксперимент.

Если новая методика исследования требует использовать оборудование, не предусмотренное заранее в конструкции экспериментальной установки и программах, это приведет также к изменению состава используемых компонентов и необходимости отредактировать часть ранее использовавшихся. При этом редактируется компонент, управляющий последовательностью выполнения операций в эксперименте. Если объединение компонентов в систему выполнялось транслятором, то эту операцию приходится выполнять заново. Чаще для упрощения процесса модификации ПО САЭ вводится специальный язык, интерпретатор языка включается в состав ПО САЭ, состав и последовательность выполнения операций в эксперименте (скрипт) описывается на этом языке списком процедур и значений параметров, а адреса процедур-компонентов и другая информация помещается в конфигурационные файлы или передается через списки параметров. Оба эти варианта вводят статическую связанность компонентов, поэтому в главе 3 диссертации для устранения этой проблемы предложены альтернативный метод управления выполнением эксперимента и способ представления методики исследования.

2.3. Классификация компонентов ПО САЭ

Анализ функционального состава компонентов ПО САЭ позволяет сделать следующие утверждения:

- Любой компонент может выступать как в роли клиента, так и в роли сервера, выполняя функции управления, исполнения действий, публикации и обработки информации.
- Логикой эксперимента предопределены роли управляющих (интерфейсы пользователя и программа управления экспериментом) и исполняющих компонентов (компоненты управления окружением образца и

подсистемы DAQ), схема их взаимодействия фиксирована. Для этих функциональных групп характерно требование гарантированной доставки команды и подтверждения завершения работы.

- В составе САЭ присутствуют другие функциональные пары – источники информации, вырабатываемой в процессе работы ПО САЭ, и потребители этой информации. 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 может быть несколько либо ни одного.
- Можно сформулировать две группы компонентов САЭ:
  - основные компоненты с детерминированным характером взаимодействия, выполняющие основные задачи эксперимента; характер взаимодействия – “один к одному”;
  - компоненты (и некоторые процедуры основных компонентов), реализующие вспомогательную логику – сервисные функции, обработку нештатных ситуаций и др., не влияющие (в штатных условиях) на выполнение основных задач; характер взаимодействия – “один ко многим”.
- В группе основных компонентов присутствует только два типа клиентов, в качестве результата вызова удаленных процедур им требуются сигнал завершения выполнения заказанного действия для синхронизации работы. Это управляющие программы – интерфейсы пользователя и программа управления экспериментом, управление – однонаправленное. Реализация обратной связи относится к вспомогательной логике.
- В группе основных компонентов присутствует только два типа компонентов, выполняющих основную работу, определяемую методикой исследования. Это подсистемы регистрации данных (DAQ) и компоненты управления устройствами, формирующими условия регистрации экспериментальных данных.

2.4. Особенности способов взаимодействия компонентов в ПО САЭ

Назначение операций взаимодействия компонентов, входящих в состав ПО САЭ, можно представить следующим списком:

- заказ на выполнение удаленной процедуры;
- передача сигнала завершения выполнения процедуры;
- периодическая передача информации;
- передача информации о событии, изменении состояния;
- передача данных мониторинга состояния объекта.

Операция вызова удаленной процедуры должна выполняться асинхронно. Периодическая передача информации используется, в частности, для подтверждения работоспособности компонента.

Приведенный состав операций взаимодействия компонентов может обслуживаться двумя типами сообщений:

- заказ на выполнение удаленной процедуры с указанием списка параметров;
- ответ исполнителя – сигнал завершения выполнения процедуры (DONE/ERROR) или публикация информации (NOTIFY) с указанием ее характера.

Таблица 2.1. Основные компоненты ПО САЭ и способ их взаимодействия

Компоненты в роли клиентов	Компоненты в роли серверов	Ответ сервера
Интерфейсы пользователя	Программа управления экспериментом	DONE/ERROR
Программа управления экспериментом	Группа подсистем DAQ	DONE/ERROR
Программа управления экспериментом	Компоненты правления условиями регистрации	DONE/ERROR

В таблице 2.1 перечислены компоненты основной логики ПО САЭ – клиенты и ассоциированные с ними серверы. Помимо типов компонентов, указанных в таблице 2.1, в состав основных входят средства обеспечения взаимодействия процессов. Взаимодействие этих компонентов может быть обеспечено одним универсальным интерфейсом для передачи текстовых сообщений с указанием их типа (удаленный вызов процедуры или ответ сервера), и при изменении количества основных компонентов изменение остальных программ, реализующих логику эксперимента, не потребуется.

Таблица 2.2. Компоненты вспомогательной логики – источники и потребители информации

Компоненты – источники информации	Периодичность сообщений	Функции компонентов – потребителей информации
Программа управления экспериментом	По чтению строки задания	1. Интерфейс пользователя 2. Протоколирование работы
Компонент архивирования данных	По записи файла	1. Визуализация данных 2. Предварительная обработка 3. Интерфейс пользователя
Средства мониторирования объекта	Определяется в задании	1. Интерфейс пользователя 2. Протоколирование работы …
Источники периодической информации	Фиксирована	1. Обслуживание взаимодействия компонентов 2. Протоколирование работы
Средства протоколирования	По мере возникновения	Регистрация событий, обработка сбоев и отказов.

В составе второй группы компонентов САЭ (таблица 2.2) компоненты-потребители информации одного и того же типа (например, визуализаторы) могут быть представлены в нескольких экземплярах (а также в нескольких

вариантах), поэтому требуется иметь возможность передавать одинаковую информацию нескольким потребителям, состав и количество которых может изменяться в процессе работы спонтанно и независимо от основной логики эксперимента. Взаимодействие всех компонентов этой группы также может быть обеспечено универсальным интерфейсом для передачи текстовых сообщений с указанием их типа (публикация данных). При изменении количества компонентов-источников информации, а также при появлении дополнительных экземпляров вспомогательных компонентов-серверов (или компонентов нового функционального назначения) изменение программ, реализующих логику эксперимента, не потребуется.

Предлагаемое группирование компонентов, фиксация состава интерфейсов и способа их использования не ограничивает возможностей развития ПО САЭ и является основой для решения проблемы связанности, указанной в разделе 2.1. Наличие в составе ПО САЭ компонентов, реализующих вспомогательную логику, и их специфика должны найти отражение в алгоритмах средств обеспечения взаимодействия компонентов.

Компонент описания методики исследования и составления задания на эксперимент отсутствует в приведенных списках основных и вспомогательных компонентов. Эти средства необходимы (и присутствуют) в любом ПО САЭ, однако реализация их в виде компонента ограничит область его применимости, а включение в виде функции в тело другого компонента внесет жесткую связанность. По этой причине в диссертации средства описания методики эксперимента выделены в специальную подсистему.

На основании анализа состава и способа взаимодействия компонентов ПО САЭ сделаны следующие выводы:

- динамическое связывание компонентов для удаленного выполнения процедур необходимо только в основной логике ПО САЭ;
- результатов удаленного вызова процедур должно быть два: 1) обязательный сигнал завершения работы, адресуемый вызывающей программе (программе управления экспериментом или интерфейсу

пользователя), и 2) детализирующая информация (список зарегистрированных файлов данных, описание состояния управляемого объекта, диагностическое сообщение и др.), адресуемая вспомогательной логике;

- реализация вспомогательной логики требует специальной дисциплины связывания компонентов, учитывающей спонтанный характер возникновения запросов на такие операции и независимость основной логики (в штатных условиях) от результатов выполнения вспомогательных операций.

Эти выводы сыграли решающую роль при разработке способа управления экспериментом и специального метода динамического связывания компонентов.

2.5. Требования к скоростным характеристикам средств обеспечения взаимодействия компонентов

1) Передача данных. Процессы взаимодействуют друг с другом путем передачи команд, сообщений и данных. Различные процессы предъявляют разные требования к скорости обмена данными. Критической в этом плане является скорость передачи экспериментальных данных между процессами их ввода, преобразования и архивирования. Максимально допустимый поток данных на входе системы (а в некоторых случаях и корректность результатов эксперимента) зависит от скорости выполнения этих процессов. Для этих трех процессов желателен непосредственный доступ к данным. Решение этой задачи представлено в главе 5. Для остальных процессов в большинстве случаев достаточна скорость, обеспечиваемая современными дисковыми накопителями и протоколом FTP.

2) Передача сообщений. Скорость передачи команд и сообщений зависит от используемого транспортного протокола и среды передачи данных и не является критическим параметром. Время передачи пакета 25 Кбайт по протоколу UDP + время получения подтверждения составляет 400..700 мкс, что

вполне достаточно для процессов, выполняемых в интересующих нас САЭ [56]. Использование протокола TCP/IP незначительно увеличивает эту оценку. Более важной является задача выбора механизма передачи, т.к. от этого зависит ряд свойств разрабатываемой системы. Разработка среды, обслуживающей взаимодействие компонентов, описана в главе 4.

2.6. Ключевые задачи при разработке унифицированного распределенного ПО САЭ

На основании выполненного в главах 1 и 2 анализа сформулированы ключевые задачи, решение которых необходимо для реализации распределенного ПО САЭ, концепция которого предложена во введении:

- разработать метод управления экспериментом, исключающий связанность компонентов и обеспечивающий возможность реализовать унифицированную программу управления экспериментом.
- разработать способ описания методики исследования и алгоритмы унифицированной подсистемы описания методики исследования и составления задания на эксперимент, инвариантной относительно изменения состава используемых компонентов управления условиями регистрации данных и подсистем DAQ;
- разработать средства обеспечения взаимодействия компонентов –поиск компонентов, автоматическое объединение их в систему, динамическое связывание компонентов для удаленного выполнения процедур и передачи сообщений.

2.7. Выводы

В данной главе на основании анализа состава и способа взаимодействия компонентов получены следующие результаты:

1. Предложена классификация компонентов.
2. Выявлены особенности взаимодействия компонентов ПО САЭ.
3. Впервые для ПО САЭ сделаны следующие выводы:

- динамическое связывание компонентов для удаленного выполнения процедур необходимо только в основной логике ПО САЭ;
- результатов удаленного вызова процедур должно быть два: 1) обязательный сигнал завершения работы, адресуемый вызывающей программе, и 2) детализирующая информация, адресуемая вспомогательной логике;
- реализация вспомогательной логики требует специальной дисциплины связывания компонентов, учитывающей спонтанный характер возникновения запросов на такие операции.

4. Сформулированы ключевые задачи, решение которых необходимо для достижения поставленной в диссертации цели:
   - разработать метод управления экспериментом, исключающий связанность компонентов и обеспечивающий возможность реализовать унифицированную программу управления экспериментом;
   - разработать способ описания методики исследования и алгоритмы унифицированной подсистемы описания методики исследования, инвариантной относительно изменения состава используемых компонентов управления условиями регистрации данных и подсистем регистрации данных DAQ;
   - разработать методы и средства обеспечения взаимодействия компонентов – поиск компонентов, автоматическое объединение их в систему, динамическое связывание компонентов для удаленного выполнения процедур и передачи сообщений.

Решение этих ключевых задач описано в последующих главах диссертации.

Глава 3. Разработка методов и алгоритмов управления выполнением эксперимента

3.1. Традиционные способы управления экспериментом и причина потери возможности использования компонентов ПО САЭ в разных экспериментах

Автоматизация работы экспериментальных установок существенна для результативного выполнения исследований. При этом существенным является не только уровень автоматизации работ, но и сроки разработки ПО САЭ. В ряде случаев эти сроки оказываются не адекватными темпам и уровню развития аппаратной базы и средств программирования. Наиболее существенная причина длительных сроков – проблемы при попытке использовать компоненты ПО САЭ в других экспериментах и системах [7,15]. Одна из причин потери такой возможности – изменение методики исследования. При возникновении необходимости выполнить эксперимент с использованием оборудования, не предусмотренного ранее в составе экспериментальной установки, приходится дополнять состав программного обеспечения. При этом включение нового компонента приводит к необходимости внести изменения в средства управления выполнением эксперимента (программу управления экспериментом, конфигурационные файлы, интерфейс пользователя и др.) а также в другие составные части ПО САЭ. В первую очередь это касается состава компонентов, управляющих условиями регистрации данных (сменой мишеней, температуры, давления, коллиматоров и т.д.). В некоторых системах количество управляемых параметров может исчисляться тысячами [57]. В нашем случае, благодаря наличию централизованных служб управления базовыми установками и их инфраструктурой [58,59], количество устройств управления окружением образца не превышает нескольких десятков (например, в [60,61] – 15 и 25 соответственно).

Традиционных способов управления экспериментом два:

1. представление программы управления на языке программирования фиксированной группой процедур, каждая из которых реализует определенную методику исследования, а значения параметров определяются в диалоге или данными из файла [62,63];
2. использование в составе ПО САЭ интерпретатора, на языке которого составляется программа управления экспериментом (скрипт) в виде списка вызовов внешних процедур и значений параметров [64,65].

Недостатком обоих методов является жесткая связанность компонентов. Помимо этого, практика показывает, что в 70% случаев способ настройки ПО САЭ с помощью скриптов содержит опечатки и может занимать до двух суток работы специалистов [66].

3.2. Модель программы управления выполнением эксперимента

В связи с тем, что традиционные способы управления экспериментом затрудняют использование компонентов в ПО разных САЭ, в диссертации предложен альтернативный метод управления экспериментом [11,13]:

- программа управления экспериментом выполняет последовательно два процесса – формирование условий эксперимента и регистрацию данных в этих условиях;
- конкретизация функционального наполнения этих процессов (состав используемых компонентов) выполняется динамически с использованием файла с описанием условий регистрации данных, создаваемого подсистемой описания методики исследования [11,13].

Для выполнения каждого из указанных процессов программе управления нужны списки компонентов, которые выполнят конкретную работу. Такая информация, дополненная значениями параметров, детализирующих условия эксперимента, является описанием методики эксперимента и может быть сформирована средствами, внешними по отношению к программе управления экспериментом. Конкретную работу по формированию нужных условий эксперимента и регистрацию данных выполнят соответствующие компоненты.

Для синхронизации последовательности выполнения процессов от каждого из них требуется только сигнал завершения работы. Компоненты должны выполнять интерпретацию параметров, а программа управления экспериментом (а также среда обслуживания взаимодействия компонентов) оказывается прозрачной для списка параметров, и это обеспечивает ей универсальность.

Предложенная модель программы управления экспериментом позволяет исключить вызов программы регистрации данных из описания задания на эксперимент и выполнять этот вызов по умолчанию. Этот вывод имеет существенное значение для составления файла задания на эксперимент.

В диссертации средства задания методики исследования по причине, указанной выше, выделены в специальную подсистему. Вариант подсистемы подготовки задания с единым для ПО различных САЭ интерфейсом пользователя и управляющей программой, настраивающейся на нужную методику эксперимента с помощью файла описания задания, ранее использован диссертантом в работе [12].

3.3. Анализ опыта эксплуатации варианта подсистемы составления задания на эксперимент

В работе [12] подробно описана подсистема подготовки задания, в составе которой была база данных (БД) и программа подготовки задания PSJ (Preparation of Single Job). БД обеспечивала информационную поддержку для специалиста (экспериментатора), описывавшего конфигурацию драйверного слоя ПО САЭ и составлявшего задание на эксперимент и содержала:

- описания готовых компонентов драйверного слоя, доступных для использования;
- описание конфигурации (список компонентов) драйверного слоя разрабатываемого ПО САЭ;
- таблицы параметров и др. данные.

Отметим следующие ограничения такого варианта, стимулировавшие выполнение разработки новой подсистемы:

- в БД заносилось явное описание конфигурации драйверного слоя конкретного ПО САЭ. Такая фиксация вносит связанность, ограничивает гибкость ПО САЭ и не является необходимой. Целесообразно использовать динамическое подключение компонентов управления окружением образца к системе, используя соответствующие технологии и описание методики эксперимента;
- в состав описания управляемого заданием драйверного слоя (и в таблицу задания на эксперимент) включался компонент, выполнявший ввод и регистрацию экспериментальных данных (DAQ). Вызов этого компонента должен быть исключен из описания методики и выполняться по умолчанию по следующим причинам:
  1. настройка такого компонента, способ использования, состав команд управления существенно иные, чем у компонентов управления условиями эксперимента. В связи с этим для работы с оборудованием ввода и регистрации данных должна формулироваться отдельная подсистема DAQ и для настройки ее использоваться специальная дисциплина;
  2. состав подсистемы DAQ фиксирован для спектрометра, во всех состояниях установки должен выполняется один и тот же процесс регистрации данных.
- БД была представлена бинарным файлом со структурой, определенной программистом. С целью обеспечения лучшей читаемости и масштабируемости следует использовать общепринятые форматы записи данных, например, JSON, XML или БД (например, SQLite).

3.4. Разработка алгоритмов подсистемы описания методики исследования

Для описания методики исследования и составления задания на эксперимент требуется информация об устройствах и работающих с ними

компонентах (паспорта устройств). Рассмотрены два альтернативных алгоритма получения этой информации:

1. хранить паспорта устройств в базе данных (БД) и выбирать их из БД специальной диалоговой программой до запуска ПО САЭ и начала эксперимента;
2. хранить паспорта устройств в обслуживающих их компонентах, динамически составлять список нужных активных компонентов (после запуска ПО САЭ) путем поиска их по типу с помощью механизма обслуживания взаимодействия компонентов и запрашивать эти данные у компонента специальной диалоговой программой до начала эксперимента;

Алгоритм составления списков активных компонентов по их типам присутствует в разработанном механизме обслуживания взаимодействия компонентов (алгоритм DiCME, описан в главе 4), с его помощью выполняется поиск:

- интерфейсом пользователя – программы управления экспериментом;
- программой управления экспериментом – компонентов управления окружением образца и компонентов DAQ;
- подсистемой мониторинга – нужных компонентов системы и обнаружение возможных коллизий (например – несколько управляющих программ);
- поиск Менеджера событий и др.

Однако для реализации подсистемы описания методики исследования и формирования задания на эксперимент выбран первый вариант по следующим причинам:

- задание на эксперимент можно подготовить заранее на любой ЭВМ без запуска ПО САЭ;
- можно подготовить к выполнению пакет заданий;
- вариант лучше защищен от ошибок пользователя.

Разработанная подсистема описания методики исследования [11] включает базу данных и две диалоговые программы: 1) программу составления паспортов компонентов, работающих с устройствами управления окружением образца, используемую программистами; 2) программу подготовки задания PSJ, используемую экспериментаторами.

1) Программа составления паспортов компонентов создает описание (документацию) в формате JSON. Формат JSON выбран из ряда альтернатив (XML, JSON, SOAP и др.) по критериям простоты, достаточности, компактности и наличия средств поддержки в популярных языках программирования (C++, Delphi, C++ Builder и др.). Паспорт содержит следующие данные:

1. версию паспорта;
2. имя контроллера устройства;
3. уникальный ID (GUID), используемый для адресации компонента средствами коммуникаций;
4. перечень параметров устройства (не может быть пустым); возможно задание взаимоисключающих групп параметров. Каждый параметр имеет:
   - имя (уникальное, неизменное);
   - тип значения;
   - значение по умолчанию (если задано, то используется при инициализации компонента);
   - диапазон допустимых значений.

Описание атрибутов параметра может быть опущено, однако не рекомендуется это делать, т.к. будет отключен контроль значений при вводе. На рис. 3.1 представлен пример паспорта устройства.

```
{
     "version":1,
     "controller-id":"6F9619FF-8B86-D011-B42D-00CF4FC964FF",
     "controller-name":"Kolhida-Motors",
     "devices":[
          {
               "name":"polarizer",
               "parameters":[
                    {
                         "name":"angle",
                         "type":"int",
                         "range-from":0,
                         "range-to":90,
                         "default-value":45
                    }
               ]
          },
          {
               "name":"analyzer",
               "parameters":[
                    {
                         "name":"angle",
                         "type":"int",
                         "range-from":0,
                         "range-to":90,
                         "default-value":0
                    }
               ]
          }
     ]
}
```

Рис. 3.1. Пример паспорта устройства

Для многокоординатных устройств (гониометры и др.) более удобно (и соответствует стилю задания на эксперимент) каждую ось формулировать, как отдельное устройство, даже в тех случаях, когда управление всеми осями выполняется через один контроллер.

Важные свойства данного подхода:

- разработка компонента, работающего с оборудованием, считается завершенной только после внесения документации в БД;
- реализация компонента возможна на любом языке, для любой ОС;
- использование компонента возможно на любой ЭВМ локальной сети без фиксации этого адреса в конфигурационных файлах или программах;
- состав параметров (прикладной протокол работы с компонентом) не фиксируется и определяется разработчиком компонента.

2) Программа составления задания использует список доступных устройств (и компонентов) из БД. Из этого списка пользователь в диалоге выбирает нужные в данном эксперименте компоненты и составляет два списка. Первый – список устройств, которым должны быть отправлены (однократно) команды инициации (и фиксации) состояния части осей многокоординатных устройств (например, гониометров), по которым в этом эксперименте не предполагается использовать сканирование, а также заказ на периодическую передачу информации о состоянии устройства и удержание значения заданного параметра. Второй список – список устройств, для которых в процессе эксперимента будут задаваться несколько состояний в соответствии с нужной методикой исследования. Последовательно выбирая названия устройств из этого списка, пользователь может задать для каждого список значений управляемых параметров. На основании этих данных PSJ создает таблицу, в которой пользователю предоставляется дополнительная возможность изменить состав и последовательность работы компонентов управления оборудованием, а также состав используемых значений параметров. Вид окна программы PSJ в режиме редактирования показан на рис. 3.2.

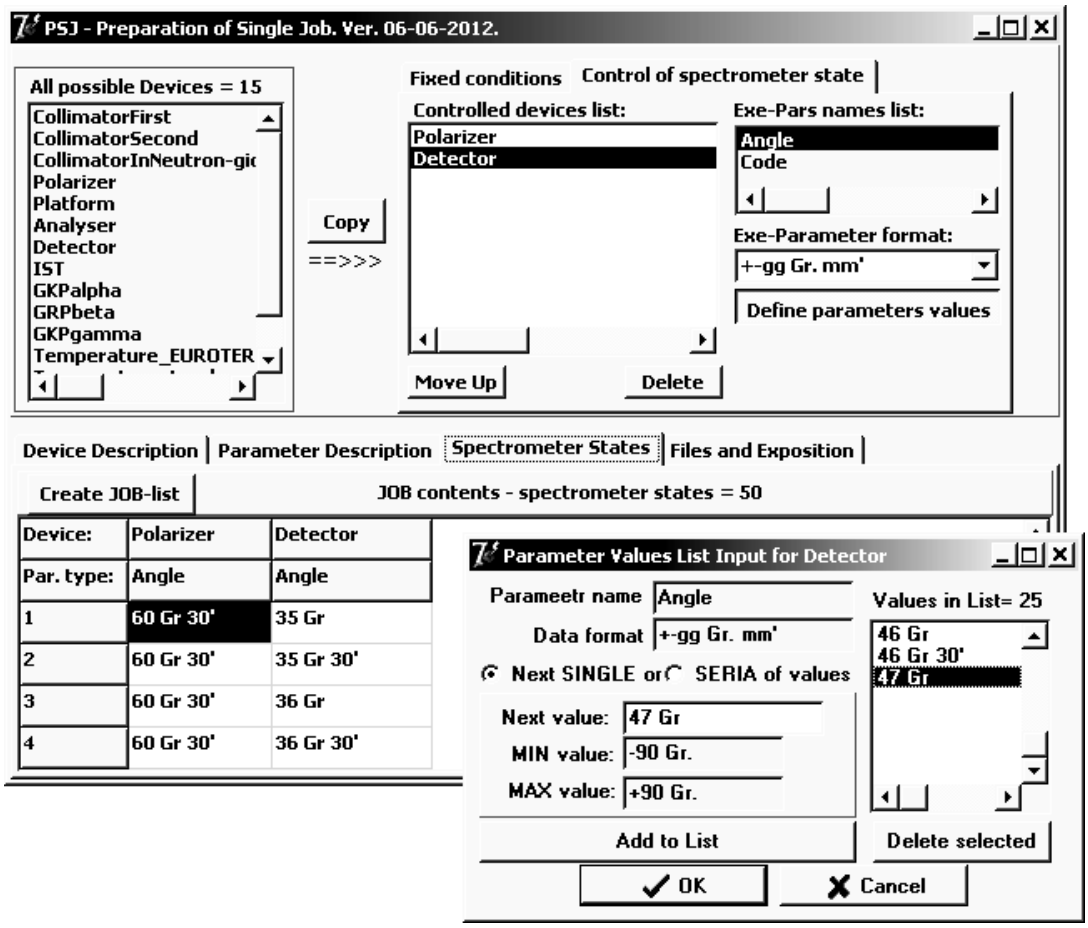

Рис. 3.2. Вид окна программы PSJ в режиме редактирования

На рис. 3.2 видно, что при составлении задания используется терминология конкретной проблемной области – терминология физика: названия узлов его спектрометра, угловые положения и т.д.

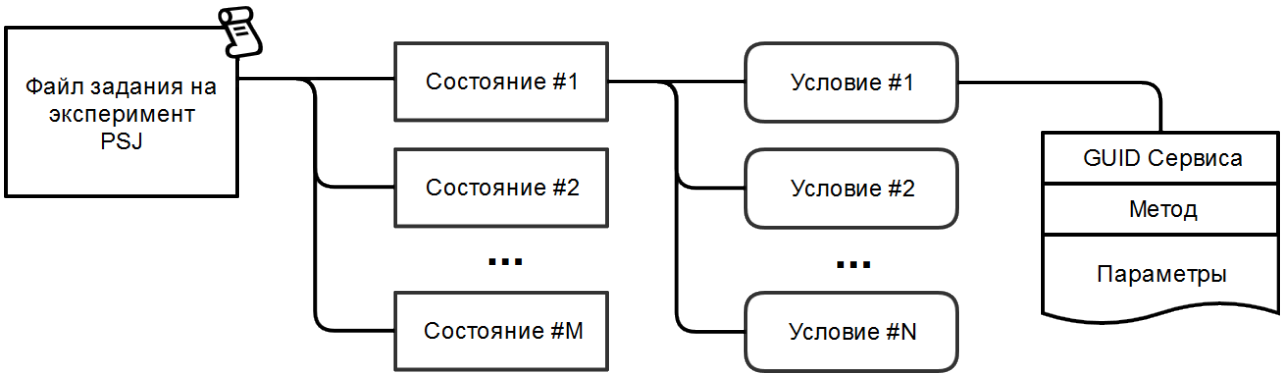

Рис. 3.3. Структура файла задания на эксперимент

Результат работы PSJ – табличное описание конечного автомата, реализующего нужные в эксперименте состояния аппаратной системы. В каждом состоянии выполняется экспозиция данных. На рис. 3.3. показана структура файла задания.

Идентификаторы компонента (GUID) и его типа автоматически заносятся в файл задания из документации (из БД).

3.5. Выводы

В данной главе:

1. Предложен новый метод управления выполнением операций основной логики ПО САЭ, основанный на модели программы, управляющей исполнением двух процессов, конкретное функциональное наполнение которых определяется динамически с использованием файла задания на эксперимент. В отличие от обычно используемых способов представления программы управления выполнением методики исследования на языке программирования или в виде интерпретируемого скрипта, предложенный метод исключил связанность компонентов и позволил разработать алгоритмы и реализовать унифицированную программу управления экспериментом.
2. Предложен способ описания методики исследования и составления задания на эксперимент в виде списка условий, в которых должна выполняться регистрация экспериментальных данных, вместо обычно используемого перечисления действий программы управления экспериментом списком процедур на языке программирования или представления последовательности действий списком названий процедур и значений параметров в виде интерпретируемого скрипта. Этот способ использован при разработке подсистемы, которая формирует информационную базу для автоматической компоновки ПО САЭ, алгоритмов позднего связывания и исполнения процессов программой управления экспериментом. Предложенный способ обеспечил

автоматическую настройку ПО САЭ на выполнение заданной методики эксперимента. Разработаны алгоритмы и реализована подсистема описания методики исследования, инвариантная относительно изменений методики исследования, конструкции экспериментальной установки и состава ПО САЭ. Свойства подсистемы:

- Формат списка параметров процедуры компонента не фиксирован, определяется программистом и документируется в паспорте устройства.
- Использование подсистемы описания методики исследования существенно сократило срок настройки ПО САЭ для нового эксперимента.
- Подсистема и программа управления экспериментом пригодны для использования без изменений в различных САЭ.
- Предложенный способ подготовки и исполнения задания на эксперимент сократил объем необходимого программирования при разработке ПО САЭ при одновременном увеличении возможностей. Без дополнительного программирования реализуются: возможность вмешательства в автоматическую работу ПО САЭ с целью скорректировать содержание выполняемого задания; автоматическое определение точки входа для рестарта ПО САЭ с известными и восстановимыми потерями данных после сбоя питания, устранимого отказа ЭВМ и др.

Эффективность предложения об использовании подсистемы подготовки задания подтверждается выполненными в последние годы в других организациях работами для систем на установке J-PARC/MLF [64] и находящейся в разработке подсистемы составления задания на эксперимент для ISIS [65]. Однако предложенная в диссертации подсистема имеет принципиальные отличия, заключающиеся в том, что в работах [64,65] не ставятся задачи автоматической компоновки ПО САЭ и динамического связывания компонентов – задание на эксперимент и конфигурация ПО САЭ в

работах [64,65] описывается скриптами и конфигурационными файлами, а последовательностью операций в эксперименте управляет интерпретатор. Недостатки такого подхода обсуждены выше.

Предложенный способ описания методики эксперимента не зависит от области приложения. Программная реализация подсистемы описания методики исследования выполнена в общем виде и может использоваться при разработке ПО САЭ для экспериментов в другой проблемной области.

Глава 4. Методы построения распределенных средств обеспечения взаимодействия компонентов системы автоматизации экспериментов для физики низких энергий

4.1. Постановка задачи

Современные системы автоматизации экспериментов реализуются в виде распределенных приложений, в которых необходима организация взаимодействия между процессами (IPC), 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, т.к. позволяют вызвать метод удаленного объекта и, по сравнению с классическим RPC в конфигурации клиент-сервер, обеспечивают лучшую прозрачность. Однако, как и в RPC, взаимодействие процессов выполняется по схеме “один с одним”, в некоторых технологиях – синхронно. Для САЭ это является ограничением, т.к. 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, известные технологии разработки распределенных систем предоставляют возможность статического и динамического связывания компонентов для удаленного вызова процедур. Способ динамического связывания компонентов САЭ является критическим свойством, от которого зависит возможность использования компонентов в других проектах. Способ динамического связывания, используемый в популярных сетевых технологиях

(например, CORBA, Ice, DCOM), для САЭ является избыточно сложными и может быть существенно упрощен, если учесть специфику САЭ [7,14]. Эти (и др.) соображения стимулировали выполнение разработки специального комплекса средств, обеспечивающих взаимодействие компонентов DiCME (Distributed Components Messaging Environment) для использования в САЭ [10]. Помимо решения задачи динамического связывания компонентов для удаленного выполнения процедур, компонент DiCME должен также поддерживать технологию разработки распределенного ПО САЭ. Для этого требуется реализовать функции автоматического поиска и объединения компонентов в ПО САЭ.

Возможна и другая трактовка задачи, поставленной в данной главе. По сути, разрабатываемое ПО САЭ является распределенным пакетом прикладных программ, специализированным для решения задачи построения линейки продуктов – программных систем автоматизации разных экспериментов. Для этого пакета нужно разработать такие средства управления, которые обеспечат автоматическое построение распределенных систем автоматизации разных экспериментов из расширяемого набора унифицированных компонентов, и их работу в соответствии с описанием задания на эксперимент.

4.2. Особенности САЭ, влияющие на алгоритмы средств обслуживания взаимодействия компонентов ПО САЭ

На основании анализа функций компонентов, используемых в ПО различных САЭ, в главе 2 сформулированы две основные группы компонентов САЭ по принадлежности к логике приложения:

1. основные компоненты с детерминированным характером взаимодействия, реализующие основную задачу эксперимента (основную 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 процедуры основных компонентов), реализующие вспомогательную логику – сервисные функции, обработку нештатных ситуаций и др., не влияющие (в штатных условиях) на выполнение основной задачи.

Помимо этого, в главе 2 выявлены особенности способа взаимодействия компонентов основной логики. Для разработки алгоритмов распределенных средств обслуживания взаимодействия компонентов существенны следующие особенности:

- схема взаимодействия компонентов определена основной логикой эксперимента и фиксирована; способ их взаимодействия – “один к одному”;
- в процессе взаимодействия клиент запрашивает выполнение процедуры и нужна гарантированная доставка сообщений – запроса клиента и ответа сервиса;
- в качестве ответа (результата вызова процедуры) клиенту нужен только сигнал завершения выполнения запроса;
- нет существенных оснований к тому, чтобы в тело кода клиента вводить информацию о конкретном связанном сервере (например, адрес) и наоборот; мы можем иметь дело со слабо связанными компонентами.

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

1. схема взаимодействия вспомогательных компонентов не фиксируется;
2. способ взаимодействия “один ко многим”;
3. в составе группы два типа компонентов – источники и потребители информации; в процессе взаимодействия компоненты-источники публикуют информацию, а не вызывают процедуры потребителей;
4. источник информации 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 основной логики эксперимента.

4.3. Требования к средствам обеспечения взаимодействия компонентов

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:

- автоматический поиск и динамическое связывание компонентов; автоматический поиск существенно улучшает эксплуатационные свойства системы
- асинхронный механизм вызова процедур, т.к. 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;
- возможность передать информацию нескольким другим процессам;
- одна и та же среда взаимодействия должна обрабатывать все обмены в рамках ПО САЭ, т.к. гомогенную систему намного легче программировать и поддерживать;
- 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 в сети;
- возможность перемещения процесса с одной машины на другую; это облегчит устранение аварийных ситуаций, приводящих к выходу из строя ЭВМ;

- 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 и другого типа;
- потеря процесса, ЭВМ или разрыв сетевой связи не должна приводить к разрушению остальной части системы;
- автоматическая адаптация к используемой конфигурации.

Реализация средств обеспечения взаимодействия, отвечающих перечисленным требованиям, значительно упростила разработку, облегчила эксплуатацию и развитие САЭ, улучшила эффективность и надежность разрабатываемых систем.

4.4. Известные решения

DIM. Приведенному составу требований в некоторой степени соответствует среда взаимодействия компонентов DIM (Distributed Information Management), описание которой опубликовано в работе [67]. Однако эта среда предназначена для использования в физике высоких энергий, используется в крупных системах (например, [68,69], насчитывающих 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, и возникает источник ошибок оператора;
- заполнение списков серверов разработчиками вносит статическую связанность, в результате реализация среды взаимодействия теряет универсальность и ухудшаются условия использования компонентов в других системах. Для уникальных систем [68,69 и др.] потеря этой возможности 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 основной и вспомогательной логике, и это усложняет развитие прикладной системы, а также уменьшает возможность использования компонентов в других системах;
- отсутствует базовая технологическая поддержка ПО САЭ.

CORBA. Одна из важнейших для ПО САЭ характеристик средств обеспечения взаимодействия – способ связывания компонентов, гарантирующий нормальное функционирование приложения. Изменение методики эксперимента свойственно самому процессу экспериментальных исследований. Поэтому для ПО САЭ нужен механизм, обеспечивающий динамическое связывание компонентов в соответствии с изменениями основной логики эксперимента. Наиболее популярная технология CORBA [70] реализует динамическое связывание через специальный набор средств: DII на стороне клиента, и DSI на стороне сервера. Через эти интерфейсы для настройки каждого взаимодействия осуществляется ряд вызовов, во время которых выясняются имя процедуры сервера, типы и значения аргументов, тип результата, выполняется вызов процедуры и, наконец, получение результата. Для поддержки сценария взаимодействия клиентский и серверный компоненты дополняются избыточным кодом, не связанным с реализацией логики приложения. Технология CORBA обеспечивает практически полную свободу при модификации приложения, однако цена этой свободы следующая:

- Средства обслуживания коммуникаций и дополнительный код в клиентском и серверном компонентах привязываются к данной технологии.
- Существенно усложняется программирование реализации, как средств обеспечения 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 [29].

Анализ, выполненный автором в работе [7], показал, что для САЭ динамическая настройка взаимодействия компонентов может быть выполнена по более простому алгоритму. К подобному выводу пришли также авторы работы [47] и разработали свой алгоритм (Inversed Injectijon), упрощающий сценарий настройки удаленного вызова процедур благодаря передаче инициативы в настройке вызова серверу вместо клиента и использованию карты взаимосвязей компонентов. Использование карты взаимосвязей вносит статическую связанность. Поэтому, в отличие от работы [47], в диссертации для устранения связанности, унификации компонентов и упрощения их разработки полностью устранено участие и клиента, и сервера в настройке удаленного выполнения процедур.

В связи с этими было принято решение о выполнении разработки распределенных средств обеспечения взаимодействия компонентов DiCME, учитывающих специфику задач автоматизации экспериментальных исследований. Обоснование решения приведено диссертантом в работах [7,14].

4.5. Структура ПО САЭ и его базовая технологическая поддержка

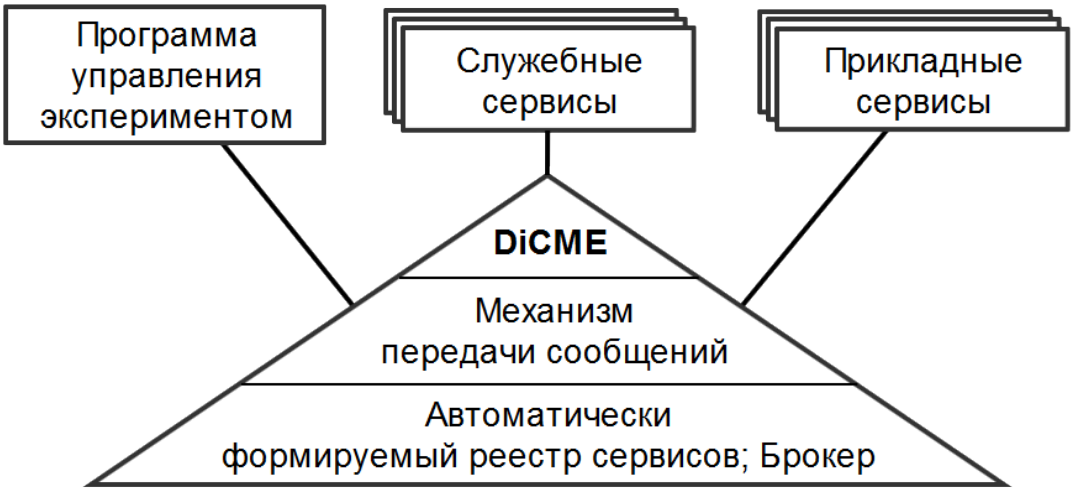

Рис. 4.1. Структура ПО САЭ

В соответствии с выводами, сделанными в главе 1, при разработке структуры ПО САЭ использованы принципы, сформулированные в концепции SOA. Помимо механизма передачи сообщений, потребовались функции формирования реестра компонентов, брокер сообщений и некоторые другие службы (рис. 4.1).

Эти функции являются традиционными для технологии разработки распределенных систем. Базовую технологическую поддержку этой структуры обеспечивает компонент DiCME. Для обеспечения автоматического формирования реестра компонентов (и объединения компонентов в систему) каждый компонент на этапе инициализации своего интерфейса к DiCME активирует функцию Маяк. Через Маяк компонент периодически посредством multicast-сообщения информирует другие части ПО САЭ о своей доступности – сообщает свой идентификатор (GUID) и тип.

Для подключения к DiCME в компонентах разного типа используется от 4 до 6 операторов. Использование открытых сетевых технологий поиска компонентов и передачи сообщений существенно упростило реализацию компонента DiCME.

4.6. Метод динамического объединения компонентов в ПО САЭ

Для динамического объединения компонентов в ПО САЭ средствами DiCME выполняется поиск компонентов при помощи блока, реализующего протокол SLP (Service Location Protocol). Данный блок содержит функцию Маяк, которая 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 компонентов, 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 идентификатору (как в случае с поиском компонентов, перечисленных в задании на эксперимент), либо по его типу (например, поиск программы управления экспериментом, представляемой в составе ПО САЭ в единственном экземпляре). Для компонентов, использующих протоколы, отличные от JSON-RPC (например, файловый сервер), используется только часть функционала DiCME, отвечающая за механизмы динамической компоновки (функция Маяк). Это обеспечивает возможность использования сторонних компонентов без их модификации или создания "прослоек" для адаптации сторонних компонентов.

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

4.7. Интерфейсы и логика взаимодействия компонентов ПО САЭ

Для двух групп компонентов (основных и вспомогательных) можно определить (и фиксировать их состав) варианты интерфейсов. Для основных компонентов достаточны интерфейсы, обеспечивающие:

- удаленное выполнение процедуры сервера – передачу запроса на выполнение действия;
- передачу сигнала завершения выполнения действия;
- публикацию информации, адресованной вспомогательной логике.

Для удаленного выполнения процедуры можно использовать компоненты несколькими различными способами. В CORBA используется вызовов процедуры. Для этого необходимо выполнить ряд обращений к библиотеке интерфейсов и несколько обращений к диску, и это выливается во множество машинных циклов. В данной работе вместо вызовов процедур используется обмен сообщениями с последующей интерпретацией сообщения в теле компонента. Это дает преимущество в виде относительной независимости запрашивающего и отвечающего компонентов. Компоненты, 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 и состояния объекта не предпринимать никакого).

В рамках основного 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 основных компонентов может обслуживаться одним интерфейсом, обеспечивающим передачу текстового сообщения, содержащего идентификатор адресата, информацию о типе компонента и типе сообщения. В случае запроса на удаленное выполнение процедуры сообщение дополняется списком параметров, после завершения выполнения процедуры клиенту возвращается только сигнал синхронизации, разрешающий продолжение развития основной логики. Детализирующая информация о результатах выполнения процедуры (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;

- сигнала о возникновении ошибки при выполнении процедуры 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 сервер. Источники и потребители должны быть связаны по типу информации. Таким образом, 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.

4.8. Метод динамического связывания компонентов основной логики ПО САЭ

Предлагаемый метод динамического связывания компонентов использует идентификаторы компонентов, в отличие от использования сетевых адресов, принятого в среде DIM [67] или технологиях CORBA, Ice и др.

Основанием для разработки нового метода динамического связывания компонентов в ПО САЭ послужили сделанные в главе 2 выводы о том, что динамическое связывание компонентов для удаленного выполнения процедур необходимо только в основной логике ПО САЭ, и в качестве результата выполнения процедуры для синхронизации работы вызывающему компоненту требуется только сигнал завершения работы процедуры. Реализация этого метода связывания основана на использовании специальной модели программы управления экспериментом и описания методики эксперимента. В соответствии

с этой моделью, программа управления экспериментом выполняет два процесса и управляет их чередованием:

1. формирование условий, в которых должна быть выполнена регистрация экспериментальных данных;
2. регистрация данных.

Такой программе управления для конкретизации функционального содержания исполняемых процессов требуется описание условий регистрации, и от каждого процесса – сигнал завершения работы, который будет синхронизировать последовательность действий. В главе 3 предложен унифицированный вариант подсистемы описания методики исследования, которая создает файл задания на эксперимент в виде списка описаний условий, в которых должна быть выполнена регистрация данных. В описании условия присутствуют идентификаторы компонентов (включаются автоматически), списки параметров и их значения, определяющие нужные условия. Идентификатор введен для поиска исполняющего компонента и связывания с ним всегда одного и того же компонента – программы управления экспериментом, а список параметров должен быть передан исполняющему компоненту для интерпретации и выполнения действия. В итоге роль программы управления экспериментом, вместо вызова процедур, сводится к диспетчеризации используемых компонентов, список которых определяется методикой конкретного эксперимента. Благодаря этому исключен диалог между компонентами, используемый, например, в технологии CORBA для настройки динамического связывания компонентов и удаленного вызова процедур.

4.9. Метод динамического связывания компонентов вспомогательной логики ПО САЭ

Наиболее существенным отличием способа реализации вспомогательной логики от основной является необходимость передать информацию нескольким

процессам, состав которых, в общем случае, источнику информации неизвестен. Можно решить эту задачу несколькими способами, например:

1. поллинг – каждый потребитель периодически опрашивает источник информации и получает ее по готовности;
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 компонент обслуживает всех “подписавшихся” потребителей при появлении этой информации.

Очевидно, 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, т.к. клиенту не нужно ждать ответа сервера, и несколько клиентов получают обновления одновременно. Широковещательные сообщения с использованием протокола UDP – достаточно простой способ решить задачу, однако использование UDP не гарантирует доставку. Потеря сообщения вспомогательным компонентом не приводит к нарушению выполнения логики эксперимента. Однако в диссертации 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 ПО 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 средств обеспечения взаимодействия.

4.10. Задачи, решаемые средствами обслуживания взаимодействия компонентов

Требования к средствам обслуживания взаимодействия компонентов приведены в разделе 4.3. Средства обслуживания взаимодействия процессов компонент DiCME и библиотека функций должны решать следующие задачи:

- кэширование списка активных компонентов и обновление его с заданной периодичностью;
- автоматический поиск и подключение компонентов к системе;
- динамическое связывание компонентов;
- обеспечение асинхронного выполнения процедуры сервера;
- передачу синхронизирующих сообщений о завершении выполнения вызванной процедуры;
- публикацию информации различного типа: о зарегистрированных экспериментальных данных, возникших ошибках, отказах и др.;
- регистрацию “подписчиков” на предоставление публикуемой информации определенного типа;
- передачу подписчикам информации по мере ее появления.

Для решения этих задач в компонент DiCME включены средства передачи сообщений, блок автоматического формирования и динамической коррекции реестра активных компонентов, брокер сообщений и др.

4.11. Использование динамического связывания компонентов основной логики ПО САЭ

Для управления экспериментом оператору предоставлены команды Start, Pause, Continue, End. Команда Start от интерфейса пользователя поступает в компонент управления экспериментом, который реализует основную логику, используя файл задания. Задание на эксперимент содержит описания всех нужных состояний установки, при которых будет выполнена регистрация

экспериментальных данных, и обеспечивает информационную поддержку динамического связывания компонентов (программы управления экспериментом с компонентами управления окружением образца и DAQ).

Список состояний (описан в главе 3 и работах [11,13]) разбирает программа управления экспериментом. Алгоритм работы программы при реализации основной логики показан на рис. 4.2.

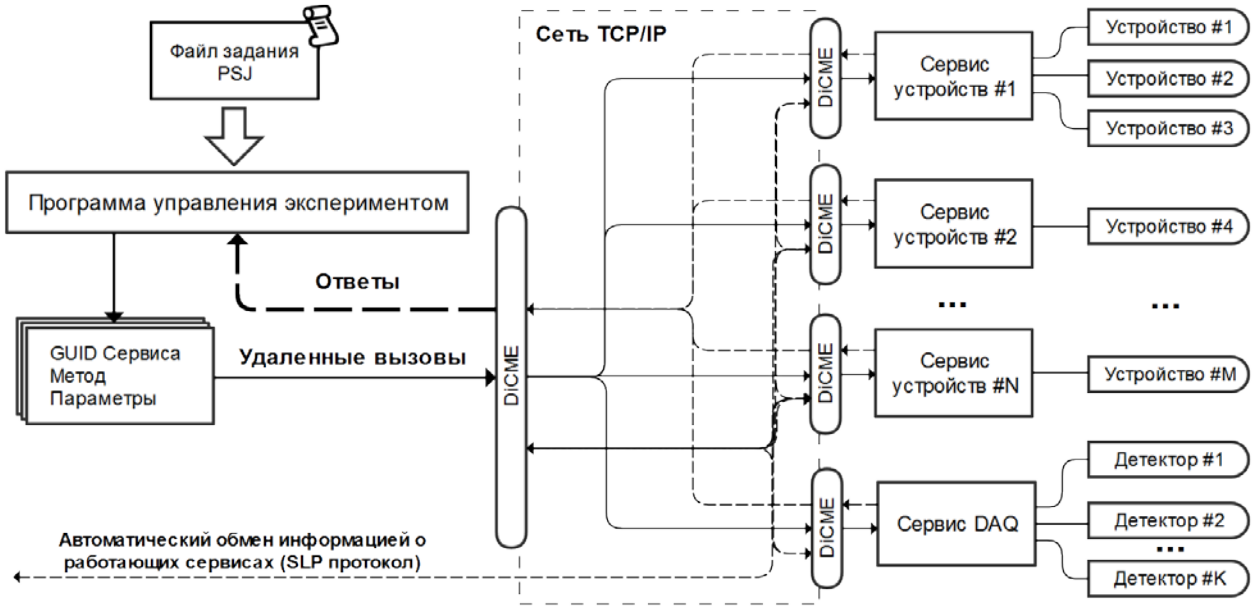

Рис. 4.2. Использование компонента DiCME при реализации основной логики

Программа управления экспериментом выбирает описание очередного состояния установки, расчленяет его на описания отдельных условий и передает описания условий компоненту DiCME. DiCME извлекает из описания условий идентификатор компонента, находит компонент и передает ему описание условий для выполнения нужных действий. После ответа всех компонентов, управляющих условиями, сигналами завершения работы, программа управления экспериментом запускает экспозицию данных и ожидает сигнал завершения 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 средства обеспечения взаимодействия DiCME.

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

4.12. Алгоритм динамического связывания компонентов вспомогательной логики ПО САЭ

Любой компонент может подписаться на получение информации определенного типа, выполнив обращение к диспетчеру событий (через специальную процедуру 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. 4.3. В использованном для вспомогательной логики механизме связывания отсутствует жесткое требование наличия потребителя публикуемой информации.

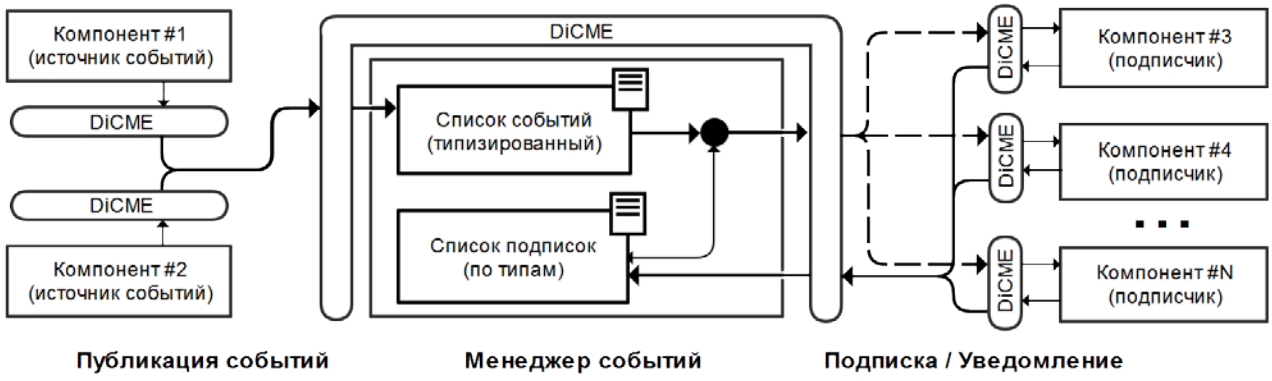

Рис. 4.3. Алгоритм связывания компонентов вспомогательной логики

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 такого сообщения.

4.13. Параметры компонента DiCME и оценка потерь процессорного времени на обслуживания взаимодействия

1. При обслуживании взаимодействия компонентов DiCME выполняет два потока, назначение которых следующее:

- передача сообщений;
- прием сообщений.

Для обеспечения устойчивости работы DiCME в случае зависания одного из обслуживаемых компонентов введен таймер, управляющий поочередным выполнением этих потоков. Благодаря этому получена возможность идентифицировать сбойный компонент, выдать диагностическое сообщение и изменить условия для продолжения работы. Период таймера параметризуется.

Для обслуживания очередей запросов в процессах передачи и приема сообщений введено две разные дисциплины, используемые в двух режимах работы DiCME:

- синхронный режим работы, при котором следующее сообщение из очереди запросов передается после получения синхронизирующего сигнала о завершении выполнения предыдущей заказанной операции:
- асинхронный, при котором DiCME выполняет передачу всех сообщений, присутствующих в очереди заказов, и лишь затем включает таймер переключения процессов.

Синхронный режим используется в том случае, когда контроллер управляет несколькими устройствами и по техническим причинам (например, ограничения по току блока питания) возможно только последовательное включение этих устройств.

2. Время инициализации DiCME определяет задержку готовности САЭ к обслуживанию процессов взаимодействия, отсчитываемую от момента старта системы. Это время зависит от следующих параметров:

tB – интервал срабатывания функции Маяк. При каждом срабатывании локальный сервис OpenSLP обновляет запись о компоненте;

tL – интервал срабатывания Локатора – функции, определяющей адрес компонента. При каждом срабатывании Локатора происходит опрос и корректировка списка активных компонентов прикладной системы;

tD – время хранения записи о компоненте в списках (параметр функции WatchDog).

Максимальное время инициализации сети, после которого все компоненты знают адреса друг друга, равняется *tL*. Чем меньше значение *tL*, тем быстрее выполняется инициализация и DiCME быстрее реагирует на появление/исчезновение компонента из сети, но при этом больше загружается процессор и сеть. Для задач спектрометрии было выбрано значение *tL* = 1мс, при котором нагрузка на систему незначительна.

Для задач, которые требуют на порядки большую скорость инициализации системы, можно отредактировать OpenSource библиотеку OpenSLP и добавить соответствующий спецификации протокола SLP механизм оповещения всех компонентов сети о наличии сервиса. В текущей версии библиотеки OpenSLP механизм регистрации информации о компонентах работает только в режиме "по запросу", несмотря на то, что механизм оповещения по сети присутствует.

3. При синхронном режиме взаимодействие выполняется в две итерации. Исключение – сообщения типа notification. Каждая итерация – это один HTTP-запрос (одно TCP-подключение) с получением ответа (результата запроса). Результат запроса – информация о корректности полученных данных, наличии запрошенной процедуры у компонента-сервиса и служебные сообщения компонента DiCME. Как правило, внутренний диалог между

вызывающим и исполняющим компонентами в средствах DiCME происходит по следующей схеме:

(ВК) → (ИК). Новое соединение: выполнить процедуру A с параметрами B.

(ИК) → (ВК). Ответ: Запрос принят. Разрыв соединения.

(ИК) Идет выполнение процедуры... Таймер переключения процессов работает.

(ИК) → (ВК). Новое соединение: Результат запроса = P.

(ВК) → (ИК). Ответ: Результат принят. Разрыв соединения. Включение таймера переключения процессов.

Максимальное время *t* выполнения пакета соединений для одиночного запуска измерения можно приблизительно посчитать по следующей формуле:

$$t = (tc + 2 \cdot tl + 4 \cdot tw) \cdot n + Tt,$$

где

tc – время установления TCP соединения;

tl – латентность сети – время, затраченное на передачу данных;

tw – интервал опроса очереди сообщений;

n – количество компонентов-исполнителей, участвующих в процессе;

Tt – сумма времен исполнения всех процедур, включенных в данный процесс.

Суммарные сетевые задержки *td*

$$td = (tc + 2 \cdot tl + 4 \cdot tw) \cdot n \quad (1)$$

по крайней мере на порядок меньше суммарного времени выполнения процедур, задающих условия регистрации *Tt*, и интервала опроса очереди сообщений, ими можно пренебречь.

4. Асинхронный режим является предпочтительным при управлении выполнением основной логики, т.к. обеспечивает более экономное использование времени работы базовой установки – источника излучения. Диалог между компонентами в этом режиме ведется по схеме:

(ВК) → (ИК-1). Новое соединение: выполнить процедуру М1 с параметрами П1. Ответ: Запрос принят. Разрыв соединения. Таймер переключения процессов остановлен.

...

(ВК) → (ИК-n). Новое соединение: выполнить последнюю в списке процедуру М-n с параметрами П-n. Ответ: Запрос принят. Разрыв соединения. Таймер переключения процессов включен.

(ИК-1...ИК-n) Идет выполнение процедур управления условиями регистрации... Таймер переключения процессов работает.

(ИК-i) → (ВК). Новое соединение: Результат запроса = R-i.

(ВК) → (ИК-i). Ответ: Результат принят. Разрыв соединения.

Для этого режима время выполнения процесса определяется временем выполнения компонентом DiCME передачи сообщений всем исполнителям процесса и временем исполнения наиболее медленной процедуры

$$t = MAX_i^n\left(\left(tc + 2 \cdot tl + 4 \cdot tw\right) \cdot i + T_i\right),$$

где

Ti – время исполнения процедуры управления условием.

Таким образом, в обоих случаях время выполнения потока определяется характеристиками исполняемых процедур.

На практике максимальная длина очереди незавершенных диалогов в рассматриваемых системах находится в пределах нескольких десятков. В случае (гипотетическом), когда количество незавершенных диалогов 100 и больше, а выполнение команд занимает секунды, целесообразно увеличить задержку *tw*, чтобы снизить холостые проверки диалогов на завершенность и тем самым снизить нагрузку на процессор ЭВМ. В большинстве же случаев оптимальным значением будет *tw* = 1мс, что обеспечивает относительно высокую скорость работы средств передачи сообщений (~4мс) и низкую нагрузку на процессор.

5. Важной характеристикой средств обеспечения межкомпонентного взаимодействия является их вклад в загрузку процессора. Оценка занятости процессора выполнена на основании измерений времени взаимодействия компонентов на макете распределенной САЭ, состоящей из 50 компонентов (что заведомо превышает размеры реального ПО САЭ), и вычислений по ее алгоритмической модели. На рис. 4.4 показана суммарная загрузка, обусловленная работой подсистемы регистрации данных DAQ с использованием компонента DiCME во время работы САЭ в эксперименте (кривая DAQ), и вклад в загрузку, вызванный использованием компонента DiCME. Потери времени на сетевые операции *td* вычислялись по формуле (1).

Занятость процессора подсистемой DAQ на графике рис. 4.4 вычислена для системы, описанной в работе [19], при следующих условиях:

интенсивности событий на входе системы: ~2,5 МГц;

время обработки 1-го события (описываемого 4 байтами): 30 нс;

скорость передачи данных по каналу USB: 480 Мбит/с.

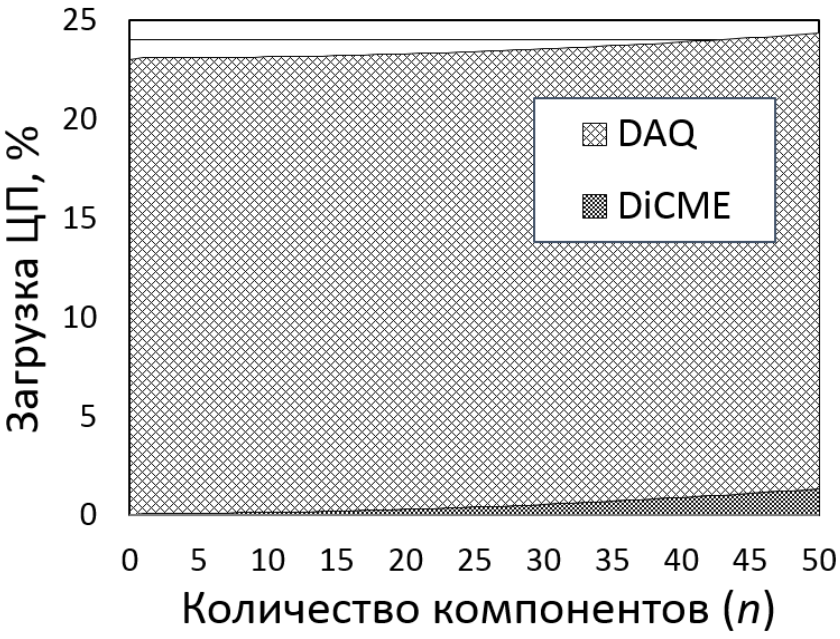

Рис. 4.4. Влияние использования компонента DiCME на загрузку ЦП

Видно, что 1) суммарная загрузка процессора составляет ~23%, т.е. на вспомогательные операции остается достаточно времени; 2) потери на

использование DiCME на порядок меньше времени обработки потока регистрируемых данных.

4.14. Примеры использования компонента DiCME

Ниже приведены тексты двух примеров подключения к компоненту DiCME и использования его возможностей клиентским и серверным компонентами.

4.14.1. Пример для компонента-клиента

```
const
// ID клиента:
  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 писка параметров из файла задания:
ParametersList  :=  '{device:"polarizer",param:"angle",value:20}';
     // JSON - формат
// Передача сообщения:
dicme.Execute(ServiceGUID,'SetParam',ParametersList,False, nil);

// Обработчики событий - прием ответа:
procedure ClientOnResponse(const Seed: Pointer; Response: String);
begin
  // some code ...
end;
procedure ClientOnError(const Seed: Pointer; Response: String);
begin
  // some code ...
end;
```

4.14.2. Пример для компонента-сервера

```
const
// ID сервиса:
  ServiceGUID: string = 'db727048-d3db-4f27-8d5c-0eeb036f6fad';
  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 процедуры передачи сообщений:
  dicme.RegisterRPCMethod('SetParam');
// Включение Маяка:
  dicme.ChangeBeaconState(True);

// Обработчики событий - прием и обработка сообщений:
procedure ControllerMethod (const MethodName: string;
const ParametersList: string;
isNotify: Boolean; var Result: string);
begin
  // При возникновении данного события параметр ParametersList
  // в данном примере будет иметь значение
  //          {device:"polarizer",param:"angle",value:20}
  // some code ...
  Result := '{status:"ok"}';              // JSON - формат
end;
procedure  ControllerOnError  (const  Seed:  Pointer;  Response:
String);
begin
  // some code ...
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.

4.15. Выводы

Средства обслуживания взаимодействия процессов является связующим слоем между компонентами, который облегчает программирование, компоновку и модернизацию программной системы в целом, повышает ее надежность, облегчает возможность использования компонентов в других проектах.

Основные результаты, полученные в данной главе [10]:

1. Разработан метод автоматической компоновки распределенного ПО САЭ в соответствии с заданием на эксперимент в условиях изменения задания при переходе от одного эксперимента к другому. Метод основан на использовании компонентами широковещательных сообщений с информацией о своих идентификаторах и сетевого протокола поиска компонентов SLP [10,15]. Автоматизация компоновки существенно облегчила работу пользователей.
2. Разработан и впервые применен новый метод динамического связывания компонентов для удаленного выполнения процедур в распределенном ПО САЭ и унифицированные средства обслуживания межкомпонентного взаимодействия [10,15]. В отличие от методов динамического связывания компонентов, используемых в технологиях DCOM, CORBA, Ice, в разработанном методе:
   - вместо сетевого адреса компонента используется его уникальный идентификатор;
   - параметры удаленного вызова процедуры интерпретируются в вызываемом компоненте, а результаты работы вызываемой процедуры представляются в двух сообщениях, содержащих: 1) обязательный сигнал завершения работы, возвращаемый вызывающему компоненту, и 2) текстовую информацию, детализирующую результаты работы, направляемую компонентам вспомогательной логики. Благодаря этому устранен диалог между компонентами, используемый в известных технологиях DCOM, CORBA, Ice для настройки удаленного вызова процедуры.

В отличие от методов динамического связывания, используемых в CORBA, DCOM, одновременно использованы две дисциплины динамического связывания: связывание компонентов основной логики с обязательным ответом компонента-исполнителя процедуры, и связывание компонентов вспомогательной логики по “подписке”, при котором компонент-источник информации не требует ответного сообщения.

Разработанный метод динамического связывания и структура ПО САЭ с использованием разных дисциплин выполнения основной и вспомогательной логики предоставляют свободу в развитии основной и вспомогательной логики ПО САЭ без изменения других компонентов ПО САЭ и существенно упрощает алгоритмы средств, обеспечивающих взаимодействие компонентов.

Впервые динамическое связывание с использованием идентификатора процедуры, а также подкачка и смена используемых подпрограмм (по сути – автоматическая компоновка прикладной программы) были реализованы в интерпретирующей системе ИС-2, разработанной М.Р. Шура-Бура для ЭВМ М-20 [17]. Динамическое связывание в системе ИС-2 выполнялось для вычислительных программ, компонуемых в адресном пространстве одной ЭВМ, вызов процедур выполнялся в синхронном режиме. Отличие методов, разработанных в диссертации, заключается в следующем:

- динамическое связывание и объединение компонентов в систему развиты для использования в распределенном ПО САЭ;
- реализован асинхронный режим исполнения процедур;
- вместо используемой в ИС-2 фиксированной таблицы характеристик библиотеки процедур использован динамически составляемый реестр активных компонентов;
- результат выполнения процедуры представлен двумя сообщениями: 1) обязательное сообщение о завершении работы процедуры, адресуемое вызывающей программе, и 2) необязательное сообщение, адресуемое вспомогательным программам.

Численные оценки на модели ПО САЭ, использующей компонент DiCME, показывают: 1) в синхронном и асинхронном режимах управления устройствами сетевые задержки намного меньше времени работы исполняющих устройств; 2) трафик незначительно нагружает процессор.

Разработанные средства взаимодействия позволяет динамически изменять состав и адреса основных и вспомогательных компонентов, что облегчает восстановление работоспособности системы при отказах, улучшает условия эксплуатации и управления.

Компонент DiCME и функции средств 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 прикладных областях.

Глава 5. Алгоритмы, структура компонентов основной логики ПО САЭ и сетевая архитектура системы

5.1. Общие свойства компонентов распределенного ПО САЭ

Каждый компонент разработанного ПО САЭ содержит:

- группу команд (4 – 6 операторов) подключения к DiCME при запуске компонента;
- интерпретатор описаний условий, если в компоненте имеется несколько процедур, которые могут использоваться другими компонентами;
- процедуры, реализующие предусмотренную функциональность.

5.2. Интерфейсы пользователя

5.2.1. Desktop-интерфейс. Интерфейс пользователя передает программе управления экспериментом команды оператора START, PAUSE, CONTINUE, END. Процедуры передачи этих команд относятся к основной логике. В САЭ может присутствовать несколько интерфейсов пользователя одновременно, но активным может быть только один. На рис. 5.1 показан Desktop-интерфейс, используемый экспериментаторами в случае, когда оператор САЭ присутствует в экспериментальном зале [18-23].

Этот же интерфейс используется при проверке системы и отладке новых компонентов.

Помимо операций управления основной логикой (START, …, END), в интерфейс могут быть включены элементы для управления функциями вспомогательной логики (например, операции визуализации состояния системы, данных и др.) – в случае специфического технического задания, однако универсальностью обладает вариант алгоритма, динамически составляющего список вспомогательных компонентов и предоставляющий возможность вызова любого из них. В этом (унифицированном) варианте

используется функция DiCME, с помощью которой в интерфейс считывается список всех активных компонентов и создается выпадающее меню, позволяющее открыть окно интерфейса любого из сервисных компонентов. На рис. 5.2 для примера показано окно компонента визуализации спектров, открытое с помощью выпадающего меню Open SERVICE (см. рис. 5.1).

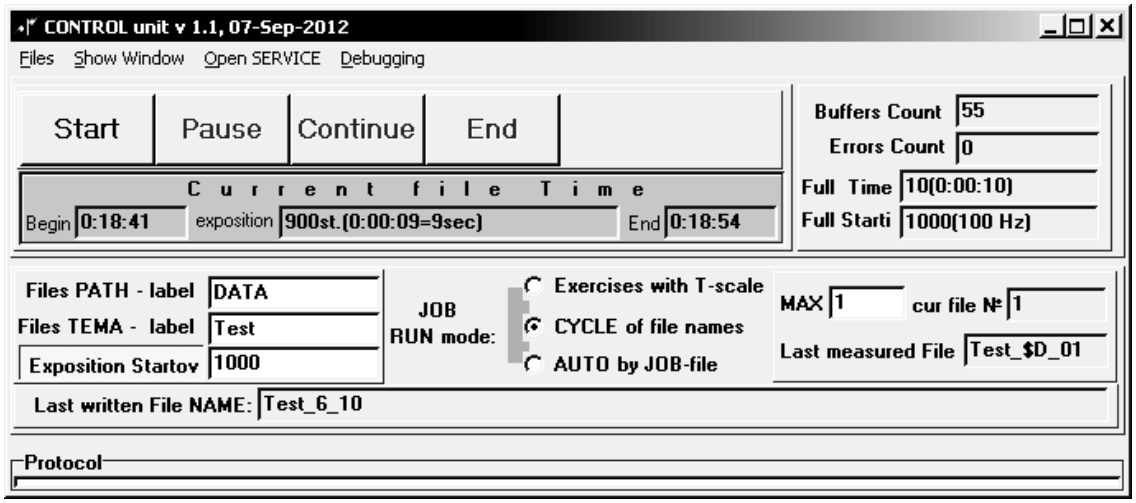

Рис. 5.1. Desktop-интерфейс пользователя

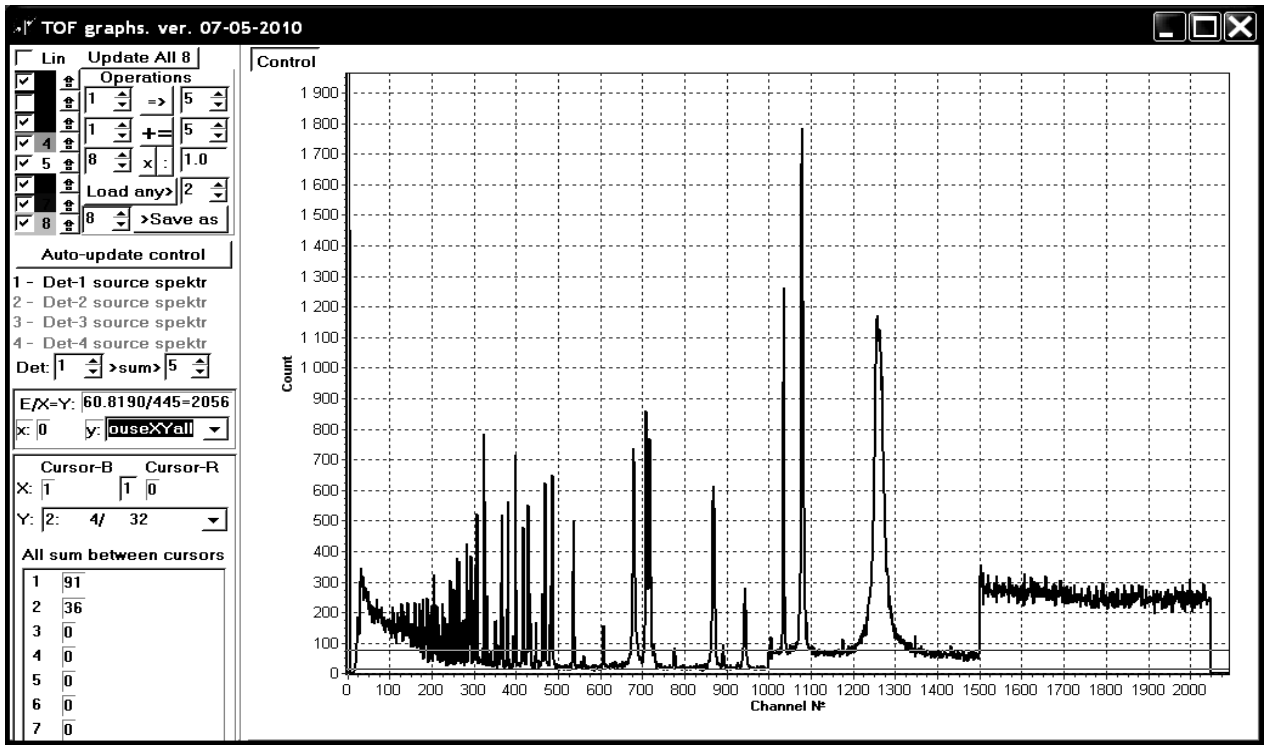

Рис. 5.2. Окно компонента визуализации спектров

5.2.2. Web-интерфейс пользователя. Первый вариант программы дистанционного управления экспериментом в ЛНФ был разработан

соискателем 10 лет назад. В связи с разработкой новых методов построения ПО САЭ, появлением новых сетевых технологий, а также с учетом накопленного опыта дистанционного управления экспериментом, потребовалась новая реализация Web-интерфейса (см. рис. 5.3).

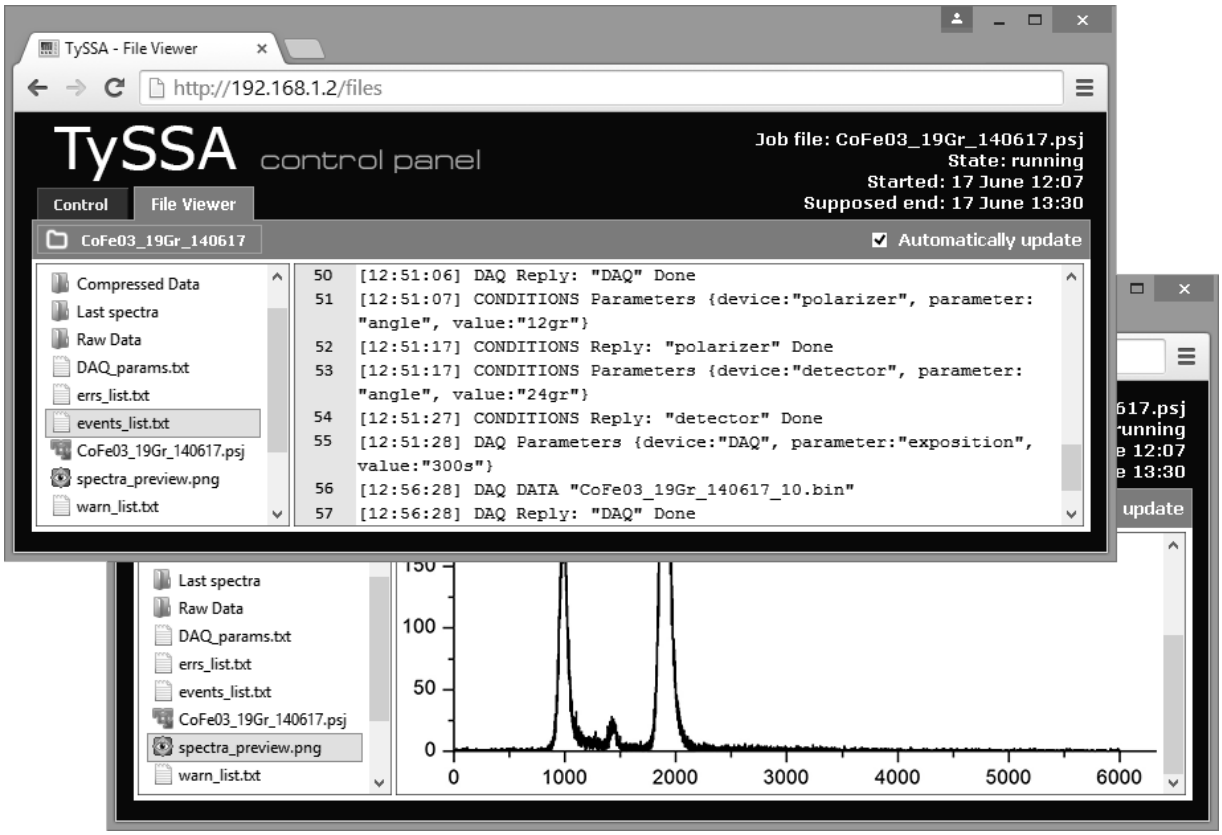

Рис. 5.3. Вид окна Web-интерфейса в режимах визуализации информации менеджера событий и последнего измеренного спектра

В настоящее время в окне интерфейса оставлены две страницы, которые выполняют:

1. взаимодействие с программой управления экспериментом – передачу команд “Start”, “Pause”, “Continue”, “End” (страница “Control”);
2. визуализацию управляющих файлов; информации, регистрируемой менеджером событий (события, ошибки) и др.; поиск, обеспечение доступа и визуализацию персонализированных данных из центрального хранилища (страница “File Viewer”).

Эти возможности интерфейс обеспечивает любой системе, ПО которой построено в соответствии с разработанными в диссертации методами.

Дальнейшее развитие Web-интерфейса автором ведется с использованием DiCME в направлении, не нарушающем его универсальность.

Во время длительных экспериментов оператор, как правило, уходит из экспериментального зала. В связи с этим, для быстрого информирования о событиях, мешающих нормальному прохождению эксперимента, введен упрощенный интерфейс пользователя для мобильных устройств (смартфоны и планшеты), который позволяет дистанционно получить актуальную информацию о ходе эксперимента. Помимо этого, пользователю предоставлена возможность заказать передачу SMS-сообщения на телефон при возникновении событий из состава настраиваемого списка.

5.3. Программа управления выполнением эксперимента и эффективность работы САЭ

5.3.1. Основной алгоритм выполнения задания на эксперимент подробно описан в разделе 3.2. Практика эксперимента требует некоторых добавлений:

- в данный алгоритм легко встраивается возможность пошагового выполнения работы, отключения регистрации или архивирования данных для целей отладки и др. непринципиальные дополнительные операции, характерные для конкретной прикладной области, информация о которых в данной работе для краткости опущена;
- обычной практикой прецизионных ядерно-физических экспериментов является расчленение многочасовой экспозиции в заданных условиях на несколько (обычно одинаковых) более коротких, результаты которых могут быть просуммированы. Для этого в задании указывается, сколько раз повторить все пункты задания (на рис. 5.4 – “цикл по проходам”). Благодаря этому, сравнивая данные в отдельных файлах, полученных при одинаковых условиях, можно обнаружить возможный дрейф фона, эффективности детекторов и др. эффекты и отфильтровать некорректные данные. На рис. 5.4 представлена схема работы программы управления экспериментом.

Данная программа протоколирует работу: запоминает последнюю команду оператора, номер последнего завершенного состояния системы (выполненной строки задания) и некоторые другие данные и использует их для автоматического формирования названий файлов. Благодаря этому при сбоях возможно перезапустить систему – она продолжит работу, начиная с последнего незавершенного состояния.

Рис. 5.4. Схема работы программы управления экспериментом

5.3.2. Важной характеристикой ПО САЭ является эффективность его работы во время эксперимента. Стоимость времени работы источников нейтронов высока (для реактора ИБР-2 – >1200 р./час для каждого из ~10 спектрометров). В таких условиях оценку эффективности ПО САЭ дает коэффициент использования времени базовой установки K_e:

$$K_e = \frac{tr}{tr + ts},$$

где

tr – продолжительность регистрации данных,

ts – время подготовки условий регистрации данных.

На рис. 5.5 показаны результаты расчетов эффективности K_e по формуле

$$K_e = \frac{tr}{tr + n \cdot tk + \sum_{i=1}^{n} td_i},$$

где

$$ts = tk \cdot n + \sum_{i=1}^{n} td_i$$

– время подготовки состояния экспериментальной установки.

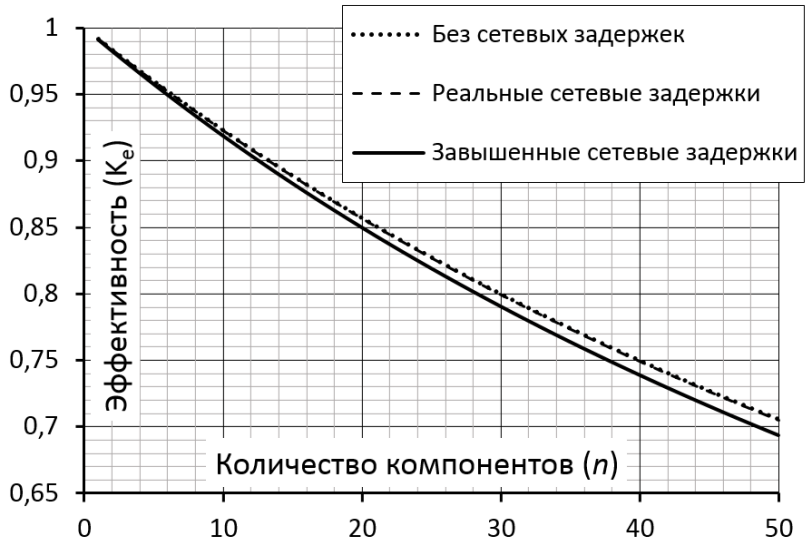

Рис. 5.5. Оценка влияния использования компонента DiCME на эффективность работы САЭ

Точками проведена кривая эффективности K_e без учета сетевых задержек. Пунктирная кривая, вычисленная с учетом сетевых задержек, сливается с кривой без сетевых задержек. Сплошной линией показана кривая эффективности, вычисленная с завышенными сетевыми задержками для иллюстрации характера зависимости вклада сетевых задержек в K_e от количества компонентов. Видно, что использование унифицированного механизма сетевого межкомпонентного взаимодействия, реализуемого в DiCME, вместо прямого вызова процедур, в реальных условиях оказывает пренебрежимое влияние на эффективность работы ПО САЭ.

Многократное выполнение операций управления условиями регистрации данных, характерное для прецизионных экспериментов, может привести к значительным потерям в использовании времени источника нейтронов. Для грубой оценки эффективности работы САЭ при планировании конкретного эксперимента может быть использовано упрощенное выражение

$$K_e = \frac{tr}{tr + td},$$

где $td = MAX(td_i)$ при параллельной работе устройств управления условиями регистрации и $td = \sum td_i$ при последовательной.

5.4. Подсистема регистрации данных DAQ

Состав данной подсистемы следующий:

- группа компонентов, выполняющих ввод, преобразование и архивирование потока данных от детекторной системы;
- диалоговая программа параметризации компонентов подсистемы.

Подсистема регистрации данных является основной составной частью САЭ. Для простых экспериментов, не требующих автоматического изменения условий регистрации данных во время эксперимента, достаточно этой подсистемы и простого интерфейса пользователя для ее управления, чтобы выполнить измерения в фиксированных условиях. Но в настоящее время такое использование DAQ встречается только в отладочных работах.

В главе 2 отмечено, что только между компонентами этой подсистемы требуется обеспечить быструю передачу больших объемов данных. Для остальных компонентов САЭ достаточна длина сообщения в пределах десятков килобайт и скорость передачи, обеспечиваемая протоколом TCP/IP. В связи с этим было принято решение группировать компоненты ввода, преобразования и архивирования данных в подсистему (такое группирование применяется и в других работах, например, работу [71]), и для этой подсистемы разработать средства быстрого обмена данными. Такое решение имеет побочное положительное следствие – упрощаются алгоритмы, и повышается надежность реализации DiCME.

В разделе 5.4.2 перечислены часто встречающиеся варианты способов включения подсистем DAQ в САЭ, способы синхронизации их работы и описана разработка алгоритма быстрой передачи данных между входящими в состав DAQ компонентами.

На одном и том же спектрометре может быть использовано более одного детектора [72,73]. Несмотря на многообразие детекторных систем, может быть

введено конечное количество форматов регистрации сырых данных (например, регистрация событий и/или гистограмм, разреженных матриц [74]). Это является достаточной основой для построения унифицированного интерфейса к используемым в организации средствам регистрации данных (DAQ) и для разработки типовых программных модулей сжатия и архивирования данных.

5.4.1. Обмен данными между компонентами DAQ и пропускная способность САЭ

Выше было отмечено, что скорость обмена данными между компонентами DAQ является критическим параметром подсистемы. Для трех ее процессов (ввод, преобразование, архивирование данных) желателен непосредственный доступ к данным. Очевидно, это самый быстрый способ доступа. Передача больших объемов данных между процессами в DAQ обеспечена путем предоставления им непосредственного доступа к одному и тому же участку оперативной памяти с использованием возможностей операционной системы. В ОС Windows такой общий ресурс реализуется в виде файла, отображенного в поименованный участок памяти. Работа с таким общим участком памяти организована двумя модулями:

- Модуль с функцией GetSharedMem(SMemoryName: string), которая возвращает адрес общего участка памяти MemoryPTR, зарегистрированного в ОС под именем SMemoryName (если он еще не существует, функция его создает). Данный модуль реализован в виде dll.
- Модуль SMemStruct, описывающий структуру данных, адрес которой MemoryPTR.

Участок памяти, начинающийся с адреса MemoryPTR и имеющий структуру, описанную в модуле SMemStruct, мы будем называть ОПНД (Общая Память с Непосредственным Доступом). Для программ, которым требуется использовать общие данные, функция GetSharedMem и модуль SMemStruct включаются в тело программы линкером, и вызов GetSharedMem выполняется однократно при их инициализации. Всем программам, подключившимся к

ОПНД, предоставляются одни и те же значения всех переменных, описанных в структуре данных. Любая программа может изменять значения этих переменных, используя семафор для устранения конфликтов при одновременном обращении к одной переменной.

ОПНД сохраняется до тех пор, пока к ней подключен хоть один пользователь (процесс). Благодаря этому во время работы САЭ могут загружаться различные программы и получать быстрый доступ к одному и тому же содержанию данных в ОПНД.

Функция GetSharedMem не зависит от конкретного варианта описания структуры данных (содержания SMemStruct) и может быть использована в различных проектах без изменения. Текущий образ ОПНД может сохраняться в файле и использоваться в процедурах быстрого автоматического восстановления работоспособности САЭ после сбоев и аварий.

Более сложная проблема возникает в случае, когда мощность ЭВМ (быстродействие + объем памяти), на которой работает DAQ, недостаточна, чтобы в реальном времени выполнять последовательно ввод, преобразование и архивирование данных. В этом случае будет происходить частичная потеря данных, и возникает необходимость решать проблему не только программными, но и аппаратными средствами – например, путем распараллеливания ввода данных в несколько ЭВМ, использования массива быстрых накопителей (RAID [75]), и использования группы процессоров, параллельно обрабатывающих поток данных. Такие решения позволяют увеличить пропускную способность системы более чем на порядок [76,77]. Конкретные варианты решения этой проблемы зависят от многих факторов и выходят за рамки темы данной работы.

5.4.2. Структура компонента ввода данных и способы синхронизации.

Возможная структуры компонента ввода данных следующая:

- группа команд подключения к DiCME, которая обеспечивает взаимодействие САЭ с данным компонентом (4..6 операторов);
- пакет управляющих параметров;

- интерпретатор команд;
- реализация процедур.

В отличие от других компонентов САЭ, компонент ввода данных после завершения экспозиции не шлет ответ вызывающей программе, а передает управление цепочке преобразования и архивирования данных, и лишь после завершения архивирования вызывающей программе передается синхронизирующий сигнал разрешения продолжить работу.

Пакет управляющих параметров помещается в ОПНД, структура ОПНД должна включать описания:

- группы управляющих параметров для модуля ввода данных;
- таблиц, управляющих преобразованием (сортировкой, сжатием) данных;
- буферов спектров, заполняемых в реальном времени программами ввода, сортировки и др.;
- буферов для временного (до переноса в архив) хранения сегментов потока событий.

Стандартный набор выполняемых команд – INIT, START, PAUSE, CONTINUE, END – в комментариях не нуждается. Команда INIT выполняется автоматически, остальные инициируются оператором САЭ при помощи интерфейса пользователя, или программы управления экспериментом.

В общем случае компонент ввода данных может работать со многими детекторами, при этом возможны варианты:

1. Один или группа детекторов с одинаковой структурой данных подключаются через один интерфейс [21,22]. Это самый простой случай, ниже (см. рис. 5.6) представлено окно программы параметризации такой подсистемы регистрации данных.
2. Детекторы с разной структурой данных подключены через один интерфейс [78]. Очевидно, в этом случае просто увеличивается объем программы параметризации подсистемы регистрации.
3. Детекторные группы подключены через разные интерфейсы. Для каждого интерфейса, подключающего группу детекторов с иной структурой

данных, потребуется дополнительный модуль ввода данных и соответствующие модули сжатия данных и параметризации интерфейса.

При работе с многодетекторными спектрометрами специальные меры для синхронизации времени старта детекторов обычно не нужны. Как правило, начало и конец регистрации синхронизируется с фронтом синхроимпульсов от источника излучения [79]. Помимо этого, задержка во времени передачи последовательных команд START используемым интерфейсам постоянна и не приводит к искажению данных. Если по условиям эксперимента такая синхронизация необходима (например, используется пространственно распределенная установка), то возможны два варианта:

- использование системы GPS при вводе данных от детекторов, не имеющих электрического соединения, дает возможность выполнить синхронизацию начала и окончания регистрации данных с погрешностью до 100 нс (метод подробно описан в работе [80]);
- если требуется синхронизация с погрешностью <100 нс, то задача решается с помощью дополнительной электроники и требуется электрическое соединение детекторов. Один из блоков (master) должен запрещать/разрешать прохождение синхроимпульсов от источника нейтронов к остальным блокам регистрации данных [79].

Продолжительность экспозиции обычно отсчитывается устройством регистрации данных [21].

Проиллюстрируем возможности конкретной подсистемы регистрации времяпролетных спектров [19] примером окна интерфейса пользователя к программе параметризации подсистемы. На рис. 5.6 представлено окно параметризации подсистемы, работающей с 8 детекторами, которые формируют данные с одинаковой структурой и подключены через один интерфейс. Пользователь получает возможность сообщить режимы работы источника нейтронов и временного кодировщика, подключить нужные детекторы (<=8), задать параметры гистограммирования (выделить несколько участков на оси номеров каналов, задать нужные ширины каналов) и др.

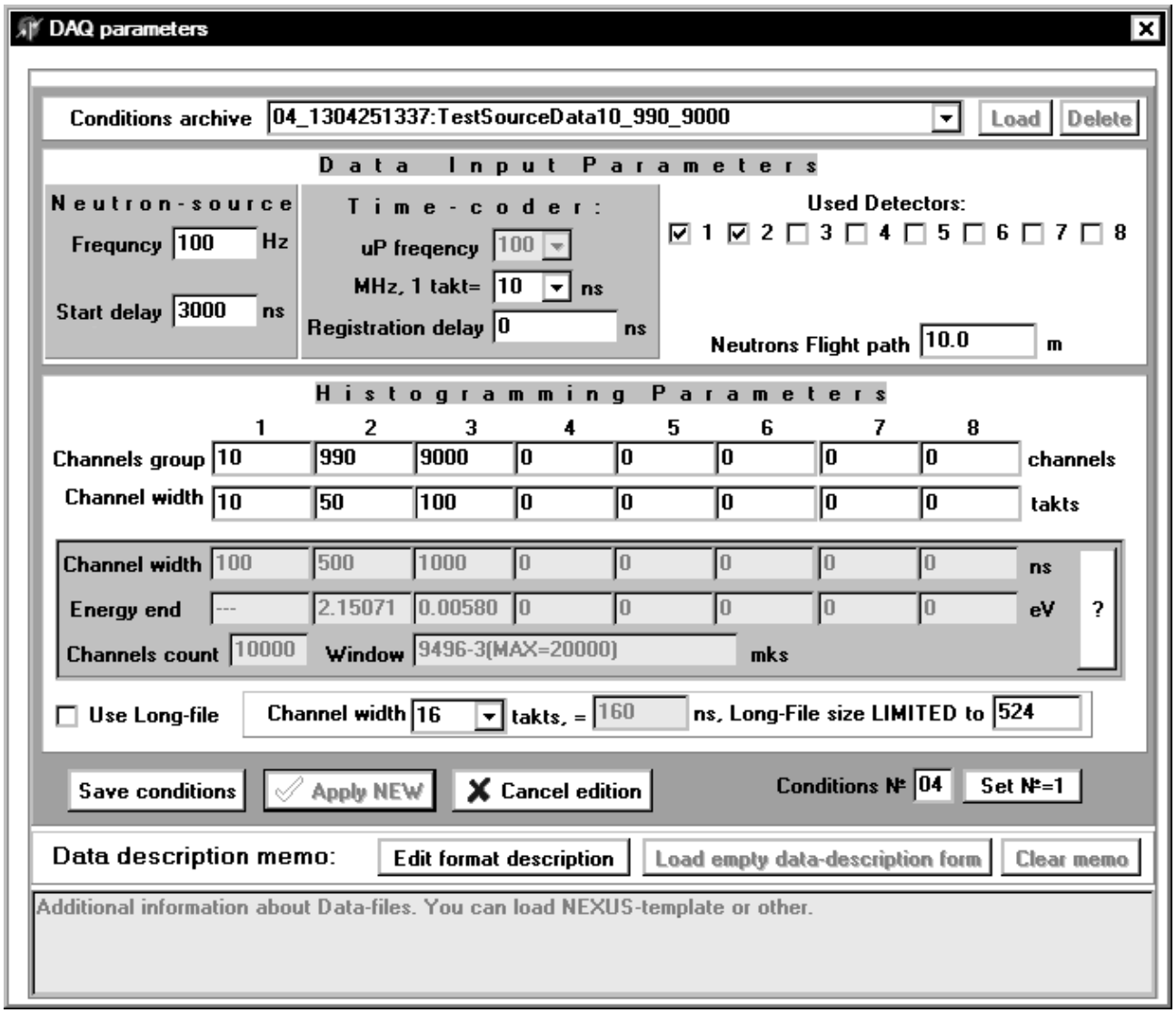

Рис. 5.6. Окно программы параметризации подсистемы регистрации данных

Значения параметров, заданных пользователем, помещаются в соответствующие поля структуры ОПНД а также в архив. На основании этих данных формируются также таблицы, управляющие алгоритмом гистограммирования. Использование ОПНД и разработанного быстрого алгоритма при сортировке данных в спектр с переменной шириной канала для окна в 20 мс при дискретности 10 нс позволяет за 20..30 нс выполнить обработку одного события для любого канала на ЭВМ с частотой процессора 2 ГГц.

Помимо задания значений параметров, управляющих процессом регистрации данных, окно программы предоставляет возможность ввести дополнительные данные, которые могут быть использованы для управления

форматом представления данных и для математической обработки. После настройки подсистемы регистрации пользователю приходится изредка менять лишь формат гистограммирования.

5.5. Структура компонента управления условиями регистрации данных

Компонент управления устройством окружения образца содержит:

- группу команд подключения к средствам DiCME;
- интерпретатор описания условий регистрации данных;
- процедуры, реализующие предусмотренные функции.

Протокол управления устройством (состав описаний условий и параметров) определяется разработчиком компонента и документируется в паспорте устройства в БД. Реализация обязательна только основной процедуры компонента, выполняющей перевод устройства в заказанное состояние. Состав остальных процедур определяется конструкцией оборудования. В отличие от других работ, в которых выдвинуты жесткие требования к структуре компонентов управления условиями регистрации данных (например, [81]), предоставленная программисту свобода выбора решений существенно облегчила разработку компонентов.

После выполнения описанной в сообщении клиента работы компонент шлет (асинхронно) вызывающей программе сигнал завершения работы и публикует для вспомогательной логики детализирующую информацию (например, информацию о состоянии, диагностическое сообщение об ошибке и др.).

В компоненты данного типа удобно поместить процедуры мониторинга состояния управляемого объекта и удержания заданного командой значения параметра.

Как указано выше, компонент DiCME прозрачен для сообщений, описывающих требуемые условия регистрации данных, их интерпретация выполняется в компоненте.

5.6. Сетевая архитектура САЭ

На рис. 2 (стр. 7) показана сетевая архитектура САЭ. Функциональные компоненты САЭ с целью обеспечения безопасности ограничены локальной сетью. Включение специальных средств обеспечения безопасности может ухудшить условия и скорость разработки САЭ, т.к. привлекаются третьи лица. В связи с этим было решено придерживаться идеологии доверенной сети. Все соединения и операции происходят без аутентификации (либо с паролями по умолчанию). Таким способом, возможно избежать избыточной (не имеющей отношения к работе экспериментатора) параметризации системы – все компоненты находят друг друга сами, не задавая лишних вопросов. Тем не менее, при выборе протоколов и процедур предусмотрена возможность будущего "упрочнения". Например, для транспорта запроса на выполнение процедуры вместо протокола HTTP может быть использован HTTPS, а авторизация с СУБД будет происходить по заранее установленной паре логин-пароль.

Во внешнюю сеть с соответствующими мерами фильтрации доступа вынесен Web-интерфейс пользователя и дополнительные средства визуализации состояния САЭ и данных.

Для осуществления удаленного доступа через Интернет безопасность поднята на надлежащий уровень:

- интерфейс пользователя создан в виде Web front-end к системе;
- специально настроенный firewall даёт доступ через Интернет только на соответствующие порты Web-сервера;
- авторизация и работа возможна только по безопасному подключению (HTTPS).

5.7. Выводы

В главе описаны алгоритмы и структура компонентов основной логики ПО САЭ. Разработаны унифицированные компоненты, которые могут использоваться без изменения в различных экспериментах и системах:

1. Интерфейс пользователя, который динамически настраивается на состав активных компонентов и предоставляет возможность активировать работу нужного компонента для выполнения вспомогательных (сервисных) функций.
2. Унифицированная программа управления экспериментом, предоставляющая возможность выполнения прецизионных экспериментов, работы в автоматическом, пошаговом, отладочном режимах работы и др.
3. Способы подключения подсистем DAQ к ПО САЭ и синхронизации их работы.
4. Алгоритм быстрой передачи средствами ОС Windows данных между входящими в состав DAQ компонентами. Данный алгоритм может быть реализован и в других операционных системах (например, [82]).
5. Структура компонентов управления условиями регистрации данных. К структуре не выдвигается жестких требований, формат сообщений для удаленного выполнения процедур компонентов определяет разработчик компонента, что существенно упрощает выполнение разработки.

Заключение

Основные результаты работы могут быть сформулированы следующим образом:

1. Выполнена классификация функционального состава и способов взаимодействия компонентов в ПО САЭ. Выявлены особенности и различия в способах взаимодействия компонентов и сделаны выводы, использованные при разработке алгоритмов взаимодействия компонентов и структуры ПО САЭ, обеспечивающей разные дисциплины одновременного выполнения основной и вспомогательной логики. Такая структура существенно увеличивает гибкость системы и удобство работы пользователей.
2. Предложен способ описания методики управления экспериментом списком условий регистрации экспериментальных данных вместо традиционно используемого перечисления действий программы управления экспериментом вызовами процедур на языке программирования или представления последовательности действий списком названий процедур в виде интерпретируемого скрипта.
3. Предложен метод управления выполнением основных функций ПО САЭ, основанный на управлении процессами, конкретное функциональное наполнение которых определяется динамически с использованием задания на эксперимент.
4. Предложен метод автоматической компоновки распределенного ПО САЭ в условиях изменения задания при переходе от одного эксперимента к другому (на одной и той же исследовательской установке), основанный на использовании компонентами широковещательных сообщений с информацией о своих идентификаторах и сетевого протокола поиска компонентов.
5. Предложен метод динамического связывания компонентов для удаленного выполнения процедур в распределенном ПО САЭ и

унифицированные средства обслуживания межкомпонентного взаимодействия, основанные на использовании открытых сетевых технологий. В отличие от методов динамического связывания компонентов, используемых в технологиях DCOM, CORBA, Ice и др., для обслуживаемой проблемной области предложенные методы обладают рядом преимуществ.

В итоге для автоматизации экспериментов в области спектрометрии нейтронов впервые поставлена и решена задача разработки методов построения ПО САЭ с использованием сетевых технологий, которые обеспечивают унификацию компонентов ПО САЭ, возможность применять компоненты в разных экспериментах и разных САЭ без изменения, что обеспечило сокращение сроков разработки и модификации ПО САЭ и способствовало повышению надежности работы систем и эффективности работы исследователей.

Разработанное на основе предложенных методов ПО САЭ является проблемно-ориентированным распределенным пакетом прикладных программ (РППП), включающим компоненты управления пакетом и унифицированные прикладные компоненты. Унификация компонентов обеспечена стандартизацией интерфейса доступа к процедурам, реализующим функциональность компонентов, и разработанными в диссертации методами, на которых основаны компоненты управления РППП – программа управления экспериментом, средства обеспечения сетевого взаимодействия компонентов и подсистема составления задания. Предложенная структура пакета и компоненты управления пакетом могут быть использованы при построении распределенного ПО для применения в иной предметной области, например, для автоматизации технологических процессов.

Результаты диссертации применены при разработке ПО САЭ для нескольких спектрометров. Эти САЭ используются тремя организациями (ЛНФ, ЛЯП ОИЯИ, ИЯИ г. Троицк) и прошли испытание в экспериментах на

источниках нейтронов ИБР-2 и ИРЕН в ОИЯИ, с их помощью получены важные научные и практические результаты [18,19,21,22].

Всего по теме диссертации опубликовано 11 работ [7,10-16,19-21]. Результаты диссертации представлены четырьмя статьями в рецензируемых научных журналах [7,10,13,19], в виде сообщений ОИЯИ [7,10-15,19], докладывались на семи международных конференциях, семинарах и совещаниях [12,13,16,20,21].

Литература

1. Ermilov V.G., Ivanov V.V., Korolev V.S. et al. The System for Diagnostics and Monitoring of the IBR-2 Reactor State. Data Acquisition, Accumulation and Storage of the Information // Proc. of the 2nd Intern. Workshop DANEF-2000 (Dubna, June 5-7, 2000), Dubna, E10-2001-11, Dubna, 2001, p. 176-185.
2. Levis P., Cooper G., Trouw F. et al. Data Acquisition and Instrument Control at the Lujan Center: An Update // NOBUGS2010, Preliminary Agendas NOBUGS34, 2010.
3. Kraimer M., Anderson J., Johnson A. et al. EPICS Input/Output Controller Application Developer's Guide. Release 3.14.8, 2005. Argonne National Laboratory, http://www.aps.anl.gov/epics.
4. Велихов Е.П., А.Н. Выставкин А.Н. Проблемы развития работ по автоматизации научных исследований // УСиМ, 1984. №4, стр. 3-12.
5. Доррил Тафе Что нам готовит год 2027-й // PC Week/RE №98 (614) 11-17 марта 2008.
6. Scott Ambler. Types of Reuse in Information Technology // www.ambysoft.com/essays/typesOfReuse.html
7. Саламатин И.М., Саламатин К.М..
   - Разработка компонентной САЭ для физики низких энергий на основе использования сетевых технологий // ОИЯИ Р13-2013-74, Дубна, 2013, 33 с.;
   - Сетевые технологии в программных системах автоматизации спектрометрии нейтронов // Прикладная информатика, 2014. № 5(53), с. 60-80.
8. Драгунов Ю.Г., Третьяков И.Т., Лопаткин А.В. и др. Модернизация импульсного исследовательского реактора ИБР-2 // АЭ, 2012. т.113, Вып. 1, стр. 29-34.

9. Belikov O.V., Belozerov A.V., Becher Yu. et al. Physical startup of the first stage of IREN facility // “ISINN-17” (Dubna, May 27-29, 2009), Dubna E3-2010-36. Dubna: JINR, 2010, p. 10-16.
10. Саламатин К.М. DiCME – Распределенная среда взаимодействия компонентов системы автоматизации экспериментов для физики низких энергий
    - // Программная инженерия, 2014. №3, с. 3-11;
    - // ОИЯИ Р13-2013-91, Дубна, 2013, 19 с.
11. Саламатин К.М.. PSJ – Унифицированная подсистема описания методики эксперимента // ОИЯИ Р13-2013-92, Дубна, 2013, 11 с.;
12. Мазный Н.Г., Саламатин И.М., Саламатин К.М.
    - Генерация программ автоматизации экспериментов из модулей в формате загрузки // ОИЯИ Р13-2007-93, Дубна, 2007, 15с;
    - Generation of experiment automation programs from modules in loadable format // Book of abstracts LVII Intern. conf. on nucl. phys “NUCLEUS 2007”, Voronezh June 25-29, 2007, p.282. СПб:СПбГУ, зак. №544/с.
13. Игнатович В.К., Саламатин И.М., Саламатин К.М., Сеннер А.Е.
    - Автоматизация экспериментов в области спектрометрии нейтронов с использованием сетевых технологий // ОИЯИ Р13-2014-33, Дубна, 2014, 8 с. (0.5 п.л.)
    - Unification of Experiment Procedure Control Tools for the Experiment Automation Systems in the Field of Neutron Spectrometry Using Network Technologies // Abstracts of the Seminar ISINN-22 (Dubna, May 27-30, 2014), Dubna E3-2014-27, Dubna: JINR, 2014, p. 50.
    - Automation of neutron spectrometry experiments using network tecnologies // Book of abstracts of ICANS XXI: 21st Internat. Collaboration on Advanced Neutron Sources 29Sep-3Oct in Mito, Ibaraki, Japan, p. 244.

- Автоматизация экспериментов в области спектрометрии нейтронов с использованием сетевых технологий // Информационные технологии, 2014 г. №12, с.63-68.

14. Саламатин К.М. Выбор технологии построения компонентной системы для автоматизации экспериментов в области спектроскопии нейтронов // ОИЯИ Р13-2013-72, Дубна, 2013, 30 с.;
15. Саламатин К.М. Вариант построения компонентной системы автоматизации экспериментов для спектрометрии с использованием сетевых технологий // ОИЯИ Р13-2013-86, Дубна, 2013, 21 с.
16. Salamatin K.M. Development of Component System for Neutrons Spectrometry Automation Through the Use of Network Technologies // “ISINN-21” (Alushta, Ukraine, May 20-25, 2013), Dubna E3-2013-40. Dubna: JINR, 2013, p. 77.
17. Луховицкая Э.С., Езерова Г.Н. Информатика в ИПМ им.М.В.Келдыша. 1960-е годы // Препринты ИПМ им.М.В.Келдыша. 2013.
18. Enik T.L., Mitsyna L.V., Popov A.B. et al. AURA Setup Testing at the IREN Neutron Beam // “ISINN-21” (Alushta, Ukraine, May 20-25, 2013), Dubna E3-2013-40. Dubna: JINR, 2013, p. 33.
19. Швецов В. Н., Алпатов С. В., Астахова Н. В.,…, Саламатин К. М. и др. 8-Входовая система для нейтронно-ядерных исследований по методу времени пролета
    - // ПТЭ, №5, с. 54-61, 2012.
    - // ОИЯИ, Р13-2011-96, Дубна, 2011.
    - // Instruments and Experimental Techniques, Vol. 55, No. 5, pp. 561–568, 2012.
20. Швецов В.Н., Астахова Н.В., Гундорин Н.А., …, Саламатин К.М. и др.
    - Многовходовая система TOF для нейтронно-ядерных исследований по методу времени пролета нейтронов // Междунар. конф. “ІЕФ’2011”, Украина, г.Ужгород, 24-27 мая 2011г. Науковый виснык Ужгородського университету. Серия Физика. Выпуск 30-2011, стр.136-142.

- Multichannel system TOF for neutron-nuclear investigations at IREN neutron source // Abstracts of the Seminar ISINN-17 (Dubna, May 27-30, 2009), Dubna E3-2009-30, Dubna: JINR, 2009, p. 57.

21. Shvetsov V.N., Alpatov S.V., Astahova N.V., …, Salamatin K.M. и др. Multiinput Encoder for Recording Spectra of Scattered Neutrons Using Time-of-Flight Method // ISINN-19 (Dubna, May 25-28, 2011), Dubna E3-2012-30. : JINR, 2012, p. 279-283.
22. Tsulaia M.I., Salamatin I.M., Sirotin A.P. et al. The Kolkhida Setup Upgrade // “ISINN-21” (Alushta, Ukraine, May 20-25, 2013), Dubna E3-2013-40. Dubna: JINR, 2013, p. 77.
23. Kuznetsov V.L., Kuznetsova E.V., Sedyshev P.V. Investigation of Possibilities for the Measurement of Parity Violation in Neutron Diffraction at the IBR-2M Reactor // Proc. of Intern. Seminar “ISINN-20” (Dubna, May 27-29, 2012), Dubna E3-2013-22. Dubna: JINR, 2013, p. 66-
24. Протокол № 4152-4-11-13 о выполнении совместной научно-исследовательской работы Объединенным институтом ядерных исследований и Международным университетом природы, общества, человека “Дубна”, г. Дубна, Россия.
25. Галкин Г. Интеграция приложений: история подходов // СЕТЕВОЙ №6, 2002. http://www.setevoi.ru/cgi-bin/text.pl/magazines/2002/6/20
26. Телков А.Ю. Распределенные системы обработки информации: Учебно-методическое пособие, 2007 г.
    // http://window.edu.ru/library/pdf2txt/549/59549/29617
27. http://java.sun.com/j2se/1.5.0/docs/guide/rmi/index.html
28. http://www.interface.ru/magazine/tcs/Archive/198/DCOM/dcom.htm
29. CORBA http://www.rsdn.ru/article/corba/vsCORBA.xml
30. A New Approach to Object-Oriented Middleware // IEEE Internet Computing, Jan 2004; http://www.zeroc.com
31. http://www.citforum.ru/internet/xml/xml_rpc/
32. http://do.gendocs.ru/docs/index-19169.html

33. http://www.json.org
34. http://www.remake.ru/analitika/technology/soap.html
35. Ахтырченко К.В., Леонтьев В.В. Распределенные объектные технологии в информационных системах // СУБД 1997, №5-6.
36. Семихатов С. Технологии построения распределенных объектных систем. // http://www.javable.com/docs/articles/dist/
37. http://www.rsdn.ru/article/corba/vsCORBA.xml
38. А. Цимбал. Технология Клиент-Сервер 2000'2 // http://www.k-press.ru/cs/2000/2/CORBA/CORBA.asp
39. Michi Henning. The Rise and Fall of CORBA // ACM Queue, Vol. 4, Num. 5, June 2006; http://citforum.ru/SE/middleware/corba_history
40. A New Approach to Object-Oriented Middleware // IEEE Internet Computing, Jan 2004; http://www.zeroc.com
41. Дубова Н. На пути к SOA // Директор информационной службы, 2005. №8, стр. 12.
42. http://osp.ru/text/print/302/379555.html
43. Введение в сервис-ориентированную архитектуру. http://kacit.ru/upload/iblock/cbc/cbc9bcc2c17f3aa9a017e8fad70b4c4c.pdf
44. http://ru.wikipedia.org/wiki/Zeroconf
45. Тим Джонс М. Автоматизация управления клиентами с помощью Service Location Protocol // http://www.ibm.com/developerworks/ru/library/l-slp
46. http://www.ibm.cjm/developerworks/ru/library/l-slp
47. Flemming S.A., Dr. James J.A., Dr. Schneider R. et al. The Open Inspire Architecture for Control, Data Reduction and Analysis // NOBUGS2008, Paper 134, 2008.
48. Martin Fowler. Inversion of Control Containers and the Dependency Injection pattern, 2004 // http://martinfowler.com/articles/injection.html
49. XML-RPC: вызов процедур посредством XML // http://web.znu.edu.ua/bdp/cs/xmlrpc.doc

50. Соловей В.А., Савельева Т.В., Вихарев Л.Е., Колхидашвили М.А. Базовые аппаратные средства для обеспечения интегральных и времяпролетных методов измерения в экспериментальных исследованиях конденсированного состояния сред с применением нейтронов // Препринт ЛИЯФ 2599, ЛИЯФ 17-02-2005.
51. NOBUGS Conferences // http://www.nobugsconference.org/
52. International Symposium on Nuclear Electronics and Computing // http://nec2011.jinr.ru/
53. Сайт ПИЯФ // http://www.pnpi.spb.ru/
54. Proc. of the. First Intern. Workshop “Data Acquisition Systems for Neutron Experimental Facilities (DANEF’97)”// Dubna, JINR E10-97-272, Dubna, 1997.
55. Proc. of the. Second Intern. Workshop “Data Acquisition Systems for Neutron Experimental Facilities (DANEF-2000)”// Dubna, JINR E10-2001-11, Dubna, 2001.
56. Riedel R., Zolnierczuk P., Parizzi A., Sundaram M. UDP, TCP, Circular Buffers, Multi-threaded Programming and the Transmission of Event Data // NOBUGS2010, №47
57. http://www.aps.anl.gov/epics/about.php
58. Ermiliv V.G., Ivanov V.V., Korolev V.S., Pepyolyshev Yu.N., Semashko S.V., Tulaev A.B. The System for Diagnostics and Monitoring of the IBR-2 Reactor State. Data Acquisition, Accumulation and Storage of the Information // Proc. of the 2nd Intern. Workshop DANEF-2000 (Dubna, June 5-7, 2000), Dubna, E10-2001-11, Dubna, 2001, p. 176-185.
59. Nikulnikov, Prihodko V.I., Sirotin A.P., Solovyov B.N., Zuravlev V.V. Control Systems of Neutron Beam Choppers at the Spectrometers of the IBR-2 Reactor // Proc. of the 2nd Intern. Workshop DANEF-2000 (Dubna, June 5-7, 2000), Dubna, E10-2001-11, Dubna, 2001, p. 125-131.
60. Куклин А.И., Сиротин А.П., Кирилов А.С., Исламов А.Х., Петухова Т.Б., Астахова Н.В., Утробин П.К.,. Ковалев Ю.С, Горделий В.И. Автоматизация

и окружение образца модернизированной установки ЮМО // ОИЯИ 313-2004-77, Дубна, 2004.

61. Аксенов В.Л., Жерненков К.Н., Кожевников С.В. и др. Спектрометр поляризованных нейтронов РЕМУР на импульсном реакторе ИБР-2 // ОИЯИ Д13-2004-47, Дубна, 2004
62. Astachova N.V., Kirilov A.S., Salamatin I.M. Remote Control of the YUMO Spectrometer and User Interface // Proc. of the 2nd Intern. Workshop DANEF-2000 (Dubna, June 5-7, 2000), Dubna, E10-2001-11, Dubna, 2001, p. 275-278.
63. Астахова Н.В., Бескровный А.И., Богдзель А.А. и др. Программный комплекс АС (Автомптизация спектрометрии) Реализация интерфейса пользователя системы автоматизации эксперимента // ОИЯИ Р13-2003-146, Дубна, 2003, 16 с.
64. T. Nakatani, Y. Inamura, T. Ito, T. Ohhara, Y. Kawakita, T. Otomo, J. Suzuki, S. Muto, K.M. Kojima. Present status of the computing environment for the experimental instruments in J-PARC/MLF // NOBUGS2010, Preliminary Agendas NOBUGS40, 2010.
65. Matt Clarke, Takeshi Nakatani, Yasuhiro Inamura, Takayoshi Itoh, Yukinobu Kawakita, Takashi Ohhara, Takaaki Hosoya, Kenji Kojima, Akuhiro Koda, Masatoshi Arai and Toshiya Otomo. Making life easier for scientists: developing an Experiment Scheduler for the MLF // NOBUGS2010, Posters NOBUGS20, 2010.
66. M. Konnecke, N. Hauser, F. Franceshcini, T. Lam, M. Zolliker. Treepath Based Instrument Control // NOBUGS2008, №132.
67. C. Gaspar, M. D¨onszelmann. DIM – A Distributed Information Management System for the DELPHI Experiment at CERN // Proceedings of the 8th Conference on Real-Time Computer applications in Nuclear, Particle and Plasma Physics, Vancouver, Canada, June 1993.
68. C. Gaspar, J.J. Schwarz. The DELPHI Experiment Control System // Proceedings of the first IEEE International Conference on Engineering of Complex Computer Control Systems, Florida, Nov. 1995.

69. B. Franek. On-line Experiment Control System for the BaBar Detector at PEP-II at SLAC // Proceedings of the International Conference on Computing in High Energy Physics, Chicago, 1998.
70. http://kunegin.com/ref3/corba4/7.htm
71. Ketlerov V.V. PC Based Data Acquisition System with Remote Control. // Proc. of the 2nd Intern. Workshop DANEF-2000 (Dubna, June 5-7, 2000), Dubna, E10-2001-11, Dubna, 2001, p. 96-98.
72. Гундорин Н.А., Дикусар Н.Д., Мазный Н.Г., Пикельнер Л.Б., Саламатин И.М., Цулаиа М.И. Экспресс-анализ спектров в прецизионных экспериментах // ОИЯИ Р10-2007-94, Дубна, 2007
73. Богдзель А.А., Гундорин Н.А., Матвеев Д.А. Электроника 16-канального сцинтилляционного детектора на основе кристаллов BGO // Тр. XVIII Междунар. симпоз. “Ядерная электроника и компьютинг”. Варна, Болгария, 12-18 сент. 2001, ОИЯИ Д10,11-2002-28, Дубна, 2002, с. 36-39.
74. Ji-Yong So. Event Mode Data Reduction & Analysis Method for the Time-of-Flight Spectrometer // NOBUGS2010, №24.
75. http://ru.wikipedia.org/wiki/raid
76. http://www.ibase.ru/devinfo/raid.htm
77. http://citforum.ru/hardware/data/raid
78. Богдзель А.А., Гундорин Н.А., Матвеев Д.А. Электроника 16-канального сцинтилляционного детектора на основе кристаллов BGO // Book of Abstr. of the XIX Intern. Symp. on Nuclear Electronics and Computing (NEC-2003), Varna, Bulgaria, Sept, 15-20, 2003. Dubna, 2003, p. 23.
79. Korobchenko M.L., Levchanovsky F.V., Rezaev V.E. Unified VME-Based Data Acquisition Systems for the Spectrometers at the IBR-2 Pulsed Reactor. Processor Module // Proceedings of the International Workshop on Data Aquisition Systems for Neutron Experimental Facilities (DANEF-97), Dubna, 2-4 June 1997, JINR, E10-97-272, Dubna 1997, p.163-171

80. Астахова Н.В., Бордюгов Л.Г., Герасимов А.В. и др. Распределенная беспроводная система регистрации с синхронизацией потоков данных // ОИЯИ Р13-2006-41, Дубна, 2006.
81. Петухова Т.Б., Кирилов А.С. Создание мастера Visual Studio.Net для разработки модулей управления устройствами в комплексе Sonix+ // ОИЯИ Р10-2006-27, Дубна, 2006, 8с.
82. http://habrahabr.ru/blogs/nix_coding/55716/

Приложения

Список рисунков и таблиц

1. Список рисунков

2. Список таблиц

Список использованных сокращений

БД – База Данных

ОПНД – Общая Память с Непосредственным Доступом

ОС – Операционная Система

ПО – Программное Обеспечение

ПУЭ – Программа Управления Экспериментом

САЭ – Система Автоматизации Экспериментов

СКАДА – Supervisory Control And Data Acquisition

Firewall – фильтр, защищающий компьютер от несанкционированных вторжений и обеспечивающий безопасность пользования

Front-end – часть системы, взаимодействующая с пользователем

CORBA – Common Object Request Broker Architecture CPU – технология

DAQ – Data Acquisition – ввод в ЭВМ, преобразование, архивирование потока данных

DCOM – Distributed Component Object Model – технология

Desktop – основное окно графической среды пользователя.

DiCME – Distributed Components Messaging Environment

DIM – Distributed Information Management

EPICS – Experimental Physics and Industrial Control System

GUID – Globally Unique Identifier

HTTP – HyperText Transfer Protocol – протокол передачи гипертекста

HTTPS – HyperText Transfer Protocol Secure – защищенный HTTP

Ice – The Internet Communications Engine

JSON– Java Script Object Notation

PLE – Product Line Engineering – группа приложений

PSJ – Preparation of Single Job

RAD – Rapid Application Development

RAID – Redundant Array of Independent Disks – дополнительный массив независимых дисков

RFC – Request For Comments – документ из серии пронумерованных информационных документов Интернета

RMI – Java Remote Method Invocation

RPC – Remote Procedure Call

SLP – Service Location Protocol

SOA – Service Oriented Architecture

SOAP – Simple Object Access Protocol (текстовый протокол на базе HTTP)

SQLite – легковесная система управления базами данных (СУБД)

TCP/IP – стек сетевых протоколов передачи данных

UDP – User Datagram Protocol – протокол пользовательских датаграмм

URL – Uniform Resource Locator – единообразный локатор (определитель местонахождения) ресурса

XML – eXtensible Markup Language – расширяемый язык разметки (текстовый протокол на базе HTTP)

Определения некоторых использованных терминов и понятий

1. Архитектура системы, программы (объекта) – это представление о системе, программе (объекте) с его внешней стороны, доступной ее (его) пользователю.

Такому понятию «архитектуры системы» соответствует первая фраза толкования термина «Архитектура ЦВМ» в «Словаре по кибернетике» // под ред. академика В.М.Глушкова (Киев, Гл. редакция Украинской советской энциклопедии, 1979):

«Архитектура ЦВМ – система основных функциональных средств машины, доступных пользователю».

В Википедии:

«Архитектура программного обеспечения (англ. software architecture) – это структура программы или вычислительной системы, которая включает программные компоненты, видимые снаружи свойства этих компонентов, а также отношения между ними».

2. Web-сервис, Веб-служба (англ. Web service) – здесь: идентифицируемая веб-адресом (URL) программная система, программа со стандартизированным интерфейсом.

Другие толкования:

Веб-сервис – это технология организации межпрограммного взаимодействия по протоколу HTTP. Основы веб-сервисов: WSDL, UDDI, XML-RPC.

Веб-сервис – это сетевая технология, обеспечивающая межпрограммное взаимодействие на основе веб-стандартов.

Консорциум W3C определяет веб-сервис, как «программную систему, разработанную для поддержки интероперабельного межкомпьютерного (machine-to-machine) взаимодействия через сеть»

3. Метод – путь, способ, приём теоретического исследования или практического осуществления чего-нибудь. (Словарь Д.Н. Ушакова, 1949 г.)

4. Методика – это процедура (система), представляющая последовательность выполнения правил (методов, приемов) для целесообразного выполнения некоторой практической деятельности.

Другие определения:

- система правил выполнения какой-нибудь работы. (Словарь Ушакова, 1938, 1949);
- совокупность методов практического выполнения чего-нибудь. (Словарь Ожегова, 1978);
- совокупность методов, приемов целесообразного проведения какой-либо работы. (Словарь иностранных слов, 1979);
- совокупность методов, приёмов практического выполнения чего-либо (Викисловарь, 2013).

5. Информационная система.

В широком смысле информационная система есть совокупность технического, программного и организационного обеспечения, а также персонала, предназначенная для того, чтобы своевременно обеспечивать надлежащих людей надлежащей информацией [1].

Также в достаточно широком смысле [2] трактует понятие информационной системы Федеральный закон РФ от 27 июля 2006 года № 149-ФЗ «Об информации, информационных технологиях и о защите информации»: «информационная система – совокупность содержащейся в базах данных информации и обеспечивающих её обработку информационных технологий и технических средств»[3].

Одно из наиболее широких определений ИС дал М.Р. Когаловский: «информационной системой называется комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а также системный персонал и обеспечивающий поддержку динамической

информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей» [4].

Стандарт ISO/IEC 2382-1 дает следующее определение: «Информационная система – система обработки информации, работающая совместно с организационными ресурсами, такими как люди, технические средства и финансовые ресурсы, которые обеспечивают и распределяют информацию» [5].

Российский ГОСТ РВ 51987 определяет информационную систему как «автоматизированную систему, результатом функционирования которой является представление выходной информации для последующего использования».

В узком смысле информационной системой называют только подмножество компонентов ИС в широком смысле, включающее базы данных, СУБД и специализированные прикладные программы. ИС в узком смысле рассматривают как программно-аппаратную систему, предназначенную для автоматизации целенаправленной деятельности конечных пользователей, обеспечивающую, в соответствии с заложенной в неё логикой обработки, возможность получения, модификации и хранения информации [6].

В любом случае основной задачей ИС является удовлетворение конкретных информационных потребностей в рамках конкретной предметной области. Современные ИС де-факто немыслимы без использования баз данных и СУБД, поэтому термин «информационная система» на практике сливается по смыслу с термином «система баз данных».

1. William S. Davis, David C. Yen, 1998
2. Поскольку понятие информационных технологий само по себе может рассматриваться достаточно широко.
3. Федеральный закон Российской Федерации от 27 июля 2006 г. N 149-ФЗ Об информации, информационных технологиях и о защите информации
4. Перейти к:1 2 Когаловский М. Р., 2003

5. Стандарт ISO/IEC 2382-1
6. Ю.А. Маглинец. Анализ требований к автоматизированным информационным системам. // Бином, 2008. ISBN 978-5-94774-865-9

6. Программная система – это совокупность приложений и программ, конструктивно объединенных в единое изделие для выполнения определенной совокупности задач, отнесенных к одному классу задач, решаемых некоторой информационной системой. [А.Н. Данчула, Учебники по предметам.]

7. Структура системы, программы (объекта) – это совокупность составляющих ее (его) компонентов и связей между ними.

Термин «структура» в общеупотребительном языке означает «внутреннее устройство» (см. «Словарь русского языка» С.И. Ожегова (М., «Русский язык», 1978).