Системы зарубежного производства имеют средства адаптации под нужды конкретного заказчика (встроенные языки программирования бизнес-логики, генераторы отчетов, генераторы структур информационных объектов). Несмотря на высокую стоимость приобретения таких систем, стоимость их внедрения и эксплуатации гораздо выше. Поэтому заказчики очень часто останавливаются на половине пути, поняв всю бесперспективность выбранного решения. И только очень немногие доходят до логического конца, затратив огромные усилия на доводку системы, обучение персонала и реинжиниринг своей компании.

Поэтому, зарубежные системы, считаем слабо применимыми в российских условиях по целому ряду причин: высокая стоимость, сложности связанные с внедрением и сопровождением, проблемы русификации и учета российской специфики.

Несмотря на то, что продукты второй группы гораздо дешевле зарубежных, отсутствие поддержки и постоянного развития продукта, а также во многом моральное устаревание, не позволяет использовать их в качестве основы эффективного управления.

Третья группа, программы, разработанные на платформе 1С, наиболее привлекает. Платформа 1С обладает достаточной гибкостью для реализации практически любых пожеланий пользователя. На базе 1С написаны продукты для автоматизации большинства отраслей промышленности, сельского хозяйства, торговли и сферы услуг. С качеством услугу этих программ так же нет замечаний. Практически в каждом городе существует фирма, занимающаяся поддержкой программ 1С, а так же сама фирма 1С регулярно выпускает обновления своих продуктов. Стоимость приобретения и сопровождения систем на базе 1С ненамного больше стоимости аналогичных российских систем и гораздо меньше (в 10-ки раз) стоимости зарубежных продуктов.

Специализированные разработки, относящиеся к четвертой группе, имеют так же свои плюсы и минусы. Из отрицательных сторон можно выделить следующие замечания. Написанный программный продукт не имеет связи с другими платформами. Так же существует зависимость от разработчика программного продукта. Отсутствия обновлений, все доработки программного продукта производятся за отдельную плату. Из положительных сторон это, прежде всего низкая стоимость программного продукта и четкая заточенность программы под определенную организацию.

Таким образом, для авторизации сервисного центра лучше всего подойдет программы, разработанные на платформе 1С. У этих программ существует гибкость, что очень важно при автоматизации. Так же программы, относящиеся к этой группе, имеют высокое качество сервиса. Немаловажным плюсом является и регулярный выпуск обновлений своих продуктов. Ну и конечно, по ценовой характеристики она входит в диапазон, наиболее подходящий для российского рынка.

Так же можно сделать вывод, что специализированные разработки, так же можно использовать для автоматизации документооборота сервисного центра. Для этого Вам необходимо включить в рабочий состав сотрудника, который, напишет Вам данный программный продукт, и будет поддерживать его в рабочем состоянии.

СПИСОК ЛИТЕРАТУРЫ

1. Баранов В.В. Автоматизация управления предприятий. Инфра – М, 2000.

2. Петров Ю.А. Комплексная автоматизация управления предприятием. Информационные технологии – теория и практика. М.: 2001.

3. Шуремов Е.Л. Информационные технологии управления взаимоотношениями с клиентами. 1С Паблишинг, 2001.

КОМПОНЕНТНАЯ ПРОГРАММНАЯ АРХИТЕКТУРА МУЛЬТИВЕРСИОННЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

Поздняков Д.А.

Аннотация

В статье предложены методы компонентного программирования для построения мультиверсионного программного обеспечения отказоустойчивых систем управления и обработки информации.

Основное содержание

На сегодняшний день разработаны различные методы проектирования отказоустойчивого программного обеспечения. Среди них одним из наиболее перспективных является метод мультиверсионного проектирования. Он состоит в том, что в систему включаются несколько программных модулей, дублирующих друг друга по своему целевому назначению.

Мультиверсионное программирование предполагает, что большинство модулей, реализующих ту же самую функциональность, производят сбой в различных точках, так что сбои могут быть обнаружены и исправлены. Независимость сбоев различных модулей может быть достигнуто с помощью ввода разнообразия в процесс разработки. Но эта независимость сбоев находится на уровне исходных кодов – это является ключевой проблемой. На стадии выполнения мультиверсионных модулей независимость сбоев теряется из-за того, что остались не учтенными возможные взаимодействия модулей в рамках всей программной системы. Модули работают в едином адресном пространстве памяти, разделяют одни и те же ресурсы операционной системы, и из-за этого возникают дополнительные зависимости между модулями программного обеспечения. Вследствие этого, сбой одного модуля может привести к сбою других или даже к отказу всей системы в целом. Для решения этой проблемы необходимо сохранить введенное разнообразие при разработке модулей и перенести его на стадию выполнения. Для этого были сформулированы и обоснованы требования, которым должны соответствовать разрабатываемые мультиверсионные модули.

1. Динамическое подключение модулей, цель которого – реализовать возможность замены модулей во время работы программной системы.

2. Требование инкапсуляции следует из требования динамического подключения модулей. Модули

подключаются друг к другу посредством инкапсулирующих интерфейсов.

3. Требование межмодульной защиты обеспечивает безопасное взаимодействия модулей в рамках программной системы. Необходимо организовать межмодульное взаимодействие таким образом, чтобы исключить нарушения в работе модуля, вызванные внешними причинами, а именно, несанкционированным вмешательством других модулей.

4. Требование независимости модулей от языка программирования отчасти вытекает из требования об инкапсуляции, которое описано выше. Другим основанием для подобного требований является экономическая целесообразность.

Для реализации среды и модулей мультиверсионного программного обеспечения, с учетом сформулированных требований, была предложена технология компонентной объектной модели Microsoft (COM) на основе которой был реализован комплекс программ MVS.

Основной особенностью комплекса является реализация программных модулей в виде независимых компонент, которые могут исполняться в отдельных от среды исполнения процессах. Более того, программные компоненты могут находиться на разных компьютерах и взаимодействовать со средой исполнения посредством сети. Это, во-первых, решает проблему ограниченности вычислительной мощности одного компьютера, во-вторых, это обеспечивает защиту программных модулей друг от друга и от среды исполнения. Другие функциональные модули системы так же реализованы в виде отдельных компонент, что позволяет модернизировать и заменять отдельные части системы с минимальными усилиями.

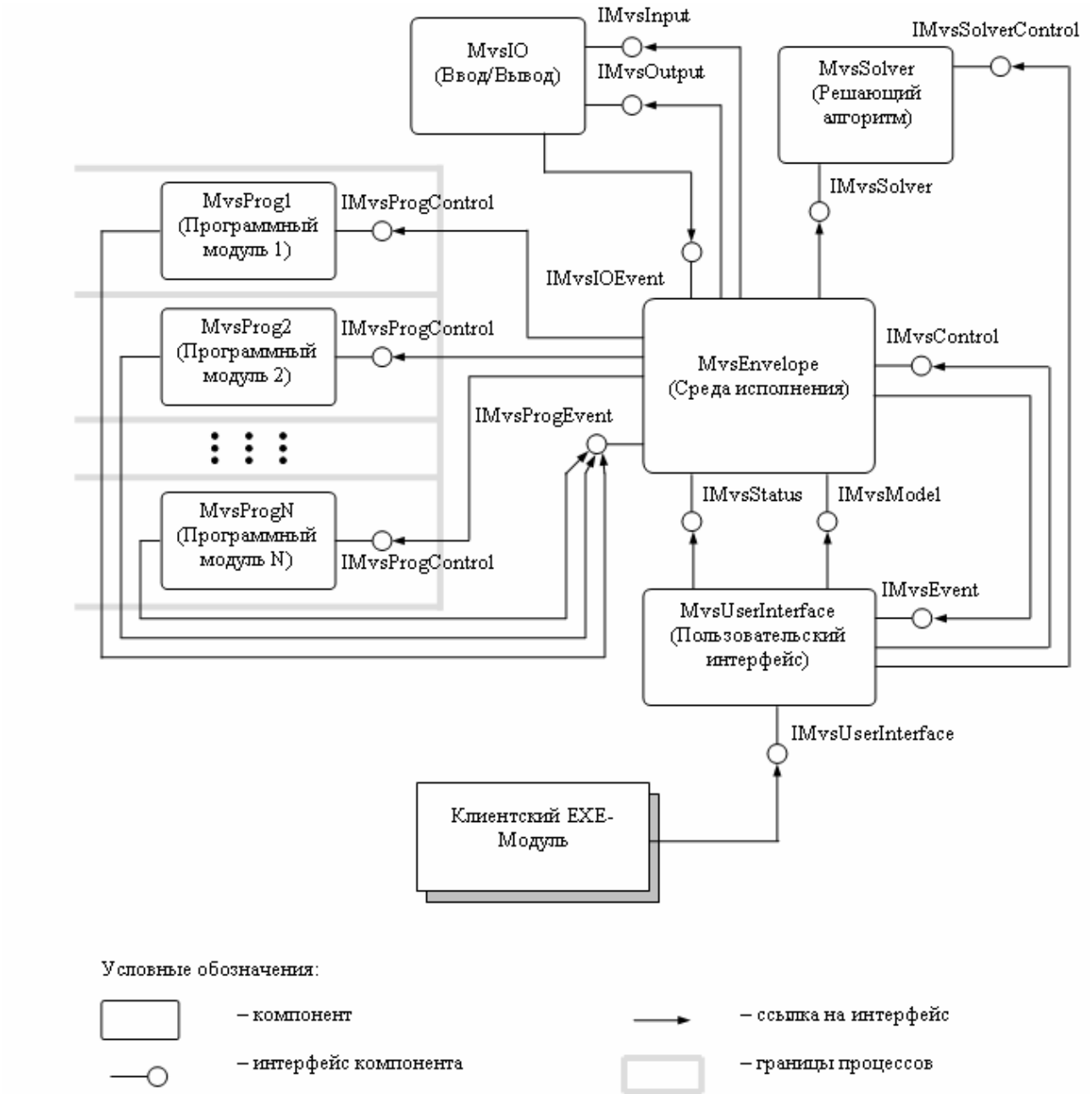

Рисунок 1. Модель мультиверсионного программного обеспечения построенного на основе технологии COM объектов

Наиболее важной задачей является защита среды исполнения от разрушительного влияния извне, поэтому взаимодействие пользователя и программных модулей со средой исполнения происходит только через специальные интерфейсы.

В случае отказа одного из программных модулей, когда работа модуля прекращается, необходим его перезапуск с восстановлением его последнего состояния. Для этого был применен метод контрольных точек и рестарта. После каждого удачного вычислительного цикла, программный модуль формирует данные контрольной точки, которые передаются в среду исполнения вместе с результатами работы модуля. Если на следующем цикле один или несколько программных компонент терпят отказ, то их состояние восстанавливается по данным предыдущей контрольной точки, выбранной решающим алгоритмом.

Разработанная компонентная архитектура мультиверсионных программных систем позволяет исключить взаимное влияние мультиверсионных программных модулей друг на друга, что обеспечивает независимость отказов отдельных модулей. Кроме того, разработанная на базе компонентной архитектуры среда исполнения, позволяет построить распределенную мультиверсионную систему, которая обеспечивает возможность подключения к среде исполнения любого количества программных модулей, распределяя вычислительную нагрузку на множество машин.

ИСТОРИЯ СЕТЕВЫХ ТЕХНОЛОГИЙ В ЛИЦАХ

Поздяев В.И, Пакшина Ю.П.
АПИ (филиал НГТУ),
Арзамас

Сетевые технологии все шире проникают во все сферы нашей жизни, поэтому изучению сетевых технологий отводится значительное место в учебных программах практически всех учебных заведений.

Изучение истории любой науки важно и актуально, как с познавательной, так и с нравственно-этической точки зрения. Но сказать, что это стало актуально именно сейчас – нельзя. Мысль эта не нова, об этом писал еще великий чешский педагог средневековья Ян Амос Коменский [1]:

"Так как изучение истории чрезвычайно радует чувства, возбуждает фантазию, украшает ученость, обогащает язык, изощряет суждение о вещах и т.п., я требую, чтобы оно постоянно сопровождало главные занятия во всех классах..... Для логического класса годится история механических проблем, дающая наслаждение уму человеческому, излагающая то, чего искали и что изобрели, то, что еще надо искать и изобретать".

К этой цитате, по сути, трудно что-либо добавить. Единственно, что современные наука и техника далеко перешагнули рамки "механических проблем" и их история стала неизмеримо богаче.

В истории информатики содержатся очень интересные материалы. Это – прежде всего творческие биографии выдающихся ученых, которые создавали и развивали кибернетику и информатику.

Говоря о любом научном или техническом достижении, чаще вспоминают об ученых, которым удалось решить ту или иную задачу, создать ту или иную программу. А предтечи, идеологи этих направлений, а также люди, которые первыми сформулировали и поставили эти задачи, очень часто остаются "за кадром".

Итак, попробуем приоткрыть завесу именно над ними. В основу Всемирной паутины положена идея гипертекста. Интересна история появления и внедрения гипертекстовой технологии.

Сама идея, как повсеместно считается, принадлежит замечательному американскому ученому Вэнниверу Бушу, который еще в 1945 году опубликовал статью "Пока мы мыслим". ("As We May Think"). И не случайно в октябре 1995 года во многих университетах США и Канады проходили торжественные конференции, посвященные 50-летию публикации в журнале "The Atlantic Monthly" этой работы Вэннивера Буша [2]. Процитируем ряд высказываний:

"…Обсудим устройство персонального назначения <…>

Имеется графический экран, клавиатура и кнопки управления. Когда пользователь ищет нужную книгу, он должен ввести ее мнемонический код и нажать нужную для поиска кнопку <…>

Появятся энциклопедии с готовыми ссылками для связывания информации и быстрого поиска…"

Статья датирована 1945 годом, в то время не было не только персональных компьютеров и сетей, вычислительная техника того времени была представлена единичными, опытными образцами, и была электромеханической и аналоговой. Как видите, задумка была замечательной.

Нисколько не умаляя заслуги Вэннивера Буша, нельзя не сказать еще об одном ученом, работавшем в Массачусетсском технологическом институте, которого тоже по праву можно отнести к идеологам Интернета. Это Норберт Винер.

Не стоит удивляться тому, что за Винером не числится никаких практических работ, связанных с компьютерами, в то время его занимали более серьезные вещи. Винер стал основателем кибернетической философии, основателем собственной школы, и его заслуга состоит в том, что эта философия была передана ученикам и последователям. Именно школе Винера принадлежит ряд работ, которые в конечном счете привели к рождению Интернета.

Винер пришел к необходимости организовать в университете еженедельный семинар с привлечением самых разных специалистов. Семинар начал работать весной 1948 г. Его участники вспоминают, что первое время он напоминал строительство Вавилонской башни, поскольку к нему были привлечены ученые разных, порой далеких друг от друга специальностей — математики, инженеры, психологи, философы, медики, биологи и т. д.

В конечном итоге удалось выработать несколько принципиальных концепций, которые можно рассматривать как первые основополагающие идеи будущей Сети.

Во-первых, на семинаре в процессе обсуждений было высказано предположение, что компьютер дол-