УДК 004.41

ОПЫТ РАЗРАБОТКИ МНОГОЦЕЛЕВОГО ТРЕНАЖЕРА В ЧАСТИ УПРАВЛЕНИЯ ДВИЖЕНИЕМ ИМИТИРУЕМОГО ОБЪЕКТА

С.А. Мешков, зав. группой; В.А. Пазюк, зав. отделом
(НИИ «Центрпрограммсистем», пр. 50 лет Октября, 3а, г. Тверь, 170024, Россия, john2211@rambler.ru, pazjuk_v@mail.ru)

Аннотация. Рассматриваются проектные решения, принятые при разработке составной части конкретного компьютерного тренажера, а именно подсистемы имитации и управления движением объекта. В частности, приводятся схема декомпозиции подсистемы и ее обоснование, техническое описание принципов ее реализации, а также выводы, сделанные на основании опыта, полученного в процессе указанной разработки.

Ключевые слова: *компьютерный тренажер, разработка, декомпозиция, проектные решения, библиотека Qt.*

В данной статье речь пойдет о компьютерных тренажерах, то есть о программно-аппаратных комплексах, предназначенных для обучения определенных коллективов людей (именуемых в дальнейшем экипажами) работе по обеспечению функционирования некоторых технических объектов. В рамках настоящей статьи под техническим объектом следует понимать *подводную лодку* (ПЛ).

Рассматриваемый тренажер структурно состоит из совокупности *рабочих станций* (РС), связанных в локальную сеть, на каждой из которых имеется программное обеспечение определенной функциональности. В частности, по функциональным свойствам можно выделить РС, выполняющие функции руководства процессом обучения (тренировки), – совокупность таких РС будем называть *пост руководства обучением* (ПРО), и РС, на которых работают непосредственно обучаемые члены экипажа, – такие РС будем называть *автоматизированные рабочие места обучаемых* (АРМО).

В процессе тренировки необходима имитация движения объекта. При этом из совокупности АРМО, составляющих тренажер, можно выделить подмножество АРМО, для которых одной из основных функций является именно управление движением. Поэтому целесообразно определить в качестве составной части тренажера упомянутое выше подмножество АРМО вместе с разрабатываемыми средствами имитации движения объекта. Эту составную часть будем называть подсистемой имитации и управления движением объекта.

В данной статье под многоцелевым тренажером понимается единый тренажер, предназначенный для проведения тренировок экипажей различных проектов ПЛ. Цель авторов – попытаться осмыслить опыт, полученный в результате разработки конкретного многоцелевого тренажера. При этом речь пойдет только о разработке подсистемы имитации и управления движением объекта.

Исходные предпосылки

Согласно требованиям заказчика, в части имитации и управления движением объекта тренажер должен обеспечивать следующие функции:

– программная имитация работы реальных пультов управления соответствующих объектов;

– визуальная имитация внешнего вида и работы органов управления соответствующих реальных панелей управления;

– имитация движения объекта в соответствии с сигналами, поступающими от программных имитаторов пультов управления;

– информационный обмен с другими составными частями тренажера.

Особенность данного тренажера в обеспечении возможности проведения тренировок экипажей различных ПЛ. Фактически в рамках одного АРМО необходимо реализовать имитацию нескольких пультов управления, относящихся к различным объектам.

Основные проектные решения

Декомпозиция разрабатываемой подсистемы имитации и управления движением объекта была осуществлена следующим образом.

1. *Контроллер движения.* Обеспечивает общую координацию работы всей подсистемы. В частности, поддерживает обмен информацией с другими составными частями тренажера, обеспечивает информационный обмен с АРМО подсистемы имитации и управления движением объекта, осуществляет на основе

заданной руководителем тренировки информации выбор конкретного объекта для проведения тренировки.

2. *Модель движения.* Обеспечивает имитацию параметров движения объекта в реальном времени тренировки с учетом начальных данных, заданных руководителем тренировки, и управляющих воздействий, поступающих от АРМО подсистемы имитации и управления движением объекта.

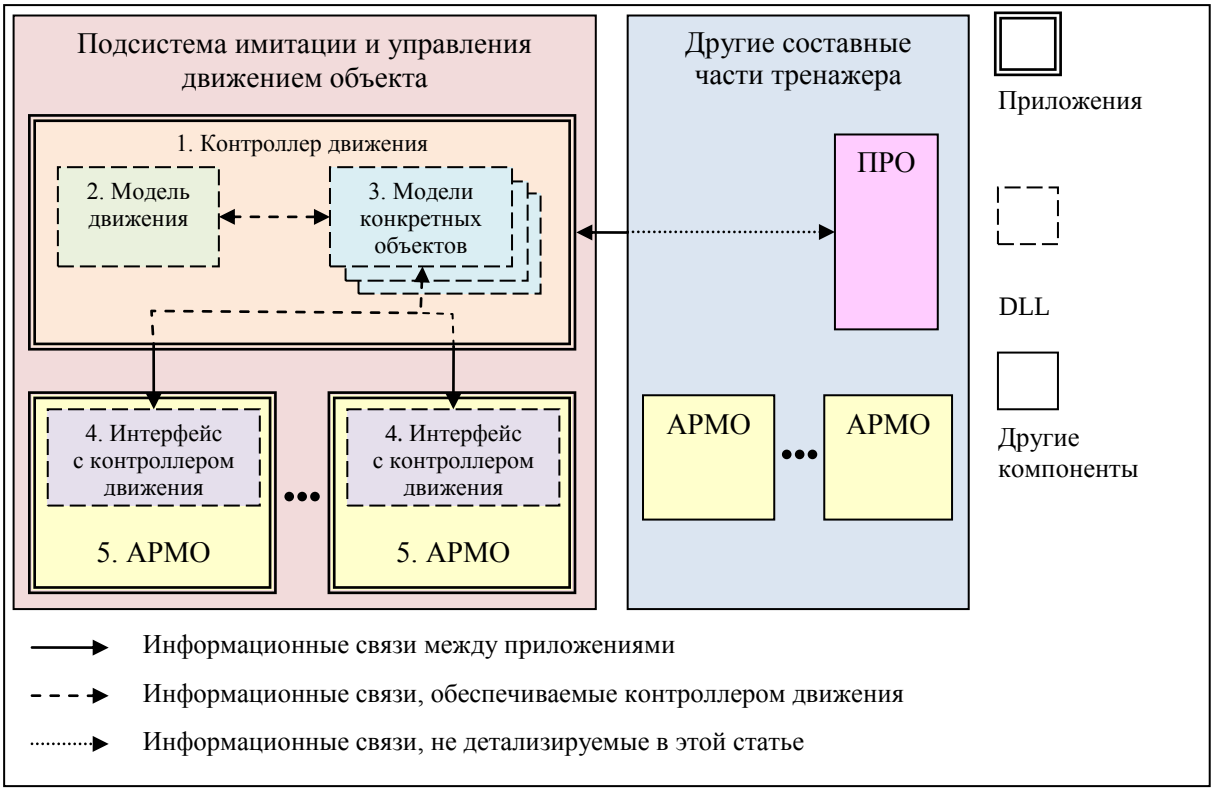
3. *Объектно-зависимые компоненты контроллера движения (модели конкретных объектов).* Обеспечивают интерпретацию управляющих воздействий, поступающих от АРМО подсистемы имитации и управления движением объекта, осуществляют имитацию отдельных процессов, специфических для конкретных объектов, и поставляют управляющую информацию для модели движения в унифицированной форме. Такой компонент разрабатывается для каждого из предусмотренных в рамках тренажера объекта тренировок.

4. *Интерфейс с контроллером движения («тонкий клиент»).* Обеспечивает на физическом уровне унифицированный интерфейс любого АРМО подсистемы имитации и управления движением объекта с контроллером движения. Этот компонент является составной частью каждого АРМО рассматриваемой подсистемы.

5. *АРМО подсистемы имитации и управления движением объекта.* Эти компоненты осуществляют программную имитацию работы реальных пультов управления соответствующих объектов и разрабатываются для каждого пульта управления каждого из предусмотренных в рамках тренажера объектов тренировок.

Разумеется, в рамках каждой отдельной тренировки могут использоваться только компоненты, ориентированные на определенный для данной тренировки объект.

Приведенную выше декомпозицию иллюстрирует рисунок.



Сделаем несколько замечаний в обоснование принятой декомпозиции.

Поскольку реальные пульты управления, имитация работы которых предусмотрена в рамках тренажера, весьма разнородны и не допускают ощутимой унификации, к тому же должны выполняться физически на нескольких РС, выбор совокупности, описанной в пункте 5, как одного из компонентов декомпозиции вполне естественен.

В качестве средства информационного обмена (учитывая наличие в составе нашей подсистемы нескольких РС) естественно выбрать один из стандартных сетевых интерфейсов.

Вместе с тем весьма тривиальный анализ показал: если структуре информационных связей между АРМО придать иерархический характер, структура сетевых сообщений допускает существенную унификацию.

Под иерархической структурой в предыдущем абзаце следует понимать запрещение информационного обмена одного АРМО непосредственно с другими и допущение такого обмена только с некоторой «высшей инстанцией» – иными словами, применение стандартной схемы сетевого взаимодействия «клиент – сервер». В этом случае поток сообщений от АРМО-клиента может содержать управляющие сигналы от оператора АРМО, а поток сообщений от сервера – информацию всем клиентам о текущем состоянии объекта моделирования (например, для отображения на приборах индикации).

Таким образом, упомянутая выше унификация структуры сетевых сообщений дает возможность выделить небольшой по объему и функциям, но полезный компонент декомпозиции, обеспечивающий для каждого АРМО унифицированный сетевой интерфейс – интерфейс с контроллером движения («тонкий клиент») (см. п. 4).

Когда для информационного обмена представляется удобным выбрать схему «клиент – сервер», возникает вопрос о выборе структуры и функций сервера. В нашей декомпозиции к ней относятся пункты 1–3.

Прежде всего следует заметить, что достижимая унификация структуры сетевых сообщений носит формальный характер: семантика этих сообщений существенно зависит от объекта, к которому эти сообщения относятся (вспомним, что тренажер многоцелевой). Поэтому логично выделить в структуре сервера компонент, зависящий от конкретного объекта имитации (см. п. 3).

С другой стороны, функция имитации движения объекта так или иначе реализует упрощенную физическую модель движения некоторого тела при довольно сложных внешних условиях и весьма разнообразных внешних воздействиях. При этом в рамках разрабатываемого тренажера как физические свойства имитируемого тела, так и предполагаемые внешние условия и воздействия допускают существенную унификацию, в силу чего унификацию допускает и сама модель. Результатом такой унификации является возможность описания специфических свойств имитируемого объекта в виде ограниченной совокупности параметров и реализации модели движения как алгоритма, настраиваемого посредством указанной совокупности параметров.

Теперь представляется логичным выделение в качестве отдельного компонента сервера модели движения (см. п. 2), не зависящего явно от имитируемого объекта.

Наконец, сам сервер как институт, объединяющий и координирующий взаимодействие своих (уже выделенных) компонентов, логично выделить в качестве собственно компонента нашей декомпозиции (см. п. 1).

Технические замечания

В соответствии с общими проектными решениями по тренажеру в целом реализация выполнялась для персональных компьютеров на платформе Intel x86 под ОС Windows 7 в качестве РС. Языком программирования выбран C++, инструментальная поддержка – среда разработки программ Microsoft Developer Studio 2008. При этом в качестве основной библиотеки поддержки разработчика была выбрана библиотека Qt версии 4.8.4. Последний выбор был обусловлен следующими обстоятельствами.

Так называемая библиотека Qt является весьма широким с точки зрения функциональности набором классов языка C++, обеспечивающим разработчика удобными инструментами для реализации гибкого графического интерфейса пользователя (что важно для АРМО), удобного и надежного сетевого интерфейса, обработки xml-файлов, создания языково-независимых приложений и многого другого.

Кроме того, библиотека Qt реализует некоторые расширения языка C++, в частности, механизм «сигнал – слот», применение которого оказалось удобным для реализации взаимодействия между компонентами нашей декомпозиции.

К тому же библиотека Qt изначально разрабатывалась как кросс-платформенный продукт, в перспективе обеспечивающий переносимость на другие платформы, в частности семейства Unix.

Наконец, не последнюю роль сыграло наличие положительного опыта использования библиотеки Qt рядом сотрудников нашего коллектива.

Подробное описание библиотеки Qt смотри, например, в [Шлее М.].

Отдельные компоненты подсистемы были реализованы следующим образом.

1. *Контроллер движения р*еализован как отдельное приложение, получающее при запуске информацию об объекте тренировки от руководителя тренировки. Имеет собственный пользовательский графический интерфейс вспомогательного характера, позволяющий контролировать процесс его выполнения.

2. *Модель движения* реализована как динамически загружаемая библиотека (DLL). Загружается контроллером движения после запуска. Информация о необходимых физических параметрах объекта тренировки содержится в предварительно подготовленном текстовом файле предопределенной структуры, имя которого определяется контроллером движения исходя из информации, полученной последним при запуске.

3. *Объектно-зависимые компоненты контроллера движения (модели конкретных объектов)* реализованы как набор динамически загружаемых библиотек (DLL) для каждого из предусмотренных объектов. Соответствующая требуемому объекту тренировки библиотека загружается контроллером движения после запуска исходя из информации, полученной последним при запуске. При загрузке контроллер движения передает ей необходимую информацию для прямой связи с (уже загруженной) моделью движения.

4. *Интерфейс с контроллером движения («тонкий клиент»)* реализован как динамически загружаемая библиотека (DLL). В обязательном порядке используется всеми АРМО для обеспечения информационной связи с контроллером движения.

5. *АРМО подсистемы имитации и управления движением объекта* реализованы как отдельные приложения. Запускаются на соответствующих РС по команде руководителя тренировки. В процессе выполнения имеют связь только с контроллером движения через унифицированный интерфейс.

Таким образом, сделаем некоторые выводы.

Поскольку разработка описанного тренажера в принципе завершена, следует отметить, что основные проектные решения в целом оказались правильными. Поэтому остановимся главным образом (но не только!) на тех моментах, которые вызвали определенные трудности и сомнения.

1. В процессе разработки оказалось целесообразным наделить контроллер движения дополнительными функциями, а именно:

– в целях автономной отладки подсистемы обеспечение возможности выполнения контроллера движения и зависящих от него АРМО подсистемы автономно, без наличия связи с остальными составными частями тренажера (в том числе ПРО).

– обеспечение получения существенной информации о состоянии объекта в рамках как модели движения, так и модели конкретного объекта в процессе тренировки средствами графического интерфейса пользователя, а также посредством построения графиков зависимости интересующих параметров от времени.

Эти функции были реализованы в процессе разработки.

2. Поскольку в каждой программе есть хотя бы одна ошибка, важное место при разработке любой программной системы занимает вопрос о ее поведении при внезапном аварийном завершении какого-либо компонента. Прежде всего это касается приложений, выполняющихся в ее составе. Для отдельных АРМО этот вопрос в рамках разрабатываемого тренажера решается достаточно безболезненно: при перезапуске внезапно завершившегося АРМО и его успешном соединении с контроллером движения оно получает всю необходимую информацию для дальнейшего нормального функционирования в условиях текущей тренировки.

При внезапном завершении контроллера движения возникает гораздо более неприятная ситуация: в этом случае вся внутренняя (относящаяся к состоянию объекта) информация теряется практически безвозвратно. После перезапуска контроллера движения она может быть восстановлена только в той части, которая может быть получена из информации, поступающей от руководителя тренировки.

Эта проблема не была решена в процессе разработки. В перспективе просматриваются два пути ее разрешения.

Во-первых, в рамках контроллера движения можно предусмотреть непрерывное в течение тренировки периодическое сохранение внутренней информации в некоторый файл (с точки зрения эффективности целесообразно использовать оперативную или виртуальную память, доступную в рамках соответствующей платформы) с последующим ее использованием при перезапуске контроллера движения с должным учетом обстоятельств запуска.

Во-вторых, опять-таки с учетом обстоятельств запуска, можно попытаться получить от присоединенных АРМО имеющуюся у них внутреннюю информацию и таким образом адаптировать контроллер движения к текущему состоянию тренировки.

3. Не очень привлекательным выглядит разделение информации об объекте на два фактически независимых компонента: совокупность информации для модели движения (унифицированная) и совокупность информации для модели конкретного объекта (специфичного формата). Целесообразность их объединения требует дополнительной проработки.

4. Несмотря на вроде бы логичный и достаточный интерфейс с контроллером движения, некоторые разработчики сочли полезным «обернуть» предоставляемый этим интерфейсом класс своим собственным классом. Возможно, следует доработать стандартный класс, предоставляемый этим интерфейсом, чтобы учесть это обстоятельство.

Авторы надеются, что изложенный в статье опыт разработки и сделанные выводы могут оказаться полезными при дальнейших разработках аналогичных программных продуктов.

Литература

Шлее М. Qt4. Профессиональное программирование на C++. СПб: БХВ-Петербург, 2007.