на ОС *Windows 2000/XP*, Мобильной системе Вооруженных сил 3.0 и защищенной ОС «Оливия». В настоящее время СИНОМ является основой информационного обмена в разрабатываемых изделиях.

Дальнейшее развитие системы заключается в автоматизации процесса создания и отправки сообщений, а также в распараллеливании и обеспечении потокобезопасного взаимодействия функциональных модулей и *sinom.dll*.

Список литературы

1. Androutsellis-Theotokis S., Spinellis D. A Survey of Peer-to-Peer Content Distribution Technologies. // ACM Computing Surveys. 2004. Vol. 36. – № 4, pp. 335–371.

2. Khan I.J., Wierzbicki A. Foundation of Peer-to-Peer Computing, Special Issue. // Elsevier Journal of Computer Communication. 2008. V 31. Issue 2, pp. 137–161.

3. Stroustrup B. The C++ Programming Language. 3rd. ed. – Addison-Wesley Publishing Company, 1997. – 1040 p.

4. Alexandrescu A. Modern C++ Design: Generic Programming and Design Patterns Applied. – Addison-Wesley. 2001.

ПОДХОД К РАЗРАБОТКЕ КРОССПЛАТФОРМЕННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

А.А. Прохоров; А.А. Карпов (НИИ «Центрпрограммсистем», г. Тверь, , prolex@cps.tver.ru)

Ключевые слова: *целевая операционная система, рабочая среда, сборка, платформо-зависимая среда, кроссплатформенное программное обеспечение.*

При смене целевой *операционной системы* (ОС) программного изделия для разработчика существуют два основных варианта организации разработки *программного обеспечения* (ПО): полностью перевести процесс разработки на целевую ОС; использовать привычную, устоявшуюся рабочую среду (соответственно, и прежнюю ОС) для разработки ПО, а в целевой ОС проводить окончательное тестирование и исправление недоработок, связанных со спецификой ОС и компиляторов.

Первый вариант организации разработки ПО потребует от разработчиков:

– отказаться от используемого длительное время инструментария разработки ПО;

– исследовать рынок готовых технологических программных продуктов для целевой ОС на соответствие требованиям и функциональным возможностям, обеспечивающим разработку ПО в срок с заданным качеством;

– обучить работе с новым технологическим ПО в целевой ОС.

Второй вариант требует от разработчиков меньшего времени на изучение использования нового технологического ПО, так как в основном будет использоваться прежняя рабочая среда.

В данной статье описан подход к организации разработки программного изделия по второму варианту, где *защищенная* ОС (ЗОС) *Мобильная система Вооруженных сил* (МСВС) является целевой ОС, а *Windows* – основной ОС, используемой при разработке ПО.

Рассмотрим задачи, которые необходимо решить при организации разработки ПО в ОС *Windows* и ЗОС МСВС:

– поддержка коллективной работы с исходным текстом и документацией по проекту (разработка структуры дерева исходного текста проекта и обеспечение работы с репозитарием проекта в обеих ОС);

– обеспечение компиляции одного и того же исходного текста проекта, сборки исполняемых файлов и создание дистрибутива в обеих ОС;

– обеспечение автоматического ежедневного создания дистрибутива программного изделия в обеих ОС;

– поддержка ресурса для фиксации ошибок, выявленных на этапах компиляции, сборки и тестирования программного изделия в обеих ОС;

– поддержка пополняемого информационного ресурса для разработчиков, раскрывающего особенности ведения разработки ПО под ОС *Windows* и ЗОС МСВС и доступного для чтения и пополнения из обеих ОС.

К платформо-зависимым элементам, используемым при разработке программного продукта, можно отнести графический интерфейс пользователя, векторную и растровую графику, работу с видео и звуком, сетевой обмен и т.д.

Для обеспечения переносимости исходного кода ПО между двумя ОС используется кроссплатформенная библиотека *Qt4* (*qtsoftware.com*), позволяющая избавиться от прямого использования API конкретной ОС – библиотека берет это на себя (рис. 1).

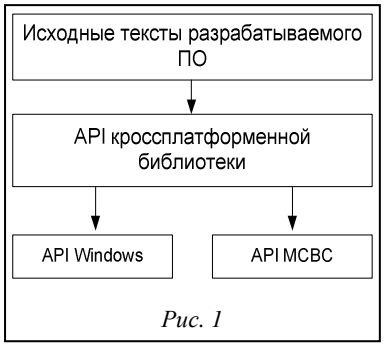

Рис. 1

Для компиляции и сборки ПО в ОС *Windows* используется *MS Visual Studio*, в ЗОС МСВС – утили-

та *qmake* из состава пакета *Qt*, использующая компилятор *C*$^{++}$ из набора компиляторов *gcc*. Для работы с документацией проекта используется пакет офисных приложений *OpenOffice* (*openoffice.org*), обеспечивающий работу в обеих ОС.

При коллективной разработке ПО возникает необходимость управления исходным текстом и документацией проекта. Для поддержки данного процесса используется система *Subversion* (*subversion.tigris.org*), работа с которой ведется из обеих ОС. Репозитарий *Subversion* включает в себя документацию и исходный текст проекта.

Прежде чем говорить о файловой структуре всего проекта, рассмотрим структуру отдельно взятого модуля (рис. 2). Назначение каталогов представленной структуры следующее: **doc** – хранение документации по модулю, его проектных решений; **include** – хранение заголовочных файлов исходных текстов модуля; **msvs** – хранение файлов проекта модуля для компиляции в среде *MS Visual Studio* в ОС *Windows*; **qmake** – хранение файлов проекта модуля для компиляции в ЗОС МСВС; **res** – хранение файлов ресурсов модуля (отсутствует в модулях, не имеющих графического отображения); **src** – хранение файлов исходных текстов модуля.

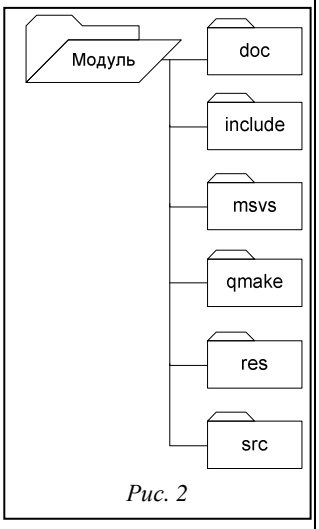

Рис. 2

Предлагаемая файловая структура всего проекта показана на рисунке 3.

Проект содержит подкаталоги: **bin** – для откомпилированных и сторонних модулей, соответственно для *debug*- и *release*-версий; **build** – для файлов скриптов при компиляции и сборке дистрибутива в ОС *Windows* и МСВС соответственно; **doc** – для документов описания используемых в проекте технологий, подходов и т.д.; **include** – для заголовочных файлов, используемых несколькими модулями проекта; **lib** – для файлов описания библиотек, используемых модулями проекта; **msvs** – для файла всего проекта (*solution*) при компиляции в среде *MS Visual Studio* в ОС *Windows*; **projects** – для каталогов всех модулей проекта; **qmake** – для файла всего проекта при компиляции в ЗОС МСВС.

Для автоматической ежедневной сборки дистрибутива в *Windows* и МСВС необходимо выделить компьютер с запущенной виртуальной машиной (то есть в *Windows* запущена виртуальная машина ЗОС МСВС) либо два компьютера (для каждой ОС в отдельности) с настроенным технологическим ПО и назначить задание на компиляцию и сборку дистрибутива. Под заданием понимается исполнение созданного скрипта компиляции и сборки, запускаемого, как правило, в ночное время. Скрипт обеспечивает скачивание обновлений исходного текста из репозитария, компилирование, сборку исполняемых файлов и создание установочного пакета (дистрибутива) разрабатываемого программного изделия.

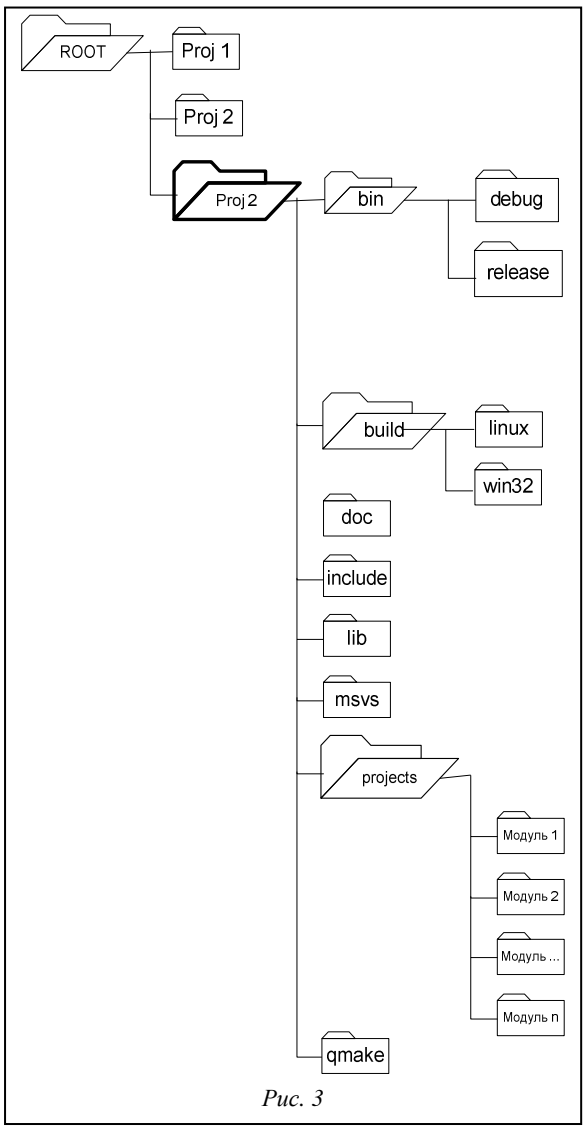

Рис. 3

Для информирования разработчиков об ошибках при сборке и тестировании отдельных модулей или проекта в целом используется система учета ошибок *Mantis* (*mantisbt.org*), доступная из обеих ОС.

Также создан внутренний информационный *wiki*-ресурс (*mediawiki.org*) для разработчиков, в рамках которого можно размещать обнаруженные непереносимые конструкции кода и советы по ведению кроссплатформенных разработок. Доступ к данному ресурсу необходимо обеспечить как из ОС *Windows*, так и из ЗОС МСВС.

Данный подход к организации процесса разработки ПО в ОС *Windows* для последующего его

использования в ЗОС МСВС, как показала практика разработки изделий, не оказывает существенного влияния на увеличение сроков выполнения работ и качество выпускаемого программного изделия.

Литература

1. Хоффман Б., Мартин К. Разработка программного обеспечения в небольшой организации (http://www.osp.ru/os/2007/03/4158385).

2. Лапыгин Д., Новичков А. Конфигурационное управление проектами разработки программного обеспечения (http://www.citforum.ru/SE/quality/configuration_management).

ВАРИАНТЫ ВЗАИМОДЕЙСТВИЯ РАБОЧИХ МЕСТ ТАКТИЧЕСКОГО ТРЕНАЖЕРА

А.Б. Шорин; И.В. Новиков (НИИ «Центрпрограммсистем», г. Тверь, oneillcps@gmail.com)

Ключевые слова: *система моделирования, АРМ, синхронизация данных, клиент модели, адекватность моделирования, активный и пассивный объекты.*

В состав тактического тренажера входят центральный пост руководства обучением, центральный *вычислительно-моделирующий комплекс* (ВМК) и *АРМ обучаемых* (АРМО).

ВМК – это распределенная система моделирования, функционирование которой строится по принципу клиент/сервер. Сервер ВМК отвечает за инициализацию системы моделирования (на основе задания на тренировку), управление единым временем тренажера, добавление и удаление моделей. Сервер ВМК в составе тренажера существует в единственном экземпляре. Клиент ВМК связывается с сервером и получает сведения о модельном времени, о состоянии тренировки и моделей тактической обстановки. Клиентов ВМК может быть множество. Как сервер, так и клиент ВМК реализуются в виде динамической библиотеки и предоставляют доступ ко всем сервисам ВМК. Данная библиотека используется также и приложениями, реализующими АРМО.

ВМК состоит из исполнительной системы и библиотеки имитационных моделей.

Моделирование тактической обстановки предполагает имитацию широкого спектра разнородных объектов, процессов и явлений. Перечень объектов моделирования нестабилен и может быть расширен в процессе разработки. Система моделирования предполагает максимальное отделение моделей от остальных компонентов системы и формализацию способов взаимодействия моделей. Исполнительная система располагает минимальным объемом информации о моделях, благодаря чему может работать с разнородным составом моделей. Данный подход позволяет произвольно масштабировать систему. Моделирование тактической обстановки с точки зрения ВМК – это процесс взаимодействия моделей, причем способ взаимодействия определяется самими моделями. Исполнительная система обеспечивает распределение моделей по узлам вычислительной сети тренажера и выполнение моделей.

Центральным понятием системы моделирования является модель объекта (системы). Все модели реализуются в отдельных модулях, которые динамически загружаются в ходе инициализации системы моделирования. Дополнение и изменение программных модулей имитационных моделей возможно как при разработке комплекса, так и при его модернизации.

Современный тактический тренажер – это распределенная вычислительная система. Результаты моделирования объектов обстановки должны быть доступными на всех вычислительных узлах сети. Для этого каждая модель реализует возможность функционирования в двух режимах – клиента и сервера. На одном узле сети функционирует сервер модели, который осуществляет собственно процесс моделирования. На остальных узлах работает клиент модели, предоставляющий доступ к результатам моделирования и передающий запросы пользователей модели серверу.

Сервис сетевого взаимодействия моделей, предоставляемый ВМК, решает вопросы синхронизации данных моделируемых объектов. Сервис предоставляет функции передачи данных с серверного экземпляра модели всем существующим в тренажном комплексе клиентским экземплярам модели (синхронизация клиентов), передачи данных от клиентского экземпляра модели серверному экземпляру (команды управления) и адресной передачи данных от любого экземпляра модели любому указанному экземпляру (адресное сообщение). Первые две функции освобождают разработчика модели от знания расположения серверного экземпляра и знания расположения и количества клиентских экземпляров моделей.

Принципиально модель реализуется в виде неделимого программного объекта, который может функционировать в одном из двух режимов – серверном и клиентском. За счет этого достигается возможность гибкого конфигурирования процесса моделирования: сервер модели может работать на любом вычислительном узле. Программы, использующие модель, принципиально не должны заботиться о том, в каком режиме она функционирует.