УДК 629.7.035:681.5

Д.И. ВОЛКОВ, В.М. ГРУДИНКИН, В.В. ДАНИЛОВ, Г.С. РАНЧЕНКО

ОАО «Элемент», Украина

СОВРЕМЕННЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ ЭЛЕКТРОННЫХ ИЗДЕЛИЙ ОТВЕТСТВЕННОГО НАЗНАЧЕНИЯ ДЛЯ АВИАЦИОННОЙ ТЕХНИКИ

На примере двигательной FADEC рассмотрены требования к её программному обеспечению, а также к процессу его разработки. Основное внимание уделено практическим аспектам, в частности архитектуре и реализации программного кода. Отмечено значение математической модели при решении задач управления и диагностирования. Показана неразрывность обеспечения требований к ПО и к процессу разработки. Затронуты вопросы проектирования, кодирования и тестирования, а также применяемый инструментарий, начиная с системы контроля версий и заканчивая автоматизированным генерированием электронной документации.

Ключевые слова: *программное обеспечение, требования, процесс разработки, тестирование, моделирование, FADEC, авиационная техника.*

Введение

Требования к программному обеспечению (ПО) можно условно разделить на требования непосредственно к ПО и на требования к процессу его разработки. Конечной целью, имеющей бизнес ценность, являются требования к ПО, однако их обеспечение невозможно без соответствующей организации процесса разработки. Учитывая то, что наивысшую категорию критичности среди электронных систем ЛА имеют системы управления, в частности двигателем, в статье рассматривается двигательная FADEC.

Итак, **целью настоящей статьи** является с одной стороны тезисное освещение практических аспектов процесса разработки FADEC, в целом определяемых [1, 2], а также возможностей встраиваемого программного обеспечения. Кроме того, хотелось бы отметить, что рассматриваемые подходы и инструментарий соответствуют практикам и экспертизе ОАО «Элемент», не являясь единственно возможными.

1. Требования к программному обеспечению

Характерные особенности ПО современной FADEC:

– математическая модель реального времени;

– многопоточность обработки данных;

– трассируемость кода;

– обработка и регистрация исключительных ситуаций;

– возможность удалённого конфигурирования и обновления ПО;

– виртуализация ресурсов смежных аппаратных узлов.

Визитной карточкой современных FADEC является использование математической модели двигателя или её частей в алгоритмах управления и диагностирования. Использование в контуре управления математической модели позволяет реализовать наблюдатели, неизмеряемых или измеряемых с недостаточной точностью параметров работы двигателя [3], таких как мощность или тяга, запас газодинамической устойчивости, ускорение ротора турбокомпрессора и т.д.

Другим характерным признаком является многопоточность/многопроцессность. Основной причиной является то, что системы обрастают всё большим количеством вспомогательных функций, не относящихся напрямую к процессу управления, но повышающих эксплуатационные характеристики двигателя и ЛА в целом. Это функции связанные с контролем и диагностикой двигателя, регистрацией и статистической обработкой полётной информации и т.д. Естественно, даже значительно возросшая производительность не позволяет выполнять весь спектр вычислений. Поэтому задачи реального времени выделяются в отдельный высокоприоритетный поток/процесс, а остальные задачи выполняются в фоновых процессах. В случае отсутствия операционной системы и аппаратной поддержки многопоточности вычислений применяются псевдомногопоточные архитектурные решения непосредственно рабочего ПО.

Отладка многопоточного ПО требует его трассируемости, предусматривающей всеобъемлющее

покрытие программного кода диагностическими сообщениями, повышающими качество тестирования ПО и упрощающими процесс его отладки за счёт более точной локализации дефектов.

Сбор сообщений реализуется во встраиваемом сервере сообщений обеспечивающим их сбор в реальном времени, сопровождая сообщения специальными временными маркерами, буферизацию и последующую передачу в КПА или другой технологический компьютер. Кроме сбора сообщений сервер обеспечивает возможность специальных тестово-технологических вызовов, что упрощает выполнение тестирования, особенно автоматизированного.

Естественно обработка сервером полного потока диагностических сообщений требует значительных вычислительных мощностей и пропускной способности технологического информационного канала, то есть обычно является невозможной. Поэтому в сервере реализуется встроенная система фильтрации, отключающая непосредственно выполнение кода, генерирующего сообщения, не соответствующие заданному профилю, что практически исключает расход вычислительной мощности. Задание профиля выполняется непосредственно в процессе выполнения ПО при помощи КПА или технологического компьютера.

Одним из наиболее тяжёлых с точки зрения отладки дефектов является редкое и тяжело воспроизводимое аварийное завершение работы ПО, сопровождающееся повисанием (маловероятно, учитывая наличие программных и аппаратных watchdog) или перезагрузкой процессора. Использование стандартных механизмов, включая сервер сообщений, в этом случае малоэффективно. Одно из решений – отправка в обработчике исключительных ситуаций диагностического сообщения, содержащего информацию о типе и месте возникновения исключительной ситуации, состоянии регистров и стека. Однако это решение является половинчатым, так как предполагает обязательную его регистрацию, что далеко не всегда является возможным. Более надёжным решением, приемлемым даже в эксплуатации является сохранение вышеозначенной информации (core dump) в энергонезависимой памяти. Содержимое стека в момент возникновения исключительной ситуации позволяет при помощи соответствующих программных средств однозначно восстановить, в том числе вложенность выполняемых методов, то есть локализовать участки кода, обуславливающие дефект. Кроме того, предусматриваются временные метрики для контроля критического возрастания вычислительной нагрузки, а также выявления и оптимизации критических с точки зрения производительности участков кода.

Другим направлением повышения отказоустойчивости является виртуализация ресурсов смежных аппаратных узлов, которая представляет собой набор программных решений, обеспечивающих удалённый доступ к информационным и вычислительным ресурсам. Использование вычислительных ресурсов не является особо актуальным в FADEC, которая как система реального времени имеет прогнозируемую вычислительную нагрузку и обязательный вычислительный резерв. Виртуализация же информационных ресурсов напротив позволяет повысить реконфигурируемость и отказоустойчивость системы. Так, например, при отсутствии виртуализации возможен переход на резервный канал управления при наличии отказа всего лишь одного измерительного канала в основном канале управления. При её использовании основной канал может продолжать полноценное функционирование при гораздо большей степени деградации аппаратных средств за счёт подмены значений параметров значениями из другого канала управления.

2. Требования к процессу разработки программного обеспечения

Процесс разработки является итеративным с короткими итерациями и характеризуется использованием:

– системы контроля версий (version control system);

– интегрированной среды разработки с библиотеками (framework);

– среды автоматизированного тестирования, интегрированной в процедуру сборки;

– среды математического моделирования;

– системы отслеживания дефектов (bugtracking system);

– непрерывного статического тестирования ПО и вносимых изменений (code review);

– инструментария генерирования электронной документации для программного кода.

Управление конфигурацией ПО и соответственно процесс разработки базируется на системы управления версиями. Применение таких систем упрощает сопровождение изменений вносимых в программный код, интеграцию отдельных частей проекта, а также маркировку и контроль версий. Яркий и широко используемый пример таких систем являют open source система SVN (Subversion) и проприетарная для коммерческого использования система Perforce.

Другим немаловажным аспектом процесса разработки является интегрированная среда. В последнее десятилетие сложилась практика, что производители процессоров предоставляют интегрированные среды разработки или тесно сотрудничают с разработчиками последних. Нередко для процессо-

ров существует несколько альтернативных сред, предоставляющие разработчику:

– средства удалённой загрузки и отладки;

– шаблоны компонентов и программные интерфейсы для их взаимодействия;

– поддержка мультипоточности и/или мультипроцессности с интегрированными средствами контроля для каждого потока/процесса;

– средства синхронизации потоков данных и управления.

Использование шаблонов программных компонентов обеспечивает унифицированное решение, снижая с одной стороны трудозатраты на программную реализацию компонента, а с другой стороны снижая риски возникновения ошибок в архитектуре или реализации ПО при его разработке.

Программные интерфейсы обычно абстрагируются двумя сущностями, такими как методы и атрибуты. Методы используются для инициативного воздействия одного компонента на другой, предоставляющий соответствующий интерфейс. Атрибуты представляют собой решения, необходимые для нотификации одного или нескольких компонентов об изменении соответствующих данных. Для снижения информационного потока часто значения атрибутов передаются при их изменении только требующим их компонентам.

Следует отметить, что нередко современные решения обеспечивают единый механизм как межпроцессного взаимодействия внутри вычислителя, так и на базе физического протокола, что позволяет применять программные решения как для централизованной, так и для распределённой систем.

Программные средства разработки и автоматизированного тестирования как интегрированные в среду разработки, так и внешние, являются неотъемлемой частью процесса разработки. Автоматизированное тестирование выполняется как непосредственно в процессе сборки на платформе, на которой реализован используемый кросс-компилятор, например, в ОС Windows, x86, так и на целевой (target) платформе, в которой исполняется рабочее ПО. Модульные и компонентные тесты исполняются на обеих платформах, причём при возможности также используют виртуальные машины, симулирующие целевую платформу. При тестировании проверяется не только выполнение функций, но и корректность использования ресурсов (отсутствие утечек памяти, превышения временных лимитов). Полноценное системное тестирование возможно только на целевой платформе в составе системы. При этом автоматизация тестирования обеспечивается как за счёт внешних программно-аппаратных решений, в том числе стенд-имитатор ГТД, так и за счёт встроенного сервера сообщений. Последний позволяет выполнять более специфические испытания и обеспечивает большую глубину контроля.

Кроме того, частично системное тестирование выполняется на персональном компьютере с использованием программных средств симуляции и моделирования. Применение этого подхода в совокупности с последующим тестированием в составе целевой системы позволяет снизить временные и финансовые затраты, расширить покрытие требований за счёт тестирования нештатных специфических ситуаций и т.д.

Моделирование систем выполняется при помощи специальных сред моделирования. Перечень таких сред достаточно широк, например, в ОАО «Элемент» применяются Matlab (Mathwork) и LabVIEW (National Instruments). Использование таких средств позволяет выполнять не только тестирование ПО в его классическом понимании, но и осуществлять непрерывный контроль параметров качества регулирования, запасов устойчивости контуров, временных требований (например, время приемистости) а также других требований, предъявляемых к управлению двигателем. Характерными особенностями вышеперечисленных сред является возможность использования языков программирования, в частности C/C++, которые весьма широко используются во встраиваемых системах. Это позволяет использовать модель регулятора, базирующуюся на оригинальном программном коде и, соответственно, максимально приближённо моделирующую реальную работу регулятора.

Полноценная организация процесса тестирования невозможна без использования системы отслеживания дефектов, которая обеспечивает как сбор и планирование устранения дефектов, так и контроль устранения дефектов. Системы широко представлены на рынке, как проприетарные, так и свободно распространяемые. Из последних имеет смысл выделить Trac, которая интегрируется с системой контроля версий SVN.

Кроме динамического тестирования немаловажным является статическое тестирование (review) программного кода, в том числе при внесении в него изменений. Статическое тестирование предусматривает проверку соответствия требованиям, отсутствия некорректных и небезопасных конструкций. Автоматизация статического тестирования обеспечивается применением такого инструментария анализа кода как lint.

Также при статическом тестировании проверяется соответствие принятому на предприятии стандарту кодирования для используемого языка программирования. Как показывает практика даже конвенция об именовании переменных и методов значительно повышает читаемость кода, упрощает совместное владе-

ние кодом для разработчиков, вовлечённых в проект, упрощает повторное использование кода.

Другой немаловажной стороной оформления кода является его комментирование. Это не только способствует вышеперечисленному, но и является основой для автоматизированной генерации электронной документации. Полноценная электронная справочная документация может быть сгенерирована при помощи doxygen, кроссплатформенной системы документирования исходных текстов. Кроме непосредственно генерации документации также выполняется дополнительный контроль оформления кода.

Заключение

Итак, рассмотрены практические требования к программному обеспечению электронных изделий авиационного назначения и к процессу его разработки. Кроме понимания изложенных аспектов, необходимо обеспечивать трассируемость требований на всех стадиях разработки, выполнять периодические аудиты выполнения процесса разработки.

Нужно также принимать во внимание то, что технологии и подходы стремительно развиваются. Поэтому внедряемые процессы должны периодически пересматриваться и оптимизировать, сотрудники повышать свой профессиональный уровень, знание применяемых процессов. Вообще человеческий фактор приобретает определяющее значение в определении нашей технологичной и наукоёмкой отрасли.

Литература

1. КТ-178В. Требования к программному обеспечению бортовой аппаратуры и систем при сертификации авиационной техники.

2. РЦ-АП 33.28-1 Электрические и электронные системы управления двигателями. Требования к программному обеспечению.

3. Волков Д.И. Оптимальный наблюдатель крутящего момента вертолётного ТВАД / Д.И. Волков // Авиационно-космическая техника и технология. – 2004. – № 8/16. – С. 131 – 135.

Поступила в редакцию 1.06.2011

Рецензент: д-р техн. наук, проф. С.А. Паложаенко, Одесский национальный политехнический университет, Одесса, Украина.

СУЧАСНІ ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЕЛЕКТРОННИХ ВИРОБІВ ВІДПОВІДАЛЬНОГО ПРИЗНАЧЕННЯ ДЛЯ АВІАЦІЙНОЇ ТЕХНІКИ

Д.І. Волков, В.М. Грудинкін, В.В. Данілов, Г.С. Ранченко

На прикладі FADEC для двигуна розглянуто вимоги до її програмного забезпечення, а також до процесу його розробки. Головну увагу приділено практичним аспектам, зокрема архітектурі та реалізації програмного коду. Відмічено значення математичної моделі задля вирішення задач управління та діагностування. Показана невідривність забезпечення вимог до ПЗ та процесу розробки. Розглянуто питання проектування, кодування та тестування, а також інструментарій, що застосовується, починаючи від системи контролю версій до автоматизованого генерування електронної документації.

Ключові слова: програмне забезпечення, вимоги, процес розробки тестування, моделювання, FADEC, авіаційна техніка.

STATE-OF-THE-ART REQUIREMENTS FOR SOFTWARE OF CRITICAL AVIATION ELECTRONIC UNITS

D.I. Volkov, V.M. Grudinkin, V.V. Danilov, G.S. Ranchenko

The requirements are considered for the engine FADEC software and for the software development process. Main focus is devoted to the practical aspects, including architecture and code implementation issue. The importance of mathematical model is highlighted with regard to control and diagnostic tasks. Unity of the software requirements and development process is shown. The design, coding and testing issues are touched upon as well as used tools like version control system, documentation system etc.

Key words: software, requirements, development process, testing, modeling, FADEC, aeronautical engineering.

Волков Дмитрий Иванович – канд. техн. наук, старший научный сотрудник, зам. главного конструктора ОАО «Элемент», Одесса, Украина, email: DIVolkov@element.od.ua.

Грудинкин Вячеслав Михайлович – зам. главного конструктора ОАО «Элемент», Одесса, Украина, email: Odessa@element.od.ua.

Данилов Всеволод Владимирович – ведущий программист ОАО «Элемент», Одесса, Украина, email: Odessa@element.od.ua.

Ранченко Геннадий Степанович – канд. техн. наук, Директор-Главный конструктор ОАО «Элемент», Одесса, Украина, email: Odessa@element.od.ua.