УДК 519.681.3

DOI: 10.15827/0236-235X.118.291-300

Дата подачи статьи: 10.11.16

2017. Т. 30. № 2. С. 291–300

ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ МОДУЛЬНОЙ АРХИТЕКТУРЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВИАЦИОННОЙ ТЕМАТИКИ

*Л.А. Чижикова, ведущий специалист, l_chizhikova@scac.ru
(Компания «Гражданские самолеты Сухого»,
ул. Поликарпова, 23б, стр. 2, г. Москва, 125284, Россия)*

Разработка ПО – довольно сложный и трудоемкий процесс, в котором проектирование корректной и надежной архитектуры (структуры) играет ключевую роль. Распределение и координация усилий по созданию ПО в группе разработчиков часто оказываются наиболее ответственными и трудными решениями, так как влияют на основной результат.

С развитием технологий для функциональности и управления бортовым радиоэлектронным оборудованием требуется увеличение объемов работ по созданию и сопровождению ПО. В сфере производства авиационных тренажеров и контрольно-проверочной техники роль проектирования, разработки и сопровождения ПО также возрастает. Все большая часть задач переносится с аппаратной части на ПО.

В статье дается анализ основных аспектов проектирования ПО авиационного назначения, сопоставляются принципы проектирования архитектуры ПО для бортового оборудования гражданского самолета и архитектуры ПО авиационного тренажера. Для представления единой модели архитектуры ПО исследованы работы по проектированию архитектуры ПО авиационного тренажера и архитектура бортового ПО самолета. Проведено сравнение подходов, выявлены их общности. Также рассмотрена модель, успешно применяемая при проектировании ПО компанией Rockwell Automation, исследован ряд работ Делфтского технического университета по рассматриваемой тематике.

В работе предлагается математическое представление модульной архитектуры программного продукта, ориентированного на использование в авиационной индустрии.

Предложенный подход к проектированию ПО для применения в авиационной отрасли позволяет унифицировать разработку программных продуктов, сократить временные затраты и трудоемкость их создания, вносить инновационные решения без трансформирования существующей структуры при условии, что программный продукт создавался с применением описанного решения.

Ключевые слова: ПО, архитектура ПО, проектирование архитектуры ПО, разработка ПО, авиационные тренажеры, авиационное ПО, математическое моделирование.

Проектирование ПО является неотъемлемой частью любого проекта, основополагающим фундаментом будущей разработки.

Программная реализация алгоритмов взаимодействия программных модулей и их информационный обмен на аппаратном уровне обеспечивают корректное функционирование разрабатываемой системы.

В группе разработчиков необходимо правильно распределять и координировать усилия по созданию ПО, при этом большое значение имеет четкое определение структуры (архитектуры) разрабатываемой программы [1].

Часто при изначально некорректном проектировании структуры программного продукта на этапе разработки или сопровождения возникает проблема регрессии программного кода, когда дальнейшее развитие и сопровождение невозможны.

Целями исследования способов проектирования архитектуры ПО были выявление, оптимизация и унификация предложенных решений по разработке ПО для применения в авиационной отрасли.

В результате изучения опыта применения модульной структуры ПО компанией Rockwell Automation, исследования зарубежных научных источников, описывающих возможность применения модульной архитектуры при разработке ПО, а

также анализа применяемых на практике технологий разработки ПО в авиационной отрасли было установлено, что основным аспектом качественного проектирования ПО для применения в рассматриваемой области является модульность. Такой подход обеспечивает гибкость и наращиваемость разрабатываемого продукта, легкость в сопровождении, экономии ресурсов.

Методы проектирования архитектуры ПО авиационного тренажера

Рассмотрим методы проектирования архитектуры ПО на примере авиационного полнопилотажного тренажера.

Использование модульной архитектуры ПО в тренажерах предполагает имитацию каждой физической системы в отдельном программном блоке.

При модульной архитектуре построения программные блоки могут создаваться независимо друг от друга и объединяться в систему для получения необходимых результатов [2].

Решением задачи корректного взаимодействия программных модулей является применение алгоритмов передачи данных от модуля к модулю. Данная задача включает как реализацию логических связей между программными блоками, так и обмен информацией системы с взаимодействующими подсистемами, при этом в архитектуре ПО необхо-

димо выделить программный модуль (модули) интерфейсного обмена [2].

При проектировании архитектуры всей системы авиационного тренажера его составляющие (программные блоки) условно можно распределить по уровням взаимодействия. Как правило, в архитектуре ПО выделяют пять уровней распределения программных блоков [1, 3].

В работе [1] представляется архитектура ПО авиационного тренажера в виде программных блоков или модулей, иерархически распределенных по уровням. Там же дано определение архитектуры ПО как структуры пакета программ моделирования самолета.

На самом верхнем уровне находятся программные блоки, предназначенные для трансформации и представления результата всей выработанной (полученной) на более низких уровнях информации в интерактивном виде. На данном уровне содержатся и подпрограммы, описывающие графический пользовательский интерфейс. Это структура, ориентированная на пользователя и содержащая информацию о подпрограммах, которые должны быть выполнены, управление заданиями. Уровень предназначен для представления, подготовки и переконфигурации всей внутренней информации в интерактивное представление для пользователя, а также для человеко-машинного взаимодействия.

Следующий, второй, уровень включает в себя начальные условия вычислений и фактический процесс моделирования. Кроме того, в начало и в конец соответственно включены две дополнительные части, где данные считываются или хранятся и где удобно осуществлять преобразования данных и перевод информации.

При моделировании полета, а также при использовании методов программной реализации в реальном времени используется цикл с отсчетом времени. Третий уровень содержит вычисления вектора состояния производной по времени и выходного вектора полной системы с приращением. На этом уровне осуществляется обмен информацией: полный результат входных сигналов, вычисленный из разных источников, передается на более низкие программные уровни.

На четвертом, программном, уровне программные модули содержат фактическую математическую модель самолета, в то время как программные модули первых трех уровней не зависят от его типа и модели. Программные модули четвертого уровня имеют доступ к БД и массивам, содержащимся на пятом уровне.

Последний, пятый, уровень содержит БД, массивы информации, необходимые для вычислений на более высоких уровнях, отдельный набор подпрограмм (программных модулей) для интерполяции данных в таблицах, конфигурационной информации, поступления новых данных.

Примером применения описанной архитектуры является реализация аэродинамических моделей тренажера самолетов Boeing 747 и Fokker F28.

Самолет можно представить как относительно малую структуру, движущуюся сквозь атмосферу под действием внешних сил и моментов. Математическая формулировка этих отношений между самолетом и внешними аэродинамическими силами и моментами обычно описана в аэродинамической модели в виде программного модуля.

Так как движение частиц когерентных систем может быть выражено обобщенно, математические модели различных типов *воздушного судна* (ВС), в первую очередь, отличаются соответствующими аэродинамическими моделями.

Первоначально данные для построения аэродинамической модели Fokker F28 были представлены в виде графов. Такое представление требует преобразований в удобную для программирования форму, в этом случае хранят отдельные наборы табличных представлений данных. Промежуточные точки могут быть найдены путем численной интерполяции [1].

Математическая модель была также линеаризована из набора ссылочных состояний с использованием процедуры численной линеаризации. Кроме того, привносятся перекрестные ссылки на оригинальное графическое представление для широкого набора исходных условий, представление линейных систем, допускается дальнейшее упрощение аэродинамической модели путем устранения незначительных отрицательных эффектов. Поскольку аэродинамические модели содержатся в отдельных программных модулях, их легко можно скопировать в целевые приложения, особенно для целевых применений с похожей модульной структурой. Обновления аэродинамической модели могут быть реализованы и путем обновления отдельно хранящихся коэффициентов или замены отдельного модуля.

Примером модульного подхода к построению ПО при разработке авиационных тренажеров [2] может служить структурная схема модуля имитатора динамики полета авиационного тренажера Ту-204, представленная на рисунке 1, где приняты следующие обозначения: ВСС-85 – вычислительная система самолетовождения, ВСУП-85 – вычислительная система управления полетом, ВСУТ-85 – вычислительная система управления тягой, СЭИ-85 – система электронной индикации, ИК – истинный курс, ПВПП – признак касания ВПП.

В данном примере тренажера имитируется движение самолета в пространстве посредством решения замкнутой системы нелинейных дифференциальных уравнений в соответствующих программных блоках, где входными параметрами являются управляющие воздействия экипажа, а выходными – вычисленные параметры полета [4].

Рис. 1. Структурная схема динамики полета с указанием взаимосвязей с другими моделями

Fig. 1. A flight dynamics structural scheme indicating interactions with other models

Для обеспечения подобия реального и моделируемого полетов самолета осуществляется непрерывное вычисление параметров полета в реальном масштабе времени. Экипажу от соответствующих модулей предоставляется визуальная, акустическая, акселерационная информация, в том числе показания приборов пилотажно-навигационного комплекса, положения и усилия на органах управления (формируемых в соответствии с параметрами, получаемыми в имитаторе динамики полета). Производится декомпозиция системы уравнений движения (продольное и боковое движение, движение по земле, модуль аэродинамических коэффициентов). Шаг интегрирования задается из условий устойчивости и мощности ЭВМ [2].

В работе [5] приведен пример схемы ПО авиационного тренажера, где программные блоки распределены по пяти уровням (рис. 2).

Рассмотрим назначение программных блоков на примере симуляции ВС Airbus серии A300-600.

Компьютер GouldSEL 32/87 – сервер, запускающий визуальные сцены для тренажера. Все программные модули в системе тренажера распределены по уровням взаимодействия и представления информации.

Опишем программные блоки первого уровня архитектуры ПО тренажера.

START. Главный программный блок загрузаемого программного модуля MAIN.LM. Используется для объявления массива и передачи данных в главную подпрограмму.

MAIN. Управляющая программа (программный модуль) для нелинейной симуляции полета в реальном времени.

INCO. Главный программный блок загрузаемого программного модуля INCO.LM. Используется для выделения памяти для обмена информацией между модулями MAIN.LM и INCO.LM, а также между INCO.LM и AERO.LM. В фазе инициализации начальные условия вычисляются посред-

ством модуля FINCO, в процессе симуляции выполняется интеграционный модуль FAIRC.

MODL. Содержит законы управления движением системы подвижности тренажера.

VIDL. Содержит законы управления, генерирующие входы (CGI-Computer Generated Imagery) для визуальной системы авиационного тренажера.

INSTR. Интерфейс инструментов ПО для запуска пилотажных приборов внутри авиационного тренажера.

Назовем программные блоки второго уровня.

FSIDS. Подпрограмма, обеспечивающая набор входных параметров для главного программного модуля MAIN и определяющая описания переменных самолета.

FINER. Подпрограмма, рассчитывающая параметры инерции в уравнениях движения. Для лучшего понимания процессов, происходящих при моделировании, в работе [5] приведены следующие математические уравнения движения самолета в связанной системе координат:

$$\begin{aligned} F_x &= -W \cdot \sin\theta + X = m \cdot (\dot{u} + qw - rv); \\ F_y &= W \cdot \cos\theta \cdot \sin\varphi + Y = m \cdot (\dot{v} + ru - pw); \\ F_z &= W \cdot \cos\theta \cdot \cos\varphi + Z = m \cdot (\dot{w} + pv - qu); \\ M_x &= L = I_x \cdot \dot{p} + (I_z - I_y)qr - J_{xz}(\dot{r} + pq); \\ M_y &= M = I_y \cdot \dot{q} + (I_x - I_z)rp - J_{xz}(p^2 - r^2); \\ M_z &= N = I_z \cdot \dot{r} + (I_y - I_x)pq - J_{xz}(\dot{p} - rq), \end{aligned}$$

где F_x, F_y, F_z – силы, действующие по координатным осям X, Y, Z ; W – вес самолета; θ – угол наклона; X, Y, Z – компоненты аэродинамической силы, приложенной по соответствующим координатным осям; φ – угол крена; $\dot{u} = u/v$ – отношение

компоненты u скорости ЛА, направленной вдоль оси X , к скорости ЛА; q – атмосферное давление; w – компонента скорости ЛА вдоль оси Z ; r – угловая скорость вдоль оси Z ; v – компонента скорости ЛА вдоль оси Z ; M_x, M_y, M_z – моменты сил, направленные вдоль координатных осей X, Y, Z ; I_x, I_y, I_z – моменты инерции, направленные вдоль координатных осей X, Y, Z ; J_{xz} – центробежный момент инерции.

FSTAT. Подпрограмма, рассчитывающая плотность воздуха на заданной высоте полета, использующая модель стандартной атмосферы.

FINCO. Подпрограмма, рассчитывающая начальные условия и использующая нелинейные уравнения движения самолета. Данный программный блок применяется прежде всего для актуальной симуляции.

INPUT Z-CARD. Интерфейс информационного обмена между Gould SEL32/87 и системами симуляции полета. Программный блок Z-CARD конвертирует 32-битные сигналы в корректный выходной сигнал.

FAIRC. Подпрограмма, рассчитывающая вектор состояния ВС (aircraft state vector) и использующая один из трех интеграционных методов: метод Адамса (конечноразностный многшаговый метод численного интегрирования дифференциальных функций), метод Гойна (Heun) и метод Рунге-Кутты.

OUTPUT Z-CARD. Подпрограмма, преобразующая входные сигналы от системы управления полетом в бортовую систему тренажера (после преобразования аналоговых сигналов в цифровые) в 32-разрядные сигналы для Gould 32/87.

Перечислим программные модули третьего, четвертого и пятого уровней.

AERO. Главная процедура данного программного модуля. Aero.LM используется для выделения области памяти для обмена информацией между подпрограммами INCO.LM и AERO.LM.

FENG4. Подпрограмма, рассчитывающая безразмерные силы и моменты, полученные от работы двигателей самолета. Номер показывает тип самолета.

LAGM4. Подпрограмма, рассчитывающая силы и моменты во время режимов авиатакси, взлета и посадки.

FDERI. В ней рассчитывается приращение времени производных самолета с использованием модели аэродинамики (FAIR4), модели двигателя (FENG4), модели шасси (LAGM4).

FCOMY. Подпрограмма, рассчитывающая воздушную скорость V , угол атаки α , угол скольжения β , угол наклона траектории γ , скорость набора (потери) высоты C и высоту H из вектора состояния $x(t)$ и выдающая выходной вектор $y(t)$.

FAIR4. Подпрограмма, вычисляющая аэродинамические силы и моменты, действующие на самолет.

FDBRD. Подпрограмма, считывающая матрицу или вектор из БД, которая содержит, например, коэффициент подъемной силы (C_l) как нелинейную функцию угла атаки (α), положения (позиции) закрылков, числа Маха (M).

FTABINT. Подпрограмма, представляющая линейную интерполяционную таблицу в виде таблиц $M \times N$ или $1 \times N$.

Таким образом, из рассмотренных примеров можно сделать вывод, что при проектировании архитектуры ПО авиационных тренажеров используется модульный метод, где каждый логический блок взаимодействует с другим посредством программного описания интерфейса обмена данными, действиями с входной и выходной информацией и может быть выражен в нескольких программных модулях. Все программные модули расположены по уровням взаимодействия в соответствии с логической структурой.

Описание архитектуры ПО самолета

Рассмотрим методы проектирования архитектуры ПО для ВС.

Традиционно техническое обслуживание ВС воспринимается на основе аппаратных средств. ПО часто рассматривается как часть аппаратного оборудования, которое при обслуживании не требует дополнительного персонала и усилий.

Исторически первоначально использование программных средств для управления и функционирования бортового оборудования самолета не требовалось.

Необходимость в использовании электроники в авиации возникла во время Второй мировой войны. Развитие бортовых *радиолокационных станций* (РЛС) с использованием магнетрона и связанных с ними технологий происходило в стремительном темпе [6].

В конце 1950–начале 1960-х гг. транзисторы вытеснили термоионные клапаны для многих применений. В военной боевой авиации улучшенная экономическая эффективность транзисторов привела к разработке в 1960–1970-х гг. цифровых систем ВС. Они использовались для систем навигации и атаки.

Развитие электронных ламп позволило создать цифровые ЭВМ, но за счет огромного количества аппаратных средств [7].

Во время Второй мировой войны британскими специалистами по криптографии была создана машина под названием Колосс (Colossus), в которой в большом количестве использовались электронные лампы с термокатодом (вакуумные трубки) для выполнения логических операций и вычислений. Машина была огромной и практически неприменимой для полетов.

Первым самолетом, разработанным с использованием цифровых технологий, был A-5 Vigilante, бомбардировщик ВМС США, введенный в эксплуатацию в 1960-е гг.

В конце 1970–начале 1980-х гг. цифровые технологии стали все шире применяться как в авиационных системах управления, так и в системах боевых действий (рис. 3).

Решающим фактором для применения было наличие экономически выгодных цифровых шин данных, таких как ARINC 429, MIL-STD-1553B и ARINC629. Эта технология в сочетании с дешевыми микропроцессорами и более совершенными инструментами разработки ПО привели к ее широкому применению на борту самолета в общепринятой практике [7].

С появлением понятия загружаемых программных модулей – технологии, применяемой на борту гражданских ВС, ПО больше не воспринимается как часть аппаратных средств, не требующих дополнительных навыков и средств для создания и управления.

С развитием авиационной индустрии число программных модулей, разработанных для систем управления самолетом и корректного функционирования полета, возрастает экспоненциально.

Такие ВС, как Boeing 737 и Airbus320, содержали примерно 30 программных блоков, которые модифицировались довольно редко. С разработкой Boeing 777 в середине 1990-х гг. их число возросло более чем на 120 модулей [8].

Производство таких самолетов, как Boeing 787, увеличило количество ПО до 500 модулей, которые, в свою очередь, загружаются в 800–900 частей аппаратных средств [8].

Рис. 3. Развитие электроники в авиации с 1950 г.

Fig. 3. Electronic engineering development in aviation since 1950

На сегодняшний день управление ПО ВС является довольно сложной задачей. При неправильно скомпилированном или загруженном в аппаратную часть ПО самолет не может совершать полеты. Поэтому в настоящее время управление ПО и его настройка (конфигурирование), скорее, не опция, а необходимость. Технология загружаемой системы включает в себя как ПО, так и аппаратную часть в виде так называемых загружаемых заменяемых блоков (LRU loadable replacement units), которые независимо друг от друга настраиваются (конфигурируются) на уровне борта самолета. Программные модули могут быть загружены в аппаратные части несколькими способами: с помощью как постоянно установленной программы-загрузчика на борту самолета, так и дополнительного оборудования вне борта.

Большинство современных самолетов, таких как Боинг 737, 747, 767, 777, на борту имеют так называемые загружаемые программные части (loadable software parts).

Авиационные бортовые системы, использующие технологию модульной загрузки ПО, позволяют техническому персоналу, обслуживающему ВС, изменять конфигурацию загружаемых систем без замены аппаратной части. Изменение функциональности системы посредством нового (обновленного) программного блока дает возможность сократить количество заменяемых аппаратных ча-

стей, повысить унификацию аппаратных частей, снизить время обслуживания.

Загружаемое ПО, предназначенное для использования в бортовой аппаратуре ВС гражданского назначения, как правило, разделяется на несколько категорий в соответствии с выполняемыми функциями: операционная система легкозаменяемых аппаратных блоков, настройки действующего ПО, БД, изменяемая информация авиалиний [9].

Операционная система заменяемого аппаратного блока ВС оперирует данными, содержащимися в конфигурационных файлах – файлах настройки для определения функции конкретного аппаратного блока. Операционная система – обычно самый большой и сложный программный модуль как по объему содержащейся в нем информации, так и по времени его загрузки в аппаратную часть.

Файл настройки операционной системы. Этот программный модуль – специализированная БД, которая определяет конфигурацию (настройки) и функцию аппаратного блока, позволяя или запрещая те или иные опциональные функции, содержащиеся в операционной системе.

Информация о настройках может передаваться по дискретным каналам связи во многие заменяемые аппаратные блоки. Файлы настройки, как правило, требуют для загрузки менее минуты.

БД представляет собой коллекцию данных, классифицированных для простоты использования

и поиска в операционной системе заменяемого аппаратного блока.

Примерами БД, используемых в аппаратных блоках с возможностью загрузки модульного ПО, могут служить БД:

- навигации ЭВМ управления полетом;
- краткого справочника взлетных скоростей ЭВМ управления полетом;
- адресно-отчетной системы авиационной связи;
- системы индикации для общего дисплея.

Навигационная БД – база, содержащая данные о навигации и маршруте полета, которые используются ЭВМ управления полетом для выполнения задач навигации. Как правило, навигационные базы обновляются каждые 28 дней и становятся доступными для загрузки за неделю до их актуализации.

Изменяемая информация авиалинии – это также небольшой файл данных, который предоставляет информацию в операционную систему заменяемых аппаратных блоков. Оператор, загружающий ПО на борт самолета, формирует (генерирует) файл данных информации авиалинии для спецификации таких функций, как формирование отчетов или предоставление сервисов для различных пассажирских зон.

Например, на некоторых ВС операционная система обращается к файлу изменяемой информации авиалиний, когда необходимо сформировать отчет, записать или отформатировать данные.

Файл изменяемой информации авиалиний – файл данных, а не программа или исполняемый файл. Тем не менее данный файл для некоторых систем содержит логические блоки, которые являются программным кодом высокого уровня. Степень изменения/модификации файла изменяемой информации авиалиний контролируется сертифицированной операционной системой заменяемых аппаратных блоков, которая не позволяет вносить оператору изменения, влияющие на безопасность полета, даже если изменения были правильными.

Технологии загрузки информации вышеописанных программных модулей в аппаратную часть борта самолета и ее обмена регламентированы соответствующими стандартами ARINC.

Наравне с методами проектирования и разработки модульной архитектуры ПО в авиационной промышленности создание ПО с применением такой технологии используется в промышленной автоматизации. Примером может служить ПО процессов управления рецептурным производством.

Компания Rockwell Automation предлагает применять термины и рекомендации производственного стандарта ISA 88.01 при разработке ПО и показывает, что для создания модульной архитектуры ПО логически всю систему следует разделить на управляющие модули, модули управления оборудованием, программные модули взаимодействия между ними [10].

Сравнение модульной архитектуры ПО самолета и симулятора

Следует отметить, что к разработке ПО ВС предъявляются более высокие требования, описанные в таких стандартах, как DO178, ARP 4754A. Это направлено на обеспечение безопасности полетов, надежности и качества программных продуктов. К ПО авиационных тренажеров данные стандарты неприменимы либо применимы частично (зависит от уровня тренажера).

Несмотря на отличие требований к разработке ПО гражданского ВС и авиационного тренажера общей является технология построения архитектуры ПО. Модульность присуща как ПО ВС, так и ПО симулятора. При разработке ПО авиационного тренажера предъявляются не такие высокие требования по отказоустойчивости и обеспечению информационного обмена, как при проектировании и разработке ПО ВС.

Обобщая подход к построению модульной архитектуры по авиационной тематике, можно сделать вывод, что для корректного проектирования архитектуры следует сформировать матрицу необходимых программных модулей, распределив их по уровням взаимодействия в зависимости от сложности разрабатываемой системы, применения и типа (типов) информационного обмена.

Логически графический пользовательский интерфейс должен находиться на самом верхнем уровне. Тем не менее, как показывают исследования построения архитектуры ПО авиационных симуляторов [1, 3, 5], управляющие программные модули связаны с графическим пользовательским интерфейсом и находятся на одном уровне. Безусловно, графический пользовательский интерфейс является отдельным программным модулем либо набором программных модулей. Взаимодействие управляющих программных модулей с программными блоками нижних уровней, как и графическое отображение операций, происходящих в системе, а также реакции системы – обратной связи на действия пользователя тесно связаны, поэтому данные программные модули находятся на одном уровне.

Архитектура ПО может быть представлена как матрица размером $M \times N$, где $M = 5$ – число программных уровней, а N – число необходимых программных модулей.

В таблице, которая является графическим представлением проектирования модульной архитектуры ПО на уровне информационного обмена, в качестве примера представлены два наиболее распространенных в авиационных приложениях вида обмена информацией – ARINC 429 и MIL-STD 1553. Следовательно, для корректного функционирования данной системы необходимо разработать два программных модуля обмена, логически соответствующих выбранным стандартам.

Графическое представление матрицы формирования архитектуры ПО

Graphical presentation of the software design architecture matrix

Уровень	Программный блок		
	1	2	N
1. Управляющие программные блоки, человеко-машинное взаимодействие	Main	Графический пользовательский интерфейс (конечное число программных блоков зависит от среды разработки и целевой платформы для реализации)	
2. Начальные условия, оперирование переменными	Блок переменных 1		
3. Информационный обмен	ARINC 429	MIL-STD 1553	
4. Математические модели самолета, реализация специфических алгоритмов для конкретной модели/типа самолета	Мат. модель 1	Мат. модель 2	Мат. модель 3
5. Массивы данных, таблицы, константы, необходимые для вычислений	Таблица 1		

Например, при разработке системного ПО уровень графического пользовательского интерфейса не применяется, но управляющие программные модули все равно присутствуют. Поэтому при проектировании такого типа ПО также необходимо оставить пять уровней, где модуль графического пользовательского интерфейса будет отсутствовать, а основные управляющие модули (например main) – нет. То есть его архитектура может быть представлена как матрица размером $(M-1) \times N = 4 \times N$, где N – число необходимых программных модулей.

Также при проектировании сложных систем общая матрица – архитектура сложной программной системы может включать составные блоки. Например, логически связанные блоки, взаимодействующие с аппаратной средой и имеющие свои алгоритмы и циклограмму выполнения задач, можно интегрировать в систему, имеющую свой графический пользовательский интерфейс, посредством внесения изменений в программные блоки либо создания новых программных модулей информационного обмена (уровень 3), переменных (уровень 2) и массивов данных (уровень 5).

Выводы

Как видно из приведенных примеров, при проектировании ПО гражданских ВС и авиационных тренажеров наиболее удобным и эффективным подходом является создание ПО авиационного назначения с применением модульной архитектуры. Главным преимуществом применения данной технологии в бортовом ПО ВС является изменение функциональности без изменения аппаратной части, что позволяет сократить количество заменяемых аппаратных частей, которые должны хранить как компании, обслуживающие ВС, так и компании-производители ВС.

Можно также выделить следующие преимущества при разработке ПО с использованием модульной архитектуры построения для разработчиков ПО.

- Снижение времени на конструирование (проектирование) архитектуры новых программных продуктов. Используя при создании ПО данную технологию и при необходимости производственный стандарт ISA 88.01, разработчики могут сконцентрироваться на функциональности управляющих программных модулей, специфических для конкретной машины, а не на технике реализации, тем самым сокращая время разработки. Данный подход к разработке также может способствовать синхронной разработке программных блоков группой разработчиков, что, в свою очередь, дополнительно ускорит выполнение каждого цикла разработки и, как результат, сократит общее время на разработку программной системы.

- Снижение экономических затрат на проектирование (конструирование), разработку, поставку (в том числе развертывание) программных продуктов. В дополнение к сокращению времени на разработку использование технологии модульной архитектуры и разработки ПО также помогает снизить затраты в течение всего его жизненного цикла, включающего конструирование (разработку) архитектуры, программирование, тестирование, введение в эксплуатацию, сопровождение.

- Возможность вносить больше инноваций в машиностроение, увеличивать производительность создаваемой техники.

Преимуществами использования разработчиками ПО технологии модульной архитектуры ПО для производителей аппаратной части оборудования являются следующие:

- гибкость процесса производства: возможность адаптировать существующие ресурсы к нуж-

дам и требованиям нового продукта с минимальными экономическими затратами и затратами времени;

– улучшение существующих ресурсов и повышение производительности: возможность упростить непрерывное усовершенствование путем вертикальной интеграции существующих ресурсов оборудования;

– безопасность и соответствие нормативным правилам: модульное программирование упрощает соблюдение нормативных и сертификационных требований, а также проверку за счет сокращения объема усилий по проверке только модифицированных модулей для аппаратной части, а не целых программных приложений [10].

Данный подход позволяет снизить трудозатраты на обслуживание уже разработанного ПО посредством замены или доработки программного модуля, не изменяя при этом всю структуру системы, повысить эффективность и качество разрабатываемого ПО, так как при модульном подходе к разработке снижается вероятность появления ошибок, которые сложно отследить в программном коде. Метод проектирования ПО с помощью модульной архитектуры стимулирует и облегчает непрерывное совершенствование программных продуктов, снижая риск отрицательного эффекта на всю программную систему в целом.

Преимуществами применения модульной архитектуры также является возможность применения при наукоемких вычислениях и решении задач параллельных вычислений.

Таким образом, при проектировании сложных систем необходимо учитывать и проектировать интерфейсы информационного взаимодействия, вносить необходимые дополнения с учетом имею-

щихся программных модулей для корректного функционирования всей системы в целом и располагать как программные модули взаимодействия, так и проектируемые программные модули в соответствии с предложенной матрицей.

Литература

1. Hoogstraten J.A., van de Moesdijk G.A.J. Modular programming structure applied to the simulation of non-linear aircraft models. URL: <http://repository.tudelft.nl/islandora/object/uuid%3A0a0508b9-7e2a-4f61-b3ce-006644642937> (дата обращения: 20.09.2016).
2. Данилов А.М., Лапшин Э.В., Гушина А.А. Информационно-вычислительные системы авиационных тренажеров модульной архитектуры с распараллеливанием вычислительных процессов // Надежность и качество: тр. Междунар. симпози. Пенза: Изд-во Пензенск. гос. ун-та, 2006. Т. 1. С. 39–44.
3. Baarspul M. Flight Simulation Techniques with emphasis on the generation of high fidelity 6 DOF motion cues. URL: <http://repository.tudelft.nl/islandora/object/uuid%3Adfd19219-afd9-4faf-b2c8-03401bfc89c0?collection=research> (дата обращения: 20.09.2016).
4. Данилов А.М., Лапшин Э.В. Теория и практика имитационного моделирования и создания тренажеров // Приборы и системы управления. 1989. № 8. С. 11–12.
5. Baarspul M. Lecture Notes on Flight Simulation Techniques Report LR-596. 1989. URL: <http://resolver.tudelft.nl/uuid:269fa44e-0b09-4fc5-9260-83b6afb7cdd> (дата обращения: 20.09.2016).
6. Lovell B. Echoes of war: the story of H2S radar. Adam Hilger Ltd., Bristol, 1991, 287 p.
7. Moir I., Seabridge A. Design and development of Aircraft Systems. UK, Wiley, 2012, 334 p.
8. Best practices for loadable software management and configuration control. Documents, IATA – Eng. and Maintenance Group, Montreal, Canada, 2013, 11 p.
9. Klein J. Onboard Loadable Software Boeing. URL: http://www.boeing.com/commercial/aeromagazine/aero_05/ps/ps02/index.html (дата обращения: 20.09.2016).
10. Integrated Architecture™: foundations of modular programming. Ref. Document, Rockwell Automation, Inc. Publ., 2010, 86 p.

Software & Systems

DOI: 10.15827/0236-235X.118.291-300

Received 10.11.16

2017, vol. 30, no. 2, pp. 291–300

DESIGN PRINCIPLES OF MODULAR SOFTWARE ARCHITECTURE IN AVIATION

L.A. Chizhikova¹, *Leading Engineer, l_chizhikova@scac.ru*

¹ JSC “Sukhoi Civil Aircraft”, Polikarpova St. 23b, bldg. 2, Moscow, 125284, Russian Federation

Abstract. Software development is a quite difficult and laborious process, which is based on correct and reliable architecture design. Distribution and coordination of duties when working in a group of developers is often a quite difficult and responsible decision due to affection the main result of development.

Nowadays, with technological development the control and functionality of aircraft avionics requires more software creation. The role of software development and management is also increasing in the flight simulation and testing equipment fields. More and more tasks are transferred from hardware to software.

The article provides the analysis and identification of basic aviation software design aspects, the comparison between the civil aircraft software design architecture and flight simulation software architecture.

In order to present a unified software architecture model, the authors made a research on papers on software architecture design in Russian companies, which make Flight Simulation devices and onboard aircraft software architecture. There is also a comparison of these solutions and their commonality. Furthermore, the article considers a model, which is successfully used

in software development by Rockwell Automation Company, and makes a research among the works of Delft Technical University on the described topic.

The paper proposes a mathematical description of modular software product architecture, which is oriented to the aviation industry.

The proposed approach allows unifying a development process, reducing the time and labor of software development, adding innovations without existing structure transformation if the software product is made using the described solution.

Keywords: software, software architecture, software architecture design, software development, flight simulation, aviation software, mathematical modeling.

References

1. Hoogstraten J.A., van de Moesdijk G.A.J. Modular programming structure applied to the simulation of non-linear aircraft models. *IMACS Symp. Simulation in Engineering Sciences*. Delft Univ. of Technology, Nantes, 1983, 11 p. Available at: <http://repository.tudelft.nl/islandora/object/uuid%3A0a0508b9-7e2a-4f61-b3ce-006644642937> (accessed September 20, 2016).
2. Danilov A.M., Lapshin E.V., Gushchina A.A. Information and computational systems of modular architecture flight simulation devices with parallel computational processes. *Trudy mezhdunar. simp. "Nadezhnost i kachestvo"* [Proc. Int. Symp. "Quality and Reliability"]. Vol. 1, Penzensky Gos. Univ. Publ., Penza, 2006, pp. 39–44 (in Russ.).
3. Baarspul M. Flight Simulation Techniques with emphasis on the generation of high fidelity 6 DOF motion cues. *Proc. 15th Congress of the Int. Council of the Aeronautical Sciences*. Delft Univ. of Technology Publ., London, 1986, 17 p. Available at: <http://repository.tudelft.nl/islandora/object/uuid%3Adfd19219-afd9-4faf-b2c8-03401bfc89c0?collection=research> (accessed September 20, 2016).
4. Danilov A.M., Lapshin E.V. The theory and practice of simulation modeling and flight simulation creation. *Pribory i sistemy upravleniya* [Instruments and Control Systems]. 1989, no. 8, pp. 11–12 (in Russ.).
5. Baarspul M. *Lecture Notes on Flight Simulation Techniques Report LR-596*. Delft Univ. of Technology Publ., 1989. Available at: <http://resolver.tudelft.nl/uuid:269fa44e-0b09-4fc5-9260-83b6affb7cdd> (accessed September 20, 2016).
6. Lovell B. *Echoes of War: The Story of H2S Radar*. Adam Hilger (Bristol) Publ., 1991, 287 p.
7. Moir I., Seabridge A. *Design and development of Aircraft Systems*. UK, Wiley Publ., 2012, 334 p.
8. *Best Practices for Loadable Software Management and Configuration Control*. IATA Int. Air Transport Association Engineering and Maintenance Group, Montreal, 2013, 11 p.
9. Klein J. Onboard Loadable Software Boeing. *AERO magazine*. 1998, no. 5. Available at: http://www.boeing.com/commercial/aeromagazine/aero_05/ps/ps02/index.html (accessed September 20, 2016).
10. Bradley A. *Integrated Architecture Foundations of Modular Programming*. Rockwell Automation. Rockwell Software Publ., 2010, 86 p.

Примеры библиографического описания статьи

1. Чижикова Л.А. Принципы проектирования модульной архитектуры программного обеспечения авиационной тематики // Программные продукты и системы. 2017. Т. 30. № 2. С. 291–300; DOI: 10.15827/0236-235X.118.291-300.
2. Chizhikova L.A. Design principles of modular software architecture in aviation *Programmnye produkty i sistemy* [Software & Systems]. 2017, vol. 30, no. 2, pp. 291–300 (in Russ.); DOI: 10.15827/0236-235X.118.291-300.