УДК 681.323

Е.В. Зотов, И.А. Печаткин

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ БОРТОВОЙ АППАРАТУРЫ

Зотов Евгений Валерьевич, *окончил Ульяновский государственный университет, начальник тематической комплексной бригады АО «Ульяновское конструкторское бюро приборостроения». Имеет работы в области авиационной тематики. [e-mail: zzz80@inbox.ru].*

Печаткин Илья Александрович, *окончил УлГУ, начальник НИО АО «УКБП». Имеет работы в области авиационной тематики. [e-mail: ilyapech@yandex.ru].*

Аннотация

В статье описаны процессы жизненного цикла программного обеспечения (ПО) бортовой аппаратуры и систем. Подробно рассмотрены процессы проектирования и кодирования ПО. Описан подход к проектированию и кодированию, который при появлении новых требований позволяет расширять функционал ПО, без кардинального изменения архитектуры ПО, в результате чего, зачастую, возникает большое количество ошибок и это приводит к длительному процессу повторной проверки и отладки ПО. Показано, как применение предложенного подхода снижает трудозатраты процессов разработки требований к ПО, проектирования, создания исходного кода и трассируемости. Сделаны выводы о преимуществах использования предложенного подхода в рамках разработки ПО бортовой аппаратуры и систем авиационной техники.

Ключевые слова: проектирование, кодирование, трассируемость.

AVIONICS SOFTWARE DESIGN

Evgenii Valerevich Zotov, *graduated from Ulyanovsk State Technical University; Head of Thematic Integrated Team at Ulyanovsk Instrument Manufacturing Design Bureau, JSC; an author of articles in the field of aviation. e-mail: zzz80@inbox.ru.*

Ilia Aleksandrovich Pechatkin, *graduated from Ulyanovsk State University; Head of Research Unit of Ulyanovsk Instrument Manufacturing Design Bureau, JSC; an author of articles in the field of aviation. e-mail: ilyapech@yandex.ru.*

Abstract

The article describes avionics software life cycle processes. Software design and encoding processes are examined. An approach to design and encoding is described, which allows to improve the software functionality without fundamental changes of software architecture causing often raise a lot of errors resulting in a lengthy process of software retesting and debugging when new requirements emerge. It is shown that the application of the approach proposed reduces labor costs related to the processes of developing the software requirements, designing and creating source code as well as traceability. Based on the findings, conclusions about advantages of the application of the proposed approach within the framework of developing software for airborne equipment and systems are drawn up.

Key words: design, encoding, traceability.

Введение

Быстрый рост использования программного обеспечения (ПО) в бортовых системах и оборудовании, применяемых на летательных аппаратах и двигателях, в начале 1980-х годов привел к необходимости разработки принятых в отрасли инструктивных материалов для удовлетворения требованиям норм летной годности. По мере роста использования ПО, развития технологий и накопления опыта использования руководящие материалы по разработке ПО пересматриваются и дорабатываются.

В настоящее время разработаны такие документы, как RTCA DO-178C «Software Considerations in Airborne Systems and Equipment Certification» [1] (DO-178C) и его отечественный аналог – КТ-178С «Квалификационные требования часть 178С [2]. Требования к программному обеспечению бортовой аппаратуры и систем при сертификации авиационной техники» (КТ-178С) предоставляют авиационному сообществу инструктивные материалы для определения единообразно и с приемлемым уровнем доверия того, что ПО бортовых систем и оборудования удовлетворяет требованиям летной годности. Документ КТ-178С является одним из основных доку-

ментов при разработке ПО бортовых систем и оборудования производства АО «УКБП», а также многих сторонних организаций, занимающихся подобным родом деятельности, например АО «РПКБ». На данный момент большое количество материалов посвящено тематике проектирования ПО [3] и управления программными продуктами [4]. В представленной работе рассмотрен подход к проектированию ПО в узкоспециализированной области разработки ПО бортовой аппаратуры.

Статья организована следующим образом:

В разделе 1 рассмотрены общие моменты процессов жизненного цикла (ЖЦ) ПО для получения представления о процессах разработки ПО для бортовых систем и оборудования. Описание подхода к проектированию, который позволяет сократить время разработки требований к ПО и создания исходного кода, изложено в разделе 2.

1 Процессы ЖЦ ПО

Процессы ЖЦ ПО разделяются на три основных составляющих:

1. Процесс планирования;
2. Процессы разработки;
3. Интегральные процессы.

Процесс планирования ПО – процесс, определяющий мероприятия процессов разработки ПО и интегральных процессов в рамках проекта. В процессе планирования разрабатываются такие документы, как план разработки, план сертификации, план верификации, стандарты на кодирование, стандарты на проектирование, стандарты на разработку требований к ПО и т. д. Планы и стандарты составляются таким образом, чтобы они служили руководством для разработчика, или группы разработчиков ПО, и обеспечивали координацию процессов разработки ПО и интегральных процессов. Процесс планирования является одним из важнейших этапов разработки ПО, но в данной статье мы не будем подробно на нем останавливаться, так как целью исследования является разработка оптимального, с точки зрения проектирования, кодирования, верификации и трассируемости подхода к разработке ПО.

Процессы разработки – совокупность процессов, в результате выполнения которых получается программный продукт. Эти процессы включают в себя процесс разработки требований к ПО, процесс проектирования ПО, процесс кодирования ПО, процесс интеграции. На этом этапе формируются требования нижнего и верхнего уровней – требования к функционированию, интерфейсам взаимодействия ПО и аппаратуры, техническим характеристикам, а также к безопасности. Разрабатывается архитектура ПО, на основе которой создается исходный код. Конечным результатом процесса разработки является интеграция ПО и аппаратуры, которая подразумевает под собой компиляцию и линковку созданного исходного кода для получения исполняемого объектного кода и загрузку его в целевой вычислитель. В следующем разделе мы более подробно рассмотрим процесс разработки архитектуры ПО и создание исходного кода на примере разработки ПО для многофункционального индикатора ИМ-16М.

Интегральные процессы – совокупность процессов, выполнение которых гарантируют корректность, управляемость и доверие к процессам ЖЦ ПО и их выходным данным. Эти процессы включают в себя процесс верификации ПО, процесс управления конфигурацией ПО, процесс гарантии качества ПО, процесс взаимодействия заявителя с сертифицирующим органом. Важно понимать, что эти интегральные процессы выполняются одновременно с процессами планирования и разработки ПО в течение всего ЖЦ ПО [2]. Верификация – проверка, подтверждение, способ подтверждения с помощью доказательств каких-либо теоретических положений, алгоритмов, программ и процедур путём их сопоставления с опытными (эталонными или эмпирическими) данными, алгоритмами и программами [5]. В контексте разработки ПО верификация – это техническая оценка выходных данных процесса планирования ПО и процессов разработки ПО, то есть рассмотрение и анализ требований, архитектуры ПО, исходного кода, а также результатов процесса интеграции. Другими словами, целью процесса верификации является подтверждение соответствия исходного кода, архитектуры ПО и требований. Под управлением конфигурацией подразумевают проведение мероприятий по идентификации конфигурации, управлению изменениями, установлению базовых версий данных ЖЦ ПО и хранению программного продукта вместе с относящимися к нему данными ЖЦ ПО. Цель процесса гарантии качества – обеспечить уверенность в том, что процессы ЖЦ ПО создают ПО, удовлетворяющее предъявляемым к нему требованиям.

2 Проектирование ПО

В качестве примера разработки архитектуры ПО и создания исходного кода рассмотрим ПО индикатора многофункционального ИМ-16М.

Индикатор многофункциональный ИМ-16М (далее индикатор) предназначен для отображения информации о параметрах и состоянии силовой установки, вертолетных систем и бортового оборудования (далее ОВО), формирования и отображения навигационно-плановой информации, метеоинформации, сигнальной информации об опасном сближении с земной поверхностью (далее НВГ) и пилотажной информации (далее ПЛТ). Внешний вид индикатора и отображаемого кадра представлен на рисунке 1. Кадр представляет собой комбинацию полуформатов. В верхней части индикатора отображена информация о состоянии силовой установки в виде мнемосхем и сигнальной информации в виде текстовых сообщений (ДВИГ). В нижней части индикатора отображена информации о состоянии системы электроснабжения (СЭС).

На основе требований к ПО верхнего уровня разработана архитектура ПО, в процессе разработки которой учтена возможность расширения функционала ПО при

Рис. 1. Формат ОВО. ДВИГ и СЭС

Таблица 1

Матрица переходов для режима ОВО

Ki\Ci	C1	C2	C3	C4	C5
K1	FO2	FONull	FO4	FO_K9	FO_K10
K2	FONull	FO1	FO3	FO_K9	FO_K10
K3	FONull	FO4	FO0	FO_K1	FO_K2
K4	FO3	FONull	FO0	FO_K1	FO_K2
K5	FONull	FO6	FO3	FO_K1	FO_K2
K6	FO5	FONull	FO4	FO_K1	FO_K2
K7	FONull	FO8	FO5	FO_K4	FO_K5
K8	FO7	FONull	FO6	FO_K4	FO_K5
K9	FONull	FO10	FO3	FO_K9	FO_K10
K10	FO9	FONull	FO4	FONull	FONull

изменении требований, а также возможность использования данного описания для проектирования ПО подобных систем и блоков. Иерархия образующих архитектуру ПО компонентов изображена на рисунке 2.

ПО индикатора состоит из компонентов верхнего уровня: «управление», «сбор», «массивы рабочих данных», «контроль» и «графическая информация», а также их потомков – компонентов второго и далее уровней, которые получают управление (процессорное время) в зависимости от состояния управляющих сигналов (событий). Компонент «управление» отвечает за начальную инициализацию, тактирование задачи, работу с охранным таймером и вызов интерфейсных функций остальных компонентов верхнего уровня:

1. Сбор – компонент отвечает за сбор данных от сопрягаемых систем и органов управления индикатора (кнопок и кремальер);
2. Контроль – компонент отвечает за контроль работоспособности индикатора во время выполнения функциональной задачи;
3. Массивы рабочих данных – компонент отвечает за обработку входных данных и формирование из них массивов данных для отображения и их нормализацию;
4. Графическая информация – компонент, отвечающий непосредственно за отображение информации на экране индикатора;
5. Графическая библиотека – компонент, содержащий функции работы с графикой.

При разработке требований к ПО индикатора и проектировании применен подход, заключающийся в описании реакций на события в виде таблицы. В столбцах данной таблицы расположены события, при возникновении которых требуется изменить режим, перейти к отображению следующего формата или произвести какое-либо действие, а в строках – формальное описание реакций на события. Разработанную таблицу назовем матрицей переходов. Тут необходимо сделать уточнение – полученная матрица переходов не имеет отношения к понятию «матрица перехода» из курса линейной алгебры. Пример матрицы переходов для режима ОВО ПО индикатора представлен в таблице 1.

Описание событий C1–C5 и реакций K1–K10 приведено в таблице 2. Описание функций-реакций на события FO1, FO_K1, FONull и др. приведено в таблице 3.

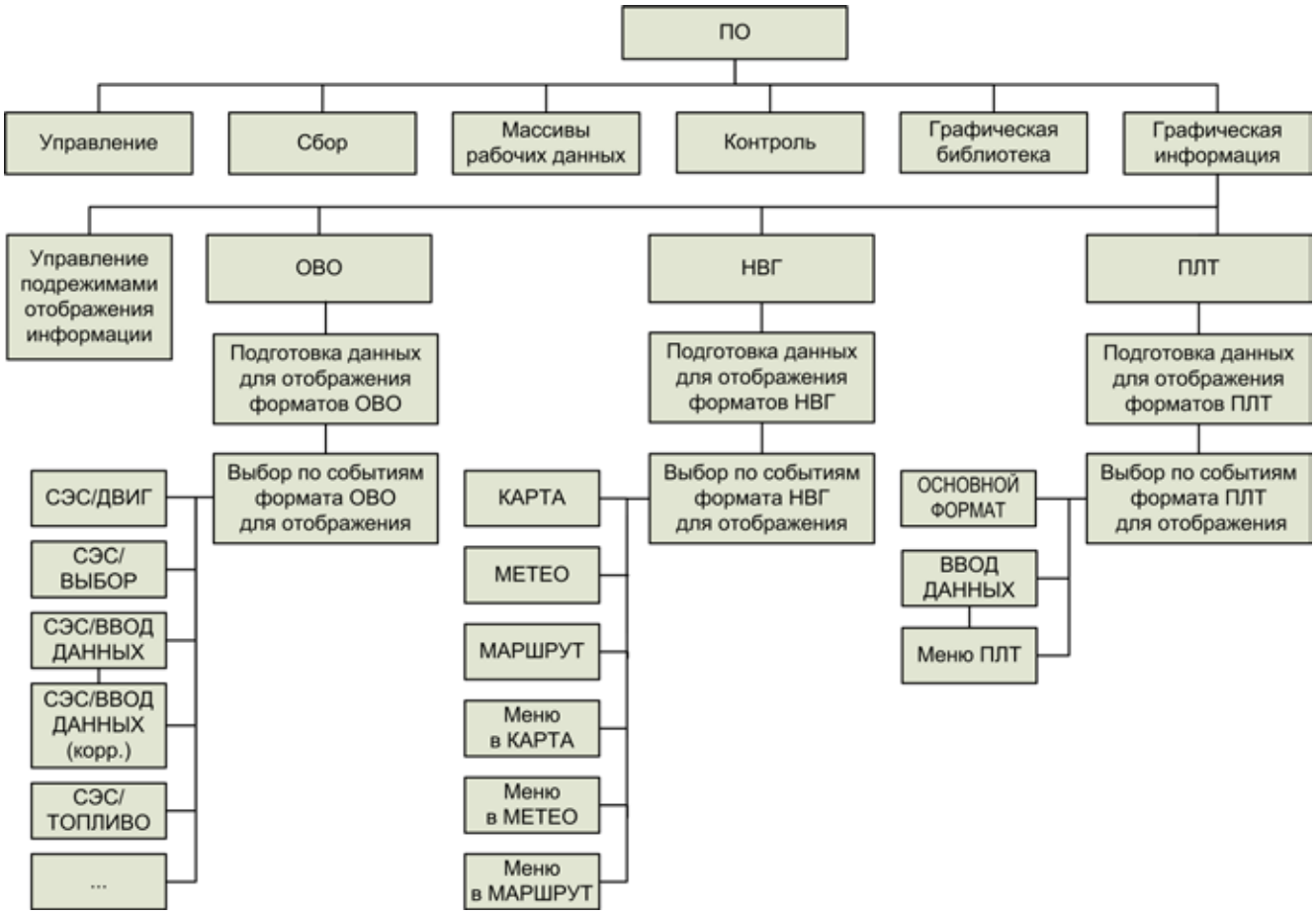

Рис. 2. Архитектура ПО индикатора

Таблица 2

Описание событий и реакций в режиме ОВО

№	Описание
С1	событие нажатия нижней кнопки № 1 ИМ-16М
С2	событие нажатия нижней кнопки № 2 ИМ-16М
С3	событие нажатия кнопки СБР ИМ-16М
С4	событие вращения кремальеры ИМ-16М влево
С5	событие вращения кремальеры ИМ-16М вправо
К1	отображение комбинированного формата СЭС и ДВИГ
К2	отображение комбинированного формата ДВИГ и СЭС
К3	отображение комбинированного формата СЭС и ВЫБОР
К4	отображение комбинированного формата ДВИГ и ВЫБОР
К5	отображение комбинированного формата СЭС и ВВОД ДАННЫХ
К6	отображение комбинированного формата ДВИГ и ВВОД ДАННЫХ
К7	отображение комбинированного формата СЭС и ВВОД ДАННЫХ с возможностью корректировки выбранного параметра
К8	отображение комбинированного формата ДВИГ и ВВОД ДАННЫХ с возможностью корректировки выбранного параметра
К9	отображение комбинированного формата СЭС и ТОПЛИВО
К10	отображение комбинированного формата ДВИГ и ТОПЛИВО

Таблица 3

Функции реакции на события

№	Описание
Функции переходов	
FO0	Переход в формат по умолчанию
FO1	Переход в формат К1, если нет запрета, иначе нет реакции на событие
…	…
FO10	Переход в формат К10, если нет запрета, иначе нет реакции на событие
Функции коррекции	
FO_K1	Перемещение по меню вверх
FO_K2	Перемещение по меню вниз
FO_K3	Подтверждение выбранной позиции меню
FO_K4	Изменение (уменьшение) выбранной позиции меню
FO_K5	Изменение (увеличение) выбранной позиции меню
FO_K6	Сброс выбранной позиции меню
FO_K7	Перемежение окна отображения зоны сигнальных сообщений в полуформатах «СИГНАЛЬНЫЕ СООБЩЕНИЯ» и «ОБОРУДОВАНИЕ» вниз
FO_K8	Перемежение окна отображения зоны сигнальных сообщений в полуформатах «СИГНАЛЬНЫЕ СООБЩЕНИЯ» и «ОБОРУДОВАНИЕ» вверх
FO_K9	Перемежение окна отображения зоны сигнальных сообщений в полуформате «ДВИГ» вниз
FO_K10	Перемежение окна отображения зоны сигнальных сообщений в полуформате «ДВИГ» вверх
FONull	Пустая функция (нет реакции на событие)

Формализовав таким образом требования, достаточно просто создать исходный код для управления событиям в режиме ОВО путем написания простейшего макроса, который преобразует таблицу матрицы переходов в исходный код на языке С [6], соответствующий стандартам [7] и правилам [8]:

```
// С1 – событие нажатия нижней кнопки №1 ИМ-16М;
// С2 – событие нажатия нижней кнопки №2 ИМ-16М;
// С3 – событие нажатия кнопки СБР ИМ-16М;
// С4 – событие вращения кремальеры ИМ-16М влево;
// С5 – событие вращения кремальеры ИМ-16М вправо;
// К1 – отображение комбинированного формата СЭС и ДВИГ;
// К2 – отображение комбинированного формата ДВИГ и СЭС;
// К3 – отображение комбинированного формата СЭС и ВЫБОР;
// К4 – отображение комбинированного формата ДВИГ и ВЫБОР;
// К5 – отображение комбинированного формата СЭС и ВВОД ДАННЫХ;
// К6 – отображение комбинированного формата ДВИГ и ВВОД ДАННЫХ;
// К7 – отображение комбинированного формата СЭС и ВВОД ДАННЫХ с возможностью корректировки выбранного параметра;
// К8 – отображение комбинированного формата ДВИГ и ВВОД ДАННЫХ с возможностью корректировки выбранного параметра;
// К9 – отображение комбинированного формата СЭС и ТОПЛИВО;
// К10 – отображение комбинированного формата ДВИГ и ТОПЛИВО;
// MANAGER_OVO – структура с управляющими данными
// Матрица переходов для режима ОВО;
void (*F_MOVE_OVO[10][5])(MANAGER_OVO *) =
{
/*      | C1    | C2    | C3    | C4    | C5    |*/
/*K1 */{&FO2    ,&FONull,&FO4    ,&FO_K9  ,&FO_K10},
/*K2 */{&FONull,&FO1    ,&FO3    ,&FO_K9  ,&FO_K10},
/*K3 */{&FONull,&FO4    ,&FO0    ,&FO_K1  ,&FO_K2 },
/*K4 */{&FO3    ,&FONull,&FO0    ,&FO_K1  ,&FO_K2 },
/*K5 */{&FONull,&FO6    ,&FO3    ,&FO_K1  ,&FO_K2 },
/*K6 */{&FO5    ,&FONull,&FO4    ,&FO_K1  ,&FO_K2 },
/*K7 */{&FONull,&FO8    ,&FO5    ,&FO_K4  ,&FO_K5 },
/*K8 */{&FO7    ,&FONull,&FO6    ,&FO_K4  ,&FO_K5 },
/*K9 */{&FONull,&FO10   ,&FO3    ,&FO_K9  ,&FO_K10},
/*K10*/{&FO9    ,&FONull,&FO4    ,&FONull,&FONull},
};
```

Следующим шагом к созданию исходного кода станет программная реализация функции обработки событий и функций-реакций на события.

Реализация функции обработки событий:

```
void F_EVENTS_OVO(MANAGER_OVO *p) //функция обработки событий
{
    if (BTN_BOTTOM_1_DOWN) // событие нажатия нижней кнопки № 1 ИМ-16М;
        {p->INDEX_TABL_OVO_MOVE_X = 0;return;}
    if (BTN_BOTTOM_2_DOWN)// событие нажатия нижней кнопки № 2 ИМ-16М;
        {p->INDEX_TABL_OVO_MOVE_X = 1;return;}
    if (BTN_SBR_DOWN)// событие нажатия кнопки СБР ИМ-16М;
        {p->INDEX_TABL_OVO_MOVE_X = 2;return;}
    if (RACK_DIRECT>0)// событие вращения кремальеры ИМ-16М влево;
        {p->INDEX_TABL_OVO_MOVE_X = 3;return;}
    if (RACK_DIRECT<0)// событие вращения кремальеры ИМ-16М вправо;
        {p->INDEX_TABL_OVO_MOVE_X = 4;return;}
    p->INDEX_TABL_OVO_MOVE_X = -1;// нет события
}
```

Реализация функции-реакции – функции перехода в формат К1 (отображение комбинированного формата СЭС и ДВИГ):

```
void FO1(MANAGER_OVO *p) // переход в формат К1, если нет запрета, иначе Null
{
    p->INDEX_TABL_OVO_MOVE_Y = 0; // номер формата в матрице переходов
    p->index_up = SES_NUMBER_up; // номер верхнего полукадра
    p->index_down = DVIG_NUMBER_dw; // номер нижнего полукадра
    p->index_extern = 1; //не расширенный формат
    p->ss_val_dv = 2; //отображаем и управляем зоной сигнальных сообщений в ДВИГ
}
```

Третьим шагом на пути к созданию исходного кода будет вызов функции обработки событий и функции матрицы переходов в основной части кода:

```
void main()
{
    //часть кода отсутствует…

    //обрабатываем события
    F_EVENTS_OVO(&p_ovo);
    //совершаем переходы
    F_MOVE_OVO[p_ovo.INDEX_TABL_OVO_MOVE_Y][p_ovo.INDEX_TABL_OVO_MOVE_X](&p_ovo);

    // часть кода отсутствует…
}
```

Разумеется, для того чтобы ПО соответствовало требованиям, необходимо реализовать компоненты верхнего уровня, определить содержание управляющей структуры, компоненты отрисовки форматов и т. д., но не будем на этом заострять внимание. Как уже говорилось выше, результатом данной работы является разработка максимально оптимизированного подхода к разработке ПО с точки зрения сокращения трудозатрат на выполнение работ по проектированию, кодированию и верификации ПО.

Заключение

В рамках исследования разработан подход к разработке требований и проектированию ПО бортовой аппаратуры и систем, который имеет ряд преимуществ по сравнению с классическим подходом:

1. Оперативность – минимальные временные затраты на создание исходного кода, соответствующего требованиям (генерация исходного кода происходит с помощью макросов автоматически, напрямую из таблиц с требованиями);
2. Применяемость – создание исходного кода для сколь угодно сложных систем, требования для которых формализованы в виде матрицы (или нескольких матриц) переходов занимает на порядок меньше времени, нежели создавать код по каждому пункту требований вручную;
3. Трассируемость – при использовании предложенного подхода трудоемкость подтверждения соответствия исходного кода требованиям существенно снижается (требования автоматически перерабатываются в исходный код);
4. Универсальность – любое изменение требований не приводит к кардинальной переработке проекта ПО и, как следствие, длительным процедурам верификации (эти изменения описываются путем модификации матрицы переходов);
5. Минимализация ошибок – при разработке требований все комбинации событий и реакций описаны таблицей и не учесть какую-либо комбинацию невозможно.

СПИСОК ЛИТЕРАТУРЫ

1. RTCA DO-178C. Software Considerations in Airborne Systems and Equipment Certification. – URL: https://www.rtca.org/content/standards-guidance-materials (дата обращения: 22.01.2018).
2. КТ-178С. Квалификационные требования часть 178С. Требования к программному обеспечению бортовой аппаратуры и систем при сертификации авиационной техники. – URL: http://kaf401.rloc.ru/TRPO/КТ-178С. pdf (дата обращения 22.01.2018).
3. Кнут Д.Э. Искусство программирования. Т. 1. Основные алгоритмы. – М. : Издательство «Вильямс», 2006. – 682 с.
4. Наместников А.М., Гуськов Г.Ю. Система управления программными проектами на основе онтологического подхода // Автоматизация процессов управления. – 2016. – № 3 (45). – С. 88–94.

5. ГОСТ Р ИСО 9000. Система менеджмента качества. Основные положения и словарь – М. : ИПК Издательство стандартов, 2015. – 48 с.

6. Керниган Б.В., Ричи Д.М. Язык С. – М. : Издательство «Вильямс», 2004. – 229 с.

7. ISO/IEC 9899:1999. Programming languages – C. – URL: http://www.dii.uchile.cl/~daespino/files/Iso_C_1999_definition.pdf (дата обращения: 22.01.2018).

8. MISRA-C: 2004. Guidelines for the use of the C language in critical systems. – URL: https://www.misra.org.uk/Activities/MISRAC/tabid/160/Default.aspx (дата обращения: 22.01.2018).

REFERENCES

1. *RTCA DO-178C. Software Considerations in Airborne Systems and Equipment Certification,* 2013. 102 p.

2. *KT-178S. Kvalifikatsionnye trebovaniia chast 178S. Trebovaniia k programmnomu obespecheniiu bortovoi apparatury i sistem pri sertifikatsii aviatsionnoi tekhniki* [Qualification Requirements. Part 178S. Avionics Software Requirements in Airborne Systems and Equipment Certification], 2016. 97 p.

3. Knuth D.E. *Iskusstvo programmirovaniia. T. 1. Osnovnye algoritmy* [The Art of Computer Programming. Vol. 1. Fundamental Algorithms]. Moscow, Williams Publ., 2006. 682 p.

4. Namestnikov A.M., Guskov G.Iu. Sistema upravlenia programmnymi proektami na osnove ontologicheskogo podkhoda [The Software Project Management System Based on the Ontology Approach] *Avtomatizatsiia protsessov upravleniia* [Automation of Control Processes], 2016, no. 3 (45), pp. 88–94.

5. *GOST R ISO 9000. Sistema menedzhmenta kachestva. Osnovnye polozheniia i slovar* [State Standard. GOST R ISO 9000 Quality Management Systems. Fundamentals and Vocabulary]. Moscow, Izdatelstvo standartov Publ., 2015. 48 p.

6. Kernighan B.V., Richi D.M. *Iazyk C* [C Language]. Moscow, Williams Publ., 2004. 229 p.

7. *ISO/IEC 9899:1999. Programming languages – C*. Available at: http://www.dii.uchile.cl/~daespino/files/Iso_C_1999_definition.pdf (accessed: 22.01.2018).

8. *MISRA-C:2004 Guidelines for the Use of the C Language in Critical Systems.* Available at: https://www.misra.org.uk/Activities/MISRAC/tabid/160/Default.aspx (accessed 22.01.2018).