УДК 004.43
DOI: 10.15827/0236-235X.126.283-289
Дата подачи статьи: 23.10.18
2019. Т. 32. № 2. С. 283–289

Информационная технология верификации специального программного обеспечения автоматизированных систем военного назначения

Р.Е. Жидков [1], *адъюнкт, r-zhidkov@rambler.ru*
Д.С. Викторов [1], *д.т.н., доцент, victorov.dmitry@yandex.ru*
Е.Н. Жидков [1], *к.т.н., eng1964@mail.ru*

[1] *Военная академия воздушно-космической обороны им. Маршала Советского Союза Г.К. Жукова, г. Тверь, 170022, Россия*

Потребность в создании информационной технологии верификации специального ПО автоматизированных систем военного назначения, позволяющей с большей эффективностью проверять требования к функциональности, обусловлена необходимостью создания качественного ПО в условиях временных и финансовых ограничений процесса разработки.

В данной статье предлагается технология, выполненная как совокупность приемов, способов и программных утилит для обработки исходного кода исследуемой программы и программной документации с целью получения на их основе информации о функциональных дефектах. В рамках процесса создания специального ПО описываемая технология может использоваться как на этапе кодирования, так и на этапах тестирования разного уровня в дополнение к существующим решениям для поиска дефектов функциональности.

Для верификации используется метод, структура которого определяется требованиями нормативных документов в области разработки ПО. Он включает методику статического анализа программ и методику расчета полноты анализа на основе характеристик программного кода. Данный метод верификации предназначен для проверки корректности реализации информационно-расчетных задач в автоматизированных системах управления войсками и боевыми средствами по принципу размерной однородности физических уравнений.

Информационная технология верификации описана в виде функциональной модели IDEF0, детализированной до второго уровня вложенности с описанием входов и выходов всех работ. Чтобы поддержать процесс верификации, предлагаются программные утилиты для автоматизации рутинных операций: формирования уникальных идентификаторов программных объектов и поиска дефектов. Данные программы созданы на основе инфраструктуры для разработки компиляторов LLVM и фронтенда Clang, позволяющих исследовать код программ на языках C, C++, Objective-C.

Ключевые слова: *информационная технология, верификация, ПО, IDEF0, функциональные дефекты, статический анализ.*

Создание специального ПО для *автоматизированных систем военного назначения* (АСВН), отвечающего современным требованиям к функциональности, обусловливает необходимость совершенствования мероприятий процесса разработки, в частности, верификации как основного вида работ по контролю качества программного продукта. Нацеленность на улучшение процесса верификации объясняется растущей сложностью специального ПО, что становится причиной наличия в нем все большего количества дефектов. Серьезность сложившейся в программной инженерии ситуации подтверждается фактом увеличения доли затрат на выявление и устранение дефектов до 80 % от общей стоимости специального ПО [1, 2].

Нарушение требований к функциональности касается в основном специального ПО, которое разрабатывается эксклюзивно для реализации возможностей вновь создаваемых АСВН, что отличает специальное ПО от унифицированных решений для общего ПО. Данная особенность до минимума сокращает возможность повторного использования уже проверенных фрагментов кода программ и влечет появление новых дефектов. Зачастую необнаруженными после отладки и тестирования остаются функциональные дефекты, которые представляют собой искажения алгоритмов, реализующих информационно-расчетные задачи, что выражается не только в нарушении порядка вычислений (дефекты в условных и циклических конструкциях), но и в ошибочной

записи расчетных выражений в операторах программы (подмена истинных значений операций и операндов). АСВН, применяемые для управления боевыми средствами и воинскими формированиями, в функциональной части имеют подсистемы боевого управления, информационную и информационно-расчетную подсистемы. Данные подсистемы включают комплекс информационно-расчетных задач, реализуемых в специальном ПО, общим для которого является оперирование данными, обладающими физическим смыслом. Проверку корректности специального ПО возможно организовать по принципу размерной однородности физических уравнений, используемых в информационно-расчетных задачах. Нарушение данного принципа будем называть *дефектами естественной семантики* (ДЕС) программных объектов (переменных, функций и т.п.) с целью акцентирования внимания на их различной природе с семантическими дефектами, вызванными нарушением требований спецификации языка программирования.

Существует возможность обнаружения ДЕС при верификации как с помощью экспертиз, предоставляющих качественную субъективную оценку некоторой стороны продукта процесса разработки, так и на основе тестирования, полнота покрытия которого и временные затраты на проведение остаются слабым местом [3]. Статический анализ при определенной доработке также способен выявлять ДЕС, при этом он не подвержен вышеописанным недостаткам и дает воспроизводимое доказательство наличия дефектов. Статический анализ позволяет снизить суммарные трудозатраты на поиск и локализацию дефектов специального ПО за счет значительной автоматизации процедуры его проведения при наличии достаточных вычислительных ресурсов [3, 4]. Подход может применяться уже на ранних этапах фазы разработки специального ПО, так как в качестве входной информации требует исходный код программы. Кроме того, возможность применения статического анализа во время кодирования позволяет сократить стоимость исправления дефектов специального ПО до пяти раз по сравнению с этапом тестирования [1].

Метод верификации специального ПО АСВН на основе статического анализа для поиска ДЕС

Структура метода обусловлена целями, стоящими перед процессом верификации согласно нормативным документам, и включает методику статического анализа для поиска ДЕС программных объектов и методику расчета полноты статического анализа для поиска ДЕС программных объектов.

Методика статического анализа основана на работе со значениями семантического домена, состоящего из утверждений о значениях свойств объектов и о зависимостях между ними [5, 6]. Выбранное для проверки принципа размерной однородности свойство объектов программы – размерность физических величин (*естественная семантика* (ЕС)) формализуется в виде *векторов ЕС* (ВЕС) и образует линейное пространство. Кроме того, семантический домен содержит зависимости между объектами как отображение множества операций с программными объектами, характерных для языков C/C++, на множество операций с ВЕС. Непосредственно проверка на наличие ДЕС по зависимостям между объектами программы происходит во время обхода внутреннего представления программы в виде абстрактного синтаксического дерева.

Расчет полноты статического анализа для поиска ДЕС программных объектов осуществляется автоматически в процессе анализа по математическим выражениям, полученным путем аналитического моделирования поведения анализатора по поиску дефектов в некоторой элементарной конструкции, что при переходе от частного к общему позволяет делать выводы о полноте статического анализа в целом. Такой подход более предпочтительный по сравнению с преднамеренным внесением искажений в программный код, который требует затрат времени, кратных числу итераций внесения дефектов, тогда как уменьшение числа итераций снижает достоверность оценки полноты [7].

Информационная технология верификации специального ПО АСВН

Информационная технология верификации специального ПО АСВН представляет собой совокупность способов обработки и представления информации о ДЕС с применением вычислительной техники и программных средств об исследуемом специальном ПО на основе исходного кода программы и постановки задач. При этом информационная технология верификации обеспечивает переход от рутинных к промышленным методам и средствам работы с информацией во время выполнения верификации специального ПО, предоставляя возмож-

ность для ее рационального и эффективного использования [8].

Целью информационной технологии верификации является инструментальная поддержка процесса верификации специального ПО АСВН на языках С и С++ при поиске нарушений функциональных требований, заключающихся в несоблюдении принципа размерной однородности физических уравнений, реализованных в программе.

Место использования предлагаемой информационной технологии верификации в жизненном цикле специального ПО АСВН – стадии фазы разработки при выполнении процессов конструирования, комплексирования и квалификационного тестирования, а также испытания АСВН различного уровня.

Информационная технология верификации специального ПО АСВН не исключает применения других мероприятий процесса верификации (экспертизы, тестирование), а дополняет работы, проводимые в рамках статического анализа программного кода, для повышения эффективности процесса верификации в целом.

Функциональная модель информационной технологии верификации специального ПО АСВН

Информационная технология верификации специального ПО АСВН представлена в виде функциональной модели IDEF0 [9], формализующей и описывающей работы процесса верификации. Применение методологии функционального моделирования IDEF0 обусловлено популярностью у предприятий промышленности процессного подхода к управлению, где IDEF0 считается классическим инструментом. На рисунке 1 представлена диаграмма А0 функциональной модели.

Работы информационной технологии верификации специального ПО АСВН включают следующие блоки:

– блок А1; осуществляется формализация требований к ЕС программных объектов, а именно формирование их ВЕС, на выходе – формальная спецификация требований к ЕС программных объектов (O_1);

– блок А2; выполняется статический анализ для поиска ДЕС, на выходе – сообщения о дефектах (O_2) и полноте анализа (O_3);

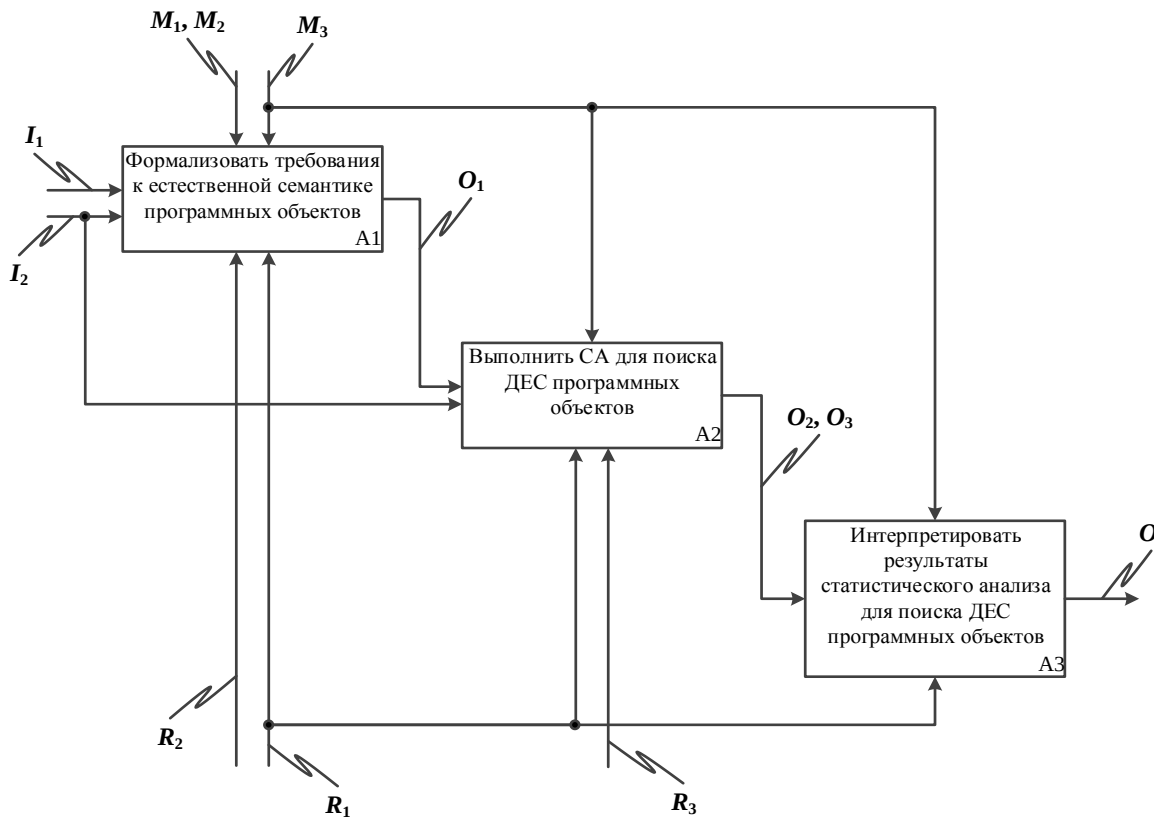

Рис. 1. Диаграмма А0 функциональной модели информационной технологии верификации

Fig. 1. An A0 diagram of a functional model for a verification information technology

– блок А3; происходит интерпретация результатов статического анализа для поиска ДЕС программных объектов, на выходе – отчет о результатах статического анализа для поиска ДЕС программных объектов (O_4).

Входные данные представляют собой документацию (I_1) и исходный код программы (I_2). Документация должна включать постановку задач, анализ которых позволит выделить и систематизировать требования к ЕС, интерпретируемой в объектах программы. Код программы должен быть представлен на исходном языке высокого уровня C, C++, Objective-C, Objective-C++ в машинно-читаемой форме, пригодной для ввода в статический анализатор.

В качестве управления выступают требования и планы в рамках верификации специального ПО, которые определяют условия, необходимые процессу для производства правильного выхода: требования к формированию уникальных идентификаторов программных объектов (M_1), требования к формированию ВЕС программных объектов (M_2), план верификации (M_3). План верификации специального ПО устанавливает временные зависимости и ограничения при выполнении работ и может быть оформлен как диаграмма Ганта.

Уникальные идентификаторы программных объектов должны включать не только имя конкретного объекта, но и имена его семантических родителей вплоть до имени файла с исходным кодом, что позволяет однозначно определить его для конкретной программы. Вариантом разрешенной таким образом неоднозначности могут быть объявление в рамках главной функции программы переменной и объявление поля с таким же именем в классе. Пример уникального идентификатора: speed | missile | function.cpp – поле speed (скорость) объявлено в классе missile (ракета) и размещено в файле function.cpp.

ВЕС программных объектов должны включать девять составляющих, каждая из которых характеризует степень сомножителя в формуле размерности физической величины. Фиксация порядка элементов в ВЕС необходима для корректного выполнения операций. Для произвольной физической величины формула размерности имеет вид:

$$\dim(A) = L^{e_1} M^{e_2} T^{e_3} I^{e_4} \Theta^{e_5} N^{e_6} J^{e_7} \alpha^{e_8} \Omega^{e_9},$$

где e_i – элемент на i-й позиции в ВЕС; L – длина; M – масса; T – время; I – электрический ток; Θ – термодинамическая температура; N – количество вещества; J – сила света; α – плоский угол; Ω – телесный угол.

В роли механизмов выполнения работ выступают специалист по верификации (R_1) и программные утилиты для верификации, позволяющие автоматизировать подготовку исходных данных для анализа (R_2) и процесс его выполнения (R_3). Данные программные утилиты предназначены для формирования уникальных идентификаторов программных объектов в рамках формализации требований и для выполнения поиска ДЕС при статическом анализе.

Процесс формализации требований к ЕС программных объектов детализируется на диаграмме А1 функциональной модели и иллюстрируется рисунком 2. Целью работ является подготовка исходных данных для статического анализа в виде списка пар «Уникальный идентификатор программного объекта – ВЕС». Работа складывается из формирования уникальных идентификаторов программных объектов и установки им соответствующих ВЕС.

Работы процесса А11: специалист по верификации специального ПО в соответствии с планом работ с помощью специальной программной утилиты в автоматическом режиме обрабатывает исходный код программы.

Входы: исходный код программы (I_2).

Выходы: уникальные идентификаторы программных объектов (O_5).

Управление: требования к формированию уникальных идентификаторов программных объектов (M_1), план верификации (M_3).

Механизмы: специалист по верификации (R_1), программная утилита для формирования уникальных идентификаторов программных объектов на базе инфраструктуры LLVM и фронтенда Clang (R_2).

Работы процесса А12: специалист по верификации, руководствуясь планом работ, в ручном режиме преобразует в векторы размерности, характерные для объектов исследуемой программы.

Входы: документация (I_1).

Выходы: ВЕС программных объектов (O_6).

Управление: требования к формированию ВЕС программных объектов (M_2), план верификации специального ПО (M_3).

Механизмы: специалист по верификации (R_1).

Работы процесса А13: специалист по верификации вручную формирует пары для программных объектов вида «Уникальный идентификатор – ВЕС».

Входы: документация (I_1), исходный код программы (I_2), уникальные идентификаторы

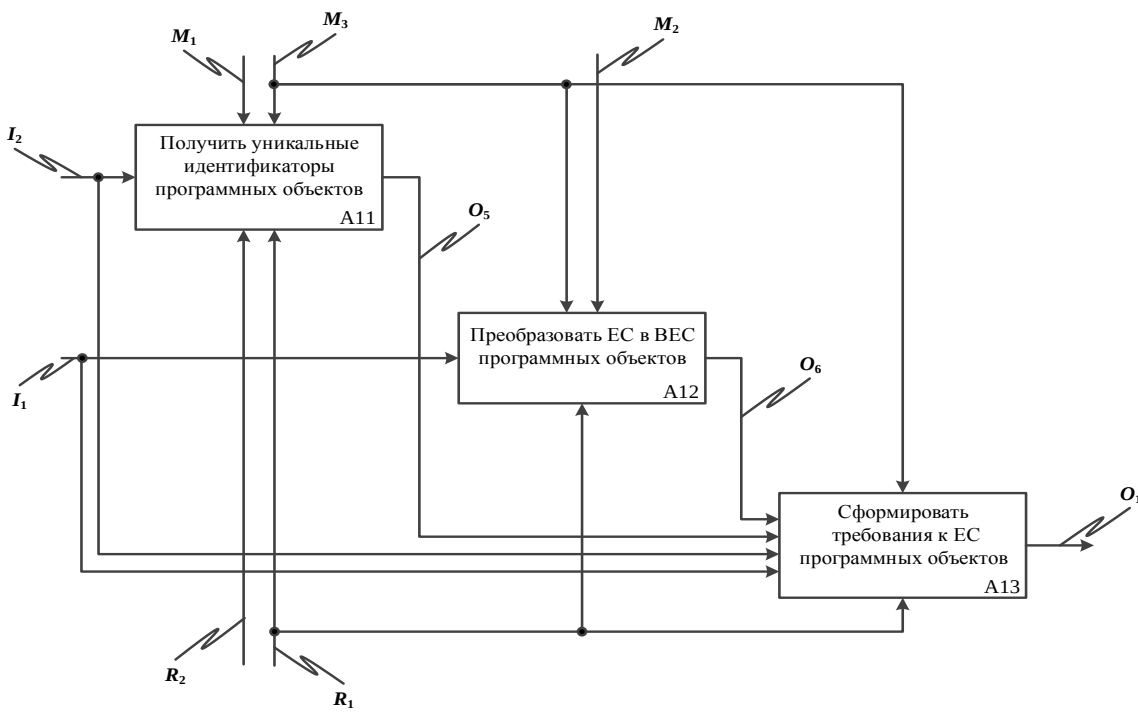

Рис. 2. Диаграмма A1 функциональной модели информационной технологии верификации

Fig. 2. An A1 diagram of a functional model for a verification information technology

программных объектов (O_5) и ВЕС программных объектов (O_6).

Выходы: формальная спецификация требований к ЕС программных объектов (O_1).

Управление: план верификации специального ПО (M_3).

Механизмы: специалист по верификации (R_1).

В рамках выполнения статического анализа для поиска ДЕС (A_2) специалист по верификации вводит в утилиту NSDdetector программный код исследуемого специального ПО (I_2), формальную спецификацию требований к ЕС программных объектов (O_1) и начинает процедуру анализа. Программная утилита опционально выводит либо на экран компьютера, либо в текстовый файл результат анализа, который включает сообщения о ДЕС, указывающие на строку в файле исходного кода и позицию в операторе (O_2), а также частные (по типам конструкций) и интегральный показатели полноты статического анализа для поиска ДЕС (O_3).

Интерпретация результатов подразумевает подготовку отчета о проведении статического анализа для поиска ДЕС (O_4), который включается в итоговый отчет о верификации специального ПО. Данные о проведении статического анализа (рис. 3) содержат название файла с исходным кодом программы, используемый язык программирования, размер программы в строках кода, время проведения анализа, локализацию ДЕС в файле исходного кода и описание корректирующих действий для его устранения; интегральный и частные показатели полноты, вывод о корректности программы с точки зрения наличия ДЕС программных объектов.

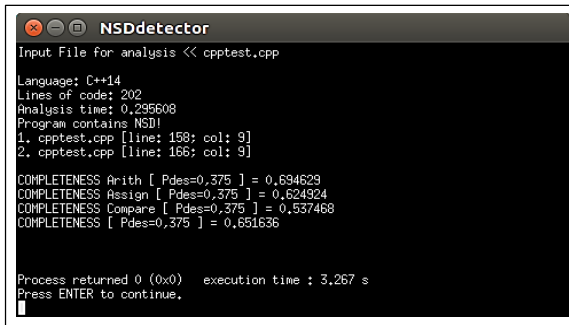

Рис. 3. Экранная форма с результатами статического анализа для поиска ДЕС программных объектов

Fig. 3. A screen view with static analysis results for searching natural semantic defects of program objects

Утилиты поддержки информационной технологии верификации специального ПО АСВН

Программные утилиты для верификации, позволяющие автоматизировать поиск ДЕС и расчет полноты статического анализа, разработаны с использованием инфраструктуры для создания промышленных компиляторов LLVM и анализатора Clang.

Поиск ДЕС возможно производить до этапов оптимизации и генерации кода, что при создании утилит позволяет ограничиться фронтендом фреймворка LLVM – Clang. Основным преимуществом Clang является его ориентация на максимальное сохранение информации о ходе процесса компиляции и коде программы, что гарантирует его точное воспроизведение в абстрактное синтаксическое дерево. Фронтенд дает возможность доступа к дереву через механизм курсоров (узлов), а также удобный инструментарий для его обхода и манипулирования, что делает доступным, кроме сбора статистических данных об исследуемом специальном ПО, вычисление производных значений ВЕС программных объектов и контроль корректности операций с ними. При обнаружении ДЕС в узле абстрактного синтаксического дерева Clang предоставляет способ локализации его в файле исходного кода с точностью до строки и позиции в операторе.

Эффективность мероприятий процесса верификации специального ПО АСВН для поиска ДЕС

В качестве показателя эффективности процесса верификации специального ПО АСВН выбрана интенсивность выявления ДЕС, связывающая полезный эффект – количество обнаруженных ДЕС вариантом содержания процесса верификации и необходимые затраты времени на вариант, каждый из которых представляет собой некоторую совокупность мероприятий для достижения цели верификации: $\overline{\lambda}_{\text{В}} = \sum_{k=1}^{K} N_k \big/ \sum_{k=1}^{K} T_k$, где N_k – количество дефектов, обнаруживаемых k-м мероприятием; T_k – время выполнения k-го мероприятия. Логично полагать, что, чем выше интенсивность обнаружения, тем эффективнее верификация.

Для выполнения оценки выигрыша в эффективности верификации рассматриваются исходный вариант, представленный тестированием, и предлагаемый как сочетание тестирования и статического анализа для поиска ДЕС. Время на работы по тестированию оценивалось по детальной методике COCOMO II [10]. Время статического анализа получено опытным путем с помощью специально разработанной программной утилиты для поиска ДЕС на основе Clang Analyzer. Время на допоиск тестированием не выявленных статическим анализом дефектов рассчитывалось по COCOMO II исходя из достигнутой полноты статического анализа. Для обоих вариантов при расчете использовались номинальные значения факторов, характеризующих время на верификацию, детальной модели COCOMO II.

Зависимость выигрыша в эффективности процесса верификации от полноты статического анализа для поиска ДЕС для программного кода размером порядка тысячи строк кода на C++ показана на рисунке 4.

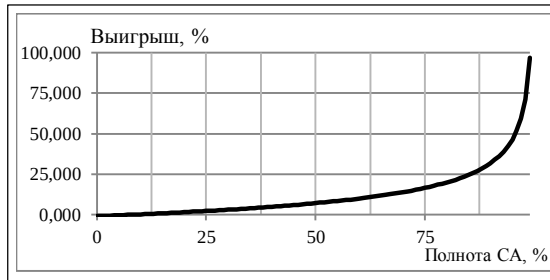

Рис. 4. Относительный выигрыш в увеличении средней интенсивности обнаружения дефектов

Fig. 4. Relative gain in increasing of an average defect detection rate

Заключение

Представленная информационная технология верификации специального ПО АСВН реализует полноценный с точки зрения требований нормативных документов метод поиска и регистрации функциональных дефектов и может использоваться как одно из мероприятий в рамках процесса верификации специального ПО АСВН для раннего выявления ДЕС с большей эффективностью. При этом использование предлагаемой информационной технологии верификации для поиска ДЕС программных объектов позволяет получить выигрыш в эффективности процесса верификации до 20–25 % для большинства практических случаев.

Литература

1. Лобанова Н.М., Дубровин Д.А. Оценка затрат на качество программного обеспечения // Вестн. ун-та. 2016. Вып. 6. С. 87–91.

2. Черников Б.В., Поклонов Б.Е. Оценка ка-

чества программного обеспечения. М.: ФОРУМ–ИНФРА-М, 2012. 400 с.

3. Кулямин В.В. Методы верификации программного обеспечения. М.: Изд-во ИСП РАН, 2008. 117 с.

4. Ицыксон В.М., Моисеев М.Ю., Цесько В.А., Захаров А.В., Ахин М.Х. Алгоритм интервального анализа для обнаружения дефектов в исходном коде программ // Информационно-управляющие системы. 2009. Вып. 2. С. 34–41.

5. Глухих М.И., Ицыксон В.М., Цесько В.А. Использование зависимостей для повышения точности статического анализа программ // Моделирование и анализ информационных систем. 2011. Вып. 4. С. 68–79.

6. Несов В.С., Маликов О.Р. Использование информации о линейных зависимостях для обнаружения уязвимостей в исходном коде программ // Тр. ИСП РАН. 2006. № 9. С. 51–57.

7. Гуров Д.В., Гуров В.В., Иванов М.А. Использование моделей надежности программного обеспечения для оценки защищенности программного комплекса // Безопасность информационных технологий. 2012. № 1. С. 88–91.

8. Андреев Г.И., Созинов П.А., Тихомиров В.А. Управленческие решения при проектировании радиотехнических систем. М.: Радиотехника, 2018. 560 с.

9. Бистерфельд О.А. Методология функционального моделирования IDEF0. Рязань: Изд-во Рязан. гос. ун-та им. С.А. Есенина, 2008. 48 с.

10. Липаев В.В. Технико-экономическое обоснование проектов сложных программных средств. М.: СИНТЕГ, 2004. 284 с.

Software & Systems
DOI: 10.15827/0236-235X.126.283-289

Received 23.10.18
2019, vol. 32, no. 2, pp. 283–289

An information technology for verifying special software of military automated systems

R.E. Zhidkov [1], *Postgraduate Student, r-zhidkov@rambler.ru*
D.S. Viktorov [1], *Dr.Sc. (Engineering), Associate Professor, victorov.dmitry@yandex.ru*
E.N. Zhidkov [1], *Ph.D. (Engineering), eng1964@mail.ru*

[1] *Military Academy of the Aerospace Defence, Tver, 170022, Russian Federation*

Abstract. There is a need to create an information technology of military automated system special software verification that allows effective checking of functionality requirements. This is due to the need to create high-quality software within time and financial constraints of the development process.

The paper proposes an information technology that is a combination of methods and software utilities for processing a source code of the investigated program and technical documentation in order to obtain information of functional defects. The described information technology might be used both at the coding stage and at different testing stages in addition to the existing solutions for searching functionality defects.

The verification method has a structure determined by the requirements of regulatory documents in the software development. This method includes a static analysis technique and a technique for assessing analysis completeness based on source code characteristics. This verification method is designed to check the correctness of the implementation of information-calculated tasks in automated systems for managing troops and military equipment based on the principle of dimensional homogeneity of physical equations.

The verification information technology is described as an IDEF0 functional model detailed to the second nesting level with a description of inputs and outputs of all works. In order to support the verification process, the authors propose software utilities for automating routine operations: generating unique identifiers of software objects, searching and intentional defects introducing. These software tools are based on the infrastructure of LLVM compiler and Clang frontend that allow checking the program codes in C, C++, Objective-C.

Keywords: information technology, verification, software, IDEF0, functional defects, static analysis.

References

1. Lobanova N.M., Dubrovin D.A. Software quality cost assessment. *University Herald*. 2016, no. 6, pp. 87–91 (in Russ.).
2. Chernikov B.V., Poklonov B.E. *Software Quality Assessment*. Moscow, "FORUM" INFRA-M Publ., 2012, 400 p.
3. Kulyamin V.V. *Methods of Software Verification*. Moscow, ISP RAN Publ., 2008, 117 p.
4. Itsykson V.M., Moiseev M.Yu., Tsesko V.A., Zakharov A.V., Akhin M.Kh. Interval analysis algorithm for detecting defects in a software source code. *Information Control Systems*. 2009, no. 2, pp. 34–41 (in Russ.).
5. Glukhikh M.I., Itsykson V.M., Tsesko V.A. The use of dependencies for improving the precision of program static analysis. *Modeling and Analysis of Information Systems*. 2011, no. 4, pp. 68–79 (in Russ.).
6. Nesov V.S., Malikov O.R. The use of linear dependency information to detect vulnerabilities in program source code. *Proc. ISP RAS*. 2006, no. 9, pp. 51–57 (in Russ.).
7. Gurov D.V., Gurov V.V., Ivanov M.A. Use of software reliability models to assess software package security. *IT Security*. 2012, no. 1, pp. 88–91 (in Russ.).
8. Andreev G.I., Sozinov P.A., Tikhomirov V.A. *Management Disisions in the Design of Radio Systems*. Moscow, Radiotekhnika Publ., 2018, 560 p.
9. Bisterfeld O.A. *IDEF0 Functional Modeling Methodology*. Ryazan, Ryaz. gos. un-t im. Esenina Publ., 2008, 48 p.
10. Lipaev V.V. *Feasibility Study of Complex Software Projects*. Moscow, SINTEG Publ., 2004, 284 p.