УДК 004.05+004.415.5

Ю.С. Манжос, В.Л. Петрик

Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Украина

ДЕСКРИПТОРНЫЙ КОНТРОЛЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КРИТИЧЕСКОГО ПРИМЕНЕНИЯ В РЕАЛЬНОМ ВРЕМЕНИ

Предложен метод, обеспечивающий контроль семантической корректности программного обеспечения на этапе стендовой отработки или в реальном времени, имеющий по сравнению с методом семантического контроля на порядок меньшую ресурсоемкость, значительно более высокую диагностирующую способность и более точную оценку вероятности существования остаточных программных дефектов. Применение метода в системах реального времени позволяет повысить надежность информационно-управляющих систем для АЭС и авиационно-космических комплексов.

Ключевые слова: семантический контроль, семантические инварианты, остаточные программные дефекты.

Введение

Безопасность современного общества определяется функциональной безопасностью программно-технических комплексов (ПТК) критического применения. Одноканальная архитектура при интенсивности отказов $10^{-4}..10^{-3}$ $ч^{-1}$, которую имеют современные устройства, не позволяет обеспечить вероятность безотказной работы выше 0.8 на протяжении 2000..200 часов [1]. В тоже время продолжительность эксплуатации таких систем исчисляется годами, что требует поиска новых решений обеспечения надежности.

Постановка задачи. Необходимый уровень надежности может быть достигнут с использованием многоверсионной избыточности [2], подразделяющейся на модельную, программную и аппаратную. Общим для них является использование нескольких программно-аппаратных каналов, а также мажоритарных элементов и коммутаторов. Параллельная работа программных версий функционального программного обеспечения (ПО) в сочетании с мажоритарным элементом позволяет формировать на выходе системы правильный выход по принципу M/N, что требует разработки нескольких версий ПО и многократного увеличения затрат и сроков разработки. Предлагается в качестве одной из программных версий использовать семантические отображения, осуществляемые программным кодом над семантиками (физическими размерностями) данных. Это позволит без разработки дополнительных версий функционального ПО, а лишь посредством контроля семантической корректности, диагностировать вычислительные процессы в реальном времени, упростить дистанционную модификацию и сопровождение ПО, и, как следствие, повысить надежность современных ПТК критического применения. Реализация семантического контроля (СК) [3, 4], использующего векторное представление физической размерности, в системах реального времени требует десятикратного увеличения объема памяти и количества операций. Кроме того СК позволяет при выполнении программы только фиксировать факт нарушения семантических инвариантов, что значительно снижает его диагностирующую способность.

Необходима разработка более эффективного, с точки зрения ресурсоемкости и диагностики, метода контроля семантической корректности ПО.

1. Целочисленное дескрипторное отображение

В основу контроля целесообразно положить (рис. 1) целочисленное семантическое дескрипторное отображение (ЦСДО) [5], эффективно представляющее физическую размерность в виде целых чисел – семантических дескрипторов (СД) и дескрипторную алгебру, позволяющую эффективно верифицировать корректность исходного кода посредством контроля сохранности семантических инвариантов [3] при операторных отображениях. ЦСДО проецирует семантическое пространство (СП) [5] на целочисленную ось – одномерное пространство целочисленных семантических дескрипторов (ЦСД), что и позволяет представить размерность программных переменных одним целым числом и снизить как объем дополнительной памяти, так и количество дополнительных операций, необходимых для реализации СК.

Реализация дескрипторного контроля (ДК) возможна для ПО на языках, допускающих переопределение операций. Аддитивные операции, к которым относятся программные операции: *присваивания, сложения, вычитания, сравнения, вызова функций;* а также математические функции: *модуль, sin, cos, tg, arcsin* и т.д, должны дополнительно оценивать корректность использования операндов и результатов вычисления с точки зрения физической размерности.

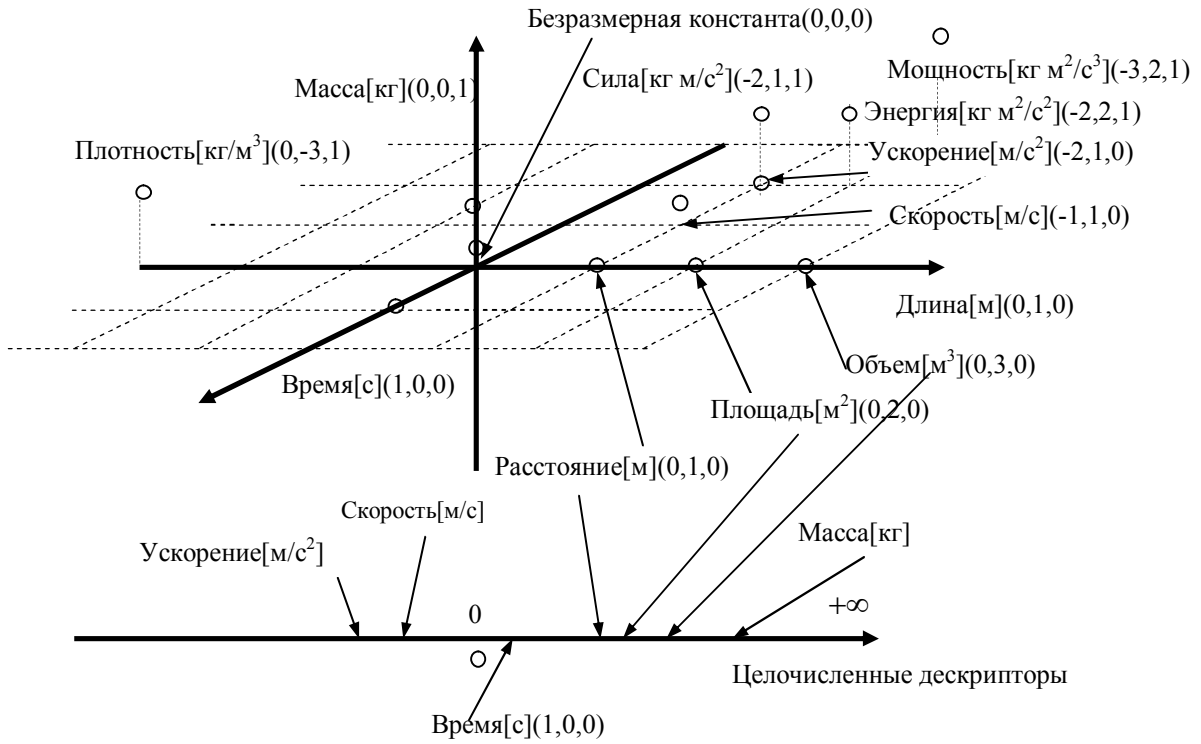

Рис. 1. Целочисленное семантическое отображение элементов семантического пространства в базисе: *Время, Длина, Масса* на целочисленную ось

При этом ЦСД операндов программных операций и формально-фактических параметров должны совпадать, а ЦСД аргументов математических функций должны иметь определенные значения, например, соответствовать размерности *радиан,* или быть безразмерными. Мультипликативные операции: *умножения, деления, возведения в степень,* а также некоторые из математических функций должны формировать новые размерности, представленные целочисленными СД, корректность которых затем должна проверяться переопределенными аддитивными операциями. Нарушение семантической корректности должно формировать исключительные ситуации либо генерировать сигналы в управляющий процесс.

Диагностирование (рис. 2) выполняется в реальном времени с точностью до аддитивной операции. Для облегчения обнаружения мест программного кода, содержащих семантические дефекты (СД), параллельно с выполнением основной задачи и диагностики восстанавливается программный код. Для этого переопределенные операции на основе своих аргументов формируют бинарное дерево арифметического выражения. При обнаружении СД сформированное дерево, приводится к текстовому представлению и перенаправляется в соответствующий поток вывода, который может быть связан как с отдельным файлом, так и с удаленным процессом. Использование ДК при стендовой отработке ПО позволяет более точно оценить вероятность отсутствия СД в верифицируемом коде. Это объясняется тем, что формула, приведенная в [4], учитывает статическое распределение данных и операций в исходном коде, в то же время ПО в разных режимах работы использует различные множества алгоритмов и данных, поэтому и вероятность существования остаточных СД будет различной для различных режимов. Переопределение операций позволяет накапливать информацию о семантическом и операционном спектрах и получать более точное значение вероятности существования остаточных СД, а также о полноте контроля [6]. В связи с тем, что в ПО используются различные типы данных, многие из которых имеют физическую размерность, для переопределения данных необходимо использовать шаблон класса, переопределяющий основные типы данных и множество шаблонов функций, переопределяющих операции над данными. Перечисленные шаблоны классов и функций оформлены в виде библиотеки, реализованной на стандарте языка С++.

2. Методика использования библиотеки классов и шаблонов дескрипторного контроля ПО

В основу методики положено использование библиотеки классов и шаблонов, обеспечивающих контроль семантических инвариантов с использованием целочисленных семантических дескрипторов, поэтому использование методики ограничено языком С++.

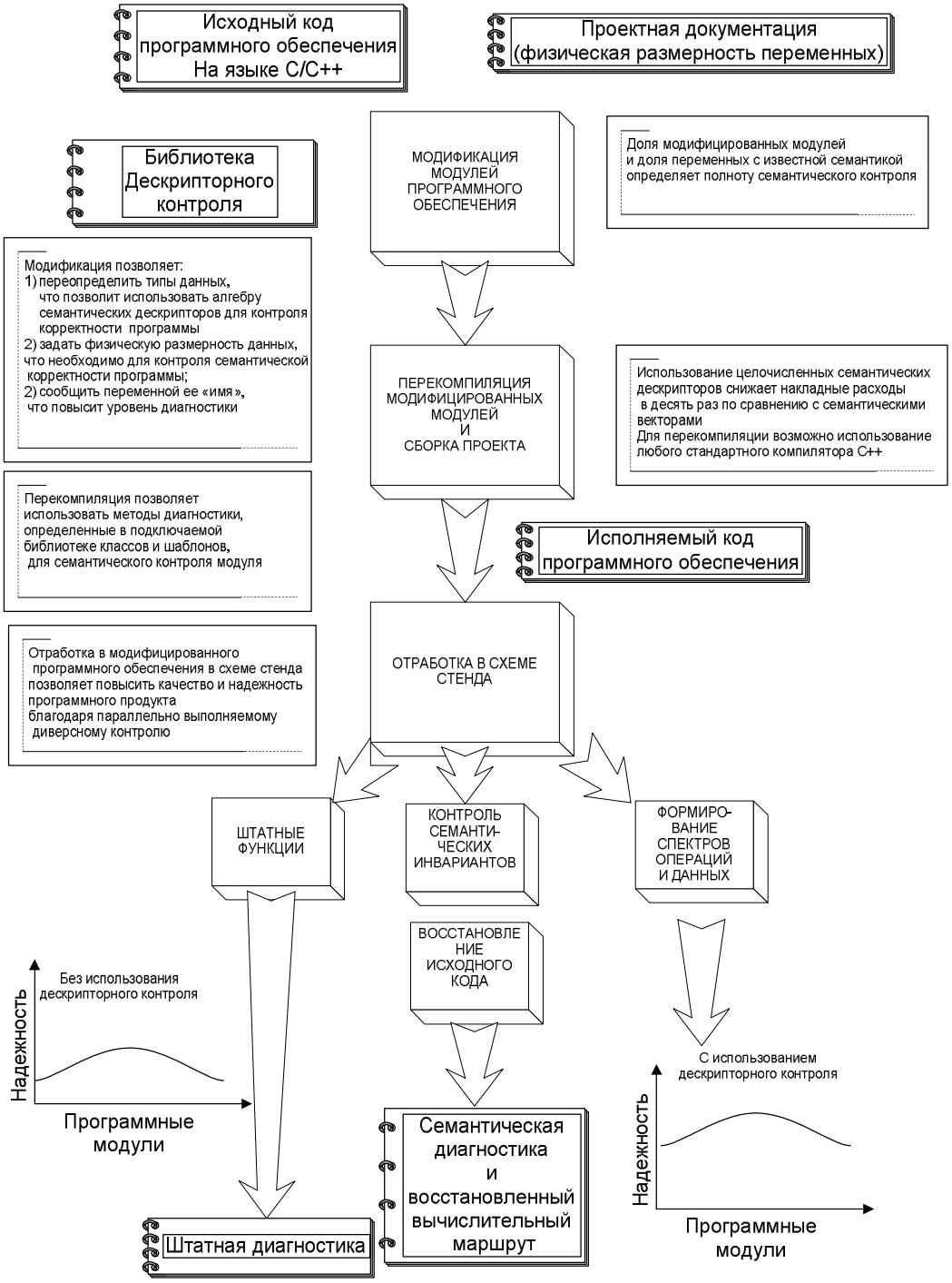

Рис. 2. Схема дескрипторного контроля ПО

Методика состоит из нескольких этапов:

1. Добавление библиотеки семантического контроля к исходному коду проекта.

2. Переопределение стандартных типов данных. Для этого:

а) необходимо в начале каждого программного модуля добавить строку

#include "idchecker.h",

подключающую библиотеку классов и шаблонов дескрипторного контроля;

б) для всех переменных, физическая размерность которых известна, необходимо в соответствии с табл. 1 переопределить типы данных.

Конструкция TIDChecker<тип> определяет инстанцирование шаблона для данного стандартного типа данных, что и обеспечивает контроль семантической корректности вычислительного процесса, восстановление исходного кода, оценку полноты контроля и уточнение вероятности остаточных СД.

Конструкция __(переменная) – определяет использование специального макроса, позволяющего передать программной переменной ее собственное имя, что необходимо для более детальной диагностики сообщений о семантических некорректностях.

Таблица 1

Форма переопределения данных

Исходный код	Модифицированный код

unsigned char a;
char b;
unsigned int c;
int d;
unsigned long e;
long f;
float g;
double h;
long double q;
	TIDChecker<unsigned char>
	__(a);
	TIDChecker<char> __(b);
	TIDChecker<unsigned int> __(c);
	TIDChecker<int> __(d);
	TIDChecker<unsigned long>
	__(e);
	TIDChecker<long> __(f);
	TIDChecker<float> __(g);
	TIDChecker<double> __(h);
	TIDChecker<long double> _(q);

Задания размерностей локальных переменных необходимо выполнять в соответствующем файле. Задание размерностей глобальных переменных лучше вынести в отдельный файл. В общем случае используются конструкции, приведенные в табл. 2.

Таблица 2

Задание размерностей программных переменных

Имя	Размерность	Исходный код
a	ньютон	a.setUnit("kg",1, "m", 1, "s", -2);
b	кило-грамм	b.setUnit("kg");
c	м/сек	c.setUnit("m",1,"s",-2);
d	ньютон	d.setSemantic (SemanticVector::Force);
e	кило-грамм	e.setSemantic (SemanticVector::Massa);
f	м/сек^2	f.setSemantic (SemanticVector::Aceleration)

Здесь переменным *a, d* присваивается размерность Ньютон, переменным *b ,e* – масса (килограмм), переменным *c, f* – ускорение. Далее модифицированные модули программного проекта необходимо вновь откомпилировать и собрать редактором связей новую версию проекта. Выполнение проекта в схеме стенда позволит получать более достоверную информацию о корректности проекта. Полнота проверок может регулироваться долей модифицированных модулей и долей известных физических размерностей переменных.

Заключение

Предложен метод, обеспечивающий контроль семантической корректности ПО на этапе стендовой отработки или в реальном времени, имеющий по сравнению с известным на порядок меньшую ресурсоемкость, значительно более высокую диагностирующую способность и более точную оценку вероятности существования остаточных программных дефектов. Применение метода в системах реального времени позволяет повысить надежность информационно-управляющих систем для АЭС и авиационно-космических комплексов. Дальнейшие исследования целесообразно проводить в направлении использования нескольких программных инвариантов [3] для контроля достоверности процессов реального времени.

Список литературы

1. Артеменко Е.А. Резервирование, диагностирование, восстановление – система обеспечения надежности сложных технических комплексов // Модели и системы. – 1999. – № 1. – С. 4-7.

2. Харченко В.С. Многоверсионные цифровые системы, важные для безопасности: анализ и перспективы // Модели и системы. – 1999. – № 1. – С. 61-64.

3. Калибровка чувствительности методов статического анализа, используемых для оценки качества и безопасности ПО ИУС АЭС / Б.М. Конорев, Ю.Г. Алексеев и др, // Межд. симпозиум «Измерения, важные для безопасности в реакторах». – М.: ИПУ, 2004. – С. 15-1–15-12.

4. Харченко В.С., Манжос Ю.С., Петрик В.Л. Статистический анализ программного обеспечения системы управления космическим аппаратом и оценка проверяющей способности семантического контроля // Технология приборостроения. – 2002. – № 2. – С. 52-59.

5. Петрик В.Л. Целочисленное семантическое отображение // Радіоелектронні і комп'ютерні системи. – 2007. – Вип. 3 (22). – С. 73-81.

6. Манжос Ю.С., Петрик В.Л. Оценка полноты семантического контроля программного обеспечения информационно-управляющих систем // Авіаційно-космічна техніка і технологія. – 2007. – Вип. 5 (41). – С. 86-93.

Поступила в редколлегию 27.02.2008

Рецензент: д-р техн. наук, проф. Б.И. Туркин, Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Харьков.

ДЕСКРИПТОРНИЙ КОНТРОЛЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КРИТИЧНОГО ЗАСТОСУВАННЯ В РЕАЛЬНОМУ ЧАСІ

Ю.С. Манжос, В.Л. Петрік

Запропонований метод, що забезпечує контроль семантичної коректності програмного забезпечення на етапі стендового відробітку або в реальному часі, має в порівнянні з методом семантичного контролю на порядок меншу ресурсоемність, значно вищу діагностуючу здатність і точнішу оцінку вірогідності існування залишкових програмних дефектів. Застосування методу в системах реального часу дозволяє підвищити надійність інформаційно-управляючих систем для АЕС і авіаційно-космічних комплексів.

Ключові слова: семантичний контроль, семантичні інваріанти, залишкові програмні дефект.

DESCRIPTOR CONTROL OF CRITICAL APPLICATION SOFTWARE IN REAL TIME

Yu.S. Manzhos, V.L. Petrik

A method, providing control of semantic correctness of software on the stage of the stand working off or in real time, having as compared to the method of semantic control on an order less ресурсоемкость, is offered, considerably more high diagnosing ability and more exact estimation of probability of existence of remaining programmatic defects. Application of method in the real-time systems allows to promote reliability of the sensor-based systems for APP and complexes of aviation-spaces.

Keywords: semantic control, semantic invariants, remaining programmatic defects, descriptor control of the real time.