УДК 681.3.06

В.И. ДУЖИЙ, И.В. ДУЖИЙ, А.В. ШОСТАК

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

РАЗРАБОТКА МНОГОВЕРСИОННОЙ ИЕРАРХИИ РЕШЕНИЙ ПРИ ПРОЕКТИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Предложена классификация многоверсионных проектов, основанная на этапах жизненного цикла. На основании данной классификации предложена графовая модель формирования множества версий многоверсионного программного обеспечения, основанная на представлении всего множества решений в виде ярусного графа. Описана методика формирования множества версий с использованием элементов решений на основе ярусного графа. Приведен пример двухверсионного проекта, представленного в виде ярусного графа и в аналитической форме.

многоверсионный проект, ярусный граф, двухверсионный проект

Постановка проблемы

Нормативные документы и практика разработки сложных аппаратно-программных систем критического применения требует использования версионной избыточности или многоверсионности, под которой понимают разработку нескольких вариантов одного и того же проекта несколькими группами разработчиков различными способами и при помощи различных средств [1].

Однако, существующие в настоящее время решения для разработки многоверсионных проектов, не в полной мере удовлетворяют требованиям разработчиков программного обеспечения (ПО) критического применения.

В стандартах различных уровней и научно-методической литературе основное внимание уделяется классификации видов версионной избыточности [2] и оценке многоверсионных проектов [3]. Однако методы формирования многоверсионных проектов не достаточно формализованы и обычно сводятся к перечислению элементов решений на каждом этапе жизненного цикла.

Поэтому, качество формирования многоверсионного проекта программного обеспечения (ПО) в значительной степени зависит от опыта и способностей участников разработки.

Целью данной статьи является разработка метода генерации множества версий программного обеспечения для их дальнейшей оценки и формирования многоверсионного проекта ПО.

Основные положения предложенной методики

Существующая в настоящее время классификация введения избыточности может быть представлена, как показано на рис. 1.

В общем виде проектирование многоверсионного программного обеспечения можно разделить на два этапа: проектирование программного обеспечения и разработка программного обеспечения.

Многоверсионность, вносимая на этапе проектирования, минимально зависит от непосредственных исполнителей, будем называть этот вид версионной избыточности внешней проектной многоверсионностью, поскольку она вносится на этапе проектирования (design multiversity). Внутренняя многоверсионность полностью определяется используемыми техническими решениями, выбираемыми разработчиками, непосредственно реализующими данное программное обеспечение.

Факторы, определяющие внешнюю многоверсионность, приведены в табл. 1.

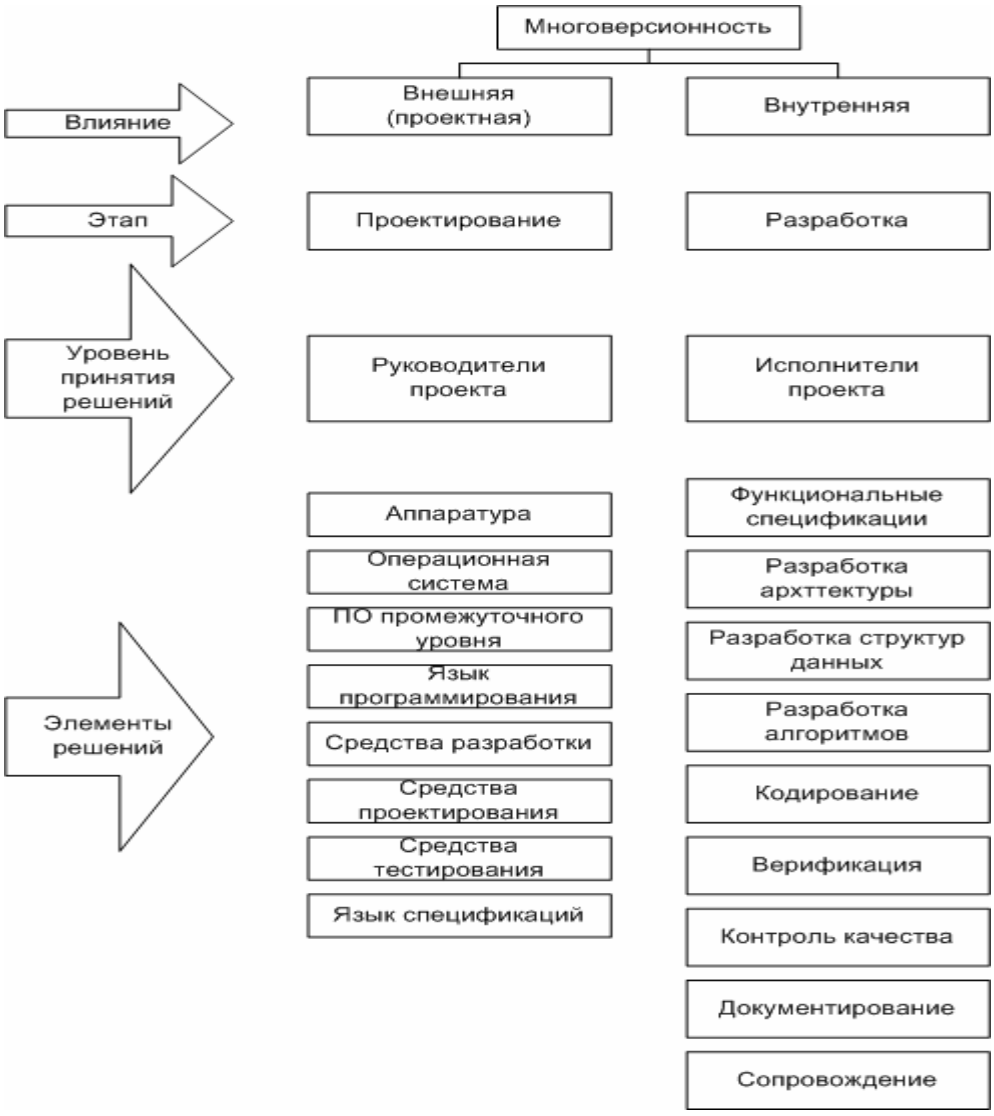

Рис. 1. Классификация видов версионной избыточности ПО

Формирование многоверсионного проекта ПО, использующего факторы внешней многоверсионности, выполняют в три этапа:

1) формирование всего множества факторов, которые будут использованы для разработки многоверсионного ПО;

2) формирование иерархии факторов многоверсионного проекта ПО в виде неориентированного ярусного графа;

3) генерация множества версий.

На первом этапе анализируется все множество факторов, которые могут быть использованы для формирования многоверсионного проекта. Эти факторы группируют по какому-либо критерию в группу, и совместно они образуют один ярус неориентированного графа.

Все множество решений для любого многоверсионного ПО представляется в виде многоярусного неориентированного графа. После формирования графа из каждой группы необходимо удалить варианты решений, которые непригодны или недопустимы для использования в данном проекте по какому-либо критерию. Например, заказчик может отдавать предпочтение технологическим решениям определенного производителя, технические решения другого производителя могут плохо интегрироваться с имеющимся у заказчика ПО и другими причинами. Эти и другие причины могут привести

к тому, отдельные элементы решений следует исключить из рассмотрения при генерации версий.

Таким образом, для формирования внешних версий многоверсионного ПО имеем исходное поле решений, для которого элементы решений были определены в табл. 1.

Таблица 1

Элементы решений и их характеристика

Элемент решений	Характеристика	Множество значений	Условное обозначение
Hard	Аппаратное обеспечение	PC, Industry, R6000, ESA, Mac, ПЛИС	$h_0, h_1, \dots, h_{k-1}$
OS	Операционная система	WIN, LIN, BSD, MacOS	$o_0, o_1, \dots, o_{n-1}$
MW	ПО промежуточного уровня	NET, J2EE, J2ME, CORBA	$m_0, m_1, \dots, m_{p-1}$
Lang	Язык программирования	C#, C, C++, Java, ObjectPascal, Ada, SQL	$l_0, l_1, \dots, l_{q-1}$
IDE	Среда разработки	VS, BOR, ECL, II, OB, A, ORC, KD	$d_0, d_1, \dots, d_{r-1}$
Spec	Язык спецификаций	NAT, UML, IDEF	$s_0, s_1, \dots, s_{t-1}$
CASE	Среда проектирования	VS, TOG, RS, ECL, ORC	$c_0, c_1, \dots, c_{v-1}$
BD	СУБД	MSSQL, MY, PSG, FB, DB, ORC	$b_0, b_1, \dots, b_{w-1}$

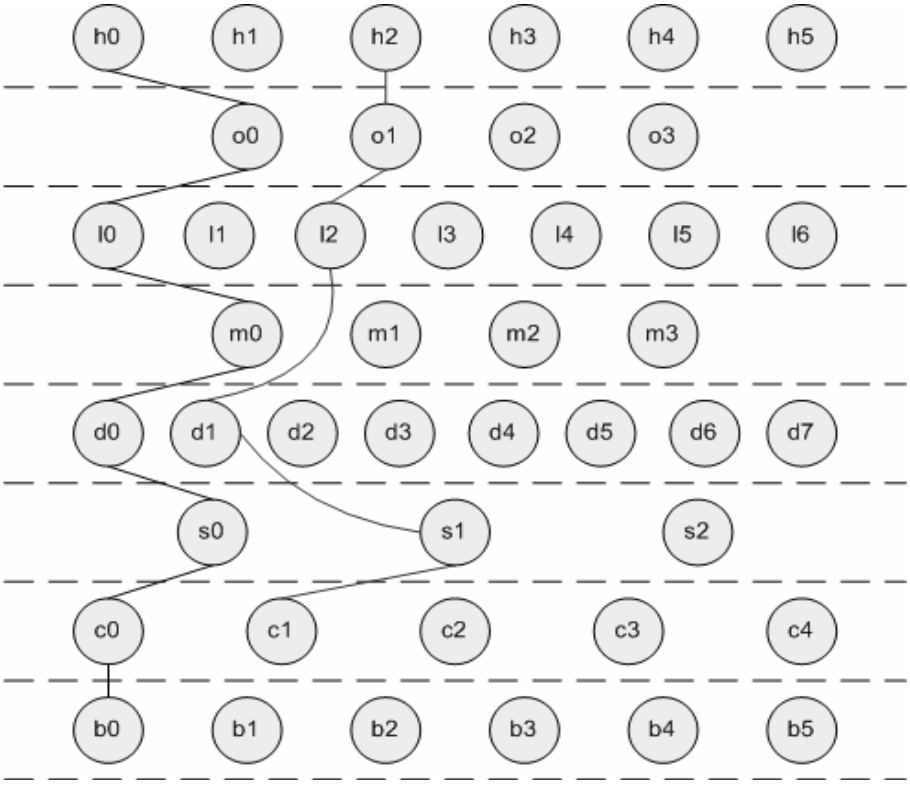

Рис. 2. Граф вариантов решений для формирования многоверсионного ПО (две версии являются полностью независимыми)

На втором этапе формируют граф допустимых комбинаций вариантов решений. Для этого при помощи ребер соединяют узлы графа, обозначающие элементы решений. Как правило, допустимые варианты решений располагаются в соседних ярусах, поэтому большинство ребер будут соединять вершины, расположенные в соседних ярусах. Тем не менее, допускается возможность связей между вершинами в несмежных ярусах графа. Таким образом, получают ярусный граф допустимых вариантов решений.

На третьем этапе выполняют генерацию версий многоверсионного ПО.

Для генерации многоверсионных проектов выбираем различные варианты решений из каждой

группы вариантов, представленные в виде вершин графа, учитывая следующие соображения:

– для каждой версии проекта из каждой группы решений выбирают только один вариант;

– выбранные варианты должны быть совместимы между собой;

– при формировании версии некоторые варианты из каждой группы решений могут отсутствовать.

Таким образом, все множество версий многоверсионного ПО – перечень путей из вершин верхнего яруса в вершины нижнего яруса. В качестве исходной точки для генерации версий можно выбрать узел на верхнем уровне и перемещаться сверху вниз. Однако, допустимо выбрать вершину на любом ярусе графа и передвигаться по графу вверх и вниз, выбирая следующий элемент решения. Каждая ветвь графа, соединяющие его вершины на разных уровнях, представляет собой одну версию проекта:

$$V_n = <h_k, o_n, m_p, l_q, d_r, s_t, c_v, b_w>.$$

Сформируем возможные варианты версий, которые могут быть получены из графа решений, представленного на рис. 2.

В двухверсионном проекте P1 используются различные элементы из каждой группы решений. Данную реализацию проекта P1 можно считать максимально диверсной и рассматривать как один из крайних случаев. Такая реализация многоверсионного проекта является наиболее дорогим и сложным решением. Это объясняется тем, что для проектов такого рода необходимо приобретать и использовать существенно различные технологии и оборудование.

Данный проект может быть описан следующими аналитическими зависимостями, описывающими каждую версию:

$$V1 = h0 \,\&\, o0 \,\&\, l0 \,\&\, m0 \,\&\, d0 \,\&\, s0 \,\&\, c0 \,\&\, b0 =$$
$$= \text{PC} \,\&\, \text{WIN} \,\&\, \text{C\#} \,\&\, \text{NET} \,\&\, \text{VS} \,\&$$
$$\&\text{NAT} \,\&\, \text{VIS} \,\&\, \text{MSSQL};$$
$$V2 = h2 \,\&\, o1 \,\&\, l2 \,\&\, d1 \,\&\, s1 \,\&\, c1 =$$
$$= \text{R6000} \,\&\, \text{LIN} \,\&\, \text{C++} \,\&\, \text{BOR} \,\&\, \text{UML} \,\&\, \text{TOG}.$$

Выводы

Таким образом, в данной статье:

– предложена уточненная классификация методов реализации многоверсионного ПО, основанная на учете этапов жизненного цикла ПО;

– предложено разделение факторов, определяющих многоверсионность ПО, на внешнюю и внутреннюю многоверсионность;

– предложен графовый метод формирования версий многоверсионного проекта ПО.

Кроме того, приведен типовой вариант многоверсионного проекта ПО.

Дальнейшим развитием в данном направлении является разработка формальных методов оценки качества сгенерированных версий при помощи метрик многоверсионности, определяемых на основе ярусного графа множества решений.

Литература

1. Головкин Б.А. Многовариантное программирование и его применение. // Автоматика и телемеханика. – 1986. - № 7. – С. 5-39.

2. Preckshot G.G. Method for Performing Diversity and Defense-in-Depth Analyses of Reactor Protection Systems // Lawrence Livermore National Laboratory, California USA December, 1994. – 45 p.

3. Харченко В.С., Пискачева И.В., Скляр В.В. Метрики диверсности: Классификация, анализ и применение для оценки надежности и безопасности компьютерных систем управления // Открытые информационные и компьютерные интегрированные технологии. – Х.: НАКУ «ХАИ», 2001. – № 9. – С. 194-214.

Поступила в редакцию 2.03.2006

Рецензент: д-р техн. наук, проф. Г.Н. Жолткевич, Харьковский национальный университет им. В.Н. Каразина, Харьков.