УДК 004.94
DOI 10.21685/2072-3059-2016-3-8

*Ю. И. Евсеева, А. С. Бождай*

# МЕТОД СТРУКТУРНО-ПАРАМЕТРИЧЕСКОГО СИНТЕЗА АДАПТИВНЫХ ПРОГРАММНЫХ КОМПОНЕНТОВ ВИРТУАЛЬНОЙ ОБРАЗОВАТЕЛЬНОЙ СРЕДЫ[1]

**Аннотация**.

*Актуальность и цели.* Современная виртуальная образовательная среда является сложной системой, включающей как методические и дидактические, так и информационные и программные компоненты. Поскольку к числу программных компонентов могут относиться различные интерактивные тренажеры и тестирующие программы, обладающие адаптивными свойствами, актуальным является вопрос разработки методов синтеза подобных компонентов. Целью работы является разработка метода структурно-параметрического синтеза адаптивных программных компонентов виртуальной образовательной среды.

*Материалы и методы.* Для решения поставленной задачи был применен математический аппарат теории графов и гиперграфов, методы морфологического анализа и синтеза сложных систем, методы моделирования изменчивости программных систем.

*Результаты.* Разработан метод структурно-параметрического синтеза адаптивных программных компонентов виртуальной образовательной среды, основанный на совместном применении технологии моделирования изменчивости и теории графов.

*Выводы.* Разработанный метод позволяет значительно сократить ресурсозатраты на разработку адаптивных программных компонентов виртуальной образовательной среды, а также увеличить их жизненный цикл.

**Ключевые слова**: адаптивное обучающее программное обеспечение, ориентированный гиперграф, автоматизированное проектирование, моделирование изменчивости, виртуальная образовательная среда.

*Yu. I. Evseeva, A. C. Bozhday*

# THE METHOD OF STRUCTURAL AND PARAMETRIC SYNTHESIS OF ADAPTIVE SOFTWARE COMPONENTS OF THE VIRTUAL LEARNING ENVIRONMENT

**Abstract**.

*Background.* The modern virtual learning environment is a complex system that includes didactic and methodical components, as well as information and software components. As program components may include various interactive simulators and testing programs with adaptive properties, an important question is the development of methods of synthesis of these components. The aim of the work is to develop a method of structural and parametric synthesis of adaptive software components of the virtual learning environment.

*Materials and methods*. To solve this problem the authors used the mathematical apparatus of the graphs and hypergraphs theory, methods of morphological analysis

[1] Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 15-07-01553.

and synthesis of complex systems and methods of variability modeling of software systems.

*Results*. The researchers have developed a method of structural and parametric synthesis of adaptive software components of the virtual learning environment, based on the application of variability modeling techniques and the graph theory.

*Conclusions*. The developed method allows to significantly reduce resource consumption on the development of adaptive software components of the virtual learning environment, and to increase their life cycle.

**Key words**: adaptive educational software, oriented hypergraph, computer-aided design, variability modeling, virtual learning environment.

## Введение

Современная виртуальная образовательная среда представляет собой информационно-образовательное пространство, построенное с помощью интеграции традиционных информационных носителей и компьютерных технологий [1]. К последним относятся распределенные базы данных, виртуальные библиотеки, оптимизированный учебно-методический комплекс (УМК), расширенный аппарат дидактики и т.д. В наши дни достаточно серьезное развитие получают интерактивные компоненты виртуальной образовательной среды, способные адаптироваться к индивидуальным особенностям учащегося. К числу компонентов подобного рода относятся различные обучающие программы, среди которых могут быть достаточно сложные в разработке виртуальные тренажеры. В связи с этим актуальность обретает вопрос разработки метода синтеза обозначенных интерактивных и адаптивных компонентов виртуальной образовательной среды.

## 1. Математическая модель адаптивного программного компонента виртуальной образовательной среды

В основе предлагаемого метода лежит математическая модель изменчивости, базирующаяся на гиперграфовом представлении диаграммы характеристик или структуры программного компонента [2, 3]. Диаграмма характеристик представляет собой модифицированное И/ИЛИ-дерево, которое, помимо стандартных взаимоотношений между структурными элементами системы по типам И и ИЛИ, включает в себя также:

1) отношение множественного ИЛИ;

2) отношение обязательного включения дочернего компонента;

3) отношение опционального включения дочернего компонента;

4) отношения, связывающие между собой компоненты, не являющиеся по отношению друг к другу родительскими или дочерними.

Предлагаемая математическая модель должна обеспечить формализацию следующих процессов и задач:

1) ручной и автоматический синтез отдельных состояний работы программного компонента;

2) верификация структурных решений, полученных в результате ручного или автоматического синтеза состояний работы программного компонента;

3) переключение между состояниями программного компонента в процессе его выполнения с учетом ряда показателей (таких, например, как аппа-

ратная конфигурация устройства, состояние среды выполнения, поведение пользователя и др.);

4) автоматическая генерация состояний не только в процессе проектирования программы, но и в процессе выполнения с учетом показателей, указанных в предыдущем пункте;

5) учет обширного списка разнородных объектов, составляющих программную систему: моделей, программных функций, переменных и т.д. (все перечисленные объекты подлежат реорганизации в процессе реализации адаптивного поведения);

6) обеспечение свойства инвариантности программного компонента к предметной области;

7) обеспечение прозрачности алгоритмов самоадаптации для конечного пользователя.

С учетом перечисленных требований математическая модель адаптивного программного компонента виртуальной образовательной среды имеет вид

$$M = (F, S, X),$$

где $F$ – гиперграфовое представление общей структуры программы, изначально определенной с помощью диаграммы характеристик; $S = \{S_1, S_2, ..., S_k\}$ – конечное множество конфигураций (подмножеств элементов) диаграммы характеристик, каждая из которых является описанием определенного состояния адаптивного программного компонента (подграфом исходного гиперграфа); $X$ – матрица переходов между состояниями программного компонента.

Общая структура программы представляет собой набор основных компонентов программы и взаимосвязей между ними, определенный в качестве морфологического множества, для описания которого используется диаграмма характеристик. Пример такой общей структуры представлен на рис. 1.

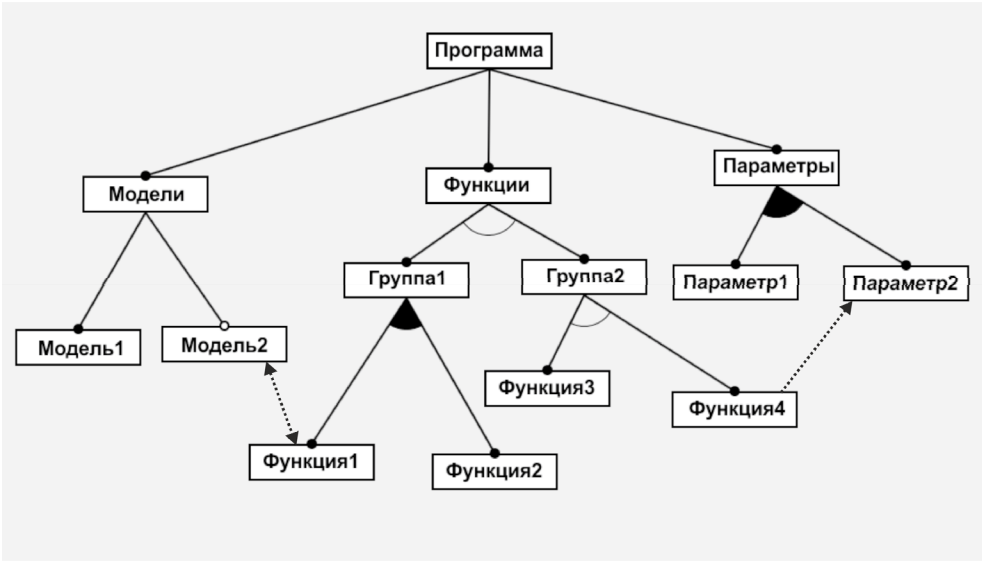


Рис. 1. Общая структура некоторой адаптивной обучающей программы

При преобразовании диаграммы характеристик в гиперграфовую форму необходимо руководствоваться следующими правилами [4]:

1) множество характеристик модели будет отображено на множество вершин соответствующего гиперграфа $Features \to V$;

2) множество взаимоотношений модели будет отображено на множество гиперребер, соединяющих вершины гиперграфа (характеристики модели) $\mathrm{Re}lations \to E$.

Гиперграфовое определение диаграммы характеристик будет иметь следующую форму:

$$F = (N, E, \Delta, \Psi),$$

где $N = \{N_1, N_2, ..., N_n\}$ – конечное множество вершин графа (характеристик исходной модели); $E = \{E_1, E_2, ..., E_m\}$ – конечное множество ребер графа (взаимоотношений исходной модели); $\Delta \in N$ – корневая вершина графа (корневая характеристика модели); $\Psi : E \to M, M \subset N \times N$ – функция маркировки, присваивающая значение мощности $mv(E_i) = (\min, \max) \in M$, где $M$ – множество значений мощности диаграммы, $N$ – множество вершин гиперграфа, каждому ребру $E_i$, так что $\min, \max \in \mathrm{Z} \wedge \min \geq 0, \quad \max > 0,$ $\min \leq \max \wedge \max \leq q_i$, где $\mathrm{Z}$ – множество целых чисел.

Под значением мощности понимается минимальное и максимальное количество вершин из головного множества гиперребра (т.е. множества вершин, в которые «входит» ребро), которые могут быть включены в конфигурацию описываемой гиперграфом модели характеристик (подграф исходного гиперграфа).

Матрица переходов содержит элементы $x_{ij}$, каждый из которых представляет собой интервал $[a; b)$ либо пустое множество $\varnothing$, причем $\bigcap x_{ij} = \varnothing$ и $\bigcup x_{ij} = [0; \infty)$, $j = 1...k, i$, фиксированы. Из состояния $p$ осуществляется переход в состояние $q$, если показатель работы пользователя с программным компонентом $C \in x_{pq}$. Если $x_{pq} = \varnothing$, то переход из состояния $p$ в состояние $q$ невозможен.

Как видно, предложенная математическая модель изменчивости является достаточно наглядным и удобным инструментом синтеза структуры адаптивной программной системы. Элементы данной модели служат для описания пространства возможных состояний программы.

## 2. Метод структурно-параметрического синтеза адаптивных программных компонентов

Структурно-параметрический синтез – это процесс, целью которого является определение структуры объекта и нахождение значений параметров составляющих ее элементов таким образом, чтобы были удовлетворены условия задания на синтез (в большинстве случаев – технического задания). Методы структурно-параметрического синтеза активно используются в автоматизированном проектировании, CASE-средствах и CALS-технологиях. В свою очередь подходы к структурно-параметрическому синтезу использу-

ют технологии программирования, инженерии знаний, теории проектирования, искусственного интеллекта и т.д. В основе процесса структурно-параметрического синтеза систем любого назначения и сложности лежат принципы системного подхода [5, 6].

В предлагаемом методе структурно-параметрического синтеза можно выделить 5 этапов: этап морфологического анализа системы, этап теоретико-множественного преобразования структуры системы, этап генерации состояний, этап верификации системных конфигурации, этап определения взаимосвязей между состояниями. Рассмотрим каждый этап подробнее.

**1. Этап морфологического анализа адаптивного программного компонента**. В ходе выполнения этапа разрабатывается общая структура программы в форме диаграммы характеристик. Для построения диаграммы нужно выделить основные компоненты разрабатываемой программной системы. Далее необходимо определить возможные варианты реализации компонентов, а также оценить их совместимость друг с другом. В последнюю очередь выполняется определение пространства параметров элементов диаграммы. Расчет множества возможных параметров элемента является задачей, специфической для каждой конкретной программы и не рассматривается в обобщенном описании метода. Данный этап является частью процесса структурно-параметрического синтеза, его результатом является множество возможных структур адаптивного программного компонента, определенное в форме диаграммы характеристик, и множество возможных параметров элементов.

**2. Этап теоретико-множественного преобразования общей структуры программного компонента**. Результатом выполнения данного этапа является гиперграфовое представление полученной на предыдущем этапе общей структуры программы. Таким образом, заданное графически множество возможных структур приобретает формализованное представление, на основе которого обеспечивается проведение последующих этапов структурно-параметрического синтеза адаптивной программной системы.

**3. Этап генерации состояний программного компонента (этап формирования системных конфигураций)**. Результатом выполнения этапа является множество возможных конфигураций исходной диаграммы характеристик, описывающей общую структуру программы. В формализованном виде конфигурации представлены как подграфы исходного гиперграфа, являющегося отображением базовой диаграммы характеристик. В отличие от предыдущих этапов, этап осуществляется не только в процессе разработки программной системы, но и в процессе ее выполнения. В ходе разработки программной системы пользователем вручную определяются необходимые структурные конфигурации, являющиеся базовыми для синтезируемой структуры программы. Во время выполнения на основе имеющихся базовых конфигураций формируются производные промежуточные конфигурации. Рассмотренный процесс является частью более широкого процесса адаптации в ходе выполнения программной системы.

Рассмотрим более подробно процедуру генерации промежуточных состояний адаптивного программного компонента. Пусть $X$, $Y$ – два состояния программы, которым соответствуют одноименные непустые множества вершин гиперграфа, представляющего общую структуру программной системы.

Введем понятие расстояния между состояниями системы:

$$D(X,Y) = 1 - \frac{|X \cap Y|}{max(|X|,|Y|)}.$$

Оно обладает следующими свойствами:

1) $D(X,Y) = D(Y,X)$;
2) $D(X,X) = 0$;
3) $D(X,Y) = 1$, только если $X \cap Y = \varnothing$;
4) $D(X,Y) \in [0;1]$.

Понятие расстояния можно интерпретировать следующим образом: чем больше у состояний $X$ и $Y$ общих вершин, тем ближе состояния друг к другу; если состояния содержат одни и те же вершины, то расстояние между ними равно нулю; если состояния не содержат ни одной общей вершины, то расстояние между ними равно единице.

Пусть требуется сгенерировать состояние $Z$, промежуточное по отношению к $X$ и $Y$. Тогда для состояния должно выполняться следующее условие:

$$D(Z,Y) = D(Z,X) = \frac{D(X,Y)}{2}.$$

Таким образом, состояние $Z$ будет находиться «между» состояниями $X$ и $Y$ на одинаковом расстоянии от них, равном половине расстояния между $X$ и $Y$. Такое состояние можно получить, если взять, например, половину вершин из состояния $X$, а другую половину – из состояния $Y$.

Если требуется сгенерировать $N \geq 1$ последовательных промежуточных состояний $Z_i$, $i = 1,2,...,N$, то для состояний должны выполняться условия

$$D(Z_i,X) = \frac{i}{(N+1)} \cdot D(X,Y), \quad D(Z_i,Y) = \frac{(N+1-i)}{(\mathrm{N}+1)} \cdot D(X,Y).$$

Таким образом, промежуточные состояния будут расположены равномерно между состояниями $X$ и $Y$.

**4. Этап верификации системных конфигураций**. Основное назначение этапа – поиск и исправление ошибок в системных конфигурациях, описывающих отдельные состояния адаптивного программного компонента [6]. Ошибки могут возникать в результате нарушения правил диаграммы как в результате автоматической генерации состояния системы, так и в случае его формирования пользователем вручную. С этой целью используются специализированные алгоритмы на гиперграфах.

**5. Этап определения взаимосвязей между состояниями программного компонента**. Результатом выполнения этапа является синтезируемая структура программы. Для определения синтезируемой структуры необходимо задание матрицы переходов. Как именно задавать матрицу переходов – непосредственно пользователем или посредством промежуточных расчетов – вопрос, имеющий второстепенный характер.

Итоговую структуру адаптивной программной системы, полученной в результате синтеза, можно представить в виде ориентированного графа следующим образом:

$$D = (V, E, H),$$

где $V = S = \{S_1, S_2, .., S_k\}$ полностью соответствует переменной $S$ в определении математической модели и представляет собой множество возможных состояний программного компонента, каждое из которых является гиперграфом; $E = \{E_1, E_2, .., E_m\}$ – множество ориентированных ребер, связывающих между собой состояния; $H = \{H_1, H_2, .., H_n\}$ – множество характеристик ориентированных ребер.

На рис. 2 представлено графическое изображение некоторой синтезированной структуры программного компонента в форме графа.

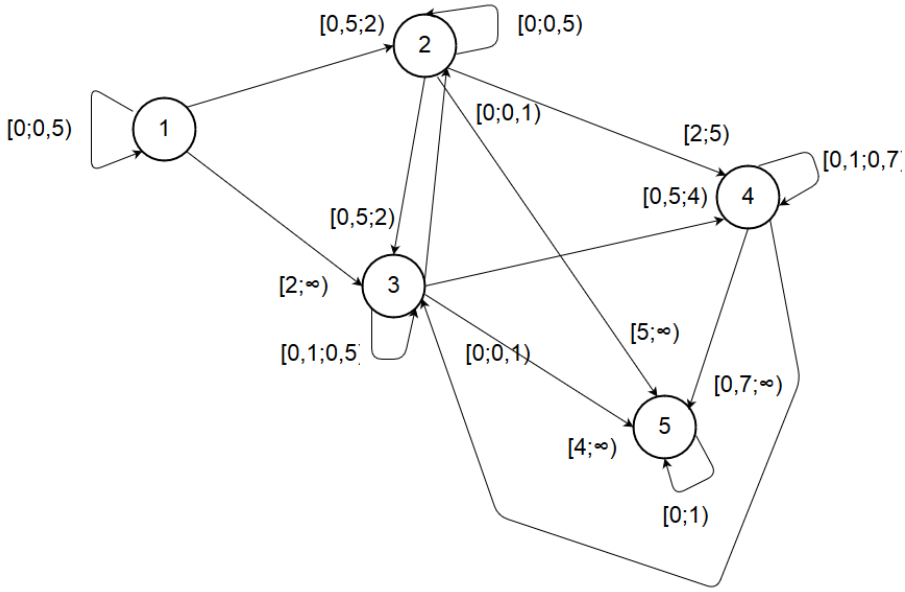


Рис. 2. Графовое представление синтезируемой структуры программного компонента

Представление синтезируемой структуры программы в форме ориентированного графа позволит применять методы дискретной математики для решения ряда задач, таких как поиск кратчайшего пути или циклов. Данные задачи могут быть актуальны при построении программного компонента определенного типа.

## Заключение

Предложенный метод структурно-параметрического синтеза адаптивных программных компонентов виртуальной образовательной среды позволяет формализовать сложную математическую процедуру задания изменчивости наглядным, простым и интуитивно понятным образом с использованием средств визуального проектирования. Использование метода на практике позволит увеличить жизненный цикл обучающих систем и снизить ресурсо-

затраты на их создание, поддержать мобильность электронного образования, улучшить адаптивные свойства обучающего программного обеспечения.

### *Список литературы*


1. **Пелюшенко, А. В.** Обучающие среды и интеллектуальные обучающие системы: возможности использования в образовательном процессе / А. В. Пелюшенко // Известия Волгоградского государственного технического университета. – 2006. – № 8. – С. 48–50.
2. **Younis, O.** Systems variability modeling: a textual model mixing class and feature concepts / O. Younis, S. Ghoul, M. H. Alomari // International Journal of Computer Science & Information Technology. – 2013. – № 5. – P. 127–139.
3. **Baresi, L.** Dynamically evolving the structural variability of dynamic software product lines / L. Baresi, C. Quinton // Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. – New York : ACM, 2015. – P. 57–63.
4. **Schobbens, P. E.** Feature diagrams: a survey and formal semantics / P. E. Schobbens, P. Heymans, J. C. Trigaux // 14th IEEE International Requirements Engineering Conference (RE'06). – Washington : IEEE Computer Society, 2011. – P. 139–148.
5. **Кумунжиев, К. В.** Теория систем и системный анализ : учеб. пособие / К. В. Кумунжиев. – Ульяновск : УлГУ, 2003. – 240 с.
6. **Финогеев, А. А.** Оценка информационных рисков в распределенных системах обработки данных на основе беспроводных сенсорных сетей / А. А. Финогеев, А. Г. Финогеев, И. С. Нефедова // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2016. – № 2 (38). – С. 49–60.


### *References*


1. Pelyushenko A. V. *Izvestiya Volgogradskogo gosudarstvennogo tekhnicheskogo universiteta* [University proceedings. Volga region. Engineering sciences]. 2006, no. 8, pp. 48–50.
2. Younis O., Ghoul S., Alomari M. H. *International Journal of Computer Science & Information Technology*. 2013, no. 5, pp. 127–139.
3. Baresi L., Quinton C. *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. New York: ACM, 2015, pp. 57–63.
4. Schobbens P. E., Heymans P., Trigaux J. C. *14th IEEE International Requirements Engineering Conference (RE'06)*. Washington: IEEE Computer Society, 2011, pp. 139–148.
5. Kumunzhiev K. V. *Teoriya sistem i sistemnyy analiz: ucheb. posobie* [The theory of suystems and system analysis: tutorial]. Ulyanovsk: UlGU, 2003, 240 p.
6. Finogeev A. A., Finogeev A. G., Nefedova I. S. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2016, no. 2 (38), pp. 49–60.



***Евсеева Юлия Игоревна***
ассистент, кафедра систем автоматизированного проектирования, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: shymoda@mail.ru

***Evseeva Yuliya Igorevna***
Assistant, sub-department of CAD systems, Penza State University (40 Krasnaya street, Penza, Russia)

***Бождай Александр Сергеевич***
доктор технических наук, профессор, кафедра систем автоматизированного проектирования, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: bozhday@yandex.ru

***Bozhday Aleksandr Sergeevich***
Doctor of engineering sciences, professor, sub-department of CAD systems, Penza State University (40 Krasnaya street, Penza, Russia)

---

УДК 004.94
**Евсеева, Ю. И.**

**Метод структурно-параметрического синтеза адаптивных программных компонентов виртуальной образовательной среды** / Ю. И. Евсеева, А. С. Бождай // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2016. – № 3 (39). – С. 84–92. DOI 10.21685/2072-3059-2016-3-8