ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

УДК 004.94

Ю. И. Евсеева

АЛГОРИТМ ВЕРИФИКАЦИИ СТРУКТУРЫ ТРЕХМЕРНОГО АДАПТИВНОГО ПРИЛОЖЕНИЯ

Аннотация.

Актуальность и цели. Вопрос создания автоматизированной системы синтеза трехмерных адаптивных программ (ТРАП) довольно актуален в настоящее время. Данная система особенно полезна в сфере образования, так как позволит преподавателям, не имеющим навыков программирования, самостоятельно разрабатывать тренажеры и обучающие программы, использующие преимущества трехмерной графики и возможность адаптации к ученику. Однако так как такая система подразумевает работу с неопытным пользователем, необходимо введение методов верификации составленных им алгоритмических решений. Цель данного исследования – разработка таких методов на основе математической модели ТРАП, используемой в предлагаемой системе.

Материалы и методы. Для решения поставленной задачи был применен математический аппарат теории графов, в частности гиперграфовое представление структуры ТРАП.

Результаты. Разработанный алгоритм является модификацией алгоритма полного обхода ориентированного гиперграфа. Формализованы основные процедуры работы с частичными конфигурациями – добавление и удаление произвольных вершин, завершение частичной конфигурации и преобразование ее в полноценную путем добавления недостающих вершин из головных множеств гиперребер. Разработанный алгоритм также позволяет эффективно обрабатывать последовательность конфигураций. Применение алгоритма позволяет находить и исправлять ошибки, допускаемые непрофессиональным пользователем при проектировании трехмерной адаптивной программы.

Выводы. Разработанный алгоритм позволяет не только обнаружить неточности, допущенные пользователем в ходе проектирования логической структуры, но и внести в некорректную конфигурацию соответствующие исправления.

Ключевые слова: трехмерное адаптивное приложение, ориентированный гиперграф, проектирование программного обеспечения, моделирование изменчивости, итерационный алгоритм верификации.

Yu. I. Evseeva

STRUCTURE VERIFICATION ALGORITHM OF A THREE-DIMENSIONAL ADAPTIVE APPLICATION

Abstract.

Background. The problem of creating an automated synthesis system of three-dimensional adaptive applications (TDAA) is quite relevant now. Such system is especially useful in education, because it allows teachers, who do not have programming skills, to develop their own simulators and training programs. Such programs may use the advantage of three-dimensional graphics and the ability to adapt to a student. However, since such system involves working with an inexperienced user, it is necessary to develop methods of user's algorithmic solutions verification. The aim of this work is to develop such methods.

Materials and methods. To solve this problem the author applied the mathematical apparatus of the graph theory, in particular, the hypergraph representation of software structure.

Results. The developed algorithm is a modification of the full crawl oriented hypergraph algorithm. The researcher formalized basic procedures for working with a partial configuration - adding and removing an arbitrary vertex, completion of the partial configuration and its transformation into a full configuration by adding the missing vertex from the head sets of hyperedges. The developed algorithm also allows to effectively handle a sequence of configurations. The use of the algorithm allows to find and correct mistakes made by non-professional users in design of three-dimensional adaptive programs.

Conclusions. The developed algorithm makes it possible not only to detect inaccuracies in software logical structure design, but also to revise an incorrect configuration.

Key words: three-dimensional adaptive application, oriented hypergraph, software design, variability modeling, verification iterative algorithm.

Введение

В настоящее время все большее распространение получают различные виртуальные тренажеры, компьютерные обучающие программы, системы виртуальной реальности, сложные компьютерные игры. Все чаще программы данного типа используют трехмерную графику (медицинские тренажеры и тренажеры управления техникой) и стратегии адаптации к пользователю (выработка индивидуальной траектории обучения). Все это обусловливает актуальность вопроса о создании автоматизированной системы синтеза трехмерных адаптивных приложений (ТРАП), предполагающую построение программ такого класса неопытным пользователем в максимально короткие сроки. Например, подобная система будет полезна преподавателям, желающим оперативно разработать обучающее приложение без необходимости написания сложного программного кода. Однако пользователь в процессе разработки может допускать ошибки, некоторые из которых могут привести к тому, что разрабатываемая программа будет нерабочей. Для решения данной проблемы необходимо введение методов проверки корректности разрабатываемой программы.

1. Структура ТРАП и математическая модель ТРАП

Для наглядного представления логической структуры ТРАП предлагается использование методов моделирования изменчивости (variability modeling [1]), в частности диаграмм характеристик (feature diagrams [2]).

С точки зрения используемого подхода под структурой ТРАП понимается схематичное описание набора основных, наиболее важных со структур-

ной точки зрения компонентов приложения (моделей, функций и т.п.) и взаимосвязей между ними.

Структура ТРАП в таком случае основана на трех основных категориях объектов [3]:

1. Трехмерные модели.
2. Основные программные функции.
3. Параметры.

На рис. 1 представлен фрагмент модели характеристик, изображающий взаимосвязи между трехмерными моделями в структуре программы.

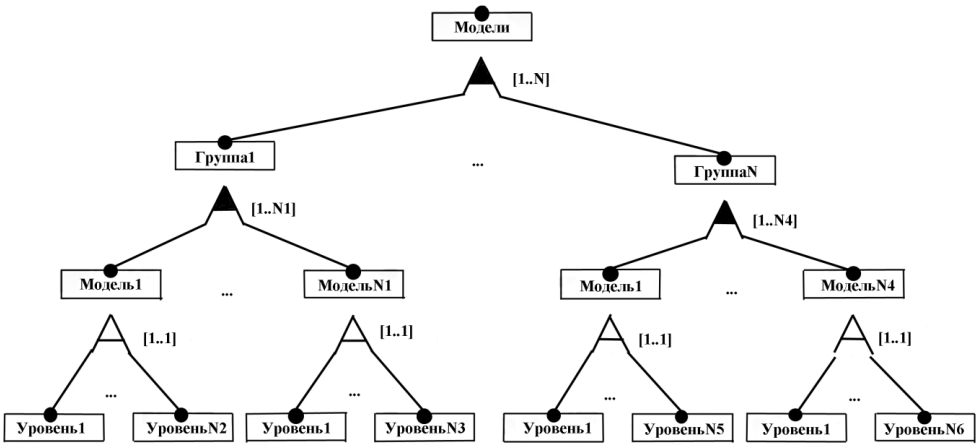

Рис. 1. Фрагмент структуры ТРАП

Представление структуры ТРАП в форме модели характеристик позволяет пользователю «визуально» проектировать структуру приложения на каждом этапе его работы, определяя тем самым его структурную изменчивость. Тем самым осуществляется решение сразу двух значимых для построения и эффективного использования предлагаемой системы задач – задачи реализации адаптивного поведения в программах, относящихся к широкому классу ТРАП, не ограниченному конкретной предметной областью, и задачи реализации достаточно дружественного по отношению к непрофессиональному пользователю интерфейса для задания такого поведения.

Математическая модель ТРАП при этом имеет вид

$$M = \{F, S, X, D(C, X)\}, \quad (1)$$

где F соответствует описанию структуры ТРАП в форме гиперграфа. Под гиперграфом в математике принято понимать обобщение графа, в котором каждым ребром могут соединяться не только две вершины, но и любые подмножества вершин [4]. Необходимо также отметить, что используемый гиперграф является F-графом, т.е. графом, из любой вершины которого может исходить только одно ребро; $S = \{S_1, S_2, S_3, ..., S_k\}$ – множество подграфов исходного графа F, каждый из которых соответствует определенной конфигурации модели характеристик. Каждая конфигурация модели характеристик соответствует определенному шагу работы адаптивного приложения; X – матрица переходов между шагами работы приложения; $D(C, X)$ – функция,

задающая динамическую изменчивость, в которой C – показатель качества прохождения шага работы приложения.

Структура ТРАП в виде гиперграфа описывается следующим образом:

$$F = (N, E, \Delta, \Psi), \quad (2)$$

где $N = \{n_1, n_2, n_3, ..., n_n\}$ – конечное множество характеристик (узлов); $E = \{E_1, E_2, E_3, ..., E_m\}, E_i \subseteq N \ \forall \ i = 1,...,m \ \wedge |H(E_i)| = q_i$ – множество гиперребер, каждое из которых связано с головным множеством (множество узлов, в которые «входит» гиперребро) мощностью q_i; $\Delta \in N$ – корневая характеристика модели характеристик. Корневая характеристика является единственным узлом гиперграфа, который не связан ни с одним головным множеством, т.е. в него не «входит» ни одно гиперребро). По отношению к корневому множеству R применимо:

– $R \subseteq N \wedge R = \{\Delta\}$;

– $BS(\Delta) = \emptyset$ (множество гиперребер, чьи головные множества включают в себя вершину Δ, является пустым);

– $BS(n) \neq \emptyset, \ \forall n \in (N - R)$ (для любой вершины гиперграфа, за исключением корневой, множество гиперребер, чьи головные множества включают в себя данную вершины, не является пустым);

$\Psi : E \to MV, \ MV \subset N \times N^*$ – функция маркировки, которая присваивает значение мощности $mv(E_i) = (\min, \max) \in MV$ (MV – множество всех значений мощности модели, N – конечное множество вершин) каждому F-гиперребру E_i, так что $\min, \max \in \mathbf{Z} \wedge \min \geq 0, \ \max > 0, \ \min \leq \max \wedge \max \leq q_i$, где Z – множество целых чисел. Под значением мощности понимается число вершин из головного множества гиперребра, которое необходимо выбрать, чтобы сформировать корректную конфигурацию модели. В данном случае речь идет о диапазоне значений, так как отношения в диаграмме характеристик (И, ИЛИ) могут связывать различное, в зависимости от выбора пользователя, число характеристик. Таким образом, данная функция устанавливает гиперребру значение его мощности в диапазоне от min до max. Данное значение всегда будет представлять собой целое число, положительное или равное нулю, или равно нулю и не превышающее мощность головного множества q_i.

Матрица переходов имеет размерность $k \times k$, где k – число конфигураций, номера столбцов и строк соответствуют номерам конкретных шагов работы ТРАП. Каждый элемент $X_{ij} \forall i = 1,..., \ k \ \forall \ j = 1,.., \ k$ представляет собой некое числовое значение, которое необходимо для определения того, на какой именно шаг, имеющий номер, соответствующий номеру столбца, перейдет программа с текущего шага, имеющего номер, соответствующий номеру строки.

Функция динамической изменчивости $D(C, X)$ необходима для определения того, на какой шаг должно будет перейти приложение в процессе своей работы. Принимая на вход значения показателя качества прохождения C и матрицу переходов, функция возвращает номер следующего шага.

За счет использования предложенной математической модели адаптивного приложения возможна реализация двух типов изменчивости (адаптивности) – структурной (различия в наборе компонентов, составляющих ТРАП на

определенном шаге его работы), задаваемой пользователем, проектирующим ТРАП посредством работы с множеством конфигураций S, и динамической, реализуемой за счет использования функции $D(C, X)$ в процессе выполнения программы.

2. Описание предлагаемого алгоритма

Так как конфигурация структурной модели полностью определяется действиями пользователя, необходимо введение дополнительных средств проверки корректности.

При составлении конфигурации структурной модели, соответствующей конкретному шагу работы ТРАП, пользователь выбирает ряд характеристик, необходимых для наиболее полного описания функционирования программы на данном шаге.

Описываемый алгоритм в цикле проходит по каждой конфигурации, ищет и исправляет допущенные пользователем ошибки. Получая на каждом шаге множество выбранных пользователем характеристик модели (вершин гиперграфа), для каждой выбранной вершины сначала выстраиваются допустимые пути от нее до корневой вершины (построение частичной конфигурации). Таким образом осуществляется проверка на то, не включил ли пользователь в конфигурацию вершины-потомки, не включив при этом соответствующие вершины-родители. Затем для каждого отношения (гиперребра) частичной конфигурации алгоритм осуществляет проверку на то, была ли достигнута его минимальная мощность, т.е. достаточное ли количество характеристик было выбрано пользователем.

В общем виде предлагаемый алгоритм верификации выглядит следующим образом:

1. Начало цикла проверки конфигураций. Выбор очередной конфигурации $S_i = (N_{S_i}, E_{S_i})$, где $i = 1,.., k$, $N_{S_i} \subseteq N$, $E_{S_i} \subseteq E$, из множества составленных пользователем на основе гиперграфового представления структуры ТРАП F.

2. Этап генерации частичных конфигураций. Под частичной конфигурацией понимается конфигурация, в которой из каждой включенной в нее вершины существует корректный путь до корневой вершины, но при этом минимально допустимое количество вершин из головного множества гиперребра может быть еще не выбрано (минимальная мощность гиперребра не достигнута). Имея в качестве входных данных множество выбранных характеристик N_{S_i}, на данном этапе необходимо найти корректный путь $P_{\Delta n}$ между каждой характеристикой $n \in N_{S_i}$ и корневой характеристикой Δ.

3. Этап завершения частичных конфигураций. Частичная конфигурация должна пройти данный этап, если в ней есть одно или более гиперребро, чье хвостовое множество (множество вершин, из которых «исходит» гиперребро) включено в нее, но при этом число включенных характеристик из головного множества данного гиперребра меньше минимальной мощности отношения. Какие именно характеристики должны быть выбраны – вопрос, на который нельзя дать однозначного ответа, так как многое зависит от потребностей пользователя. Данный алгоритм будет выбирать характеристики из

головного множества гиперребра до тех пор, пока не будет достигнута минимальная мощность отношения.

4. Переход к первому пункту алгоритма.

Следует отметить, что в случае, если ход работы приложения разбит на более чем один шаг, нет смысла каждый раз заново вычислять множество частичных конфигураций, так как в силу определенной схожести смежных шагов различия между множествами S_i и S_{i+1} будут минимальны. По этой причине более экономным в отношении ресурсов компьютера подходом будет вычисление для каждой смежной пары множеств S_i и S_{i+1} множества добавленных на $i+1$ шаге вершин $N_{S_i+} = N_{S_{i+1}} - N_{S_i}$ и множества удаленных вершин $N_{S_i-} = N_{S_i} - N_{S_{i+1}}$. При этом полноценную процедуру вычисления множества частичных конфигураций достаточно будет провести только на шаге $i = 0$, вычисление же данного множества на каждом шаге $i+1$ будет осуществляться, во-первых, за счет удаления из результатов, полученных на предыдущем шаге, вершин, принадлежащих множеству N_{S_i-}, а во-вторых, за счет добавления в формируемое множество конфигураций вершин из N_{S_i+}.

Ниже приведено описание алгоритма в форме псевдокода, сделанное с учетом данного замечания. Алгоритм принимает на вход множество конфигураций S, а на выходе формирует множество корректных конфигураций $C = \{C_1, C_2, C_3, ..., C_k\}$.

```
процедура проверкаКонфигураций(S,C)
  s:=выбратьПервыйЭлемент(S);  // выбор конфигурации из списка
  N:=получитьВершины(S);// получить вершины конфигурации
  добавитьВершины(N,C1);  // добавить вершины в конфигурацию
   k:=получитьМощность(S); // получить число элементов в списке
  завершитьКонфигурацию(C1); // функция завершения конфигурации,
                                  // подробно рассмотрена ниже
   i:=2;
   S:=S-{s}; // удаление рассмотренной конфигурации из  списка
  пока  i ≤ k выполнять // цикл по конфигурациям
          s:=выбратьПервыйЭлемент(S);
          Nnew:=получитьВершины(s);
          Nplus :=Nnew  -N; // получение добавленных на новом шаге вершин
           Nminus:=N-Nnew;    // получение удаленных на новом шаге вершин
          Ci:=C(i-1);  // начало модификации множества вершин,
                    //полученного на предыдущем шаге
          добавитьВершины(Nplus, Ci);  // функция добавления вершин
                                        // в конфигурацию. Подробно
                                        // рассмотрена ниже
          удалитьВершины(Nminus, Ci);  // функция удаления вершин
                                        // из конфигурации. Подробно
                                        // рассмотрена ниже
          завершитьКонфигурацию(Ci);
           i=i+1; // переход к следующему шагу
  конец цикла // завершение цикла перебора конфигураций
конец процедуры
```

Имея обобщенную модель ТРАП $M = \{F, S, X, D(C, X)\}$, гиперграфовое представление ТРАП $F = (N, E, \Delta, \Psi)$, а также множество выбранных на конкретном шаге характеристик $N_{S_i} \subseteq N$, будем понимать корректную конфигурацию структурной модели на *i*-м шаге работы $C_i = (N_{C_i}, E_{C_i}, \Delta)$ как подграф ограниченной модели характеристик $C_F = (N, E', \Delta, \Psi')$. При этом под ограниченной моделью характеристик будем понимать модель характеристик, где:

- $E' = E \cup E_r \cup E_m$:
  - $E_r = \{r_1, r_2, r_3, ..., r_l\} \wedge r_i \in N \ \forall \ i = 1, ..., l$ – множество отношений включения (отношение, при котором включение одной характеристики, например модели, требует обязательного включения другой, например функции), смоделированных в форме гиперребер, для которых в целом справедливо $|H(r_i)| \geq 1$, где H – головное множество гиперребра;
  - $E_m = \{m_1, m_2, m_3, ..., m_p) \wedge m_j \in N \in j = 1, ..., p$ – множество отношений исключения (связанные данным отношением характеристики не могут быть частью одной конфигурации), смоделированных в форме гиперребер, для которых в целом справедливо $|H(m_j)| \geq 2 \ \wedge T(m_j) = \Delta$, где T обозначает хвостовое множество гиперребра;
- $\Psi' = \Psi \cup \Psi_r \cup \Psi_m$:
  - $\Psi_r : E_r \to MV$ – функция маркировки, ставящая в соответствие каждому гиперребру r_l пару $mv(r_l) = (\min, \max) \in MV$, где MV – множество всех значений мощности модели, так что $\min = |H(r_l)| = \max$. Когда отношение включения затрагивает две вершины, $mv(r_l) = (1,1)$;
  - $\Psi_m : E_m \to MV$ – функция маркировки, ставящая в соответствие каждому гиперребру m_p пару $mv(m_p) = (\min, \max) \in MV$, так что $mv(m_p) = (0,1)$.

3. Процедура добавления вершины в частичную конфигурацию

Данная процедура необходима для построения частичной конфигурации. Она осуществляет добавление вершины в конфигурацию, добавляя при этом (в случае необходимости) вершины-предки.

На основе имеющейся ограниченной модели характеристик $C_F = (N, E', \Delta, \Psi')$ множества выбранных характеристик $N_{S_i} \subseteq N$ (либо множества выбранных для добавления на текущем шаге вершин $N_{S_i+} \subseteq N$) и начальной конфигурации $C_0 = (N_0, E_0) / N_0 = \{\Delta\} \wedge E_0 = \emptyset$ (либо уже составленной на предыдущих шагах работы алгоритма конфигурации $C_{i-1} = (N_{C_{i-1}}, E_{C_{i-1}}, \Delta)$) предлагаемая процедура работает следующим образом:

1. Выбор и удаление характеристики $n \in N_{S_i}$ из множества N_{S_i}. Добавление $n \in N_{S_i}$ в конфигурацию C_i.

2. Для каждого гиперребра $e \in E$, чье головное множество содержит выбранную характеристику n и чья максимальная мощность еще не была достигнута, алгоритм выбирает хвостовое множество $T(e)$, если оно еще не выбрано, добавляет его к множеству N_{S_i} и совершает рекурсивный вызов (с обновленными множеством N_{S_i} и частичной конфигурацией C).

3. Если $N_{S_i} = \varnothing$, алгоритм завершает работу; в противном случае он продолжает свою работу с первого шага.

Аналогичным образом алгоритм работает и для множества $N_{S_i+} \subseteq N$.

Алгоритм, реализованный в форме псевдокода, выглядит следующим образом:

```
процедура добавитьВершины(Ns,C)
    пока  Ns≠Ø выполнять // цикл по множеству вершин
        // Выбор и удаление узла n ∈ Ns
        n:=выбратьПервыйЭлемент(Ns);
        Ns:=Ns-{n};
        // Отметить узел n ∈ Ns как выбранный
        включитьВконф (n,C);
        // Действительное число родителей
        числоРодителей:=0;
        для каждого e ∈ BS(n) и тип(e) ≠ ”ограничение” // цикл
                                                    //по гиперребрам
            // Выбрать гиперребро, если максимальная мощность
            //отношения еще не достигнута
            если использовано(e) < максМощность(e) тогда
                числоРодителей:=числоРодителей+1;
                использовано(e):= использовано(e)+1;
                // Включение хвостового множества
                // в конфигурацию
                если не включеноВконф(хвост(e), C) тогда
                    включитьВконф(хвост(e), C );
                конец условия
                Ns:=Ns ∪ хвост(e);
                добавитьВершины(Ns,C);
            конец условия
        конец цикла
    конец цикла
    // Завершение пути
    C:=void;
    // Проверка на некорректные конфигурации
    если числоРодителей=0 тогда
        возврат;
    конец условия
конец процедуры
```

4. Процедура удаления вершины из частичной конфигурации

Данная процедура необходима для построения частичной конфигурации. Она осуществляет удаление вершины из частичной конфигурации, удаляя при этом (в случае необходимости) ее потомков.

На основе набора данных, в который входят ограниченная модель характеристик $C_F = (N, E', \Delta, \Psi')$, множество выбранных для удаления на текущем шаге вершин $N_{S_i-} \subseteq N$ и составленная на предыдущих шагах работы алгоритма конфигурация $C_{i-1} = (N_{C_{i-1}}, E_{C_{i-1}}, \Delta)$, предлагаемая процедура работает следующим образом:

1. Характеристика $n \in N_{S_i-}$ выбирается и удаляется из множества N_{S_i-} и конфигурации C_i.

2. Для каждого структурного гиперребра $e \in E'$, чье хвостовое множество содержит выбранную характеристику n, алгоритм выбирает головное множество $H(e)$, удаляет его из конфигурации C_i, если это необходимо, и совершает рекурсивный вызов (с обновленными множеством N_{S_i-} и частичной конфигурацией C_i).

3. Если $N_{S_i-} = \varnothing$, алгоритм завершает работу; в противном случае он продолжает свою работу с первого шага.

Алгоритм, реализованный в форме псевдокода, выглядит следующим образом:

```
процедура удалитьВершины(Ns,C)
    пока Ns≠Ø выполнять // цикл по вершинам
        // Выбор и удаление узла n ∈ Ns
        n:=выбратьПервыйЭлемент(Ns);
        Ns:=Ns-{n};
        исключитьУзел(n,C);
        // исключение потомков узла из конфигурации
        // цикл по исходящим из удаленного узла ребрам
            для каждого e ∈ FS(n) и тип(e) ≠ ”ограничение”
            // рассмотрение головного множества ребра
             UN=голова(e);
            // проверка дочерних узлов из головного множества ребра.
            для каждого n1 ∈ UN // если узел не включен в конф.
                    //или у него есть включенные родители, то
                    // исключить его из дальнейшего рассмотрения
                        если не(включеноВКонф(n1,C) ) или
                                числоРодитВКонфиг(n1,C)>0
                                UN:=UN-{n1};
                        конец условия
                конец цикла
                Ns:=Ns ∪ UN;
                удалитьВершины(Ns,C);
            конец цикла
        конец условия
    конец цикла
конец процедуры
```

5. Процедура завершения частичной конфигурации

Данная процедура служит для завершения частичной конфигурации. Для каждого включенного в конфигурацию гиперребра (отношения модели

характеристик) она осуществляет проверку на достижение минимальной мощности отношения и в случае необходимости добавляет недостающие вершины в конфигурацию.

На основе составленной на *i*-м шаге частичной конфигурации $C_i = (N_{C_i}, E_{C_i}, \Delta)$ предлагаемая процедура работает следующим образом:

1. Генерируется список структурных гиперребер P, чье хвостовое множество уже включено в конфигурацию, но чья минимальная мощность еще не достигнута.

2. Для каждого гиперребра $e \in E_i$ алгоритм выбирает столько характеристик из головного множества, сколько требуется для достижения минимальной мощности, и выполняет рекурсивный вызов.

3. Если $P = \emptyset$, алгоритм завершает работу; в противном случае он продолжает свою работу с первого шага.

Алгоритм, реализованный в форме псевдокода, выглядит следующим образом:

```
процедура завершитьКонфигурацию (Ci)
    для каждого  e ∈ E и включеноВКонфиг(хвост(e)) выполнить
        P:=P ∪ {e};
    конец цикла
    пока  P ≠ Ø выполнять
        // Выбор и удаление p ∈ P
        p:=выбратьПервыйЭлемент(P);
        P:=P-{p};
        // Выбор характеристики из головного множества гиперребра
        // для достижения его минимальной мощности
        для каждой n ∈ голова(p) выполнять
            // Проверка, достигнута ли мин. мощность ребра
            //и не включена ли уже n в конф.
            если использовано(p) < минМощность(p) и
                не(включеноВКонф(n, Ci) тогда
                включитьВКонф(n, Ci);
                использовано(p):= использовано(p)+1;  // проверка
                                        //на корректность
                если корректно (Ci) тогда
                    // Поиск частичной конфигурации
                    //для отношения ограничения
                если мощность(получитьОграничения(Ci)) > 0 тогда
                Civ:= добавитьВершину(получитьОграничения(Ci), Ci);
                    иначе
                        Civ:= Ci;
                    конец условия
                    // Завершение конфигурации
                    для каждого c ∈ Civ выполнять
                        C'iv := завершить (c);
                        если корректно(C'i) тогда
                            Ci:= Ci ∪ { C'i};
                        иначе
                        // Конфигурация некорректна
```

```
                                        Ci:=пусто;
                                    конец условия
                                конец цикла
                            иначе
                                // Конфигурация некорректна
                                Ci:=пусто;
                            конец условия
                        конец условия
                    конец цикла
                конец цикла
конец процедуры
```

Заключение

В статье впервые предложен итерационный алгоритм верификации конфигураций математической модели ТРАП. Алгоритм основан на работе с представлением данной модели в форме ориентированного гиперграфа. Формализованы основные процедуры работы с частичными конфигурациями – добавление и удаление произвольных вершин, завершение частичной конфигурации и преобразование ее в полноценную путем добавления недостающих вершин из головных множеств гиперребер. Разработанный алгоритм также позволяет эффективно обрабатывать последовательность конфигураций. Применение алгоритма позволяет находить и исправлять ошибки, допускаемые непрофессиональным пользователем при проектировании трехмерной адаптивной программы.

Список литературы

1. **Apel, S.** Feature-Oriented Software Product Lines: concepts and implementation / S. Apel, D. Batory, C. Kästner, G. Saake. – Berlin : Springer, 2013. – С. 36–45.
2. **Czarnecki, K.** Feature Diagrams and Logics: There and Back Again / K. Czarnecki, A. Wasowski // SPLC '07 Proceedings of the 11th International Software Product Line. – Washington, USA : IEEE Computer Society Washington, 2007. – P. 68–70.
3. **Бождай, А. С.** Моделирование изменчивости в автоматизированном проектировании адаптивных обучающих программ / А. С. Бождай, Ю. И. Евсеева // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2015. – № 1 (33). – С. 5–18.
4. **Емеличев, В. А.** Лекции по теории графов / В. А. Емеличев, О. И. Мельников, В. И. Сарванов, Р. И. Тышкевич. – М. : Наука, 1990. – С. 90–97.

References

1. Apel S., Batory D., Kästner C., Saake G. *Feature-Oriented Software Product Lines: concepts and implementation*. Berlin: Springer, 2013, pp. 36–45.
2. Czarnecki K., Wasowski A. *SPLC '07 Proceedings of the 11th International Software Product Line*. Washington, USA: IEEE Computer Society Washington, 2007, pp. 68–70.
3. Bozhday A. S., Evseeva Yu. I. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2015, no. 1 (33), pp. 5–18.
4. Emelichev V. A., Mel'nikov O. I., Sarvanov V. I., Tyshkevich R. I. *Lektsii po teorii grafov* [Lectures on the graph theory]. Moscow: Nauka, 1990, pp. 90–97.

Евсеева Юлия Игоревна
аспирант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: shymoda@mail.ru

Evseeva Yuliya Igorevna
Postgraduate student, Penza State University (40 Krasnaya street, Penza, Russia)

УДК 004.94
Евсеева, Ю. И.

Алгоритм верификации структуры трехмерного адаптивного приложения / Ю. И. Евсеева // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2015. – № 4 (36). – С. 5–16.