

ТЕХНОЛОГІЇ РОЗРОБКИ ТА СУПРОВОДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.91

Кузнецова Ю.А.

Национальный аэрокосмический университет
им. Н.Е. Жуковского «ХАИ»

ВИЗУАЛИЗАЦІЯ УПРАВЛЯЮЩИХ АЛГОРИТМОВ В СИСТЕМАХ АВТОМАТИЗАЦІЇ ИСПЫТАНИЙ СЛОЖНЫХ ТЕХНИЧЕСКИХ КОМПЛЕКСОВ

Показано, что для повышения эффективности функционирования, снижения стоимости разработки и эксплуатации АСУ СТК, обработка информации в которых происходит в режиме реального времени, особое внимание необходимо уделять разработке управляющих алгоритмов с учётом временных ограничений. Показано, что на современном этапе развития АСУ СТК требуются новые подходы к разработке программного обеспечения, реализующего управляющий алгоритм технологического процесса испытаний, которые повысят эффективность контроля и управления оператора за ходом выполнения управляющего алгоритма, а также обеспечат возможность анализа его результатов. В работе предложен подход, основанный на применении компьютерной визуализации. Показана важность использования метафор визуализации для точного и ясного представления информации о ходе выполнения управляющего алгоритма. При разработке визуальных представлений в качестве основного подхода используется представление управляющего алгоритма на основе графовых моделей, отображающих поток управления. Опираясь на теорию множеств, разработана семантика управляющего алгоритма реального времени (УА РВ), которая стала основой для синтеза таких графовых моделей, как многовходовая и многовариантная модели управляющего алгоритма реального времени. Показано, что визуализация программного обеспечения, реализующего управляющий алгоритм, опирается на ряд стандартных шаблонов визуализации, удовлетворяющих требованиям системы и способных отобразить поведение управляющих алгоритмов.

Теория графов предлагает свои решения для задачи формального синтеза, анализа и верификации модели управляющего алгоритма. Проблема корректного визуального представления УА ТП испытаний решается с помощью формальной спецификации визуальных языков. В настоящей работе применяется текстовый метод спецификации синтаксиса, основанный на расширенной форме Бэкуса-Наура (EBNF).

Показано, що для підвищення ефективності функціонування, зниження вартості розробки та експлуатації АСУ СТК, обробка інформації в яких відбувається в режимі реального часу, особливу увагу необхідно приділяти розробці керуючих алгоритмів з урахуванням часових обмежень. Показано, що на сучасному етапі розвитку АСУ СТК потрібні нові підходи до розробки програмного забезпечення, що реалізує керуючий алгоритм технологічного процесу випробувань, які підвищать ефективність контролю та управління оператора за ходом виконання керуючого алгоритму, а також забезпечать можливість аналізу його результатів. У роботі запропоновано підхід, заснований на застосуванні комп'ютерної візуалізації. Показано важливість використання метафор візуалізації для точного і ясного подання інформації про хід виконання керуючого алгоритму. При розробці візуального подання в якості основного підходу використовується представлення керуючого алгоритму на основі графових моделей, що відображають потік управління. Спираючись на теорію множин, розроблена семантика керуючого алгоритму реального часу (КА РЧ), яка стала основою для синтезу таких графових моделей, як багатовходова і багатоваріантна моделі керуючого алгоритму реального часу. Показано, що візуалізація програмного забезпечення, що реалізує керуючий алгоритм, спирається на ряд стандартних шаблонів візуалізації, що задовольняють вимогам системи і здатних відобразити поведінку керуючих алгоритмів. Теорія графів пропонує свої рішення для задачі формального синтезу, аналізу та верифікації моделі керуючого алгоритму. Проблема коректного візуального представлення КА ТП випробувань вирішується за допомогою формальної специфікації візуальних мов. У цій роботі застосовується текстовий метод специфікації синтаксису, заснований на розширеній формі Бекуса-Наура (EBNF).

It is shown that to improve the efficiency of functioning, to reduce the development cost and operation of automated control systems (ACS) of complicated technical complexes (CTC), in which the information processing occurs in real time, a special attention should be paid to the development of supervised algorithms taking time limits into consideration. It is shown that at the present stage of CTC ACS development the new approaches are required to develop software that implements the supervised algorithm of the operational process of tests that will improve the operator's check and control efficiency for the progress of the supervised algorithm, as well as provide the ability to analyze its results. The paper proposes the approach based on the computer visualization. The importance of visualization metaphors is shown in this paper for an accurate and clear representation of information according to the progress of the supervised algorithm. Based on the theory of sets, we developed the real-time supervised algorithm (RT SA) semantics, which

became the basis for the synthesis of such graph models as a multi-input model and a multivariate model of the RT SA. It is shown that the visualization of software that implements the SA, is based on a range of standard visualization templates which satisfy the system requirements and displaying the behavior of the SAs. Graph theory offers its solutions for the problem of the formal synthesis, analysis and verification of the SA model. The problem of the correct visual representation of test technological process SA is solved with the formal specification of visual languages. This paper applies the text method of syntax specification, based on the Extended Backus-Naur Form (EBNF).

Ключевые слова: АСУ СТК, реальное время, управляющий алгоритм, программное обеспечение, визуализация, графовые модели, расширенная форма Бэкуса-Наура.

Введение

Необходимость автоматизации испытаний сложных технических комплексов (СТК) обусловлена сложностью объектов испытаний, разнообразием видов и типов испытаний, высокой информационной насыщенностью процесса испытаний, жесткими требованиями к качеству испытаний. Автоматизированная система управления (АСУ) испытаниями СТК относится к классу систем сбора данных и диспетчерского управления (ССДДУ), при этом специфика процессов испытаний требует реализации в этой системе функциональности, присущей лабораторным информационно-управляющим и информационным производственным системам.

Внутренняя структура ССДДУ обычно характеризуется многоуровневой иерархией и многообразием связей между различными элементами. Кроме того, системы данного класса целиком (или определенные её подсистемы) могут изменять своё состояние во времени по определенному закону. Любая сложная система служит для решения определенной комплексной задачи (или ряда задач).

Задачи, выполняемые ССДДУ, в общем виде можно сформулировать следующим образом: по полученным данным о процессе составить прогноз его дальнейшего хода и такой план управляющих воздействий (например, изменения режимов), чтобы в определенный момент состояние процесса отвечало некоторому экстремальному значению обобщенного критерия качества. В ССДДУ принципиально важной является работа в реальном масштабе времени.

Обычно задача синхронизации возлагается на отдельно выделяемый прибор, который осуществляет все управляющие и диспетчерские функции. В качестве такого прибора используется микропроцессор или электронно-вычислительная машина (ЭВМ). Управляющий микропроцессор должен иметь управляющие и информационные связи со всеми основными подсистемами. Для

обеспечения синхронной и бесперебойной работы всех подсистем он должен взаимодействовать с другими приборами по строго определенному алгоритму. Такой алгоритм называется *управляющим алгоритмом* (УА).

Учитывая высокие требования к надёжности и качеству управляющих алгоритмов СТК, а также большое число различных элементов, становится очевидной сложность управляющего алгоритма.

Программа, реализующая такой алгоритм, включает сотни тысяч, а иногда и миллионы команд. Естественно, что проектирование и разработка такого алгоритма является сложной и трудоёмкой задачей, возможной только с применением той или иной технологии автоматизации программирования. Одной из наиболее перспективных и удобных для пользователя технологий является технология визуального программирования, реализуемая в SCADA-системах.

Таким образом, программные средства, используемые в процессе автоматизации испытаний СТК относятся к классу систем реального времени (РВ). Одно из наиболее важных требований к ним – описание типового технологического (ТП) испытаний, включающее несколько тысяч действий (наборов операций), выполняемых в последовательностях, находящихся в сложных причинно-следственных связях. SCADA-системы не обеспечивают решение этой задачи и не позволяют выполнять отработку и верификацию ТП испытаний на имитационной математической модели. Всё это требует создания моделей и методов, обеспечивающих эффективное решение указанных задач.

Следовательно, на современном этапе развития АСУ СТК требуются новые подходы к разработке программного обеспечения, реализующего управляющий алгоритм технологического процесса испытаний, которые повысят эффективность контроля и управления оператором за ходом выполнения управляющего алгоритма, а также обеспечат возможность анализа его результатов.

Организация человеко-компьютерного взаимодействия с учётом человеческого фактора невозможна без применения компьютерной визуализации.

I. Особенности визуализации управляющих алгоритмов

В классическом определении под компьютерной визуализацией понимают методику перевода абстрактных представлений об объектах в геометрические образы, что даёт возможность исследователю наблюдать результаты компьютерного моделирования явлений и процессов.

Традиционно выделяют информационную и научную визуализацию, а также визуализацию ПО. В визуализации ПО в качестве модельного объекта обычно выступает либо программа, либо алгоритм. На рис. 1 представлена диаграмма Венна, отражающая взаимосвязь между различными подобластями визуализации ПО.

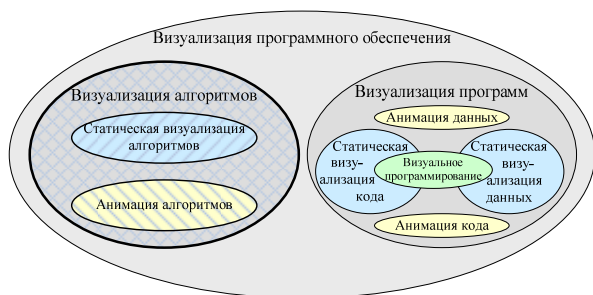


Рис. 1 Классификация средств визуализации по назначению: диаграмма Венна

По своей сути задача визуализации управляющих алгоритмов относится к области визуализации ПО, так как содержит в себе как элементы визуализации алгоритмов, так и визуализации программ. Для классификации, перечисления и описания типов и характеристик визуализации ПО, а также для сравнения и оценки программных средств визуализации ПО, служит таксономия визуализации ПО.

Одной из главных проблем визуализации является разработка метафор визуализации. Метафора визуализации определяет спецификацию визуализируемых процессов, их атрибутов, взаиморасположения и взаимодействия этих процессов. Визуальное представление предназначено для облегчения понимания визуализируемого алгоритма.

При разработке визуальных представлений в данной работе в качестве

основного подхода используется представление управляющего алгоритма на основе графовых моделей, отображающих поток управления.

Можно выделить несколько основных моделей представления управляющих алгоритмов с помощью графов: управляющий граф, граф операционно-логической истории, информационный граф и граф истории реализации.

В настоящей работе будут изложены графовые модели, наилучшим образом подходящие для представления и синтеза управляющих алгоритмов РВ – логико-временные схемы программ и многоходовые модели. Они сочетают в себе, с одной стороны, изобразительные возможности управляющего графа, с другой – таймированной системы переходов. При этом исчезает проблема «взрыва» количества состояний реальных программ, присущей модели Крипке.

II. Семантика управляющего алгоритма реального времени

Управляющий алгоритм должен обеспечить независимо от своих внутренних состояний в каждый заданный момент времени (или на протяжении заданного промежутка времени) выдачу корректного управляющего воздействия $Y(t_i)$, соответствующего текущему состоянию объекта управления и внешней среды, задаваемого $\bar{I}(t_i)$.

Вектор $\bar{I}(t_i)$ вместе с набором «флагов», формируемым УА отражается в значениях соответствующих логических переменных, которые являются компонентами логического вектора $\bar{I}(t_i)$.

С точки зрения правильности работы УА РВ его корректность может быть определена как успешное выполнение системой требуемых целевых задач при любом возможном развитии ситуации, что может быть записано в обозначениях пред- и постусловий:

$\{ U(D_0, t_0) \} YS \{ B(D_1(t_1), D_2(t_2), \dots, D_k(t_k)) \}$,
то есть в момент времени t_0 начала функционирования SCADA-системы истинно условие корректного задания исходных данных D_0 , а к моменту t_k завершения работы управляющего алгоритма YS истинно условие B , означающее успешное выполнение всех целевых задач D_k во все заданные моменты времени t_1, t_2, \dots, t_k , на всем интервале функционирования системы.

Управляющий алгоритм должен также учитывать изменение факторов внешней среды, который относятся к случайным факторам. Особенностью является также то, что управляющее воздействие (информация, выходные данные) программы реального времени через некоторый интервал времени \square_n устаревает, и ценность такого действия утрачивается.

Многие управляющие воздействия требуют своего осуществления не однократно (пример – единовременная выдача команды управления на бортовой аппаратуре космического аппарата), а в течение определенного промежутка времени (пример – работа реактивного двигателя для сообщения космического аппарата заданного импульса).

Отсюда логически следует представление семантики УА РВ в виде набора четверок объектов:

$$\text{УА РВ} = \{ \langle f_i, t_i, \tau_i, \bar{1}_i \rangle \}, i = \overline{1, N}, \quad (1)$$

где f_i – функциональная задача (действие); t_i – момент начала выполнения действия (целое неотрицательное число); τ_i – длительность действия (целое неотрицательное число); $\bar{1}_i$ – логический вектор, обуславливающий действие.

Элементами логического вектора являются значения входящих в него логических переменных – ИСТИНА, ЛОЖЬ, НЕОПРЕДЕЛЕННОСТЬ. Рассмотрение сразу целого логического вектора, а не отдельных логических переменных, обуславливающих действие, целесообразно, потому что одной из задач УА РВ является задача построения всех возможных вариантов исполнения алгоритма с отбрасыванием недостижимых вариантов. Полное множество вариантов имеет размерность $2M$, где M – общее количество логических переменных. Актуальным также является применение не обычной двузначной логики, а трехзначной логики, как в модифицированной системе алгебраических алгебр Глушкова.

Таким образом, семантически управляющий алгоритм должен обеспечивать на некотором не пустом множестве временных меток («включений») выполнение определенных действий по управлению испытаниями сложными техническими системами, зависящих от текущей ситуации в системе, отражаемой вектором значений

логических переменных. В этом заключается выполнение целевых задач, решение которых обеспечивается УА.

III. Многовходовая и многовариантная модели управляющего алгоритма реального времени

Рассмотрим описанные в (1) четверки объектов, задающие семантику УА РВ. Зафиксируем какое-то конкретное значение логического вектора, и выпишем соответствующие данному значению тройки (то есть фактически построим проекцию по данному компоненту). Получим следующий набор:

$$\text{ММ} = \{ \langle f_i, t_i, \tau_i \rangle \}, i = \overline{1, N}. \quad (2)$$

Каждый элемент вышеприведенного множества по семантике соответствует одному включению (одному моменту времени, в который управляющий алгоритм выполняет некоторую функциональную задачу).

Приведенная формула есть фиксация семантики одного из возможных вариантов управляющего алгоритма. При этом отличием данного представления от ранее предложенных, является то, что указывается не просто последовательности действий (шагов), а точные значения времени начала и длительностей выполнения отдельных действий.

Представление вида (1) может быть преобразовано в *многовходовую модель* УА РВ путём применения следующей процедуры:

Шаг 1. В полученном множестве троек объектов выделим множество несовпадающих друг с другом моментов времени запуска ФЗ $t_{\text{н}}$ (в общем случае в один момент времени может запускаться несколько ФЗ). Назовём каждый такой момент времени *входом*.

Шаг 2. Упорядочим полученное множество моментов времени в порядке возрастания. В полученном списке ближайшие по времени старта входы назовем соседними.

Шаг 3. Найдем разницу во времени между каждой парой соседних входов.

Шаг 4. Сформируем ориентированный взвешенный граф: каждому входу сопоставим вершину графа, между двумя соседними входами проведём дугу с весом, равным разнице по времени между входами.

Пример. Пусть мы имеем следующую семантику УА РВ, включающую пять ФЗ $f_1 - f_5$ и логический вектор, состоящий из одной компоненты f_j :

$$\{ \langle f_1, 0, 20, (\alpha_1=I) \rangle, \\ \langle f_2, 0, 100, (\alpha_1=II) \rangle, \\ \langle f_3, 20, 200, (\alpha_1=I) \rangle, \\ \langle f_4, 220, 10, (\alpha_1=H) \rangle, \\ \langle f_5, 180, 50, (\alpha_1=H) \rangle \}.$$

Фиксируя значение логического вектора ($\square_i=I$), случаи ($\square_i=H$), когда нет зависимости от значения данной логической переменной, тоже покрываются этим вариантом, получим следующее представление:

$$\{ \langle f_1, 0, 20 \rangle, \\ \langle f_3, 20, 200 \rangle, \\ \langle f_4, 220, 10 \rangle, \\ \langle f_5, 180, 50 \rangle \}.$$

Список моментов времени стартов ФЗ будет следующим: (0, 20, 180, 220). В момент времени $t=0$ запускается ФЗ f_1 , в момент времени $t=20$ – ФЗ f_3 , в момент $t=180$ – ФЗ f_5 , и в момент $t=220$ – ФЗ f_4 .

Многовходовая модель в узком смысле, или набор входов (моментов выполнения действий, соединенных дугами с весами, соответствующими разделяющие входы интервалам времени) для данного примера изображена на рис. 2:

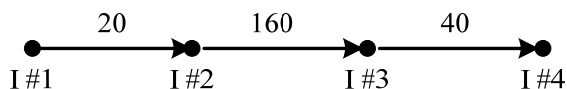


Рис. 2 – Многовходовая модель в узком смысле

Помимо *многовходовой модели в узком смысле*, представляющей один возможный вариант исполнения управляющего алгоритма, можно описать многовходовую модель, отражающую все возможные варианты исполнения данного УА РВ. В этом случае первый шаг (фиксация значения логического вектора и отбрасывание всех несоответствующих ему четверок из семантики УА) не осуществляется, и все входящие в УА РВ различные моменты времени, в которые включаются функциональные задачи, отображаются во входы многовходовой модели. Многовходовые модели первого типа будут соответствовать операционно-логической истории УА при фиксации некоторых значений логических переменных, а второго типа – просто управляющему графу программы.

Поскольку в УА РВ сразу несколько функциональных задач могут иметь одно и то же время запуска, хотя и с разными

логическими векторами, обуславливающими их выполнение (то есть они будут принадлежать одному входу), для реализации выполнения этих функциональных задач можно построить некоторый управляющий граф программы – с операторами и проверками логических условий. Таким образом, вход будет иметь собственную внутреннюю структуру (возможно, достаточно сложную). Такую многовходовую модель, в которой расписаны внутренние структуры входов, и она описывает при этом все варианты исполнения управляющего алгоритма, то есть включает в себя проверки логических условий и цветные дуги, будем называть *расширенной многовходовой моделью*, или *многовходовой моделью УА (ММ)*.

Замечание. В реальности управляющие алгоритмы для систем автоматизации испытаний СТК могут иметь более сложную структуру, в частности, один вход может запускать несколько других (рис. 3). При этом соответствующая семантика УА РВ не изменяется. Многовходовые модели, представленные на рис. 2 и 3, реализуют одну и ту же семантику УА РВ.

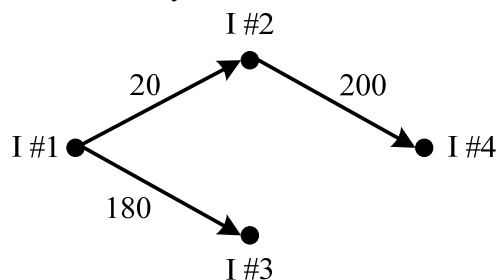


Рис. 3 Расширенная многовходовая модель

Применяя к семантике УА РВ операцию проекции путем фиксации момента времени, получаем *многовариантную модель (МММ)* одного включения (входа) УА РВ, играющую важную роль, в частности, при отладке управляющих программ, где важен точный набор возможных вариантов выполнения:

$$MVM = \{ \langle f_i, \tau_i, \bar{l}_i \rangle \}, i = \bar{1}, \bar{N} \quad (3)$$

Многовариантная модель (3) включает в себя описание выполнения логического вектора для каждой функциональной задачи. Если удалить в ней все строки, в которых логические векторы совпадают, и отбросить первые два компонента каждой триады, то фактически получается таблица возможных вариантов исполнения данного входа. По этой таблице, применяя специальный алгоритм, можно построить оптимизированную по числу

проверок логических переменных (и, соответственно, количеству логических ветвей), алгоритмическую структуру, или реализацию, входа.

Таблица вариантов исполнения имеет большое значение и применяется для решения ряда теоретических и практических задач в системах автоматизации испытаний сложных технических систем.

С точки зрения использования графовых представлений, возможно также формальное описание графовых моделей и диаграмм управляющих алгоритмов и правил их преобразования с помощью аппарата формальных грамматик, основанных на расширенной форме Бэкуса-Наура.

IV. Шаблоны визуализации

Шаблоны визуализации определяют основные типовые модели визуализации, которые неоднократно используются для визуализации УА, и охватывают всё интересное поведение УА.

Для разрабатываемой подсистемы визуализации достаточно таких базовых конструкций, как:

- линейный шаблон – для визуализации последовательной смены состояний;
- шаблон выбора – для отображения переходов из одного состояния в другое по определенному списку условий;
- условный шаблон представляет собой частный вид шаблона выбора и может быть необходим, если при определенных условиях автоматический процесс управления не проходит определенные состояния;
- цикл – шаблон, отображающий цикл (нет необходимости в шаблонах циклов с постусловием и предусловием, так как условия перехода задаются на дугах, а не на отдельных вершинах графа);
- шаблон обмена информацией необходим, если используется несколько распараллеленных процессов автоматического управления и состояние одного процесса автоматического управления запрашивает параметры определенного состояния второго процесса автоматического управления.

В таблице 1 будет представлено детальное описание шаблонов визуализации.

V. Формальное описание графовых моделей и диаграмм управляющих алгоритмов

Для формального описания модели системы необходимо создать словарь предметной области и на этой основе выделить множества терминальных и нетерминальных символов.

Подсистема визуализации управляющих алгоритмов планирует выполнение УА, то есть распределяет процессорное время между несколькими одновременно существующими в системе УА, а также занимается созданием и уничтожением процессов, обеспечивает УА необходимыми системными ресурсами, поддерживает взаимодействие между УА, которые характеризуется определенными состояниями. Таким образом, ключевыми понятиями для формализации управляющих алгоритмов являются:

- управляющий алгоритм;
- их качественные и количественные свойства;
- управляющие воздействия.

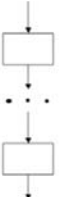
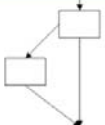
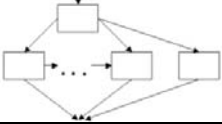

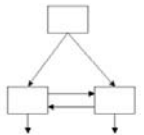
В соответствии с этим, можно сказать, что для определения качественных и количественных свойств УА необходимо определить основные состояния УА, параметры, характеризующие данные состояния, переходы и условия переходов из одного состояния в другое в соответствии с данными параметрами.

Выполнение перехода подразумевает сбор информации о текущем состоянии управляющего алгоритма, проверка заданного набора условий для оценки ситуации и переход в соответствии с ними в новое состояние. Описание УА для назовем планом процесса автоматического управления.

На первом этапе пользователь создает шаблон – план процессов автоматического управления, в котором описывает все типы объектов и состояний, которые могут использоваться в его графовой модели. Процесс аналогичен процессу декларации переменных. При описании типов объектов нужно указать имя типа, атрибуты, состояния, переходы, условия перехода и формат отображения данного типа в экземпляре графовой модели.

Таблица 1

Шаблоны визуализации

Визуальное отображение	Словесное описание	Математическая модель
	Линейный шаблон (линейная последовательность узлов графа)	$V_2: (ResV_1, \{InpV_2\}) \rightarrow ResV_2$. Вершина V_2 соединяется с вершиной V_1 , если V_2 использует результат V_1 в качестве аргумента
	Условный шаблон	$V_2: (ResV_1, \{InpV_2\}, U_2) \rightarrow ResV_2$. Вершина V_2 соединяется с вершиной V_1 , если V_2 использует результат V_1 в качестве аргумента и подчиняется управляющему воздействию U_2 .
	Шаблон выбора	$V_i: (ResV_1, \{InpV_i\}, U_i) \rightarrow ResV_i$. Вершина V_i соединяется с вершиной V_1 , если V_i использует результат V_1 в качестве аргумента и подчиняется управляющему воздействию U_i .
	Самоцикл шаблон (Самоцикл – узел, зацикливающийся сам в себя)	$V_1': (ResV_1, \{InpV_1'\}, U_{out}) \rightarrow ResV_1'$. Вершина V_1 зацикливается до тех пор, пока результат полученный при прохождении данной вершины не удовлетворяет условию выхода U_{out} .
	Шаблон обмена информацией	$V_i: (DataV_j, \{InpV_i\}) \rightarrow DataV_j$. $V_j: (DataV_i, \{InpV_j\}) \rightarrow DataV_i$. Вершина V_i обменивается информацией с вершиной V_j и наоборот, если V_i использует данные V_j в качестве аргумента и наоборот.

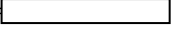
Шаблон опирается на формальную грамматику визуализации [44]. *Формальной грамматикой* называется упорядоченная четверка $G = \langle N, T, S, P \rangle$, где T – конечное множество *терминальных*, или основных символов; N – конечное множество *нетерминальных*, или вспомогательных символов ($T \cap N = \emptyset$); S – начальный символ, или аксиома, $S \in N$; P – конечное множество правил или продукций вида $\phi \rightarrow \psi$, где $\phi \in (T \cup N)^* V_N (T \cup N)^*$, $\psi \in (T \cup N)^*$ – различные цепочки, а \rightarrow – специальный символ, не принадлежащий $V_T \cup V_N$ и служащий для отделения левой части правила ϕ от правой ψ . В правилах вывода фигурные скобки «{}» означают произвольное повторение $0 \dots M$, а квадратные «[]» – необязательную часть.

Существуют различные методы описания формальных грамматик. Наиболее популярные – основанные на форме Бэкуса-Наура (либо расширенной форме Бэкуса-Наура). Однако эти методы предназначены для описания текстовых языков, а для описания визуальных языков данные методы

необходимо расширять. Ниже представлены два варианта формальных грамматик, использующих различные дополнения к форме Бэкуса-Наура.

Первый из них *основан на стандартной форме Бэкуса-Наура*, дополненной несколькими конструкциями, позволяющими описывать графическую информацию языка. *План процесса автоматического управления*, схемы текущего состояния и предыстории реализации процесса автоматического управления представляют собой графические диаграммы, которые имеют следующий графический синтаксис:

```

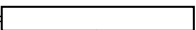
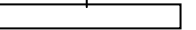
< diagram > ::= < symbol > { < heading > }
< symbol > ::= < frame symbol >
< frame symbol > ::= 
< heading > ::= < name >
    
```

Каждая диаграмма может иметь свое имя $\langle \text{name} \rangle$, которое является необязательным. Диаграмма также содержит в себе графовую модель, графический синтаксис которой представлен ниже:

```

< instance area > ::= < instance head area > < instance body are >
    
```


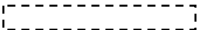
```

<instance head area> ::= <instance head
symbol> <instance heading>
<instance heading> ::= <instance name>
<instance head symbol> ::= 
<instance name> ::= <name>
<instance body area> ::= <instance state-
transition symbol> {<instance state area>
{<instance end symbol> | <stop symbol> } }
<instance axis symbol> ::= <instance state
area> <transition in area>
<instance end symbol> ::= 

```

Символ заголовка графа описывает начало процесса автоматического управления. Символ окончания графа описывает завершение процесса автоматического управления. Но данные символы не описывают создание и уничтожение процесса автоматического управления. Каждая графовая модель содержит в себе состояние instance state area, графический синтаксис которого представлен ниже:

```

< instance state area > ::= <action area>
<action area> ::= <action symbol> <action
text>
<action symbol> ::= 


```

```

<action text> ::= <text>

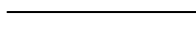

```

Семантика:

- события локальны по отношению к процессу, который их проходит;
- семантика события определяется при построении плана процесса автоматического управления;
- событие происходит «мгновенно».

Переход моделирует взаимодействие (т.е. обмен информацией или управление) между состояниями процесса. Переход может иметь имя transition instance name. Дополнительно допускается описывать передачу информации при переходе. С переходом может быть связан список условий перехода. Каждый параметр моделирует передачу конкретной информации от одного состояния к другому. Ниже представлен графический синтаксис перехода transition in area.

```

<transition out area> ::= <flow line
symbol>
<flow line symbol> ::= 
<transition in area> ::= <transition symbol>
<transition identification>
<transition symbol> ::= 

```

```

<external message area> ::= <message
symbol>
<transition identification> ::= [<transition
instance name>] [<condition list>]
<condition list> ::= <condition name>
<condition parameter name><condition
operator><condition value>[,<condition list>]
<condition name> ::= <name>
<condition parameter name> ::= <name>
<condition operator> ::= <=> | == | > | < | <>
<condition value> ::= int | real | boolean
<transition instance name> ::= <name>

```

Дополнительные правила построения диаграмм. Ограничения на взаимодействие:

- граф состояний и переходов должен быть ациклическим;
- состояние не может переходить само в себя;
- для каждого состояния и перехода должно быть задано порождающее состояние и порожденное состояние.

Второй вариант основан на расширенной форме Бэкуса-Наура. Идея данного метода взята из работы, в которой описывается метод спецификации визуальных языков моделирования на примере языка ADORA.

Суть выбранного метода спецификации:

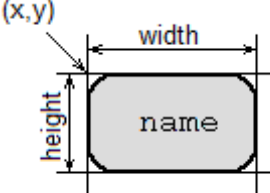

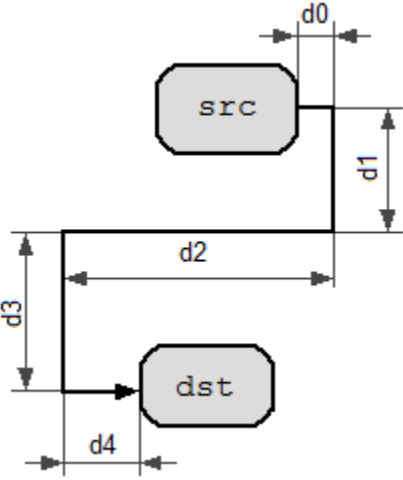

Положим, VL – заданный графический язык, синтаксис которого требуется определить. Для начала определяется текстовый язык TL так, что существует изоморфное отображение из каждого элемента языка VL на соответствующий элемент языка TL. Отображение определяется в табличной форме (таблица 2).

Имея TL и изоморфное отображение VL → TL, любое формальное определение синтаксиса языка TL является также определением синтаксиса исходного языка VL.

Для спецификации синтаксиса текстового языка TL используется расширенная форма Бэкуса-Наура (EBNF – Extended Backus-Naur Form), которая проста в восприятии для человека и для которой имеются проверенные и эффективные методы построения синтаксических анализаторов. Пример спецификации приведен ниже:

Таблица 2

Пример отображения элементов графического языка на элементы текстового

Графический символ	Текстовая конструкция
	<pre>state (state_id) name at (x,y) size (width,height) end</pre>
	<pre>start_state end</pre>
	<pre>transition (transit_id) from src:state_id to dst:state_id with (d0,d1,d2,d3,d4) end</pre>
	<pre>start_transition to dst:state_id end</pre>

Таким образом, в начале оператор выбирает конфигурационный файл управляющего алгоритма для автоматизации испытаний. Визуализируется план автоматического процесса управления. После этого запускается управляющий алгоритм. После этого подсистема визуализации отслеживает состояния и переходы УА и сверяет их с условиями перехода. По текущим данным переменных OPC-сервера (OPC – OLE for Process Control) визуализируется текущее состояние УА. По данным накопленным в ОЗУ (ОЗУ – оперативное запоминающее устройство) и условиям перехода конфигурации визуализируется схема предыстории реализации управляющего алгоритма.

Заключение

Таким образом, в настоящей работе была исследована проблема разработки управляющих алгоритмов в АСУ СТК, доказана необходимость применения компьютерной визуализации в программном обеспечении, реализующего управляющий алгоритм, а также важность использования метафор визуализации. Опираясь на теорию множеств, была разработана семантика управляющего алгоритма реального времени (УА РВ), которая стала основой для синтеза таких графовых моделей, как многоходовая и многовариантная модели управляющего алгоритма реального времени. Визуализация УА технологического процесса (ТП) испытаний осуществляется путем выделения «интересных» состояний и условий перехода из одного состояния в другое.

Визуализация программного обеспечения (ПО), реализующего управляющий алгоритм, опирается на ряд стандартных шаблонов визуализации, удовлетворяющих требованиям системы и способных отобразить поведение УА ТП испытаний. Теория графов предлагает свои решения для задачи формального синтеза, анализа и верификации модели УА. Проблема корректного визуального представления УА ТП испытаний решается с помощью формальной спецификации визуальных языков. В настоящей работе применяется текстовый метод спецификации синтаксиса, основанный на расширенной форме Бэкуса-Наура (EBNF).

Предложенный подход будет способствовать повышению эффективности ПО, реализующего УА ТП испытаний, что приведёт к уменьшению трудозатрат человека-оператора, непосредственно участвующего в процессе управления такими сложными системами, как АСУ СТК.

Список использованных источников

1. Douglas M. "Testability of Complex, Middleware-Based Systems" [Internet source] / Douglas M. Wells, Robert E. Bernstein, Amarendranath Vadlamudi // Paper presented at the International Conference on Dependable Systems & Networks, Washington, DC: National Defense University, 23-26 June, 2002. – URL: <http://www.opengroup.org/RI/recent/papers/WDMS2002.handout.pdf>.
2. Бенькович Е.С. Практическое моделирование сложных динамических систем [Текст] / Е.С. Бенькович, Ю.Б. Колесов, Ю.Б. Сениченков. – СПб.: БХВ-Петербург, 2002. – 464 с.
3. Подчуфаров Ю.Б. Физико-математическое моделирование систем управления и комплексов [Текст] / Ю.Б. Подчуфаров. – М.: Изд-во физ.-мат. литературы, 2002. – 168 с.
4. Тюгашев А.А. Процессор выходных документов в САПР управляющих алгоритмов [Текст] / А.А. Тюгашев // Представлено на 7-ом Всероссийском семинаре с международным участием по управлению движением и навигации летательных аппаратов, Самара, СГАУ, 1995. – С. 74 – 76.
5. Калентьев А.А.. Исчисление управляющих алгоритмов: Математические методы и модели в САПР [Текст] / А.А. Калентьев // Межвузовский сборник научных трудов. – Самара, 1991. – С. 4 – 10.
6. Борисов В.В. Компьютерная поддержка сложных организационно-технических систем [Текст] / В.В. Борисов, И.А. Бычков, А.В. Дементьев. – М.: Горячая линия Телеком, 2002. – 154 с.
7. Одинцов И.О. Профессиональное программирование: Системный подход [Текст] / И.О. Одинцов. – СПб.: БХВ-Петербург, 2002. – 312 с.
8. Гречишников А.В. Алгоритмизация и визуальное программирование управленческих задач [Текст] / А.В. Гречишников, А.П. Данчул, А.П. Павлов. – М.: Изд-во РАГС, 2001. – 95 с.
9. Авербух В.Л.. К теории компьютерной визуализации [Текст] / В.Л. Авербух // Вычислительные технологии. – Т. 10, №4. – 2005. – С. 21–51.
10. Авербух В.Л. Разработка средств визуализации программного обеспечения параллельных вычислений. Визуальное программирование и визуальная отладка параллельных программ [Текст] / В.Л. Авербух, А.Ю. Байдалин // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. – №. 4, 2003. – С. 68–80.
11. Price V.A. A Principled Taxonomy of Software Visualization [text] / V.A. Price // Journal of Visual Languages and Computing, 1993. – P. 211 – 266.
12. Авербух В.Л. Разработка видов отображения в специализированных системах компьютерной визуализации [Текст] / В.Л. Авербух // Труды 12-й Международной конференции по компьютерной графике и машинному зрению "ГрафиКон-2002". – Нижний Новгород, сентябрь 2002. – С. 184 – 189.

13. Воеводин В.В. Параллельные вычисления [Текст] / В.В. Воеводин, Вл.В. Воеводин. – СПб: БХВ-Петербург, 2002. – 253 с.

14. Касьянов В.Н. Графы в программировании: обработка, визуализация, применение [Текст] / В.Н. Касьянов, В.А. Евстигнеев. – СПб.: БХВ-Петербург, 2003. – 1104 с.

15. Новиков Ф.А. Дискретная математика для программистов [Текст] / Ф.А. Новиков. – СПб.: Питер, 2002. – 304 с.

16. Глушков В.М. Алгебра. Языки. Программирование [Текст] / В.М. Глушков, Г.Е. Цейтлин, Е.Л. Ющенко. – Киев: Наукова думка, 1989. – 446 с.

17. Жоголев Е.А. Графические редакторы и графические грамматики [Текст] / Е.А. Жоголев // Программирование. – № 3, 2001. – С. 30 – 42.

18. Ларман К. Применение UML и шаблонов проектирования: Пер. с англ. [Текст] / К. Ларман. – М.: Издательский дом «Вильямс», 2001. – 496 с.

19. Кузин С.Г. Сетевая грамматика как средство описания профессионального языка и модель функционирования языкового процессора [Текст] / С.Г. Кузин, Н.М. Миллер // Межвуз. тематич. сб. научн. тр. – Изд-во Горьк. гос. ун-та, 1989. – 229 с.

20. Мансуров Н.Н. Методы формальной спецификации программ: язык MSC и SDL [Текст] / Н.Н. Мансуров, О.Л. Майлингова // М.: Московский государственный университет им. М.В. Ломаносова, 1998. – 263 с.

21. Xia Y. A Syntax Definition Method for Visual Specification Languages [text] / Y. Xia, M. Glinz // in Technical Report. – Zurich University: Department of Information Technology, March 2002. – 171 p.

Сведения об авторе:



Кузнецова Юлия Анатольевна – м.н.с., ассистент кафедры инженерии программного обеспечения Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ». Научные интересы: инженерия программного обеспечения.
e-mail: JK-Sv@yandex.ru.