ДОКЛАДЫ АН ВШ РФ
2014 январь–март № 1 (22)

ТЕХНИЧЕСКИЕ НАУКИ

УДК 004.4'22

АЛГОРИТМ АВТОМАТИЧЕСКОЙ ТРАНСЛЯЦИИ ДИАГРАММЫ АКТИВНОСТИ В СЕТЬ ПЕТРИ

Марков А.В.[1], Романников Д.О.[2]
[1]*Новосибирский государственный технический университет*
[2]*ООО «Параллелз», филиал в г. Новосибирске*

Для разработки качественного программного обеспечения (ПО) предлагается применять средства проектирования, а именно пользоваться UML диаграммами для моделирования систем и сетями Петри для анализа полученных диаграмм. Поскольку трансляция между диаграммами и сетями выполняется вручную, существует необходимость автоматического преобразования. Данную процедуру предлагается осуществить при помощи схожей структуры форматов, в которых могут быть сохранены UML диаграммы активности и сети Петри. Для корректного преобразования приводятся требования к проектированию диаграмм – задание такой структуры имен, которая позволит передать необходимые данные для трансляции в сеть Петри, а также выделяются схожие сущности, между которыми происходит обмен информацией – места, переходы, взаимосвязи между ними. Правила, по которым происходит преобразование, представляются в алгоритмическом виде, что способствует их формализации. Алгоритм преобразования по схожим форматам представляется на примере диаграммы активности и сети Петри, которые содержат набор приведенных правил. Поскольку данные правила являются базовыми для построения диаграмм любой сложности, предложенный алгоритм может быть использован для проектирования и анализа большинства систем.

Ключевые слова: UML, сети Петри, автоматическое преобразование, формат «*.xmi*», формат «*.cpn*», диаграмма активности, анализ спроектированных диаграмм.

Введение

В настоящее время высокая работоспособность предприятий зависит от используемых программных продуктов, качество и безотказность которых должны быть высокими. Создание приложений такого уровня реализуют при помощи различных инструментов проектирования, моделирования, анализа, а также языков программирования. Яркими представителями разработки ПО являются UML диаграммы и сети Петри [1–12]. Существует множество способов и правил совместного использования диаграмм и сетей, но в каждой из них не хватает либо самых правил преобразования одной сущности в другую, либо их формализации. В работах [3, 4] в графическом виде предлагаются правила преобразования, но отсутствует упоминание о формализации данных правил. Работы [8, 9] посвящены разработке методик совместного проектирования UML диаграмм вариантов использования, классов, объектов, состояний, активности и сетей Петри, а также разработке правил для трансляции диаграмм в сети Петри, которые представлены в иллюстрированном виде и не формализованы. Применяя UML диаграммы вариантов использования, классов, активности, проектируются системы с последующим анализом при помощи сетей Петри поведенческих диаграмм [11, 12].

Наиболее популярными пакетами для моделирования UML диаграмм являются Rational Rose от компании IBM и MagicDraw от компании No Magic. Данные пакеты имеют большой арсенал для моделирования диаграмм, связи их в единые проекты и преобразования полученных наборов в коды. Object Management Group был предложен формат «*.xmi*», в котором может храниться сжатая, но полная информация как о смоделированных проектах, так и об отдельных диаграммах. Данный формат имеет синтаксис языка *xml*, но на сегодняшний момент возмож-

ность сохранить в данном формате присутствует не у всех пакетов визуального моделирования. Для проектирования сетей Петри наиболее популярным пакетом является CPN Tools. Данная программа находится в свободном доступе, имеет интуитивно понятный интерфейс и возможность не только моделировать сети, но и анализировать их через генерацию пространства состояний. Сохраненные проекты CPN Tools имеют формат «*.cpn*», для записи данных также используется синтаксис языка *xml*.

Таким образом, связка UML и сети Петри позволяет использовать язык графического описания для объектного моделирования и математический аппарат сетей как инструмент проверки спроектированных диаграмм. Целью работы является разработка алгоритма для реализации автоматической трансляции между диаграммой активности и сетью Петри, который основан на формализации правил, представленных в графическом виде, и выполнении требований при проектировании диаграммы активности. Инструментом для автоматического преобразования диаграммы активности в сети выберем преобразование схожей структуры двух форматов «*.xmi*» и «*.cpn*», что позволит сократить время на ручное преобразование диаграмм.

1. Правила преобразования UML диаграмм активности в сети Петри

Рассмотрим наиболее часто встречающиеся правила преобразования UML диаграмм активности в сети Петри [8] и предложим алгоритмическую форму представления данных правил: состояние ожидания трансформируется в место, а состояние действия преобразуется в переход, который начинает действие (алгоритм 1); разделение и слияние потоков управления UML диаграмм преобразуется в соответствующую сеть Петри (алгоритм 2); ветвление на диаграмме активности, обозначаемое символом решения, преобразуется в соответствующий эквивалент сети Петри (алгоритм 3); условия ограничения показывают с помощью условий ограничения переходов (алгоритм 3).

Алгоритм 1. Преобразование последовательности состояний из UML диаграммы в сеть Петри

$SS := SP$ // SS – начальное состояние на диаграмме активности, SP – начальное место в сети Петри

$ES := EP$ // ES – конечное состояние на диаграмме активности, EP – конечное место в сети Петри

$UST := PST$ // UST – стартовый переход на диаграмме активности, PST – начальный переход в сети Петри

$UET := PET$ // UET – конечный переход на диаграмме активности, PET – конечный переход в сети Петри

$A := \{AS_i\}$ // переменная A – множество всех состояний действия на диаграмме активности

$T := \{UT_i\}$ // переменная T – множество всех переходов системы на диаграмме активности

while $A \neq \emptyset$ *do* // пока A не равна $\emptyset$, выполняем …

select an $AS_i \in A$ // выборку всех состояний действия

$AS_i := PP_i$ // трансформируем каждое состояние действия диаграммы активности в эквивалентное место сети Петри

return PP_i // возвращаем значение место сети Петри

while $T \neq \emptyset$ *do* // пока T не равна 0, выполняем …

select an $UT_i \in T$ // выборку всех переходов диаграммы активности

$UT_i := PT_i$ // и приравниваем их к соответствующим переходам сети Петри

return PT_i // возвращаем значения перехода сети Петри

Алгоритм 2. Преобразование элементов разделения и слияния из UML диаграммы в сеть Петри

$A := \{AS_i\}$

while $A \neq \emptyset$ *do*

select an $AS_i \in A$

$AS_i := PP_i$

$UF := PF$ // переход UF (разветвление диаграммы действия), переход PF (разветвление в сети Пети)

$UJ := PJ$ // переход UJ (слияние диаграммы действия), переход PJ (слияние в сети Пети)

Алгоритм 3. Преобразование элементов ветвления и условий из UML диаграммы в сеть Петри

$S := (A, T)$ // связка множеств состояний и переходов между ними

$A := \{AS_i\}$

$T := \{UT_i\}$

while $A \neq \emptyset$ *and* $T \neq \emptyset$ *do* // пока A и T не равна $\emptyset$, выполняем …

if $(AS_i, UT_i, UJun) \in S$ *then* // условие: если существует множество, состоящее из последовательной цепочки: состояние действия, переход, блок условия, тогда

$A := A \setminus \{AS_i\}$ *and* $T := T \setminus \{UT_i\}$ *and* // из множества A и T удаляем соответствующие элементы и …

$(AS_i, UT_i, UJun) := PJun$ // трансформируем последовательную цепочку в эквивалентное место в сети Петри

else select an $AS_i \in A$ // иначе делаем выборку всех состояний действия

$AS_i := PP_i$

select an $UT_i \in T$ // делаем выборку всех переходов диаграммы активности

$(UT_i \mid condition) := (PT_{(i-1)} \mid condition)$ // добавляем условия к переходам сети Петри, которые являются выходящими для $PJun$

Практически в каждой разработанной системе можно встретить последовательность и условие, разветвление и слияние. Следовательно, преимуществом описанного набора правил является то, что он содержит рекомендации по взаимному получению более часто встречающихся элементов в алгоритмах UML диаграмм активности и сетей Петри. Представление правил в алгоритмическом виде способствует их формализации. Сокращение времени на проектирование при помощи упомянутой выше связки можно добиться, реализовав автоматическое преобразование между ними.

2. Описание и сравнение форматов «*.xmi*» и «*.cpn*», требования к проектированию диаграммы активности

Для реализации автоматического преобразования UML диаграмм активности и сетей Петри необходимо использовать не только правила, описанные в [8], но и прибегнуть к дополнительным приемам. Поэтому предложим правила преобразования структуры файлов «*.xmi*» и «*.cpn*», а также требования к моделированию элементов диаграммы активности, способной показать динамику работы системы. Выделим начальное место, конечное место, обычное место и обычный переход в структуре файла диаграммы активности и построенной на основе ее сети Петри, представим логику нахождения и преобразования псевдосостояний: разветвлений, слияний, условий. Дадим рекомендации по именованию состояний и переходов для диаграммы активности, поскольку в сетях Петри встречаются свойства мест: начальная маркировка, тип данных, и переходов: условие срабатывания перехода, который в языке UML не предусмотрен, но для сетей является обязательным.

При проектировании UML диаграмм активности структура системы имеет три вида элементов: состояния, псевдосостояния (начало, условие, слияние и разветвление), переходы (дуги между местами). Также для каждого элемента могут быть определены входные и выходящие элементы. При моделировании проекта в программной среде CPN Tools также существует три вида элементов: места, переходы и дуги, а взаимосвязь вершин происходит через описание дуг. Как таковых псевдосостояний CPN Tools не предусматривает, их заменяют различные вариации связей мест и переходов.

Приведем дополнительные требования и рекомендации к моделированию системы, которые реализуем на диаграмме активности, включающей выполнение условия, слияние и разделения (рис. 1), с последующей трансформацией в сеть Петри. Начальное состояние в UML у формата «*.xmi*» является псевдосостоянием (в контексте синтаксиса «*.xmi*») и имеет имя *StartState__1`1__INT*. В данном случае имя содержит больше информации, чем просто название – в него входит информация о начальной маркировке этого состояния и о типе данных в этом состоянии. Чаще всего дуга в UML является переходом в CPN Tools. В данном примере она имеет имя *Start__i__i__i*, в котором представлена информация о названии, об имени для входной и выходной дуг и условии для перехода CPN Tools соответственно. Обычное состояние, например *Activity1*, имеет имя, схожее с именем начального состояния: *Activity1__0`0__INT*. В нем также присутствует название, начальная маркировка и тип данных. Дуги с именем *empty__i__i* добавляются на диаграмму активности перед и после псевдосостояний со свойством формата «*.xmi*» *kind*: *fork*, *join*, *junction*. В CPN Tools эти дуги никакой информации не несут и не отображаются. Разветвление на рис. 1 является псевдосостоянием, а в формате «*.xmi*» имеет дополнительное свойство *kind* с атрибутом *fork*, и будет представлено в CPN Tools как переход с именем *Activity1__i__i__i*. Первая часть имени – название, вторая – имя входящей дуги, третья – имя выходящей дуги, четвертая – условие срабатывание перехода. Слияние моделируется и преобразовывается в CPN Tools аналогично разветвлению.

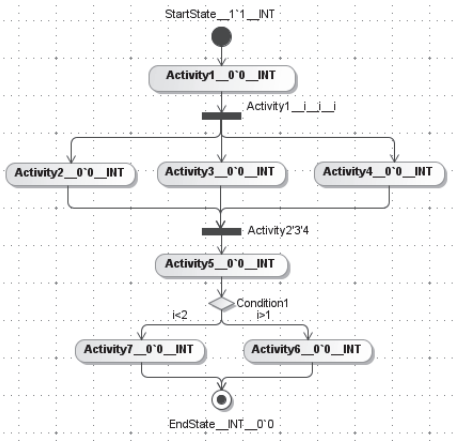

Рис. 1. Диаграмма активности

Псевдосостояние со свойством *kind* и атрибутом этого свойства *junction* имеет имя *Condition1*. Данная сущность не отображается в сети Петри, смоделированной в программной среде CPN Tools, эту роль будут играть место *Activity5* и переходы, образующиеся при помощи выходящих дуг из пвсевдосостояния *Condition1*. Выходящая дуга из псевдосостояния *Condition1* имеет имя *Activity5_1__i__i__i>1*, где первая часть – название перехода в CPN Tools, последующие два элемента – название входной и выходной дуги соответственно, а четвертый элемент отвечает за возможное условие, предъявляемое к срабатыванию перехода. Конечное состояние в UML диаграмме активности имеет имя *EndState__0`0__INT* и является обычным состоянием системы. Для сравнения двух форматов «*.xmi*» и «*.cpn*» вручную преобразуем данную диаграмму в сеть Петри (рис. 2). Основные свойства построенной сети: *colset INT = int; var i : INT.*

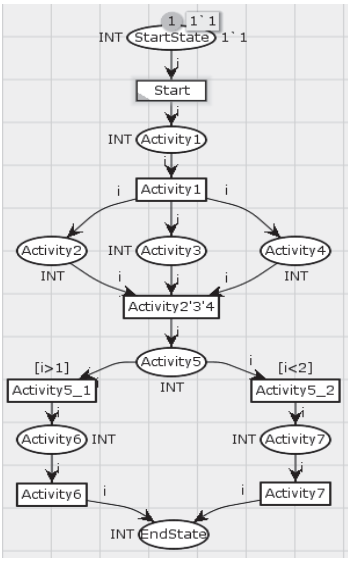

Рис. 2. Сеть Петри, полученная из диаграммы активности

В CPN Tools существует три типа элементов: места, переходы и дуги. У места есть имя, тип данных, который возможен в этом месте, и начальная разметка этого места. В переходе заложена информация об имени и возможном условии, а дуги – об ориентации, начальной и конечной вершины: либо места и перехода, либо перехода и места, в зависимости от атрибута свойства *orientation*. Так как данные диаграммы являются аналогами друг друга, сравним структуру сохраненных фалов и выделим схожие части форматов «*.xmi*» и «*.cpn*», а именно: конечные места, обычные места, обычные переходы и дуги для формата «*.cpn*». Более детально разберем описание и сравнение начального места. Первым элементом на диаграмме активности (рис. 1) и сети Петри (рис. 2) является начальное место. Структура данного элемента приведена в таблице.

Начальное место

формат «.cpn»	формат «.xmi»
1. <place id="ID1420720921"> 2–5. … 6. <text>StartState</text> 7–11. … 12. <type id="ID1420720922"> 13–16. … 17. <text tool="CPN Tools" version="3.4.0">INT</text> 18. ... 19. <initmark id="ID1420720923"> 20–23. … 24. <text tool="CPN Tools" version="3.4.0">1`1</text> 25–26. …	1. <UML:Pseudostate xmi.id = '-64--88-0-100--4260c10:1410c69956e:-8000:0000000000000869' 2. name = 'StartState__1`1__INT' isSpecification = 'false' kind = 'initial'> 3. <UML:StateVertex.outgoing> 4. <UML:Transition xmi.idref = '-64--88-0-100--4260c10:1410c69956e:-8000:0000000000000874'/> 5. </UML:StateVertex.outgoing> </UML:Pseudostate>

У формата «*.xmi*» описание начального состояния более краткое по сравнению с описанием начального места формата «*.cpn*», тем не менее данный факт не сказывается на информативности. В строке 6 формата «*.cpn*» (см. таблицу) задается имя начального места, которое соответствует имени в названии начального состояния у формата «*.xmi*». В строке 17 формата «*.cpn*» задается тип данных, соответствующий второй части в названии начального состояния у формата «*.xmi*». Строка 24 формата «*.cpn*» содержит информацию об исходной разметке начального состояния, а у формата «*.xmi*» это третья часть в названии. Во второй строчке формата «*.xmi*» помимо названия содержится свойство *kind* с атрибутом *initial*. Это означает, что данное псевдосостояние является начальным. В строке 3 формата «*.xmi*» говорится о существовании выходящей дуги, направленной из этого состояния, а ниже указывается ее *ID*, который уникален в проекте. После описания и сравнения двух форматов выделяем основные признаки: имя элемента, тип данных, присущих данному элементу и маркировка каждого элемента.

Описание общей идеи преобразования диаграммы активности приводилось в [8], но данное представление затрудняет трансляцию диаграммы активности в сети Петри в автоматическом виде. При помощи формализации правил и использовании форматов со схожей структурой разработан алгоритм для автоматического преобразования, что способствует сокращению времени на проектирование систем при помощи этих сущностей. В последующих работах будет описан программный продукт и его применение к различным системам. Уточним, что в данном случае форматы использовались как один из способов реализации трансляции между диаграммой активности и сетями Петри.

Заключение

Описано совместное использование унифицированного языка моделирования UML и математического аппарата сетей Петри, которое заключается в верификации спроектированных диаграмм активности при помощи сетей Петри. Сети Петри разрабатывались специально для систем с параллельными процессами с целью синхронизации их взаимодействия и выявления логических ошибок: зацикливания, характеризующиеся циклическими процессами, и тупиковые состояния, которые вызывают непреднамеренное завершение работы системы в смоделированных алгоритмах. Между диаграммой активности и сетью Петри существует сходство, обусловленное тем, что данные сущности моделируют динамические свойства системы. Диаграмму активности предложено проектировать с добавлением к каждому состоянию информации о разметке (о количестве и типе ресурсов системы), которая будет соответствовать разметке места сети. При получении положительных результатов после анализа спроектированных диаграмм при помощи сетей Петри можно с уверенностью сказать о корректном построении диаграммы активности. Затруднительным видится ручное преобразование диаграммы активности в сеть Петри, но используя автоматическую трансляцию, можно сократить временные ресурсы, выделяемые на проектирование различных систем. Данную процедуру рекомендовано выполнять через схожую структуру двух форматов «*.xmi*» и «*.cpn*» соответственно. Для ее реализации были выделены и сравнивались начальное и конечное место, обычное место, обычный переход, полученные из исследования форматов, упомянутых выше; описаны основные нюансы: количество и в отдельности каждый тип элементов, реализация взаимосвязи между элементами. Сходством этих форматов является наличие мест и переходов у сетей Петри и UML диаграмм активности, а основным отличием – наличие у формата «*.cpn*» тега <*arc*>, который отвечает за связку между вершинами сети. А у формата «*.xmi*» данный тег отсутствует, но информация о взаимо-

связях между состояниями на диаграмме активности заложена в тегах *<UML:…State>* и *<UML:Transition>*.

Формализация правил [8] преобразования UML диаграмм активности и сетей Петри осуществлена через представление их в виде алгоритмов, что позволяет закодировать данные правила [8, 9, 11, 12].

Опираясь на полученные результаты, можно сказать, что реализация автоматической трансляции UML диаграмм активности и сетей Петри выглядит весьма простым, а время, выделяемое на анализ и ручное преобразование построенных диаграмм, сокращается, что является несомненным преимуществом.

ЛИТЕРАТУРА

[1] **ZHU** Lian-Zhanga, KONG Fan-Sheng. Automatic Conversion from UML to CPN for Software Performance Evaluation // *Procedia Engineering*. – 2012. – No. 29. – Pp. 2682–2686.

[2] **Campos J., Merseguer J.** On the Integration of UML and Petri Nets in Software Development // *ICATPN*. – 2006. – Pp. 19–36.

[3] **Yann Thierry-Mieg, Lom-Messan Hillah.** UML behavioral consistency checking using instantiable Petri nets // *Innovations Syst Softw Eng*. – 2008. – No. 4. – Pp. 293–300.

[4] **Eichner C., Fleischhack H., Meyer R., Schrimpf U., Stehno C.** Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets // *SDL*. – 2005. – Pp. 133–148.

[5] **Nianhua Yang, Huiqun Yu, Hua Sun, Zhilin Qian.** Modeling UML sequence diagrams using extended Petri nets // *Telecommun Syst*. – 2012. – Vol. 51. – Pp. 147–158.

[6] **Basile F., Chiacchio P., Grosso D.D.** A two-stage modelling architecture for distributed control of real-time industrial systems: Application of UML and Petri Net // *Computer Standards & Interfaces*. – 2009. – Vol. 31. – Pp. 528–538.

[7] **Miyamoto T., Kurahata H., Fujii T., Hosokawa R.** Synthesis of state machine diagrams from communication diagrams using Petri nets // *Innovations Syst Softw Eng*. – 2010. – Vol. 6. – Pp. 39–46.

[8] **Коротиков С.В.** Применение сетей Петри в разработке программного обеспечения центров дистанционного контроля и управления: дис. …. канд. техн. наук. – Новосибирск, 2007. – 216 с.

[9] **Романников Д.О., Марков А.В.** Пример применения методики разработки ПО с использованием UML-диаграмм и сетей Петри // *Научный вестник НГТУ*. – 2012. – № 1(67). – С. 175–181.

[10] **Воевода А.А., Романников Д.О.** Применение UML диаграмм и сетей Петри при разработке встраиваемого программного обеспечения // *Научный вестник НГТУ*. – 2009. – № 4 (37). – С. 169–174.

[11] **Марков А.В., Воевода А.А.** Развитие системы «Перемещение манипулятора в пространстве с препятствиями» при помощи рекурсивных функций // *Автоматика и программная инженерия*. – 2013. – № 2. – С. 35–41.

[12] **Марков А.В., Воевода А.А.** Анализ сетей Петри при помощи деревьев достижимости // *Сборник научных трудов НГТУ*. – 2013. – № 1. – С. 78–95.

ALGORITHM OF AUTOMATIC CONVERSION OF THE ACTIVITY DIAGRAM INTO PETRI-NET STRUCTURE FORMATS

Markov A.V.[1], Romannikov D.O.[2]

[1]*Novosibirsky State Technical University, Novosibirsk, Russia*
[2]*Parallels Ltd, Novosibirsk branch, Novosibirsk, Russia*

To develop high quality software it is proposed to apply design tools, namely UML diagrams, for modeling systems and Petri nets for the analysis of the obtained diagrams. Since the translation between diagrams and nets is done manually, there is a need in automatic conversion. It is proposed to carry out this procedure by using a similar format structure in which UML activity diagrams and Petri nets can be saved. To carry out correct conversion, requirements for design

diagrams i.e. setting such a name structure which will make it possible to transmit the necessary data for the translation into the Petri net are stated. In addition, similar entities between which the exchange of information such as places, transitions and relationship between them are given. The rules by which the conversion occurs are presented in an algorithmic form, which facilitates their formalization. The conversion algorithm by similar formats is given by the example of the activity diagram and Petri nets which contain a set of the given rules. Since these rules are basic for modeling diagrams of any complexity, the proposed algorithm can be used for the design and analysis of most systems.

Keywords: UML; Petri nets; automatic conversion; *xmi* format; *cpn* format; activity diagram; analysis of designed diagrams.

REFERENCES

[1] **ZHU** Lian-Zhanga, KONG Fan-Sheng. Automatic Conversion from UML to CPN for Software Performance Evaluation. *Procedia Engineering*, 2012, no. 29, pp. 2682–2686.

[2] **Campos J., Merseguer J.** On the Integration of UML and Petri Nets in Software Development. *ICATPN*, 2006, pp. 19–36.

[3] **Yann Thierry-Mieg, Lom-Messan Hillah.** UML behavioral consistency checking using instantiable Petri nets. *Innovations Syst Softw Eng*, 2008, no. 4, pp. 293–300.

[4] **Eichner C., Fleischhack H., Meyer R., Schrimpf U., Stehno C.** Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets. *SDL*, 2005, pp. 133–148.

[5] **Nianhua Yang, Huiqun Yu, Hua Sun, Zhilin Qian.** Modeling UML sequence diagrams using extended Petri nets. *Telecommun Syst.*, 2012, vol. 51, pp. 147–158.

[6] **Basile F., Chiacchio P., Grosso D.D.** A two-stage modelling architecture for distributed control of real-time industrial systems: Application of UML and Petri Net. *Computer Standards & Interfaces*, 2009, vol. 31, pp. 528–538.

[7] **Miyamoto T., Kurahata H., Fujii T., Hosokawa R.** Synthesis of state machine diagrams from communication diagrams using Petri nets. *Innovations Syst Softw Eng*, 2010, vol. 6, pp. 39–46.

[8] **Korotikov S.V.** *Primenenie setey Petri v razrabotke programmnogo obespecheniya tsentrov distantsionnogo kontrolya i upravleniya. Diss. kand. tekhn. nauk* [Application of Petri nets in software development centers, remote monitoring and control. PhD eng. sci. diss.]. Novosibirsk, 2007. 216 p.

[9] **Romannikov D.O., Markov A.V.** Primer primenenija metodiki razrabotki PO s ispol'zovaniem UML-diagramm i setej Petri [An example of the application of the software development methodology using UML diagrams and Petri nets]. *Nauchnyj vestnik NGTU*, 2012, no. 1, pp. 175–181.

[10] **Voevoda A.A., Romannikov D.O.** Primenenie UML diagramm i setej Petri pri razrabotke vstraivaemogo programmnogo obespechenija [Applying UML diagrams and Petri nets in developing embedded software]. *Nauchnyj vestnik NGTU*, 2009, no. 4, pp. 169–174.

[11] **Markov A.V., Voevoda A.A.** Razvitie sistemy «Peremeshhenie manipuljatora v prostranstve s prepjatstvijami» pri pomoshhi rekursivnyh funkcij [Development of the "Manipulator motion in space with obstacles" system using recursive functions]. *Avtomatika i programmnaja inzhenerija*, 2013, no. 2, pp. 35–41.

[12] **Markov A.V., Voevoda A.A.** Analiz setej Petri pri pomoshhi derev'ev dostizhimosti [Analysis of Petri nets by means of reachability trees]. *Sbornik nauchnyh trudov NGTU*, 2013, no. 1, pp. 78–95.

СВЕДЕНИЯ ОБ АВТОРАХ

Марков Александр Владимирович – родился в 1988 году, окончил Новосибирский государственный технический университет (НГТУ), с 2011 года аспирант кафедры автоматики НГТУ. Область научных интересов: UML диаграммы, сети Петри. (Адрес: 630073, Россия, Новосибирск, пр. Карла Маркса, 20. Email: Muviton3@gmail.com)

Markov Alexandr Vladimirovich (b. 1988) – graduated from the Novosibirsk State Technical University (NSTU), Post-graduate Student of Automation De-

partment of the NSTU. Area of research: UML diagrams, Petri-nets. (Address: 20, Karl Marx Av., Novosibirsk, 630073, Russia. Email: Muviton3@gmail.com)

Романников Дмитрий Олегович – родился в 1987 году, канд. техн. наук, инженер-программист ООО «Параллелз», филиал в г. Новосибирске. Область научных интересов: UML диаграммы, сети Петри. (Адрес: 630007, Россия, Новосибирск, Октябрьская маг., 4. Email: Rom2006@gmail.com)

Romannikov Dmitry Olegovich (b. 1987) – PhD (Eng.), Software Engineer, Parallels Ltd, Novosibirsk branch. His research interests are currently focused on the UML diagrams, Petri-nets. (Address: 4, October highway, Novosibirsk, 630007, Russia. Email: Rom2006@gmail.com)

Статья поступила 12 февраля 2014 г.
Received 12 Feb. 2014

To Reference:
Markov A.V., Romannikov D.O. Algoritm avtomaticheskoi translyatsii diagrammy aktivnosti v set' Petri [Algorithm of automatic conversion of the activity diagram into petri-net structure formats]. *Doklady Akademii Nauk Vysshei Shkoly Rossiiskoi Federatsii* [*Reports of Russian Higher Education Academy of Sciences*], 2014, no. 1(22), pp. 104–112. (in Russ.).