УДК 004.43

К. ЯНЧЕК, Е. КОЙЧЕВА, Р. В. ХАЗАНКИН, А. М. МОРОЗОВ

ИНТЕРПРЕТАТОР СКРИПТОВОГО ЯЗЫКА JAVASCRIPT ДЛЯ СИМУЛЯТОРА ОБОБЩЕННЫХ СЕТЕЙ

В статье рассмотрена задача внедрения интерпретатора скриптового языка JavaScript в программное приложение. В статье рассмотрены порядок и особенности внедрения интерпретатора Mozilla SpiderMonkey в программный пакет GNTicker, в котором происходит моделирование. *Обобщенные сети*; *интерпретатор языка программирования*; Javascript; *Spidermonkey*

Обобщенные сети могут применяться в областях искусственного интеллекта, теории систем, медицины, экономики, транспортировки, химии и других. В последнее время этот инструмент приобретает популярность у многих исследователей.

Инструмент для моделирования процессов различной природы, обобщенные сети, расширение сетей Петри, появился в 1982 г.

Отличительной особенностью обобщенных сетей от других расширений сетей Петри является то, что описание сети включает некоторые логически изложенные условия, а также то, что токен является информационным носителем.

Основные исследования, касающиеся обобщенных сетей, принадлежат профессору К. Атанассову из Болгарской академии наук (София). Программное обеспечение для симуляции обобщенных сетей GNTicker позволяет моделировать обобщенную сеть, описание которой включает в себя скрипты на Lisp – подобном языке программирования для изложения логики сети. Проблемой внедрения интерпретатора скриптов в GNTicker ранее занимались его разработчики из Софийского университета «Св. Климент Охридски». В результате описание скриптов стало возможным на Lisp-подобном языке [1–3].

Контактная информация: (347) 273-79-67

Исследования частично поддержаны грантом РФФИ 09-07-00408-а «Распределенная интеллектуальная система поддержки принятия решений при выполнении проектов фундаментальных исследований сложных систем» и выполнялись в рамках хоз. договорной темы ИФ-ВК-09-01-03 «Исследование интеллектуальных технологий поддержки принятия решений и управления для сложных социально-экономических объектов».

Обобщенные сети стали привлекать все больше внимание исследователей из разных областей науки. Появилась необходимость использования более простого и понятного языка программирования в таком приложении для того, чтобы сделать процесс моделирования с помощью обобщенных сетей более простым, наглядным и доступным для освоения.

Данная статья посвящена вопросам внедрения такого языка в GNTicker. В качестве скриптового языка выбран JavaScript, рассматривается задача создания объектной модели обобщенных сетей в JavaScript и разработки модуля, который реализует эту модель, обеспечивает интерактивный обмен данными со средой GNTicker, а также обеспечивает поддержку жизненного цикла объектов в среде JavaScript.

1. ОСОБЕННОСТИ ОБОБЩЕННЫХ СЕТЕЙ И ПО ДЛЯ ИХ МОДЕЛИРОВАНИЯ

Обобщенные сети (ОС) – это разновидность сетей Петри, которые используются для моделирования сложных параллельных процессов. ОС состоит из трех типов элементов: позиции (places), токены (tokens) и переходы (transitions). Позиция может содержать в себе токены. Токен (или фишка) – это носитель информации, где информация определяется набором характеристик. Характеристика определяется названием и значением. Каждая позиция имеет ассоциированную характеристическую функцию, которая меняет характеристики поступающих в позицию токенов.

Токены двигаются согласно правилам (предикатам), определенным в переходах.

Простой пример обобщенной сети изображен на рис. 1. На рисунке используются обозначения:

Позиции – Input, Output, Iterate;

Переход – Controller.

В начальной конфигурации токен находится в позиции Input и имеет характеристику $I = 0$.

Предикатная функция перехода Controller проверяет наличие фишки в Input и разрешает ей переход в Iterate. При наличии фишки в Iterate, проверяет значение характеристики I, и если оно больше 10, разрешает фишке переход в Output, иначе разрешает переход в ту же позицию Iterate.

Характеристическая функция позиции Iterate увеличивает характеристику токена I на 1.

При выполнении сети фишка пройдет через позицию Iterate 11 раз (1 раз из позиции Input и 10 раз из самой же позиции Iterate) и попадет в позицию Output.

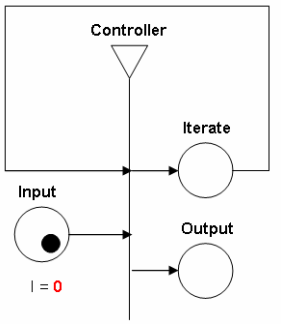

Рис. 1. Пример обобщенной сети

GNTicker – это программный пакет для симулирования обобщенных сетей. Это приложение загружает обобщенную сеть и исполняет ее. Оно моделирует движение токенов в сети и изменение их характеристик, выполнение характеристических функций, предикатов и функций слияния токенов. В итоге вся симуляция сети сохраняется в выходной XML-файл, содержащий набор событий произошедших при выполнении сети.

До внедрения интерпретатора JavaScript, GNTicker поддерживал специальный язык в стиле Lisp для программирования характеристических функций и предикатов – GNTCFL.

2. ВЫБОР СКРИПТОВОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ

Языка GNTCFL было достаточно, пока обобщенные сети не начали приобретать популярность у исследователей, в том числе у тех, которые не имеют опыта программирования и не знакомы с языками программирования вроде Lisp. Было принято решение внедрить в GNTicker другой язык программирования, более понятный обычному человеку, желающему имитировать какие-либо процессы на GNTicker.

Существует множество скриптовых языков программирования с различными синтаксисами и областью применения. Условно их можно поделить на встроенные в другое ПО средства для внутреннего использования (VBA, AutoLISP, ActionScript и т. п.) и доступные для внедрения в любые проекты (JavaScript, Lua и т. п.).

Большинство из доступных для внедрения языков основано на JavaScript. Эти языки, как правило, имеют различные расширения для удобства использования в тех или иных областях.

В силу того, что рассматриваемая задача не предполагает особых требований к скриптовому языку, был выбран JavaScript. Также этот выбор сделан из соображений простоты использования и освоения языка пользователями системы.

Пользователю системы, т. е. тому, кто моделирует обобщенную сеть, в среде JavaScript должны быть доступны структура и конфигурация сети.

- Структура сети:

– множество позиций. Каждая позиция характеризуется ее названием (идентификатором), приоритетом (числовое значение), именем характеристической функции, входной и выходной транзакцией;

– множество переходов. Переход характеризуется его названием (идентификатором), 2 списками: входных и выходных позиций, предикатной функцией.

- конфигурация сети. Определяется номером текущего шага, множеством фишек и их характеристик. Фишка характеризуется названием (идентификатором), позицией, в которой она находится в данный момент времени и набором разнотипных именных характеристик, значения которых могут изменяться характеристическими функциями.

Структура сети остается неизменной во время выполнения ОС, поэтому соответствующие элементы достаточно единовременно загрузить в среду JavaScript.

Конфигурации сети на каждом шаге отличаются друг от друга, поэтому необходимо реализовать механизм снабжения среды JavaScript объектами конфигурации обобщенной сети.

3. ВЫБОР ИНТЕРПРЕТАТОРА

В качестве интерпретатора был выбран API Spidermonkey от Mozilla (https://developer.mozilla.org/En/SpiderMonkey) [4]. Во-первых, он реализован на C++, как и сам проект GNTicker, что упрощает внедрение, во-вторых, он сопровожда-

ется хорошей документацией, и, наконец, он используется такими приложениями, как всемирно-известный браузер Mozilla FireFox, что гарантирует его высокую отказоустойчивость.

Компилятор/интерпретатор языка JavaScript spidermonkey – это набор функций для C++, с помощью которых среда выполнения скриптов может быть адаптирована под конкретные задачи.

Для того чтобы работать с компилятором, необходимо создать runtime (среда выполнения скриптов) и context (контекст выполнения скриптов). Runtime – это рабочая область, где хранятся используемые приложением переменные, объекты и контексты скриптов. Context – это область для хранения состояния выполнения скрипта, соответствующая одному потоку или скрипту. Связь runtime и context показана на рис. 2.

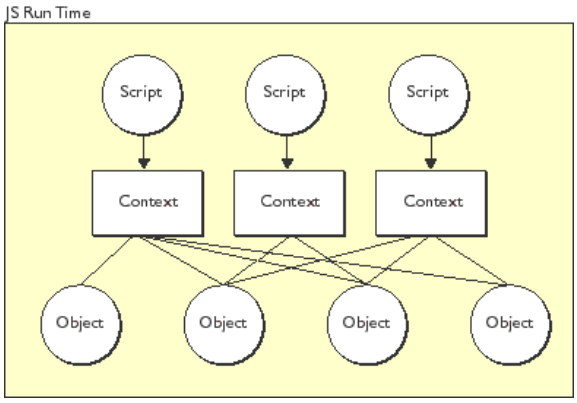

Рис. 2. Связь runtime и context в SpiderMonkey

Для создания runtime и context нужно вызвать API-функции:

- JSRuntime * JS_NewRuntime(uint32 maxbytes);

maxbytes – количество байт, выделяемых под хранение переменных и объектов JavaScript.

- JSContext * JS_NewContext(JSRuntime *rt, size_t stackchunksize);

Так же, после работы со средой JavaScript, необходимо уничтожить эти объекты с помощью функций:

```
void JS_DestroyRuntime(JSRuntime *rt);
void JS_DestroyContext(JSContext *cx);
```

Для инициализации объектно-ориентированной структуры среды JavaScript необходимо создать глобальный объект, к которому в последствии можно будет привязывать другие объекты, как его свойства. Для этого нужно описать объект JavaScript (JSObject *) и выполнить API-функцию:

```
JSBool JS_InitStandardClasses(JSContext *cx,
    JSObject *obj),
```

которая устанавливает объект obj глобальным, а так же инициализирует стандартные объекты в контексте cx: Array, Boolean, Date, Math, Number и String.

JavaScript – язык с автоматическим преобразованием типов данных. Типы данных в SpiderMonkey API: JSOBJECT, JSINT, JSDOUBLE, JSSTRING, JSBOOLEAN.

Все типы можно приводить к универсальному jsval и обратно.

Существует набор функций для определения и ручного преобразования типов.

Все стандартные функции этого языка принимают на вход переменные с любыми типами данных. В интерпретаторе содержатся механизмы для автоматического преобразования данных из одних типов в другие.

Основной способ для предоставления доступа к данным приложения из среды JavaScript-создание объектов, при вызове методов или обращении к свойствам которых, фактически выполняются функции, находящиеся непосредственно в приложении (так называемая перегрузка функций и свойств, wrapping). Таким образом можно по запросу из среды JavaScript возвращать данные из приложения в скрипт (чтение), а также выполнять соответствующие изменения в приложении по запросу скрипта (запись).

Любому классу в JavaScript можно назначить перехватчики событий. В перехватываемые события входят:

- Get (getter). Событие вызывается при попытке чтения любого свойства объекта данного класса, вне зависимости от того, существует ли свойство с таким идентификатором или нет. Позволяет назначить или подменить запрашиваемое значение.
- Set (setter). Событие вызывается при попытке изменения любого свойства объекта данного класса, вне зависимости от того, существует ли свойство с таким идентификатором или нет. Позволяет проанализировать присваиваемое значение, подменить его, или отменить изменение.
- Resolve. Событие вызывается при попытке чтения несуществующего свойства объекта данного класса. Позволяет назначить запрашиваемое значение. После такого запроса соответствующее свойство появится у объекта.
- Enumerate (Enumerator). Событие вызывается при выполнении конструкции for (a in x) где *x* – объект данного класса. Позволяет перечислять идентификаторы свойств, даже если некоторые из них отсутствуют в данный момент в

контексте среды. Во время исполнения скриптов SpiderMonkey с определенной периодичностью проверяет переменные в среде на предмет их использования скриптом. Если переменная перестает использоваться, она уничтожается сборщиком мусора для освобождения памяти.

Существует механизм защиты переменных и объектов среды от сборки мусора с помощью функции JS_AddRoot. Если объект (переменная) защищен, то все его свойства также будут автоматически защищены от сборки мусора.

4. ОБЪЕКТНАЯ МОДЕЛЬ ОБОБЩЕННЫЙ СЕТИ

Внедрение осуществляется в 2 этапа, содержащие соответствующие пункты:

1. Разработка независимого модуля, который обеспечивает интерактивный обмен данными со средой GNTicker, а также обеспечивает поддержку жизненного цикла объектов в среде JavaScript:

● разработка объектной модели обобщенных сетей в JavaScript;

● реализация механизмов доступа к данным в среде JS:

– механизм инициализации структуры ОС в JS;

– механизм снабжения среды JavaScript элементами конфигурации ОС;

– механизм обратной связи для присвоения новых значений характеристикам;

– механизмы преобразования данных из их представления в среде GNTicker в JavaScript и обратно.

● Реализация механизма запуска пользовательских скриптов по запросу;

● Реализация механизма управления сборкой мусора для корректной и эффективной работы с памятью.

2. Внедрение этого модуля в приложение GNTicker:

● реализация механизмов доступа к данным ОС из модуля:

– функции передачи модулю структуры и конфигурации ОС;

– функция присвоения новых значений характеристикам фишек;

● вызов необходимых механизмов модуля для работы с пользовательскими скриптами.

Объектная модель, с одной стороны, должна соответствовать логике структуры ОС, с другой – должна быть эргономична и интуитивно понятна пользователю.

Разработанная объектная модель обобщенной сети показана на рис. 3.

GN – головной объект, через который осуществляется доступ к сети.

Через этот объект можно получить доступ к номеру текущего шага и ко всем объектам конфигурации и структуры сети. Позиции и переходы представлены в виде статических объектов – массивов, а набор фишки с текущими характеристиками – динамических объектом – массивом, элементы которого генерируются на лету во время выполнения скрипта. Все объекты обобщенной сети имеют необходимые свойства и методы для написания характеристических функций.

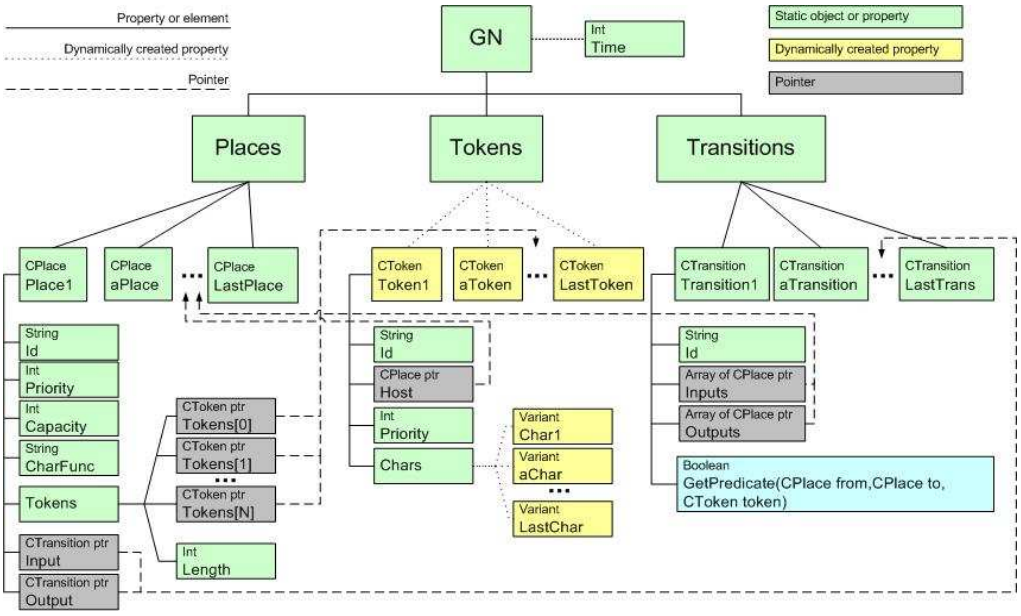

Рис. 3. Объектная модель обобщенной сети в JavaScript

5. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Программная реализация выполнена в двух модулях: GN_JS – реализующий функционал первого этапа и GN_JS_Linker – реализующий ункционал второго. GN_JS_Linker интегрирован в GNTicker, GN_JS – является независимым от GNTicker и использует GN_JS_Linker в качестве шлюза. Оба модуля реализованы на C++, так как GNTicker разработан на этом языке программирования.

Модуль GN_JS_Linker можно назвать шлюзом между GNTicker и модулем GN_JS, реализующим работу со средой JavaScript.

Он плотно интегрирован в GNTicker и предоставляет модулю GN_JS интерфейс для взаимодействия с ним. GN_JS_Linker включает в себя функции для получения структуры и текущей конфигурации сети из GNTicker и функции для установки новых значений характеристикам фишек в GNTicker.

Модуль GN_JS выполнен так, что для связи с GNTicker он использует только функции модуля Linker, не содержит каких-либо структур данных из GNTicker и не обращается к данным из него напрямую.

Таким образом, модуль является независимым от GNTicker и, в принципе, может быть использован вместе с другими симуляторами обобщенных сетей при условии, что для них будет реализован модуль Linker.

Модуль GN_JS содержит функции для построения структуры сети в среде JavaScript, функции для подгрузки конфигурации сети в среду JavaScript, функцию выполнения пользовательских скриптов в среде JavaScript, функции конвертирования данных в/из формата JavaScript, а также ряд служебных функций для инициализации среды, для завершения работы со средой, для обеспечения жизненного цикла объектов в среде и для обработки ошибок.

Схема и принцип работы модуля GN_JS изображены на рис. 4.

Реализованная структура классов для работы со средой JavaScript имеет вид, представленный на рис. 5.

Перехватчики событий типа Enumerate, Resolve, Getter и Setter используют функции модуля GN_JS_Linker для получения данных во время выполнения скрипта.

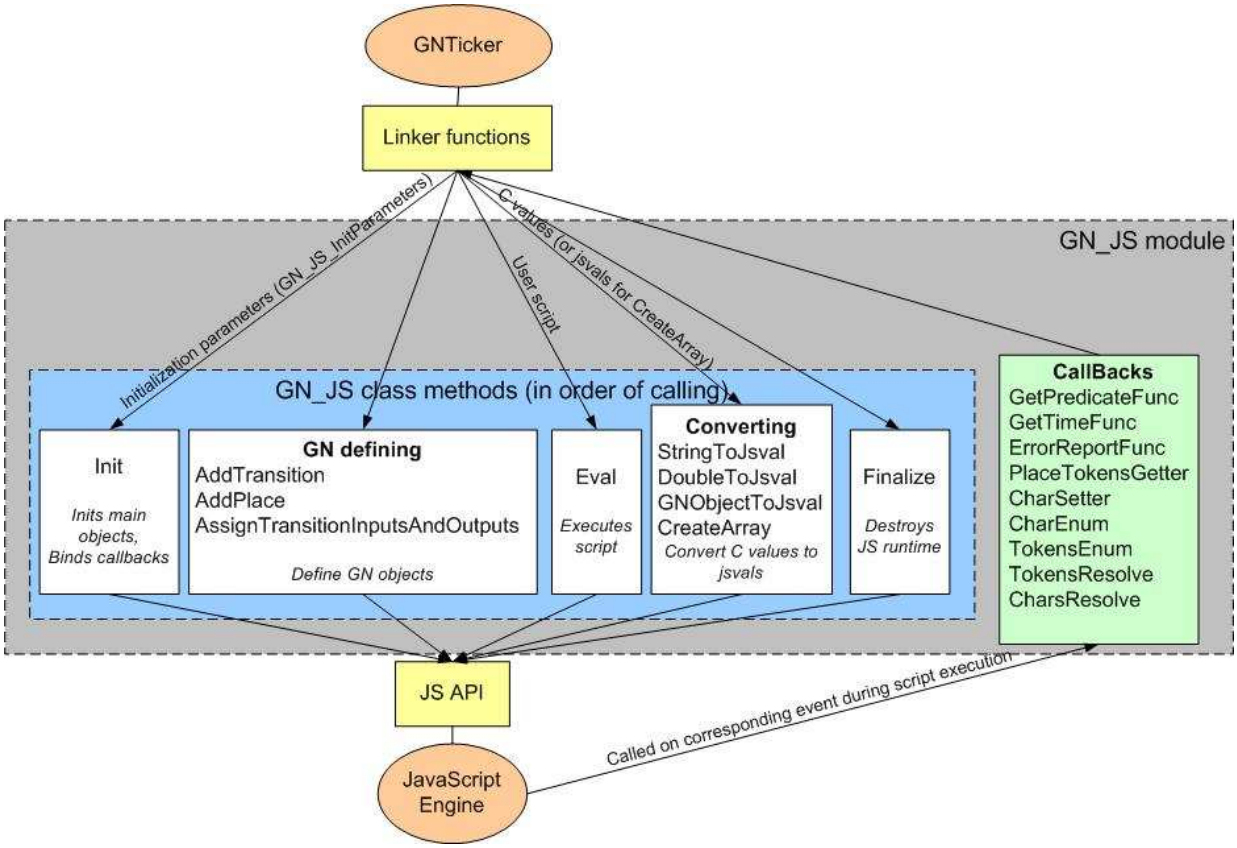

Рис. 4. Схема и принцип работы модуля GN_JS

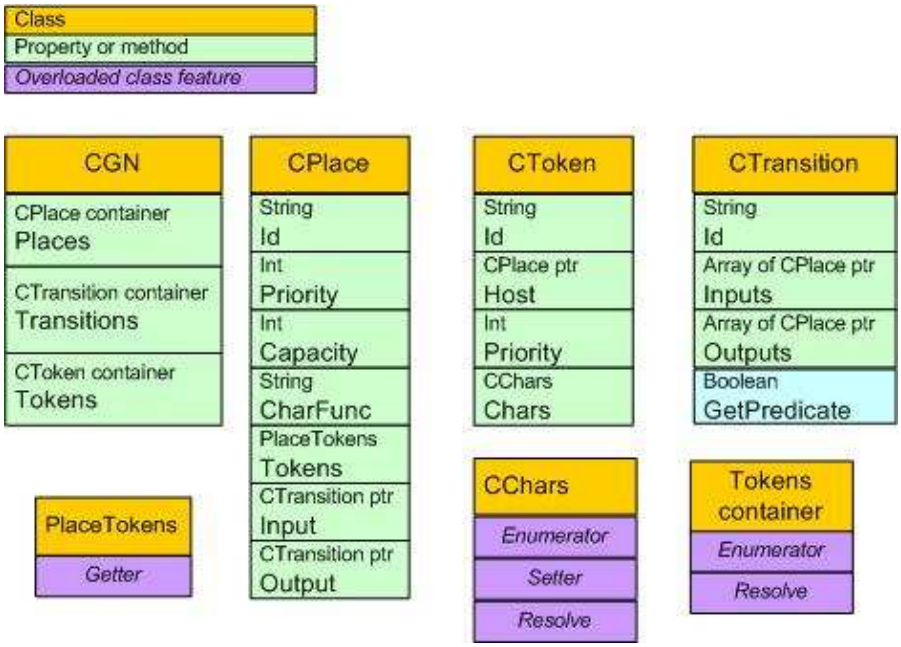

Рис. 5. Структура классов в среде JavaScript

6. РЕЗУЛЬТАТЫ ВНЕДРЕНИЯ

Итоговым результатом разработки и внедрения является новая версия программы GN Ticker способная обрабатывать модели обобщенных сетей с характеристическими функциями, реализованными на языке JavaScript.

Был разработан основной модуль, осуществляющий работу с интерпретатором JavaScript SpiderMonkey, спроектирована и реализована объектная модель для работы с обобщенными сетями. Было проведено необходимое тестирование этой системы, реализована система обработки ошибок.

Разработанный программный модуль

- позволяет программировать характеристические функции обобщенной сети на языке JavaScript;
- упрощает процесс моделирования и симулирования обобщенный сетей.

Таким образом, данный программный продукт может увеличить интерес к обобщенным сетям, так как его после внедрения интерпретатора JavaScript в использовании он стал более доступным для широкого круга пользователей.

Пример возможных функций для GNTicker на JavaScript:

```
function
  Adding_New_Chars_And_Objects_Example()
{
      print("\n------Example3:
       Adding new characteristics and
      GN.Tokens["Token1"].Chars["Obj"]=
      GN.Places["P1"];
       vtr=GN.Tokens["Token1"].Chars["Vtr"];
       vtr[3]=GN.Places["P1"];
       GN.Tokens["Token1"].Chars["Vtr"]=vtr;
}

function Show_Places_Example()
{

      print("\n--------------------
      Example5:Show Places----------\n");
      print("\nPlaces\n----------\n");
      for(i in GN.Places)
      {
            place=GN.Places[i];
            print( "\n\nId="+place.Id);
            print( "\nPriority="+place.Priority);
            print( "\nCapacity="+place.Capacity);
            print( "\nCharFunc="+place.CharFunc);
            if(place.Input)
            print( "\nInput.Id="+place.Input.Id);
            if(place.Output)
            print( "\nOutput.Id="+place.Output.Id);
            print( "\nTokens in Place:")
            for(j=0;j<place.Tokens.Length;j++)
            {
                  print("  \n
                  Token.Id="+place.Tokens[j].Id);
            }
      }
}
GN.Tokens["Token1"].Chars["NewStr"]="New String";
```

ЗАКЛЮЧЕНИЕ

Рассмотрены вопросы внедрения скриптового языка в симулятор обобщенных сетей, задача создания объектной модели обобщенных сетей в среде JavaScript, а также концепция модуля, реализующего эту модель и обеспечивающего интерактивный обмен данными со средой эмулятора обобщенных сетей.

Работа выполнялась в рамках договора о сотрудничестве между кафедрой ВМиК УГАТУ и институтом автоматизации факультета электроники и информатики технического университета Дрездена (Германия), в рамках совместного исследовательского проекта «Entwurfsbegleitende Funktions- und Leistungstests mit Generalisierten Netzen» («Сопровождающее тестирование функциональности и производительности проекта с помощью Обобщенных сетей»).

СПИСОК ЛИТЕРАТУРЫ

1. **Atanassov K.** Generalized Nets, World Scientific: Singapore – New Jersey – London, 1991.

2. Официальный ресурс обобщенных сетей [Электронный ресурс] (http://www.generalised.net).

3. Официальный ресурс GNTicker [Электронный ресурс] (http://gnticker.generalised.net/).

4. Официальный ресурс Spidermonkey [Электронный ресурс] (http://www.mozilla.org/js/spidermonkey).

ОБ АВТОРАХ

Янчек Клаус, управляющий Инта Автоматизации, возглавляет факульт. Электротехники и Инф. Технологий в Техн. Ун-те Дрездена. Иссл. в обл. проектирования систем автоматизации, телемеханизации, мобильной робототехники, навигации, обработки оптической информации.

Койчева Евелина, сотр. факульт. Электротехники и Инф. Технологий Техн. Ун-та Дрездена. Дипл. инж.

Хазанкин Роман Вениаминович, асп. каф. вычисл. матем. и кибернетики УГАТУ. Дипл. инж. по матем. обеспечению и администрированию инф. систем (УГАТУ, 2007).

Морозов Андрей Михайлович, аспирант той же каф. Дипл. инж. по матем. обеспечению и администрированию инф. систем (УГАТУ, 2007).