УДК 004.05

Генерация пользовательского интерфейса на основе технологии распределенной виртуальной среды

И.О. Шальнев – аспирант, программист,
лаборатория автоматизации научных исследований,
Санкт-Петербургский институт информатики и автоматизации РАН
E-mail: elias.shalnev@gmail.com
А.Ю. Аксенов – к.т.н., ст. науч. сотрудник,
лаборатория автоматизации научных исследований,
Санкт-Петербургский институт информатики и автоматизации РАН
E-mail: a_aksenov@iias.spb.su

Аннотация

Постановка проблемы. Развитие современных информационных систем определяется насущными требованиями общества и рынка. Увеличивающиеся объемы обрабатываемой информации, необходимость постоянного доступа к информационным системам, возникающая реальность автономных систем обостряет проблему взаимодействия компонентов сети и их взаимной интеграции. Каждый человек фактически постоянно включен в пространство личных и корпоративных информационных взаимодействий, непосредственно принимает участие в формировании информационных потоков и управленческих сигналов, взаимодействует с другими участниками системы. Огромное число информационных систем создает препятствие для единовременного и комфортного их использования, не говоря уже о том, что все системы не уместить в одном устройстве. В качестве решения еще в 1960-х годах был предложен терминальный подход в качестве устройства доступа к информационной системе.
Цель. Разработать новый подход к реализации приложений с терминалом для решения проблемы, поставленной современным цифровым пространством.
Результаты. Генерация пользовательского интерфейса на основе использования технологии распределенной виртуальной среды, на взгляд авторов, показала свою эффективность как на концептуальном уровне, так и в производительности. Интерпретация управляющих команд виртуальной машины оказалось эффективней процессинга текстового формата HTML, что делает такой подход конкурентоспособным. В предложенном подходе пользовательский интерфейс генерируется динамически на основании работы сервиса приложения, находящегося на другом сетевом узле. Таким образом, пользователь имеет единую точку входа к массиву информационных систем.
Практическая значимость. Предлагаемое решение найдет свое применение во многих инфокоммуникационных мобильных системах, особенно в системах, использующих для своего реконфигурирования технологию активных данных.

Ключевые слова

Языковые виртуальные машины, терминальные системы, генерация интерфейса, объектное взаимодействие, клиент-серверное взаимодействие.

Представленные результаты исследований получены при поддержке бюджетной темы № 0073-2019-0005 и гранта РФФИ № 16-29-12965.

DOI: 10.18127/j20700814-201905-06

Введение

Развитие современных информационных систем определяется насущными требованиями общества и рынка. Изначально программы были простыми и решали маленькую, сугубо практическую, часто вычислительную задачу, например, из области математики. С развитием компьютерных технологий стали появляться задачи, которые принципиально невозможно решить в рамках одной изолированной вычислительной системы. Необходимость интерфейсного доступа группы пользователей к мейнфрейму в 1960-х годах определила пути развития терминальных систем. Нужда в совместном использовании файлов и записей в них, упорядочении, поиске и сортировке сформировала вектор развития систем управления базами данных. Необходимость доступа к структурированным документам и справочным системам предвосхитила появление гипертекста [1] (Теодор Нельсон) и определила направление создания современной сети таких документов – World Wide Web (WWW).

Современный глобальный экономический и социальный мир ставит задачи иного уровня сложности. Увеличивающиеся объемы обрабатываемой информации, необходимость постоянного доступа к информационным системам, возникающая реальность автономных систем обостряет проблему взаимо-

действия компонентов сети и их взаимной интеграции. Каждый человек фактически постоянно включен в пространство личных и корпоративных информационных взаимодействий, непосредственно принимает участие в формировании информационных потоков и управленческих сигналов, взаимодействует с другими участниками системы. Причем следует подчеркнуть, что под «другими участниками взаимодействия» будем понимать не только людей, но и разные информационные системы, базы данных, системы искусственного интеллекта, взаимодействующие автономные системы (массивы датчиков, контроллеров, беспилотные транспортные средства, роботехнические комплексы).

Ц е л ь р а б о т ы – разработать новый подход к реализации приложений с терминалом для решения проблемы, поставленной современным цифровым пространством.

Описание подхода

В данной статье описан подход, предлагающий динамическое генерирование интерфейса, способного осуществить реализацию терминального доступа к информационной системе через пользовательское (рабочее) устройство. Рабочее устройство в таком случае не имеет собственных алгоритмов и моделей целевой задачи, оно лишь является интерпретатором визуального отображения. Реальная же бизнес-логика находится в серверном окружении, которое помимо всего прочего определяет пользовательский интерфейс.

Решения, основанные на взаимодействии распределенных виртуальных систем [2–4], оказываются чрезвычайно гибкими для использования на максимально широком спектре прикладных задач [5]. При этом распределенная виртуальная система является набором установленных на каждом сетевом узле участников программной среды языковых виртуальных машин [6–8].

Вся предметная область задачи может быть представлена набором динамически взаимодействующих объектов, существующих и функционирующих в распределенной виртуальной среде. Описываемый объект реализуется как единая совокупность двух связанных между собой компонентных объектов, раздельно существующих по разные стороны сетевого взаимодействия. С одной стороны, существует реализация объекта (его функциональная часть), а с другой стороны, существует программный интерфейс.

Компонентные объекты существуют в пространстве виртуальной среды, где обмен управляющими командами представлен последовательностью команд языковой виртуальной машины, которые в процессе исполнения изменяют как состояние самой виртуальной среды, так и состояние объектов [9].

В процессе работы программы у объектов могут возникать сигналы, оповещающие систему о возникновении события. Для реализации механизма оповещения о возникшем событии в программном коде может использоваться, например, событийно-ориентированная архитектура фреймворка Qt. Возникающие в ходе работы объекта сигналы соединяются с обработчиками. Данная архитектура отлично вписывается в асинхронную модель сетевых технологий, в которой результат вызванной функции может вернуться через некоторое время.

Программный код, исполняемый на разных узлах сети, при таком подходе можно считать единым и консистентным. Здесь компонентные объекты является объединяющим звеном между программным кодом узлов в сети. Компонентные объекты работают с распределенной виртуальной средой, которая, с одной стороны, занимается ассемблированием и отправкой управляющей информации, а с другой стороны, исполнением и приемом этой информации. Распределенная виртуальная среда при обнаружении сообщения о событии функционального объекта запускает процесс эмуляции события объекта программного интерфейса. Описанная методология подхода позволяет разработчику использовать возникшее единство компонентных объектов как целостный программный код, распределенный на разных узлах сети.

Экспериментальные результаты

Рассмотрим работу распределенной виртуальной среды на примере построения пользовательского интерфейса на удаленном рабочем устройстве. Программный код данной программы будет разделен таким образом, чтобы рабочее устройство имело объекты, содержащие исключительно интерфейсную логику. Целевая логика системы в таком случае находится на другом сетевом узле. Имена объектов стороны программного интерфейса начинаются с буквы V (от слова virtual).

Для демонстрации примера работы протокола создадим набор комплиментарных классов, описывающих окно пользовательского интерфейса, в котором будут следующие графические элементы:

1) кнопка;

2) надпись или метка (label) – элемент пользовательского интерфейса, который отображает текст на форме окна, как правило используемый для пояснений каких-либо текстовых полей и прочих виджетов;

3) поле однострочного ввода (line edit).

Логика работы демонстрационной программы следующая: по нажатию на кнопку взять из полей ввода имя и фамилию и поместить их в отдельную надпись (label).

Определим класс MyWindow, который наследуется от класса VWidget, хранящего методы для управления удаленным виджетом, и определим обработчик onClickedButton(), который будет обрабатывать событие нажатия на удаленную кнопку, находящуюся на рабочем устройстве (рис. 1).

```
class MyWindow : public VWidget
{
  Q_OBJECT
public:
  MyWindow(Connection* owner);
private:
  VLineEdit *mFamilyNameLineEdit;
  VLineEdit *mNameLineEdit;
  VLabel *mInputResult;
protected slots:
  void onClickedButton();
};
```

Рис. 1. Листинг определения класса MyWindow

Fig. 1. Listing of MyWindow class definition

Определим пользовательский интерфейс программы в конструкторе созданного класса. Для того чтобы команды, собирающиеся в процессе работы вызова методов программного интерфейса, ассемблировались в один сетевой пакет, был разработан объект PacketAssembler.

```
MyWindow::MyWindow(Connection *owner)
  : VWidget(owner)
{
  PacketAssembler packet(owner->getProcessor()->getScriptSender());
  VVBoxLayout *Layout = new VVBoxLayout(owner, this);
  //Создание надписи “Введите фамилию”
  VLabel *FamilyNameLabel = new VLabel (owner, QString("Введите фамилию:"), this);
  Layout->addWidget(FamilyNameLabel);
  //Создание поля ввода для введения фамилии
  mFamilyNameLineEdit = new VLineEdit(owner, this);
  Layout->addWidget(mFamilyNameLineEdit);
  //Создание надписи “Введите имя”
  VLabel *NameLabel = new VLabel(owner,QString("Введите имя:"),this);
  Layout->addWidget(NameLabel);
  //Создание поля ввода для введения имени
  mNameLineEdit = new VLineEdit(owner, this);
  Layout->addWidget(mNameLineEdit);
  //Создание кнопки
  VPushButton *NameButton = new VPushButton(owner, QString("Подтвердить"), this);
  Layout->addWidget(NameButton);
  //Соединение сигнала “clicked” исходящего от удаленной кнопки.
  vconnect(NameButton, SIGNAL(clicked()),
           this,       SLOT(onClickedButton()));
  //Создание поля для результатов
  mInputResult = new VLabel(owner,this);
  Layout->addWidget(mInputResult);
}
```

Рис. 2. Листинг определения пользовательского интерфейса

Fig. 2. Listing of user interface definition

Создадим экземпляр класса VVBoxLayout, отвечающий за горизонтальное размещение добавленных в него объектов (рис. 2).

Затем реализуется обработчик события нажатия на кнопку (рис. 3).

```
void MyWindow::onClickedButton()
{
  QString string;
  connect(mFamilyNameLineEdit, &VLineEdit::textReceived,
          [string, this](const QString text) mutable
  {
    QObject::disconnect(mFamilyNameLineEdit, &VLineEdit::textReceived, nullptr, nullptr);
    string.append(text).append(' ');
    connect(mNameLineEdit, &VLineEdit::textReceived,
            [string, this](QString text) mutable
    {
      QObject::disconnect(mNameLineEdit, &VLineEdit::textReceived, nullptr, nullptr);
      string.append(text);
      mInputResult->setText(string);
    });
    mNameLineEdit->text();
  });
  mFamilyNameLineEdit->text();
}
```

Рис. 3. Листинг реализации обработчика события удаленного объекта

Fig. 3. Listing of the realization of remote object event handler

Так как функция text() по своей сути это удаленный вызов метода, то ждать синхронного ответа не рационально, так как он может задерживать выполнение процесса. Сетевое взаимодействия требует асинхронной обработки по причине того, что ожидание ответа требует остановки потока. Поэтому во время приема ответа со стороны терминала виртуальная среда определяет, какому объекту принадлежит пришедший результат, и выпускает сигнал о событии получения текста из формы, который, в свою очередь, перехватывает обработчик.

Поскольку функция text() класса VLineEdit асинхронна, то необходимо гарантировать, что

первый пакет (в данном случае фамилия) придет на сторону программного интерфейса ранее, чем второй пакет (имя), поэтому именно в лямбда-функции, обрабатывающей поступление текста из поля ввода фамилии, делается запрос на получение текста из поля ввода имени. Во время того, как текст с именем придет на сторону программного интерфейса, установим собранный воедино текст в метку (label), находящуюся ниже кнопки.

В данном примере, описанном в классе MyWindow, рассмотрено полноценное сетевое приложение, определяющее алгоритм программы и презентационную логику на одном сетевом узле, тогда как его графический вид отображается на другом. Несмотря на то, что виртуальные методы вызываются на стороне одного сетевого узла, в действительности вызов методов функциональных объектов происходит на стороне другого.

Для тестирования производительности произведен эксперимент, включающий в себя создание тысяч компонентных объектов и вызовы тысяч удаленных методов. Для этого в цикле из 1000 итераций создадим те же графические элементы, что были приведены выше, и поместим их в область прокрутки (scroll area) для их корректного отображения.

На рис. 4 показан реализованный графический интерфейс.

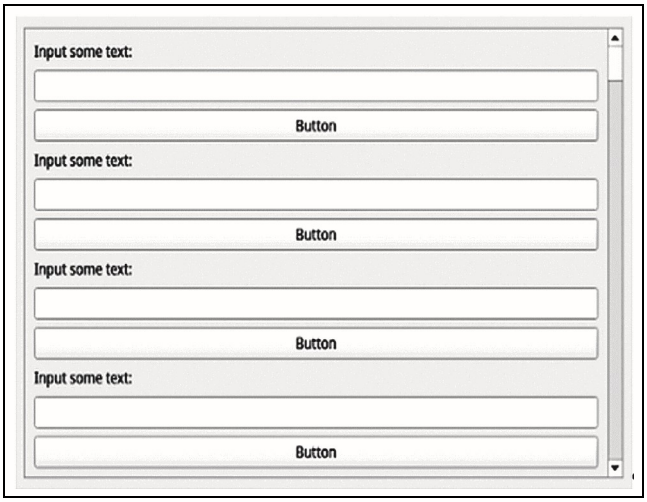

Рис. 4. Интерфейс эксперимента
Fig. 4. User interface of experiment

Эксперимент показал, что для интерпретации графической сцены, состоящей из нескольких тысяч команд программного интерфейса, виртуальному процессору понадобилось 5 мс. Размер передаваемых данных составил 27009 байт.

Также произведено сравнение производительности отрисовки графической сцены с типовыми подходом к решению аналогичной задачи, используемым в web, где файл с расширением .html вкупе с протоколом HTTP интерпретируется браузером и отрисовывается в удаленный пользовательский интерфейс. Важно отметить, что подходы к задаче отображения графической сцены протокола данной работы и протокола HTTP отличаются принципиально. Протокол HTTP оперирует в основном двумя функциями GET и POST. По запросу клиента сервер отправляет html-файл вместе с таблицей стилей и скриптами, которые процессируются браузером. Язык html является декларативным, в то время как подход, описанный в данной работе, является императивным и объектно-ориентированным, позволяющим отойти от задач графического отображения к более общим задачам программирования. Создадим аналогичный пользовательский интерфейс из трех тысяч графических элементов при помощи средств языков html и JavaScript:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Test</title>
</head>

<body>
</body>

<script>
  var i=0;

  while(i<1000)
  {
```

```
    var LabelDiv = document.createElement('div');
    var label = document.createElement('label');
    label.innerHTML = 'Input some text:';
    LabelDiv.appendChild(label);
    document.body.appendChild(LabelDiv);

    var InputDiv = document.createElement('div');
    var input = document.createElement('input');
    InputDiv.appendChild(input);
    document.body.appendChild(InputDiv);

    var ButtonDiv = document.createElement('div');
    var button = document.createElement('button');
    button.innerHTML = 'Submit';
    ButtonDiv.appendChild(button);
    document.body.appendChild(ButtonDiv);

    i++;
  }
</script>
</html>
```

Из рис. 5 видно, что общее время обработки тестового html-документа равно 493 мс, что почти на два порядка больше соответственной обработки управляющей информации распределенной виртуальной системы. Замеры производительности html-документа производились в браузере Google Chromium версии 65.0.3325.181.

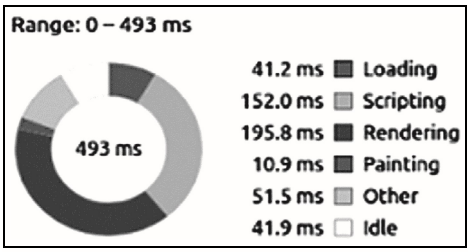

Рис. 5. Диаграмма производительности отрисовки примера в браузере

Fig. 5. Diagram of the performance of the example in the browser

Размер html-файла равен 993 байта, что приблизительно в 27 раз меньше соответствующего интерпретируемого кода виртуальной среды. На самом деле в примере html была применена небольшая хитрость, влияющая на производительность обработки документа и экономию трафика. Генерация элементов происходит в скрипте, встроенном в html-код. Такой подход является неприемлемым для большинства реальных задач, потому что обычно в html-файл помещают данные, хранящиеся в какой-либо системе хранения данных (базе данных, файлах и пр.). Этот факт делает невозможным подобную генерацию DOM из скрипта. Стандартным сценарием является генерация Back End'ом всего html-файла, хранящего все теги с информацией в них. Промоделируем такой сценарий генерацией html-файла, состоящего из 1000 элементов надписей (label), кнопок и полей ввода:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Test</title>
</head>

<body>

   <div>
      <label for="uname">Input some text: </label>
    </div>
    <div>
      <input type="text" id="uname" name="name">
    </div>
    <div>
```

```
      <button>Submit</button>
    </div>

     <div>
      <label for="uname">Input some text: </label>
    </div>
    <div>
      <input type="text" id="uname" name="name">
    </div>
    <div>
      <button>Submit</button>
    </div>
    <!-- Оставшиеся 2994 элемента -->
</body>
</html>
```

Размер такого документа составляет 214 825 байт, а оценка производительности, проиллюстрированная на рис. 6, показывает, что общее время обработки документа приближается к 593 мс, что еще медленнее, чем в предыдущем примере.

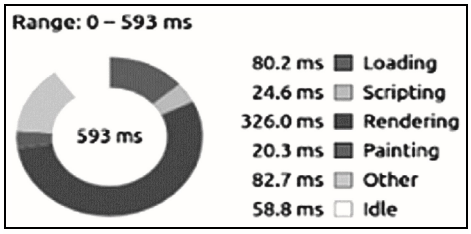

Рис. 6. Диаграмма производительности отрисовки второго html-документа

Fig. 6. Diagram of the performance of the second html document rendering

Заключение

Таким образом, можно с уверенностью сказать, что предложенная технология может не только быть конкурентоспособной в части технологий программирования, но и, прежде всего, превосходит типовое web-решение в скорости и может стать основой для разработки клиент-серверных приложений с развитым пользовательский интерфейсом. Эксперимент показал, что решение на базе предложенных технологий предоставит мобильному пользователю комфорт, сопоставимый с обычными десктоп-приложениями, при этом обладающий гибкостью, сравнимой с решениями на тонком клиенте. Предлагаемое решение найдет свое применение во многих инфокоммуникационных мобильных системах, особенно в системах, использующих для своего реконфигурирования технологию активных данных [10].

Литература

1. DEEP HYPERTEXT: THE XANADU® MODEL. URL = http://xanadu.com/xuTheModel/ (дата обращения 15.08.2019).
2. *Wei Chen, Weixia Xu, Zhiying Wang, Qiang Dou, Baokang Zhao* A Formalization of An Emulation based Co-Designed Virtual Machine // Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. 2011. P. 164–168.
3. *Кулешов С.В.*, *Цветков О.В.* Активные данные в цифровых программно-определяемых системах // Информационно-измерительные и управляющие системы. 2014. Т. 12. № 6. С. 12–19.
4. *Александров В.В.*, *Кулешов С.В.* Этерификация и терминальные программы // Информационно-измерительные и управляющие системы. 2008. Т. 6. № 10. С. 50–53.
5. *Шальнев И.О.* Реализация объектного взаимодействия с тонким клиентом на основе языковой виртуальной машины // Материалы XVI Санкт-Петербургская Междунар. конф. «Региональная информатика (РИ-2018)». СПб.: СПОИСУ. 24–26 октября 2018. С. 299–301.
6. *Sudha M.*, *Harish G.M.*, *Nandan A.*, *Usha J.* Performance Analysis of Kernel-Based Virtual Machine // International Journal of Computer Science and Information Technology. 2013. V. 5. P. 137–144.
7. Regina Preciado What Managed Runtime Environments (MRTEs). URL = https://software.intel.com/en-us/articles/what-managed-runtime-environments-mrtes-mean-to-you (дата обращения 15.08.2019).
8. Managed Runtime Technologies // Intel Technology Journal. 2003. V. 7. № 1. URL = https://software.intel.com/sites/default/files/m/d/6/a/vol7iss1_managed_runtime_technologies.pdf (дата обращения 15.08.2019).
9. *Шальнев И.О.* Подход к построению распределенных систем на основе балансировки объема исполняемого кода между сетевыми узлами // Материалы 4-й Междунар. научной конф. «Технологическая перспектива в рамках евразийского пространства: новые рынки и точки экономического роста». 2019. С. 151–158.
10. *Kuleshov S.V.*, *Zaytseva A.A.*, *Aksenov A.Y.* The conceptual view of unmanned aerial vehicle implementation as a mobile communication node of active data trans-mission network // International Journal of Intelligent Unmanned Systems. 2018. № 6(4). P. 174–183.

Поступила 18 июля 2019 г.

UDC 004.05

User interface generation based on distributed virtual environment technology

I.O. Shalnev – Post-graduate Student, Programmer,
St. Petersburg Institute for Informatics and Automation of RAS
E-mail: elias.shalnev@gmail.com
A.Yu. Aksenov – Ph.D.(Eng.), Senior Research Scientist,
St. Petersburg Institute for Informatics and Automation of RAS
E-mail: a_aksenov@iias.spb.su

Abstract

The issue of rapid and convenient access to a number of information system is exist. Another problem is that these plenty has to be integrated to each other to build one uniform information system.
The basic idea is to create terminal technology which is only the interpreter of visual presentation. There is no use to keep algorisms and business logic in a user working device. It is hard to maintain, update and most of all it is impossible to hold all created information systems in a single device. That's why the terminal access is essential.
This paper describes dynamic generation of user interface approach. The server infrastructure keeps not only the business logic but defines user interface. The distributed virtual environment interaction turned out to be a good solution for terminal system with user interface generation. In this research succeeded.
Problem statement: The issue of rapid and convenient access to a number of information system is exist. Another problem is that these plenty has to be integrated to each other to build one uniform information system. The development of modern information systems is determined by the urgent requirements of society and the market. The increasing volumes of processed information, the need for constant access to information systems, the emerging reality of autonomous systems aggravates the problem of interaction between network components and their mutual integration. Each person is actually constantly included in the space of personal and corporate information interactions, directly takes part in the formation of information flows and managerial signals, interacts with other participants in the system. As a solution a terminal approach was proposed as a device for accessing the information system.
Purpose: A huge number of information systems creates an obstacle to their simultaneous and comfortable use, not to mention the fact that not all systems can fit in one device. Search and development of a new approach to the implementation of applications with a terminal is one of the solutions to the problem posed by modern digital space.
Results: In our opinion, the generation of a user interface based on the use of technology of a distributed virtual environment has shown its effectiveness both at a conceptual level and in performance. The interpretation of the control commands of the virtual machine turned out to be more efficient than the processing of the HTML text format, which makes this approach competitive. In the proposed approach, the user interface is dynamically generated based on the operation of the application service located on another network node. Thus, the user has a single entry point to the array of information systems.
Practical relevance: The proposed solution will find its application in many infocommunication mobile systems, especially in systems using active data technology for their reconfiguration.

Keywords

Language virtual machines, terminal systems, interface generation, object interaction, client-server interaction.

The presented research results were obtained with the support of budget theme № 0073-2019-0005 and RFBR grant № 16-29-12965.

DOI: 10.18127/j20700814-201905-06

References

1. DEEP HYPERTEXT: THE XANADU® MODEL. URL = http://xanadu.com/xuTheModel/ (data obrashcheniya 15.08.2019).
2. *Wei Chen, Weixia Xu, Zhiying Wang, Qiang Dou, Baokang Zhao* A Formalization of An Emulation based Co-Designed Virtual Machine. Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. 2011. P. 164–168.
3. *Kuleshov S.V.*, *Tsvetkov O.V.* Aktivnye dannye v tsifrovykh programmno-opredelyaemykh sistemakh. Informatsionno-izmeritelnye i upravlyayushchie sistemy. 2014. T. 12. № 6. S. 12–19. (In Russian).
4. *Aleksandrov V.V.*, *Kuleshov S.V.* Eterifikatsiya i terminalnye programmy. Informatsionno-izmeritelnye i upravlyayushchie sistemy. 2008. T. 6. № 10. S. 50–53. (In Russian).
5. *Shalnev I.O.* Realizatsiya obieektnogo vzaimodeistviya s tonkim klientom na osnove yazykovoi virtualnoi mashiny. Materialy XVI Sankt-Peterburgskaya Mezhdunar. konf. «Regionalnaya informatika (RI-2018)». SPb.: SPOISU. 24–26 oktyabrya 2018. S. 299–301. (In Russian).
6. *Sudha M.*, *Harish G.M.*, *Nandan A.*, *Usha J.* Performance Analysis of Kernel-Based Virtual Machine. International Journal of Computer Science and Information Technology. 2013. V. 5. P. 137–144.
7. Regina Preciado What Managed Runtime Environments (MRTEs). URL = https://software.intel.com/en-us/articles/what-managed-runtime-environments-mrtes-mean-to-you (data obrashcheniya 15.08.2019).
8. Managed Runtime Technologies. Intel Technology Journal. 2003. V. 7. № 1. URL = https://software.intel.com/sites/default/files/m/d/6/a/vol7iss1_managed_runtime_technologies.pdf (data obrashcheniya 15.08.2019).
9. *Shalnev I.O.* Podkhod k postroeniyu raspredelennykh sistem na osnove balansirovki obieema ispolnyaemogo koda mezhdu setevymi uzlami. Materialy 4-i Mezhdunar. nauchnoi konf. «Tekhnologicheskaya perspektiva v ramkakh evraziiskogo prostranstva: novye rynki i tochki ekonomicheskogo rosta». 2019. S. 151–158. (In Russian).
10. *Kuleshov S.V.*, *Zaytseva A.A.*, *Aksenov A.Y.* The conceptual view of unmanned aerial vehicle implementation as a mobile communication node of active data trans-mission network. International Journal of Intelligent Unmanned Systems. 2018. № 6(4). P. 174–183.