Перетворимо останню нерівність таким чином:

$$2n(1-p)^2+(1-p)^2<(n+1)^2.$$

$$(1-p)^2(2n+1)<(n+1)^2$$

$$\frac{(2n+1)}{(n+1)^2}<\frac{1}{(1-p)^2};$$

$$1-\frac{n^2}{(n+1)^2}<\frac{1}{(1-p)^2}. \qquad (20)$$

Права частина нерівності (20) при будь-якому натуральному n завжди менше одиниці, а права при будь-яких припустимих значеннях p $(0\le p\le 1)$ завжди *не* менше одиниці. Нерівність (17) доведена, а тому доведена і від'ємна визначеність матриці (14).

Рішення задачі (2)-(3) $x_1^*=x_2^*=\cdots x_{n+1}^*=\frac{1}{n+1}$ свідчить про справедливість твердження, що найбільший приріст функціональної надійності системи досягається при симетричній установці перемичок, тобто у випадку, коли n перемичок розділяють кожний з двох паралельних трубопроводів на $(n+1)$ рівну частину. Однак при надмірному збільшенні числа перемичок n при незмінних значеннях p_a і p додавання нових перемичок може призвести до зворотного ефекту.

1.Ильин Ю.А. Надёжность водопроводных сооружений и оборудования. – М.: Стройиздат, 1985. – 240 с.

2.Самойленко М.І., Гавриленко І.О. Функціональна надійність трубопровідних транспортних систем. – Харків: ХНАМГ, 2009. – 184 с.

Отримано 14.04.2010

УДК 681

Б.И.ПОГРЕБНЯК, канд. техн. наук
Харьковская национальная академия городского хозяйства

ИНТЕГРАЦИЯ MICROSOFT OFFICE И INTERNET EXPLORER ДЛЯ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ДИСТАНЦИОННОГО УПРАВЛЕНИЯ БОЛЬШИМИ ИНЖЕНЕРНЫМИ СЕТЯМИ

На языке программирования VBA рассматриваются особенности программного управления браузером Internet Explorer из приложений Microsoft Office с целью получения информации из Web-страниц для повышения эффективности дистанционного управления большими инженерными сетями.

На мові програмування VBA розглядаються особливості програмного управління браузером Internet Explorer з додатків Microsoft Office з метою отримання інформації з

Web-сторінок для підвищення ефективності дистанційного керування великими інженерними мережами.

In programming of VBA language the features of programmatic management the browser of Internet Explorer are examined from the applications of Microsoft Office with the purpose of receipt of information from Web-pages for the increase of efficiency of remote-control large engineering networks.

Ключевые слова: ActiveX-Automation, OLE-Automation, сервер автоматизации, клиент автоматизации, раннее связывание, позднее связывание, VBA, объектная переменная, инженерная сеть.

Одной из замечательных особенностей многих приложений операционной системы Microsoft Windows является возможность управления одним приложением из другого. В частности, такими свойствами обладают все приложения знаменитого пакета Microsoft Office [1], а также многие браузеры, например, Microsoft Internet Explorer. Такое взаимодействие осуществляется при помощи средства Автоматизации (Automation) технологии ActiveX (старое название – OLE-Automation). При этом управление по технологии ActiveX-Automation браузером Internet Explorer из приложений Microsoft Office существенным образом отличается от аналогичного взаимодействия других приложений. Особенностям такого взаимодействия приложений на языке программирования VBA посвящена настоящая статья.

Для эффективного управления другим приложением необходимо знать, как к нему подключиться, а также его объектную модель. Технология ActiveX-Automation позволяет одному приложению программно управлять объектами другого приложения, в том числе с помощью VBA. Приложение, которое управляет объектами другого приложения, называется *контроллером автоматизации* (ActiveX Automation Controller), или *ActiveX-клиентом*, или *ведущей прикладной системой*. Приложение, которое предоставляет свои объекты для манипуляции, называется *сервером автоматизации* (ActiveX Automation Server) или *целевым приложением* [4, 5].

Некоторые приложения могут обладать как свойствами контроллера автоматизации, так и свойствами сервера автоматизации, т.е. они в одних условиях могут предоставлять свои объекты для управления другим приложениям, а в других – управлять другими приложениями. В частности, такими свойствами обладают приложения пакета Microsoft Office. Другие же приложения могут быть либо только клиентами автоматизации, либо только серверами – все зависит от того, какие свойства в них были заложены при разработке [6].

При программном управлении одним приложением из другого вначале необходимо установить между ними связь (подключение) че-

рез *библиотеку типов*. Существует два способа установки такого подключения: *раннее* или *статическое связывание* (*late binding*) и *позднее*, или *отложенное*, или *динамическое связывание* (*early binding*) [2]. В любом случае подключение устанавливается путем создания объектной переменной, ссылающейся на интересующее приложение, или конкретный объект в пределах интересующего приложения. После этого свойства и методы интересующего объекта становятся доступными в контроллере автоматизации. Выбор типа связывания влияет на такие характеристики приложения, как производительность, гибкость и удобство сопровождения [3].

В случае раннего связывания, мы как бы говорим компилятору: «Я точно знаю, чего я хочу. Поэтому жестко (статически) связывай все вызовы функций». При применении механизма позднего связывания мы как бы говорим компилятору: «Я пока не знаю, чего я хочу. Когда придет время, я сообщу что, и как я хочу». Таким образом, во время раннего связывания вызывающий и вызываемый методы связываются при первом удобном случае, обычно при компиляции. При позднем же связывании вызываемого и вызывающего методов они не могут быть связаны во время компиляции. Поэтому реализован специальный механизм, который определяет, как будет происходить связывание вызываемого и вызывающего методов, когда вызов будет сделан фактически. Очевидно, что скорость и эффективность при раннем связывании выше, чем при использовании позднего связывания. В то же время, позднее связывание обеспечивает некоторую гибкость в решении поставленной задачи.

В ранних версиях приложений Microsoft Office позднее связывание было обязательным. Оно продолжает использоваться и по сей день, так как обладает определенными преимуществами по сравнению с ранним связыванием. Одним из таких преимуществ является возможность создания кода, определяющего наличие или отсутствие необходимой библиотеки типов на исполнительном компьютере, а также создающего связи с разными версиями приложений в зависимости от принятых решений в процессе выполнения кода.

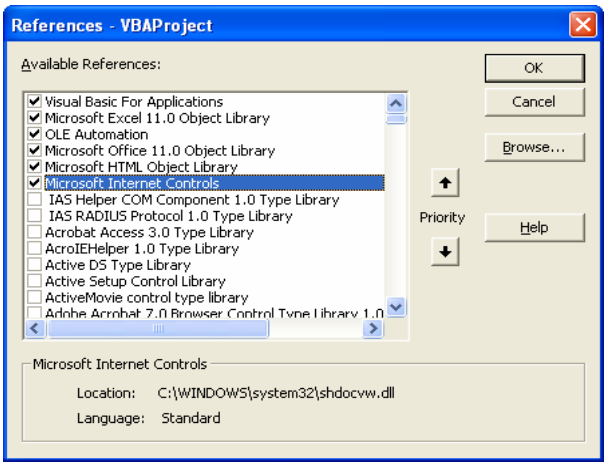
Недостаток позднего связывания – недоступность библиотеки типов интересующего приложения в процессе написания кода. Таким образом, отсутствует справочная информация о приложении, недоступны поименованные константы приложения, а ссылки на интересующее приложение не всегда действительны в процессе компиляции, так как их правильность невозможно проверить. Связи полностью проверяются и, соответственно, устанавливаются только в процессе выполнения кода, а это требует определенных затрат времени. При

этом на этапе проверки и выполнения кода могут возникнуть ошибки, которые могут привести к аварийному завершению работы приложения.

Раннее связывание поддерживается всеми версиями приложений Microsoft Office, начиная с Office 97. Использующий раннее связывание код работает быстрее, чем код на основе позднего связывания, так как библиотека типов интересующего приложения доступна в процессе написания кода.

При использовании раннего связывания код писать намного проще, так как в окне Object Browser видны объекты, методы и свойства интересующего приложения, а в окне кода, после ввода ссылки на объект, будут автоматически отображаться подсказки, например, список свойств и методов. Кроме этого, при раннем связывании можно пользоваться встроенными константами интересующего приложения.

При использовании раннего связывания в проекте VBA необходимо создать ссылку на библиотеку типов внешнего приложения. Для этого в редакторе Visual Basic необходимо выполнить команду **Tools ⇨ Reference**. В результате откроется окно диалога, показанное на рисунке.



Создание ссылок на объектные библиотеки

Для создания ссылок нужно установить флажки напротив названий необходимых библиотек. После этого можно объявлять объектные переменные интересующих типов. Например, можно объявить переменную `oIE` типа `InternetExplorer`:

```
Dim oIE As SHDocVw.InternetExplorer.
```

В отличие от других задач автоматизации различных приложений работа с Internet Explorer имеет ряд особенностей. Общая схема «создать объект приложения ⇨ открыть документ ⇨ выполнить обработку ⇨ закрыть документ ⇨ закрыть приложение» в случае с Internet Explorer неприменима. Для документов Internet Explorer не существует метода `Open`. Вместо этого используется метод `Navigate`, который только инициирует операцию открытия документа. Каждый, кто работал с браузерами, не мог не заметить, что эти приложения работают в фоновом режиме, оставляя массу свободного времени для других программ. То же самое происходит и при запуске Internet Explorer из Microsoft Office. Internet Explorer немедленно возвратит управление вызывающему модулю, хотя до окончания загрузки необходимого документа еще далеко. Чтобы не заблокировать систему, необходимо решить задачу синхронизации параллельно работающих приложений. Обычно для этой цели в VBA используется функция `DoEvents`, передающая управление операционной системе для обработки событий и выполнения других программ.

Практические особенности взаимодействия приложений Microsoft Office и Internet Explorer рассмотрим на следующем примере. Пусть, необходимо отобразить содержимое таблицы №8 на одной из Web-страниц портала Харьковской национальной академии городского хозяйства – список студентов последней просматриваемой группы.

Реализация поставленной задачи на языке программирования VBA с помощью раннего связывания представлена в Листинге 1.

```
Sub РаннееСвязывание()
    ' создать объектную переменную,
    ' которая ссылается на Internet Explorer
    Dim oIE As New SHDocVw.InternetExplorer

    With oIE
        ' начать загрузку Web-страницы
        .navigate
"http://www.ksame.kharkov.ua/portal/index.php?mnu=75"
        ' ждать завершения загрузки Web-страницы
        Do While .Busy And .readyState <>
SHDocVw.READYSTATE_COMPLETE
            DoEvents
        Loop

        ' обработка страницы
        .Visible = True ' если надо посмотреть, что загрузили
        MsgBox "Таблица 8" & Chr(13) & Chr(13) & _
            .document.all.tags("table").Item(8).innerText
```

```
        .Quit ' завершить Internet Explorer
    End With
    Set oIE = Nothing ' освободить память
End Sub
```

Листинг 1. Раннее связывание

В основе метода программного управления другим приложением лежит создание объектной переменной, которая ссылается на интересующее приложение. В данном случае такая переменная называется `oIE` и имеет тип `SHDocVw.InternetExplorer`. Использование явного указания типа объектной переменной допустимо потому, что предварительно были созданы ссылки на необходимые библиотеки: Microsoft Internet Controls и Microsoft HTML Object Library (рисунок).

При помощи метода `Navigate` Internet Explorer инициируется начало загрузки требуемой Web-страницы. Цикл `Do While … Loop` и функция `DoEvents` отслеживают окончание ее загрузки. Далее выполняется обработка загруженной страницы: метод `Visible` Internet Explorer отображает ее в окне браузера целиком, а функция `MsgBox` – только таблицу №8 из этой страницы. После завершения обработки страницы метод `Quit` закрывает Internet Explorer, а инструкция `Set oIE = Nothing` освобождает занятую переменной память.

Код реализации той же задачи, но с использованием позднего связывания, представлен в Листинге 2.

```
Sub ПозднееСвязывание()
    ' объявить объектную переменную
    Dim oIE As Object
        ' создать объект, ссылающийся на Internet Explorer
    Set oIE = CreateObject("InternetExplorer.Application")

    With oIE
        ' начать загрузку Web-страницы
        .navigate
"http://www.ksame.kharkov.ua/portal/index.php?mnu=75"
        ' ждать завершения загрузки Web-страницы
        Do While .Busy And .readyState <> 4
            DoEvents
        Loop
              ' обработка страницы
        .Visible = True ' если надо посмотреть, что загрузили
        MsgBox "Таблица 8" & Chr(13) & Chr(13) & _
            .document.all.tags("table").Item(8).innerText
        .Quit ' завершить Internet Explorer
    End With
    Set oIE = Nothing ' освободить память
End Sub
```

Листинг 2. Позднее связывание

В отличие от раннего связывания, в данном случае библиотеки типов не подключаются. Поэтому объектная переменная `oIE` объявлена универсального типа `Object`. Запуск на выполнение приложения-сервера Internet Explorer и его связывание с приложением-клиентом выполняет функция `CreateObject`. Во всем остальном данный код не отличается от кода раннего связывания, приведенного в Листинге 1.

С каждым годом все больше и больше информации переносится с традиционных печатных форм ее представления на электронные носители. Наибольшее количество информации в электронном виде представлено во всемирной компьютерной сети Интернет – особенно во Всемирной Паутине – World Wide Web (WWW). Однако, информация, представленная на Web-страницах, в первую очередь предназначена для визуального ее восприятия человеком и плохо приспособлена для автоматической обработки. На практике же довольно часто возникают задачи получения информации из Web-страниц с целью ее дальнейшей компьютерной обработки. Можно привести массу примеров, когда целесообразно в автоматическом режиме осуществлять сканирование страниц Всемирной Паутины. Например, можно автоматически отслеживать котировки мировых валют и акций мировых компаний; можно регистрировать новые сведения, появляющиеся на страницах конкурирующих компаний; можно следить за появлением новых ссылок по интересующему предмету, сканируя страницы, генерируемые различными поисковыми системами; извлекать мировые новости и генерировать собственные дайджесты; можно дистанционно, из единого центра, эффективно управлять большими инженерными сетями (тепло-, водо-, газоснабжения) и т.д.

Для решения любой из этих задач вполне подходят приложения Microsoft Office. Так, для работы с числовыми данными удобно импортировать информацию из Интернета прямо на рабочие листы Microsoft Excel, а тексты можно записывать в документы Microsoft Word. Реализовать же задуманное можно при помощи встроенного языка программирования Microsoft Office – VBA (Visual Basic for Applications). Для этого необходимо решить две задачи:

1) из приложения Microsoft Office программным путем запустить браузер, например, Microsoft Internet Explorer, и получить необходимую Web-страницу;

2) проанализировать полученный HTML-документ с целью извлечения из него необходимой информации.

1.Елманова Н. Автоматизация приложений Microsoft Office в примерах [Электронный ресурс] – Режим доступа: http://www.lib.csu.ru/dl/bases/prg/kompress/articles/2000_11_offauto/.

2.Колесов А. Особенности технологий раннего и позднего связывания в Visual Basic [Электронный ресурс] – Режим доступа: http://www.visual.2000.ru/develop/ms-vb/cp0009-1/binding.htm.

3.Арчер Том, Уайтчепел Эндою. Visual C++.NET. Библия пользователя: Пер. с англ. – М.: Изд. дом "Вильямс", 2003. – 1216 с.

4.Буллен Стивен, Боуви Роб, Грин Джон. Профессиональная разработка приложений Excel: Пер. с англ. – М.: ООО "И.Д.Вильямс", 2007. – 736 с.

5.Кашаев С.М. Офисные решения с использованием Microsoft Excel 2007 и VBA. – СПб.: Питер, 2009. – 352 с.

6.Уокенбах Джон. Microsoft Office Excel 2007: профессиональное программирование на VBA: Пер. с англ. – М.: ООО "И.Д.Вильямс", 2008. – 928 с.

Получено 14.04.2010

УДК 378.14

Н.И.САМОЙЛЕНКО, д-р техн. наук, А.Б.КОСТЕНКО, канд. физ.-матем. наук, В.А.ПОПОВ
Харьковская национальная академия городского хозяйства

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В РАСЧЕТЕ ФУНКЦИОНАЛЬНОЙ НАДЕЖНОСТИ ГОРОДСКИХ ТРУБОПРОВОДНЫХ СЕТЕЙ

Определяется математическая модель влияния установки дополнительной запорной арматуры на функциональную надёжность простейшей трубопроводной системы. Проверяется адекватность модели с помощью цифрового моделирования процесса эксплуатации такой системы.

Визначається математична модель впливу установки додаткової запірної арматури на функціональну надійність простої трубопровідної системи. Перевіряється адекватність моделі за допомогою цифрового моделювання процесу експлуатації такої системи.

The mathematical model of influencing of setting of additional constipation armature is determined on functional reliability of the simplest pipeline system. Model adequacy is checked up by the digital design of process of exploitation of such system.

Ключевые слова: городские трубопроводные сети, функциональная надежность системы, цифровое моделирование.

Установка дополнительной запорной арматуры в трубопроводных транспортных системах является одним из широко используемых структурных методов изменения системной надёжности. При этом надёжность системы в зависимости от надёжности устанавливаемой арматуры $p_{ад}$ может изменяться как в одну, так и в другую сторону.

Анализ методов расчёта функциональной надёжности (ФН) сложных трубопроводных сетей говорит о необходимости учета влияния запорной арматуры на общую надёжность сети.

Целью статьи является формализация влияния дополнительной задвижки на ФН системы и проверка адекватности полученной математической модели с помощью вычислительного эксперимента.