H-0230 (H0408-003) August 29, 2004
Computer Science

IBM Research Report

Putting Knowledge to Work - A Visual Basic Migration Case Study

Avi Yaeli, Neta Aizenbud-Reshef, Jonathan Bnayahu, Nurit Dor, Sara Porat, Asaf Yaffe
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

Putting Knowledge to Work - A Visual Basic Migration Case Study

Avi Yaeli Neta Aizenbud-Reshef Jonathan Bnayahu
Nurit Dor Sara Porat Asaf Yaffe

IBM Haifa Research Lab
{aviy, neta, bnayahu, nurit, porat, yaffe }@il.ibm.com

Abstract

Migration of legacy applications is a challenging task, even more when the target is a service oriented open platform such as J2EE, requiring reasoning about the legacy applications and understand the challenges and opportunities for a migration. This position paper presents the challenges in a VB to J2EE migration and suggests knowledge-base driven analyses to aid in the process.

1. Introduction

Over the last decade, Microsoft Visual Basic 6.0 (VB) applications have expanded beyond small-medium applications into larger projects, mostly as front-ends to legacy applications and middleware. The easy to use IDE and the large set of reusable ActiveX components (also from third parties) have turned VB into a popular platform for building distributed applications that involve graphical user interface, business logic, database access, networking, messaging, and more.

With the upcoming end of service for VB [1], the market is increasingly concerned in moving the existing VB legacy applications to state-of-the-art programming models such as .NET and J2EE. Also, the emerging trend to endorse Service Oriented Architecture (SOA) implies increasing needs for transforming legacy portfolio, including VB-based applications. Companies are now getting more and more forced or motivated to optimize and simplify existing IT infrastructure, to reduce TCO, platform-related costs through server consolidation. It's becoming then clear that enterprise software systems are often required to undergo transformations, and that tooling support to make these transformations more effective and accurate is an emerging business.

This position paper addresses the challenges in designing tooling to support assessing and planning the effort in migrating large VB applications to J2EE. More specifically we look at the complexity of migrating enterprise VB rich-client applications to thin, server-centric J2EE applications with SOA concepts in mind.

1.1. Motivation

VB is a rich language supporting many programming styles. One can find in a VB application both script-based programming patterns such as usage without declaration of variables, through limited object oriented design such as encapsulation and late binding method invocation. In addition, VB allows programmers to easily use Microsoft Windows system facilities, common applications such as Microsoft Office and third-party libraries.

Assessment and planning of migration projects is a cumbersome, complex and risky task, even more when there is a need to move between significantly different platforms, programming styles and architectures [2]. For large scale programs, the migration process focuses on a new manageable well structured application by re-architecting the existing applications. In order to do that, the manual process would require deep structural understanding, identifying components in the code, exploring dependencies between components and on external resources, abstracting dynamic behavior and interaction between components etc. Tools should therefore follow this level of understanding.

The main idea this paper introduces is the challenge in having a knowledge-base where patterns are formally described. An assessment and planning tool will analyze the applications to-be ported, detect those patterns and plan according to these findings. The challenge is to define the best and appropriate patterns. This idea lies as a base for a tool under development in IBM Haifa Research Labs.

1.2. Existing Tools

Application mining tools, e.g., CAST [3], support understanding of existing VB applications. These tools are oriented towards lower level program understanding such as determining dependencies, and change impact analysis. None of these tools provides higher-level program understanding such as layering and componentization of VB applications. In addition, they lack in their ability to provide migration issue detection and suggestions. Similar comments go to Rational Rose [4], a reverse engineering tool, generates a syntactic-driven model of the legacy application. Unfortunately, this model does not aid much in designing the target J2EE model.

Source-to-source transformation tools for VB are available. Some generate Java code, e.g, VB-Converter[2]. Unfortunately, many of them generate unmanageable code by providing wrappers to VB functionality. Microsoft .NET built-in converter for VB [5] generates an OS-dependent .NET application. Since the methodology is basically a line-to-line transformation and not re-architecting, the generated code is left with bad design practices such as on-error statements and wrappers to Active-X controls instead of usage of the equivalent .NET controls.

Current tools do not provide the two major requirements for a VB to J2EE migrations:

1. re-architecting and re-use capabilities
2. target-specific migration issues detection and recommendation

Our project aims at knowledge-based driven light static analyses that provide both requirements.

2. Leveraging a Knowledge Base

Even with the continue growth of automation and tooling for legacy transformation, a migration project is still inherently complex. The case of converting a typical VB application, OS-dependent rich client application, into a J2EE web service open platform application is even more complex. We believe that a rich knowledge base can aid and simplify the re-architecting and re-use of the legacy application.

The assessment phase, the first phase of a migration process, is to analyze and understand the legacy application. The goal is to detect the opportunities and obstacles of transforming this application by answering question such as:

- How is the code tied into the operating system? What are the possible J2EE replacements?
- Are the client tier, business logic and data access intertwined? What are the components of the application?
- Are there acute code quality issues, such as complexity, bad designs or J2EE unsupported features?
- Which parts of the code is worthwhile to transform?

Accompanied by information, even incomplete, obtained form the assessment phase, the next phases, planning and transformation, can be simplified up to the level of some automatic translation.

A Knowledge Base (KB) is the mean to capture domain expert knowledge. Many academics and commercial work tackle program understanding tasks by relaying on knowledge base. Migration tools, e.g., code-advisor [6], spot migration obstacles depending on a knowledge base. We present further ideas on how other assessment tasks can benefit from the KB. An example is available as appendix.

2.1. Migration Mining

The core of the VB language is rather small and simple, consisting of a few dozen statements and built-in control. However, VB's richness comes from the easy-of use of the thousands of system APIs and third-party Active-X controls [7]. The KB contains information about the language intrinsic and external features: its functionality, migration obstacle and severity, and the possible migration options. We envision a KB that grows as more VB-to-J2EE migrations are conducted as more features and Active-X controls are revealed.

Our tool applies a static analysis based algorithm that marks the program points as migration issues according to the KB. Due to VB's late-binding, a type analysis performed prior to the analysis provides information about the possible activation and dependencies. Understanding external dependencies provide a pick into the applications' architectural patters revealing infrastructure used by the application such as which database access, messaging protocols and more.

2.2. Classification

We propose a classification process that attaches properties to code fragments based on the KB. A basic and useful classification is layering. In the KB approach, the KB indicates for each language feature and for each external function to which layers may it belong. With this information the classification is rather straightforward, indicating if an expression belongs to a specific layer, such as presentation, business-rule or database access.

2.3. Data-Model Extraction

Date model extraction is a challenging task, requiring sophisticated data-flow and dependencies analyses. A heuristic KB-dependent light approach combines variables into "logical" classes. The idea is to state which program statement (functions) are likely to indicate a logical connection among the argument (parameters). For example, in a print statement, all arguments are marked as a data model. Another example, are text boxes members of a frame.

2.4. Componentization

Componentization, the process of partitioning a big legacy application, is another very challenging task. One option for componentization is according to the layering classification. Another option is according to the extracted data-model where the analysis attempts to detect all accesses to a given data-model.

2.5. Past Experience

Historical data from previous and similar migration projects is of significant value. It aids in the migration process as well as planning and organizing the tasks. Among the information obtained concerning features of the language, additional information regarding complexity and cost of the previous project can aid in planning of the current migration task. For example, a KB can contain historical cost models according to the transformed code size or complexity.

3. Conclusion

Based on our preliminary experience obtained by assessing a 100,000 lines of VB application, we believe that techniques presented in this paper can be of significant value. We have seen that the migration issues detection and classification are rather precise. Further work is undergoing to add additional KB-based analyses. Our current design is of a simple KB that contains basic elements such as statements and external libraries. We assume that a rule-based KB can enhance the analyses by allowing one to express more sophisticated patterns.

Bibliography

1. Microsoft, *Product Lifecycle Dates*, http://support.microsoft.com/default.aspx?scid=fh;%5Bln%5D;LifeDevToolFam
2. Declan Good, , *Legacy Transformation,* copyright Club De Investigacion Technologica 2002
3. CAST, Application Mining Suite, http://www.castsoftware.com
4. IBM, Rational Rose Enterprise Edition, http://www.ibm.com/software/rational/
5. DiamondEdge, VB Converter, http://www.diamondedge.com/products/Convert-VB-to-Java.html
6. Microsoft, Visual Basic 6.0 Code Advisor , http://msdn.microsoft.com/vbasic/downloads/codeadvisor/default.aspx
7. ComponentSource, http://www.componentsource.com
8. IBM, Migration Services for Microsoft Windows Applications Services, http://www-106.ibm.com/developerworks/websphere/services/services.html

Appendix

```
…
Global gMyConn As New MYSQL_CONNECTION
…
```

(a)

```
…
VB.TextBox txtPassword
VB.TextBox txtUserName
…
```

(b)

```
Private Sub cmdSave_Click()
      Dim rsUsers As MYSQL_RS

      Set rsUsers = gMyConn.Execute("Select Username From Users WHERE Username =
            txtUsername.text)
      If rsUsers.RecordCount > 0 Then
            MsgBox "Username & Me.txtUsername.text & already taken"
            txtUsername.SetFocus
            rsUsers.CloseRecordset
            Exit Sub
      End If

    Set rsUsers = gMyConn.Execute("INSERT INTO Users (Username,Password)
            VALUES (txtUsername.text  "," txtPassword.text  )
    rsUsers.CloseRecordset

    MsgBox "User Created Successfully"

    frmAdmin.Fill_Tree
    frmAdmin.SetFocus
    Unload Me
End Sub
```

(c)

🟨

🟩 Business Logic

🟦 User Interface

(d)

String txtUsername
String txtPassword

(e)

Fiqure 1: An example fo VB code contining: global declarations (a), local declaration (b) a event driven procedure (c). The classfication analysis shows threee intertwined layers (d) and an extracted data model as result of the second execute command (e).