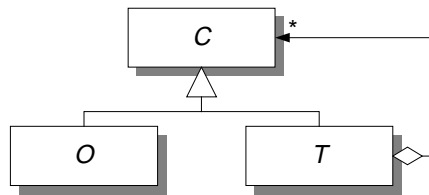


*Evaluation of COM Support in Visual  
Basic  
COT/3-6-V1.0*



Centre for Object Technology

*Centre for  
Object Technology*

Revision history: V0.1 11-08-98 First draft.  
V1.0 18-08-98 Introduction included.

Author(s): Henrik Lykke Nielsen, DTI

Status: Final

Publication: Public

Summary:

This document presents the answer to a number of specific questions regarding the support for COM based development using Microsoft Visual Basic.

© Copyright 1998 Danish Technological Institute

## Introduction

This document presents the answer to a number of specific questions regarding the support for COM based development using Microsoft Visual Basic.

This report is part of an evaluation of four development environments with respect to their support for COM based development. The main document of this evaluation is COT-3-4: Evaluation of COM Support in Development Environments.

## 1. Component Server

1.1 Which types of components does the tool support creation for?

(1.1) -

- 
- 1.1.1 Support for visual Controls ?  
(1.1.1) *Yes. Visual Basic supports Visual COM controls in the form of (what Visual Basic names as) ActiveX Controls (based on OCX servers) as well as ActiveX Documents (based on DLL or EXE servers).*

- 
- 1.1.2 Support for non-visual Controls ?  
(1.1.2) *Yes. Visual Basic supports non-visual COM controls in the form of ActiveX DLL and EXE servers.*

1.2 Types of servers

(1.2) -

- 
- 1.2.1 Can the tool generate/create In-process servers?  
(1.2.1) *Yes.*

- 
- 1.2.2 Can the tool generate/create Out-of-process servers?  
(1.2.2) *Yes.*

- 
- 1.2.3 How can you make transition between in-process and out-of-process servers?  
(1.2.3) *Changing from an in-process to an out-of-process server and vice-versa is a simple and straightforward selection in a combo-box.*

- 
- 1.2.3.1 You can do it through your IDE by clicking in some menu (where) ?  
(1.2.3.1) *Yes.*  
*Project / <ProjectName> Properties / Project Type.*

- 
- 1.2.3.2 Changing the code by hand ?  
(1.2.3.2) *Not in the application code itself. The type of the server is a property on the project. It is however possible to implement such a change in a wizard.*
-

- 1.2.3.3 Restarting the project and moving the code manually ?  
*(1.2.3.3) Not necessary but absolutely possible.*
- 

- 1.2.3.4 Other:  
*(1.2.3.4) -*
- 

- 1.2.4 Can multiple components be contained in the same server ?  
*(1.2.4) Yes. But if the components are public they should all be of the same type. It is not possible to mix different types of public components in a server in the Visual Basic environment.*
- 

- 1.2.5 Are there any limits on the number of different components in a server ?  
*(1.2.5) Yes. See 1.2.4.*
- 

### 1.3 Component construction (tool support)

*(1.3) Components are created in Visual Basic using a combination of facilities in the development environment (see 1.2.3.2) and application specific Visual Basic code. The COM specific functionality (that is the functionality necessary to create a COM component without specific application functionality) is largely managed by the development environment itself (developer-specified properties and options decide some of the functionality) or implemented using native keywords provided by Visual Basic. These keywords and the development environment encapsulate the "technical" COM functionality*

---

- 1.3.1 How are the COM components created ?  
*(1.3.1) See 1.3.*
- 

- 1.3.1.1 Not using wizard. Comments:  
*(1.3.1.1) With respect to implementation of the "technical" COM functionality please see 1.3.*
- 

- 1.3.1.2 using wizard. Comments:  
*(1.3.1.2) Visual Basic has an open architecture for adding wizards to the development environment in the form of what Visual Basic calls add-ins. A couple of add-ins for creating ActiveX components comes with Visual Basic. But these wizards are primarily used for generating application specific code and not for implementing COM specific functionality. It is relatively easy to implement such wizards (add-ins) which generates/manipulates code.*
- 

- 1.3.1.2.1 Which code-generation-wizards exists (list names and usage's)?  
*(1.3.1.2.1) The first three wizards are those, which comes as part of the Visual Basic package and Visual Modeler is available for download at [www.microsoft.com](http://www.microsoft.com):*
  - VB Class Builder Utility
  - ActiveX Control Interface Wizard
  - ActiveX Document Migration Wizard

- *Visual Modeler*
- *A lot of other third-party wizards exist.*

*It is not possible for the developer to specify how the above-mentioned wizards generate the code. So it is rather difficult (read: impossible) to generate and reverse-engineer code which uses a specific coding-standard or which uses developer specified abstractions (e.g. encapsulation). So even though application-specific code can be generated with these wizards it is necessary to customise and change the code rather intensively. Furthermore the wizards all produce code with the same level of abstraction (good or bad?) as the native application specific code most programmers use. There is - in other words - not much to gain by using the wizards. It is simple to code the components by hand and the wizards do not make it very much easier.*

- 
- 1.3.1.2.2 Description of the available options for generating the code (parameters, code-templates etc.)  
*(1.3.1.2.2) Not very relevant for the Visual Basic (see 1.3.1.2.1).*

- 
- 1.3.1.2.3 Description of the architecture of the generated code ?  
*(1.3.1.2.3) Mostly application specific and native Visual Basic code with no level of developer specified abstraction (e.g. encapsulation).*

- 
- 1.3.1.2.4 Is it easy to customise the generated code ?  
*(1.3.1.2.4) Yes.  
The above-mentioned wizards (see 1.3.1.2.1) all generates application specific Visual Basic code so it should be easy to find and recognise the relevant code. Only a very small amount of technicalities are involved.*

- 
- 1.3.1.2.5 Do the wizards allow for reverse-engineering ? To answer this question you should answer the following  
*(1.3.1.2.5) Some of them do to a certain extent. And some of these depends on comments in the generated code which they maintains themselves (the comments that is).*

- 
- 1.3.1.2.5.1 Are customisations in the generated code preserved when reverse-engineering the code ?  
*(1.3.1.2.5.1) If the wizards support reverse engineering then they preserve for customisation of the generated code. Else there doesn't seem to be much point in reverse-engineering the code. ; ^) Some of them demand that e.g. interfaces should be unchanged though.*

- 
- 1.3.1.2.5.2 Is it possible to change the fundamental strategies for the generated code ?

(1.3.1.2.5.2) *That should - with more or less work involved - be possible in all cases. But whether the wizard will still be able to reverse-engineer with a successful outcome is rather dependent on the specific wizard, the generated code and the actual changes.*

---

- 1.3.1.2.5.3 To evaluate the maintenance of the generated code (customisations, additions, changes, etc.): What changes are allowed in the generated code while still retaining the possibility of reverse-engineering it ?  
(1.3.1.2.5.3) *Depends on the specific wizard and the generated code.*
- 

- 1.3.1.2.6 Evaluation of the quality of the generated code  
(1.3.1.2.6) -
- 

- 1.3.1.2.6.1 How are the performance, stability and coding standard of the generated code ?  
(1.3.1.2.6.1) -
- 

- 1.3.1.2.6.2 Comments:  
(1.3.1.2.6.2) -
- 

- 1.3.2 Component creation using hand-coding  
(1.3.2) -
- 

- 1.3.2.1 Is it easy?  
(1.3.2.1) *Yes - very.*  
*The only instance that can be a little "demanding" for a typical Visual Basic developer is implementing encapsulated collections with custom interfaces.*
- 

#### 1.4 Support for interface's (1.4) -

---

- 1.4.1 Which types of interfaces do the tool support:  
(1.4.1) -
- 

- 1.4.1.1 Custom/vtable ?  
(1.4.1.1) *Yes, see 1.4.1.3.*
- 

- 1.4.1.2 Dispatch ?  
(1.4.1.2) *Yes, see 1.4.1.3.*
- 

- 1.4.1.3 Dual ?  
(1.4.1.3) *All interfaces created by Visual Basic are implemented as dual interfaces.*
-

- 1.4.2 How do you define an interface?  
(1.4.2) *As source code.*  
*An interface is by definition defined in Visual Basic as the collection of members in a Class Module.*
- 

- 1.4.3 Is it possible to implement multiple interfaces on a component ?  
(1.4.3) *Yes.*
- 

- 1.4.3.1 If Yes, how ?  
(1.4.3.1) *By defining and implementing abstract base classes.*
- 

- 1.4.4 Predefined interfaces:  
(1.4.4) -
- 

- 1.4.4.1 Is it possible to implement predefined interfaces, i.e. implementing interfaces defined by other people (preserving IID's)?  
(1.4.4.1) *Yes.*
- 

- 1.4.4.2 What do you use as foundation for the predefined interface ?  
(1.4.4.2) -
- 

- 1.4.4.2.1 Type lib ?  
(1.4.4.2.1) *Yes.*
- 

- 1.4.4.2.2 IDL ?  
(1.4.4.2.2) *IDL files must be compiled before the interfaces can be implemented.*
- 

- 1.4.4.2.3 Source code ?  
(1.4.4.2.3) *No.*
- 

- 1.4.4.2.4 other  
(1.4.4.2.4) -
- 

- 1.4.4.3 How do you use a predefined interface?  
(1.4.4.3) *Implementing the interface `InterfaceClass` defined in the type-library `TypeLibServer` is done as demonstrated below:*

1. *Set a reference to the type-library `TypeLibServer` defining the interface `InterfaceClass`*

2. *Implement the `InterfaceClass` interface in the `ImplementationClass` using code such as the following:*

```
Implements TypeLibServer.InterfaceClass
```

```
Private Sub InterfaceClass_Member
```

```
'Some implementation code  
End Sub
```

3. Use the interface (from the client) using code such as the following:

```
Dim interface as  
TypeLibServer.InterfaceClass  
  
Set interface = New  
TypeLibServer.ImplementationClass  
  
interface.DoSomething 'Call members
```

- 
- 1.4.5 Interface maintenance  
(1.4.5) -

- 
- 1.4.5.1 Which types of changes in a published interface are allowed ?  
(1.4.5.1) *All changes are allowed but certain changes are handled without breaking compatibility with earlier versions. If compatibility would be broken then Visual Basic generates warnings. So Visual Basic is a safe environment for developing components with regard to backward compatibility. The only exception is the "changing the name of a parameter" mentioned in 1.4.5.1.2.*

- 
- 1.4.5.1 Insert new methods and/or parameters ?  
(1.4.5.1) *All changes are allowed but certain changes are handled without breaking compatibility with earlier versions. If compatibility would be broken then Visual Basic generates warnings. So Visual Basic is a safe environment for developing components with regard to backward compatibility. The only exception is the "changing the name of a parameter" mentioned in 1.4.5.1.2.*

- 
- 1.4.5.2 Change existing methods and/or parameters ?
  - 1.4.5.3 Delete methods and/or parameters ?
  - 1.4.5.4 What are the consequences of changes on the generated component ?

- 1.4.6 Interface optimisations:  
(1.4.6) -

- 
- Which type of interface optimisation's are supported:

- 1.4.6.1 None?  
(1.4.6.1) -

- 
- 1.4.6.2 Tear-of interfaces (a tear-of interface is part of an aggregated component. The component will not be instantiated

until the interface is actually queried), How: ?  
(1.4.6.2) -

---

- 1.4.6.3 Other:  
(1.4.6.3) -
- 

## 1.5 Reference counting (1.5) -

---

- 1.5.1 A description of aspects of reference counting  
(1.5.1) -
    - 1.5.1.1 This question is intentionally left blank  
(1.5.1.1) -
    - 1.5.1.2 Is reference counting automatically supported for aggregated components?  
(1.5.1.2) *See 3.2.1.*
  - 1.5.2 How is reference counting implemented in the tool ?  
(1.5.2) -
    - 1.5.2.1 Is AddRef called automatically by a generated component when returning an interface reference ?  
(1.5.2.1) *Yes.*
    - 1.5.2.2 Is Release called automatically by a generated component when exiting scope for an interface reference received as a parameter ?  
(1.5.2.2) -
    - 1.5.2.3 What happens when the reference count drops to 0 ?  
(1.5.2.3) *The instance is deallocated.*
      - 1.5.2.3.1 Is it possible to customise what happens when the reference count drops to 0 ?  
(1.5.2.3.1) *It is not possible for the developer to implement his own reference counting scheme - if such a thing should be relevant.*
    - 1.5.2.4 How does the tools mechanism for implementing reference counting affect flexibility etc. ?  
(1.5.2.4) -
- 

## 1.6 Threading (1.6) -

---

- 1.6.1 Which threading models are supported ?  
(1.6.1) -

- 
- 1.6.1.2 Not specified ?  
(1.6.1.2) -

- 
- 1.6.1.3 Single ?  
(1.6.1.3) Yes.

- 
- 1.6.1.4 Apartment ?  
(1.6.1.4) Yes.

- 
- 1.6.1.5 Free ?  
(1.6.1.5) Yes. Not natively though, but through the windows API.

- 
- 1.6.1.6 Other:  
(1.6.1.6) -

- 
- 1.6.2 Multi-threaded components:  
(1.6.2) -

- 
- 1.6.2.1 Can a component be multi-threaded ?  
(1.6.2.1) Yes, see 1.6.1. and its subitems.

- 
- 1.6.2.2 If no, is multi-threading supported through language facilities (i.e. in native Java) or is it near API ?  
(1.6.2.2) -

---

## 1.7 Throwing errors (1.7) -

- 
- 1.7.1 Do generated components support:  
(1.7.1) -

- 
- 1.7.1.1 ISupportErrorInfo ?  
(1.7.1.1) -

- 
- 1.7.1.2 this question is intentionally left blank  
(1.7.1.2) -

- 
- 1.7.2 Direct support for HRESULT  
(1.7.2) -

- 
- 1.7.2.1 Does the tool support the use of existing HRESULT's ?  
(1.7.2.1) -

- 
- 1.7.2.2 Does the tool support the use of interface dependent HRESULTS ?  
(1.7.2.2) -
-

- 1.7.2.3 Does the tool provide support for HRESULT creation ?  
(1.7.2.3) -
- 

- 1.7.2.4 Is it easy to create/use HRESULTS ?  
(1.7.2.4) -
- 

- 1.7.3 Are there mappings between HRESULTS and native language exceptions ?  
(1.7.3) *Yes.*
- 

- 1.7.3.1 Is the mapping manual ?  
(1.7.3.1) *No.*
- 

- 1.7.3.2 Is the mapping automatic ?  
(1.7.3.2) *Yes.*
- 

- 1.7.4 How does the mapping work ?  
(1.7.4) *Visual Basic automatically adds HRESULTS to all members. These HRESULTS are handled by Visual Basic with integration with its standard error-handling scheme.*
- 

## 1.8 Datatypes

(1.8) -

---

- 1.8.1 How are data-types of the development tools mapped to COM data types ?  
(1.8.1) -
- 

- 1.8.1.1 Simple datatypes ?  
(1.8.1.1) *One-to-one.*
- 

- 1.8.1.2 Strings ?  
(1.8.1.2) *Visual Basic's native string type is BSTR.*
- 

- 1.8.1.3 Predefined types (automation compliant types)?  
(1.8.1.3) *Visual Basic uses the automation data-types.*
- 

- 1.8.1.4 User-defined types (e.g. struct, enum)?  
(1.8.1.4) -
- 

- 1.8.1.5 Object parameters (interface references) ?  
(1.8.1.5) -
- 

- 1.8.1.6 Variants ?  
(1.8.1.6) *Variant is a native Visual Basic data-type.*
- 

- 1.8.1.7 SafeArrays ?  
(1.8.1.7) -
-

## 1.9 Marshalling

(1.9) -

- 
- 1.9.1 How can marshalling be handled:

(1.9.1) -

- 
- 1.9.1.1 Don't know! ?

(1.9.1.1) -

- 
- 1.9.1.2 OLE32.DLL ?

(1.9.1.2) -

- 
- 1.9.1.3 Own Proxy/Stub DLL ?

(1.9.1.3) -

- 
- 1.9.1.3.1 Can it be attached to the component DLL?

(1.9.1.3.1) -

- 
- 1.9.1.4 Directly made by tool ?

(1.9.1.4) -

- 
- 1.9.1.5 Work around ?

(1.9.1.5) -

- 
- 1.9.2 Does the tool support IMarshal ?

(1.9.2) -

---

## 1.10 How does the server support type-libraries

(1.10) *Visual Basic always generates a type-library as part of the server. Furthermore it is possible to create stand-alone type-libraries (.tlb files).*

- 
- 1.10.1 IDL Integration ?

(1.10.1) *It is not possible in Visual Basic to directly view the .IDL file that is the basis for the type-library. But as all other type-libraries they can be reverse engineered to their IDL-format using third party tools.*

- 
- 1.10.1.1 IDL/ODL ?

(1.10.1.1) -

- 
- 1.10.2 Implicit creation of type-libraries ?

(1.10.2) *Yes.*

- 
- 1.10.3 Integration of type-libraries as resource in server ?

(1.10.3) -

- 
- 1.10.4 Are errors in the type library able to "cheat" the tools ?

(1.10.4) -

---

## 2. Compilation and Distribution

2.1 Is version compatibility checking performed during compilation and how ?

(2.1) *Yes. The developer has the option to check version compatibility by referencing a previous version of the server with which the project is to be compatible. The type-library in the compiled server is then automatically checked with the type-library to be included in the new server.*

---

2.2 Interface compatibility

(2.2) -

---

- 2.2.1 Is it possible to change an interface and keep the same IID?

(2.2.1) *Yes - sort of. See 1.4.5.1.1.*

---

2.3 Which types of target code does the tool produce (i.e. native/optimised native + runtime, p-code, nothing/interpreted) ?

(2.3) *Optional: Native, optimised native, p-code*

---

2.4 How does the tool support registration of components ?

(2.4) -

---

- 2.4.1 Components are selfregistering ?

(2.4.1) *Yes always.*

---

- If Yes then

- 2.4.1.1 DLL RegSvr32 ?

(2.4.1.1) *Yes always.*

---

- 2.4.1.2 EXE: selfregistering when run once ?

(2.4.1.2) *Yes. And by running with the /RegServer option.*

---

- 2.4.1.3 EXE: selfregistering when compiled ?

(2.4.1.3) *Yes.*

---

- 2.4.2 Registration is done through

(2.4.2) -

---

- 2.4.2.1 Installation ?

(2.4.2.1) *Yes. E.g. by use of the SetupWizard which is part of the Visual Basic package.*

---

- 2.4.2.2 Has to be done manually ?

(2.4.2.2) *It can be done manually, if so wanted.*

---

- 2.4.3 Does the tool support unregistration of components ?  
(2.4.3) *It is rather the components themselves as created by the tool that supports unregistration.*
- 

- 2.4.4 Does the tool support registration using ProgID ?  
(2.4.4) -
- 

- 2.4.4.1 Can the developer freely choose a components ProgID ?  
(2.4.4.1) *Not entirely freely. A components ProgID is defined as `ServerName.ClassName`.*
- 

- 2.4.5 Does the tool support version independent ProgID's, and if yes how ?  
(2.4.5) *Yes.*
- 

2.5 Support for component categories:  
(2.5) -

---

- 2.5.1 Is it possible to define and implement your own component categories ?  
(2.5.1) -
- 

- 2.5.1.1 How ?  
(2.5.1.1) -
- 

- 2.5.2 Does the tool have support for predefined component categories and how ?  
(2.5.2) -
- 

2.6 Can the developer freely choose CLSID for a component ?  
(2.6) *No.*

---

## 3. Component Client

### 3.1 Support for COM clients

3.1.1 Creating and deleting components  
(3.1.1) -

---

- 3.1.1.1 Is it easy to create new components ?  
(3.1.1.1) *Yes - very easy and very fast.*
- 

- 3.1.1.2 Is it easy to delete existing components ?  
(3.1.1.2) *Yes - very easy and very fast.*
- 

- 3.1.1.3 Does the tool delete registry entries for deleted components after recompilation ?  
(3.1.1.3) -
-

- 3.1.1.4 This question is intentionally left blank  
(3.1.1.4) -

---

### 3.1.2 COM Library Support (3.1.2) -

- 3.1.2.1 Does the tool allow direct access to COM functionality, like calling COM functions direct (CoCreateInstance, CreateInstance, ...). ?  
(3.1.2.1) *Not as part of the language.*
- 3.1.2.2 Does the tool encapsulate COM functionality (indirect COM support) ?  
(3.1.2.2) *Yes very much so. Encapsulation is the keyword for Visual Basic in general and especially so for COM support.*
- 3.1.2.3 Does the tool provide COM Helper Classes ?  
(3.1.2.3) -
- 3.1.2.4 Is it possible to combine native and encapsulated COM support ?  
(3.1.2.4) -

---

### 3.1.3 Acquisition of interfaces (3.1.3) -

- 3.1.3.1 Is it possible to call QueryInterface directly ?  
(3.1.3.1) -
- 3.1.3.2 Does the tool encapsulate QueryInterface ?  
(3.1.3.2) *Yes.*
- 3.1.3.3 Reference counting ?  
(3.1.3.3) *Reference counting is dealt with automatically.*
  - 3.1.3.3.1 Is AddRef called automatically by a generated client when using an interface as parameter for a (server) method ?  
(3.1.3.3.1) *Yes.*
  - 3.1.3.3.2 Is Release called automatically by a generated client when exiting scope for an interface reference ?  
(3.1.3.3.2) *Yes.*

---

### 3.1.4 Component binding (3.1.4) -

- 3.1.4.1 Is early binding supported through: ?  
(3.1.4.1) *Visual Basic always creates servers, which supports Vtable-binding. If a server supports Vtable binding then a VB client will always use it. If the server does not support Vtable binding then DispID binding will be used if present in the server.*

- 
- 3.1.4.1.1 VTable binding ?  
(3.1.4.1.1) See 3.1.4.1.

- 
- 3.1.4.1.2 DispID binding ?  
(3.1.4.1.2) See 3.1.4.1.

- 
- 3.1.4.2 Is late binding supported and how ?  
(3.1.4.2) Yes, late binding is supported using  
`CreateObject("ServerName.ClassName")`.

- 
- 3.1.4.3 Binding through ProgID  
(3.1.4.3) Yes, by using `CreateObject("ServerName.ClassName")`.

- 
- 3.1.4.3.1 Does the tool support binding of (server) component through version independent ProgID ?  
(3.1.4.3.1) Yes, on the client side by using  
`CreateObject("ServerName.ClassName")`.

- 
- 3.1.4.3.2 Does the tool support binding of (server) component through version specific ProgID ?  
(3.1.4.3.2) Yes, on the client side by using  
`CreateObject("ServerName.ClassName.Version")`.

---

3.1.5 Is the process of catching errors supported through:  
(3.1.5) -

- 
- 3.1.5.1 native COM support (i.e. using HRESULT) ?  
(3.1.5.1) No.
  - 3.1.5.2 encapsulated support (in the tool language/environment/library) ?  
(3.1.5.2) Yes. See 1.7.4.

---

## 3.2 Aggregation

3.2.1 Can the tool control COM-aggregation for produced components ?  
(3.2.1) No, COM based aggregation is not supported.

- 
- If yes, Supported methods
    - 3.2.1.1 Can you specify that a component is not aggregatable ?  
(3.2.1.1) -
    - 3.2.1.2 Is the aggregation support Wizard oriented (GUI) ?  
(3.2.1.2) -
    - 3.2.1.3 Is Manual programming of aggregation Easy/Difficult ?  
(3.2.1.3) -
-

### 3.3 Delegation

3.3.1 Support for delegation is:

*(3.3.1) Delegation is the name of the game for Visual Basic.*

- 
- 3.3.1.1 Manual ?  
*(3.3.1.1) -*

- 
- 3.3.1.2 Tool ?  
*(3.3.1.2) -*
- 

## 4. Development Environment support

4.1 Description and evaluation of supporting tools/facilities in the development tool

*(4.1) -*

- 
- 4.1.1 What kind of object browser facilities are provided ?  
*(4.1.1) A two-level (parent-child's) object-browser. OK - but not brilliant.*

- 
- 4.1.2 What kind of syntax help/info for COM-components and methods herein ?  
*(4.1.2) There are very, very good facilities for syntax help/info for COM components in Visual Basic. The options includes:*
    - *Tool-tip-like syntax for members*
    - *Listing with possible values when using enumeration's as datatypes*
    - *Auto-completion of object and member names*
    - *Automatic (but optional) listing of available members for a given instance of a component.*

- 
- 4.1.3 Does the tool provide context sensitive help for components (to be used by component integrators) ?  
*(4.1.3) Yes.*

- 
- 4.1.4 Support for interface prototypes (components implementing interfaces specified externally)  
*(4.1.4) -*

- 
- 4.1.4.1 How can you make this manually ?  
*(4.1.4.1) Yes (possible).*

- 
- 4.1.4.2 What kind of tool support is provided (the tool provides prototypes for the methods of the interface). Filling out the slots ?  
*(4.1.4.2) Yes (most often used approach).*

- 
- 4.1.5 Debugging  
*(4.1.5) -*
-

- 4.1.5.1 Which facilities exist for debugging COM-components ?  
*(4.1.5.1) The Visual Basic development environment allows for source code based debugging.*
- 

- 4.1.5.2 Is it possible in the development to debug from client into component ?  
*(4.1.5.2) Yes.*
- 

- 4.1.5.3 If any, what are the differences between debugging in-process and out-of-process servers ?  
*(4.1.5.3) An out-of-process server and its client must be in separate instances of the Visual Basic development environment. An in-process server and its client can either be in separate instances or in the same instance of the Visual Basic development environment.*
- 

## 4.2 Evaluation of the development tool *(4.2) -*

---

- 4.2.1 A description of the work load  
*(4.2.1) -*
- 

- 4.2.1.1 Does the COM-based part of the development integrate nicely and consistently into the tool ?  
*(4.2.1.1) Yes very nicely. The ActiveX technology is largely designed to be easy to use from Visual Basic. And one must admit that it has very much succeeded.*
- 

- 4.2.1.1.1 Is it easy to find out *how* to do COM-based development in the tool ?  
*(4.2.1.1.1) Yes relatively easy. For the simple parts anyway. Compared to standard Visual Basic development it is a little bit more complicated process using interface inheritance (that is use of non-default interfaces).*
- 

- 4.2.1.1.2 Is the COM-based development in the tool similar to non-COM-based development ?  
*(4.2.1.1.2) Yes. Almost identical with interface inheritance (the use of non-default interfaces) as a possible exception.*
- 

- 4.2.1.2 Is it easy for novices/experts with respect to the COM-development in general/the development tool specifically to do COM-based development in the tool.  
*(4.2.1.2) Simple COM-based development will be easy for both experts and novices with respect to the development tool. But when dealing with advanced COM-based development it will be a clear advantage to have some prior knowledge of the COM architecture.*
-

## 5. Development Processes

- 5.1 Is it possible to develop and test at the same time ?  
(5.1) *Yes. Visual Basic has a source code level debugger.*

- 
- 5.1.1 Will recompilation require change of server GUID and IID's ?  
(5.1.1) *Not necessarily. See 1.4.5.1.*
-