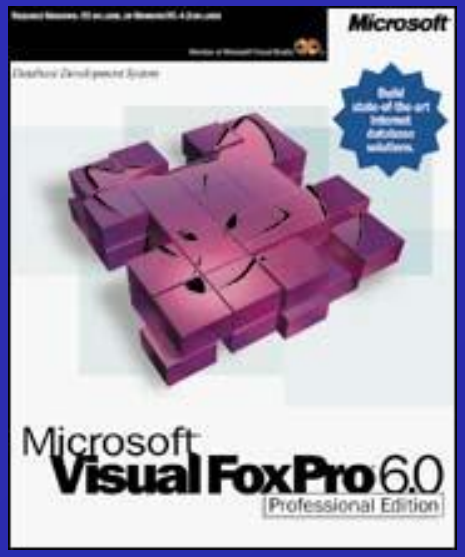
COM в Visual FoxPro® 6.0

(основные понятия)



Дроздов Михаил

Компания «ИВС Софт»

My Page: *http://vfpdmur.narod.ru/*

ICS Page: *http://ics.perm.ru/*

mailto:Drozdov@ics.perm.su

Отличия модульного программирование и с использованием классов

```
PUBLIC pVat1[,…,pVatN]
PROCEDURE Modul_name1( param1 [, …, paramN])
{
   LOCAL var1[,…, varN]
  … использование:
• param1 [, …, paramN]
• pVat1[,…,pVatN]
• CALL Modul_name1(var1[,…,varN])
   RERURN RetVar
}
```

- - Использование PUBLIC-переменных для передачи значений между процедурами приводит к потери контроля над их значениями
- - Возможность взаимодействовать с внешней средой данных, только через значения, явно указанного и фиксированного на этапе разработки списка параметров
- - Компоновка на этапе LINK в монолитное (более не изменяемое) приложение

```
DEFINE CLASS MyClass AS Custom
  PROTECTED pot_prop
  pot_prop =  'Value protected property'
  pub_prop = 'Value public property'
  Name = "MyClass"
  Version = '1.0'
  FUNCTION pubMethod(param1[,...paramN])
    … использование:
    свойств и методов класса
   RETURN RetVar
  ENDFUC
ENDDEFINE
```

- - Использование PUBLIC-переменных для передачи значений между классами приводит к потери контроля над их значениями
- + Взаимодействие между классами определяется открытым интерфейсом классов. Наследование позволяет исключить дублирование кода, а инкапсуляция - защитить данные.
- - Компоновка на этапе LINK в монолитное (более не изменяемое) приложение. (+ VFP позволяет осуществлять динамическое создание экземпляров классов из библиотек на этапе выполнения)

Проблемы традиционного программирования

- Монолитное (необходимо «целиком» переделывать всё приложения после изменений)
- Жёстко зависит от языка программирования (практически отсутствует возможность использования различных языков в одном приложении)

Что такое COM (*Component Object Model*)

Технология спецификация или стандарт создания программного обеспечения, позволяющая:

- использовать возможности любых языков программирования в рамках одного приложения (конечно, речь идёт об языках, способных создавать COM-компоненты)
- исполнять COM-компонент на множестве платформ: Microsoft® Windows®, Microsoft Windows NT™, Apple® Macintosh®, UNIX®
- изменять любой компонент не нарушая работы всего приложения, т.е. позволяет развивать возможности ПО
- компоненты COM являются прозрачно перемещаемы, следовательно позволяют оптимально распределить ПО согласно мощностям парка машин в Вашей организации

Краткая история COM:

- первоначальной целью было поддержка концепции ***связывания и внедрения*** (***object linking and embedding***) объектов, обеспечивающей существование различных документов в одном. Реализация этой концепции известна под названием OLE
- Первая версия для связи между клиентом и компонентов использовала аппарат известный под названием ***динамический обмен данными*** (***dynamic data exchange - DDE***). В то время COM не существовало и DDE был построен на основе передачи сообщений OS Windows, а версия OLE в основу которой был положен механизм DDE - OLE1.
- Позже был изобретён COM и следующая версия OLE (OLE2) была переписана на основе COM. Ещё позже она называлась просто OLE, теперь это называют ActiveX

Интерфейсы COM

- ***Интерфейс класса*** (или компонента) - это некоторое множество свойств и методов, доступных из-вне этого класса. На низком уровне - это определённая структура памяти, содержащая массив указателей на функции класса.
- Основу COM составляет ***соглашения о межкомпонентном интерфейсе,*** т.е. некоторое множество внутренних свойств и методов, присущих всем COM компонентам, используя который можно получать информацию о прикладных методах и свойствах компонента.
- Нужно иметь ввиду, что
  - компонент это не класс: компонент может быть реализован как используя несколько классов, так и вообще без класса, лишь бы он удовлетворял спецификациям COM
  - в настоящей реализации отсутствует наследование (это предполагается осуществить в версии COM+)
  - однажды созданные (опубликованные) интерфейсы COM-компонента не изменяются. (нарушение этого привело бы к несовместимости версий компонентов)
- Интерфейс COM, через который запрашиваются другие интерфейсы компонента называется *Iunknown* и он содержит в себе всего три метода: *QueryInterface, AddRef,* и Release

```
Interface IUnknown
{
    virtual HRESULT __stdcall QueryInterface(const IID& iid, void** ppv)= 0;
    virtual ULONG __stdcall AddRef() = 0;
    virtual ULONG __stdcall Release() = 0;
}
```

Интерфейсы COM

Для получения ссылки на интерфейс *Iunknown* используют функцию:

```
STDAPI CoCreateInstance(
  REFCLSID rclsid,            // ссылка на идентификатор (CLSID) объекта
  LPUNKNOWN pUnkOuter, // ссылка на Iunknown агрегирующего объекта, иначе NULL
  DWORD dwClsContext,         // определяет контекст создаваемого объекта
  REFIID riid,                //  ссылка на идент. интерфейса, который тр. получить
  LPVOID * ppv                // адрес для возвращаемой ссылки на интерфейс тр. интерфейс
)
```

Функция CoCreateInstanceEx позволяет получать интерфейсы удалённых серверов

Для облегчения работы с ком сервером создан интерфейс «фабрики классов»

```
Interface IClassFactory : IUnknown
{
   HRESULT CreateInstance(
        IUnknown * pUnkOuter, //Pointer to whether object is or isn't part of an aggregate
        REFIID riid,          //Reference to the identifier of the interface
        void ** ppvObject  //Address of output variable that receives the interface pointer requested in riid
        );
   HRESULT LockServer(BOOL fLock ); //Increments or decrements the lock count
}
```

Для «работы» с библиотекой типов (TypeLib) (которая и содержит описание интерфейса COM-компоненты) имеется интерфейс ITypelib

```
Interface ITypeLib : IUnknown
{
  FindName(); GetDocumentation(); GetLibAttr(); GetTypeComp(); GetTypeInfo();
GetTypeInfoCount(); GetTypeInfoOfGuid(); GetTypeInfoType(); IsName(); ReleaseTLibAttr();
}
```

Интерфейс IDispach

OLE Automation создано для языков, не имеющих механизмов работы со ссылками (Visual Basic, Visual FoxPro, и т.д.) и основывается на интерфейсе IDispatch

```
MIDL_INTERFACE("00020400-0000-0000-C000-000000000046")
  IDispatch : public IUnknown
  {
  public:
     virtual HRESULT STDMETHODCALLTYPE GetTypeInfoCount(
        /* [out] */ UINT __RPC_FAR *pctinfo) = 0;
    virtual HRESULT STDMETHODCALLTYPE GetTypeInfo(
        /* [in] */ UINT iTInfo,
        /* [in] */ LCID lcid,
        /* [out] */ ITypeInfo __RPC_FAR *__RPC_FAR *ppTInfo) = 0;
    virtual HRESULT STDMETHODCALLTYPE GetIDsOfNames(
        /* [in] */ REFIID riid,
        /* [size_is][in] */ LPOLESTR __RPC_FAR *rgszNames,
        /* [in] */ UINT cNames,
        /* [in] */ LCID lcid,
        /* [size_is][out] */ DISPID __RPC_FAR *rgDispId) = 0;
   virtual /* [local] */ HRESULT STDMETHODCALLTYPE Invoke(
        /* [in] */ DISPID dispIdMember,
        /* [in] */ REFIID riid,
        /* [in] */ LCID lcid,
        /* [in] */ WORD wFlags,
        /* [out][in] */ DISPPARAMS __RPC_FAR *pDispParams,
        /* [out] */ VARIANT __RPC_FAR *pVarResult,
        /* [out] */ EXCEPINFO __RPC_FAR *pExcepInfo,
        /* [out] */ UINT __RPC_FAR *puArgErr) = 0;
};
```

OLE-серверы

OLE-серверы в зависимости от памяти, в которой они выполняются, бывают двух типов:

- вне процесса клиента (Out-of-process)
  - исполняемые файлы с расширением EXE
  - обычно могут быть выполнены не только в 32-разрядной, но и в 16-разрядной OS
  - кроме того они могут содержать визуальный интерфейс для пользователя.
- в процессе клиентского приложения (In-process)
  - файлы с расширением DLL
  - только под 32-разрядной OS
  - не могут содержать визуальный интерфейс для пользователя и не может быть внешним (удалённым сервером).

Для создания OLE-компонента достаточно добавить при описании класса (DEFINE CLASS) опцию OLEPUBLIC (в диалоге Class info в редакторе классов), это позволит, в зависимости от опции

- Win32 executable / COM server (exe)
- Single-threaded COM server (dll) - каждое приложение-клиент использует свою копию сервера
- Multi-threaded COM server (dll) - несколько приложений может использовать одну копию сервера

VFP создать и зарегистрировать OLE-компоненту, при этом, помимо EXE/DLL будет созданы файлы с расширениями VBR и TLB, обеспечивающие регистрацию компоненты в системном реестре и описание интерфейсов Вашей компоненты. После первого создания Вашей компоненты будет доступна информация на вкладке Server в диалоге Project Info.

OLE-серверы (ряд функций)

- CreateObject(ClassName [, eParameter1, eParameter2, ...]) - создаёт экземпляр объекта
- CreateObjectEx(cCLSID | cPROGID, cComputerName) - создаёт экземпляр объекта удалённого компьютера
- SYS(2335 [, 0 | 1]) - включает/отключает режим допустимости модальных диалогов в COM-exe сервере (без второго параметра возвращает режим: 0-недопустимо 1-допустимо) Для COM-dll серверов всегда недопустимо, т.е. вызывает ошибку
  - команда WAIT или MESSAGEBOX( )
  - сообщения об ошибках
  - диалог открытия файлов
  - ввод пароля на соединение с внешним источником данных через ODBC
- SET OLEOBJECT включает/отключает режим поиска в системном реестре. Компоненты, помещённый в различных построителях отыскиваются безотносительно к этому режиму.
- VFP является Automation Server-ом и клиенту доступны методы и свойства объекта Application
- Объект Project имеет коллекцию Servers объектов Server позволяющий получить всю необходимую информацию о создаваемом в VFP COM-компоненте

OLE-серверы (ряд функций)

- ComClassInfo(oObject [, nInfoType]) пример:
  - 1 - Excel.Application.8
  - 2- Excel.Application
  - Microsoft Excel 97 Application
  - {00024500-0000-0000-C000-000000000046}
- CommArray(oObject [, nNewValue])
  - 0 - первый индекс 0 и передача по значению
  - 1 - первый индекс 1 и передача по значению
  - 10 - первый индекс 0 и передача по ссылке
  - 11 - первый индекс 1 и передача по ссылке
- Remote Auto Connection Manager - позволяет выполнить регистрацию COM-компонент для выполнения «удалённых по сети» компонент
- Automation Manager - обеспечивает «сетевое взаимодействие» компонент.

OLE-серверы (взаимодействие через сеть)

Адресное пространство клиента

Адресное пространство сервера

OLE-компонента
на клиенте

OLE-компонента
на сервере

Сетевой OLE-посредник
Proxy

Сетевой OLE-посредник
Proxy

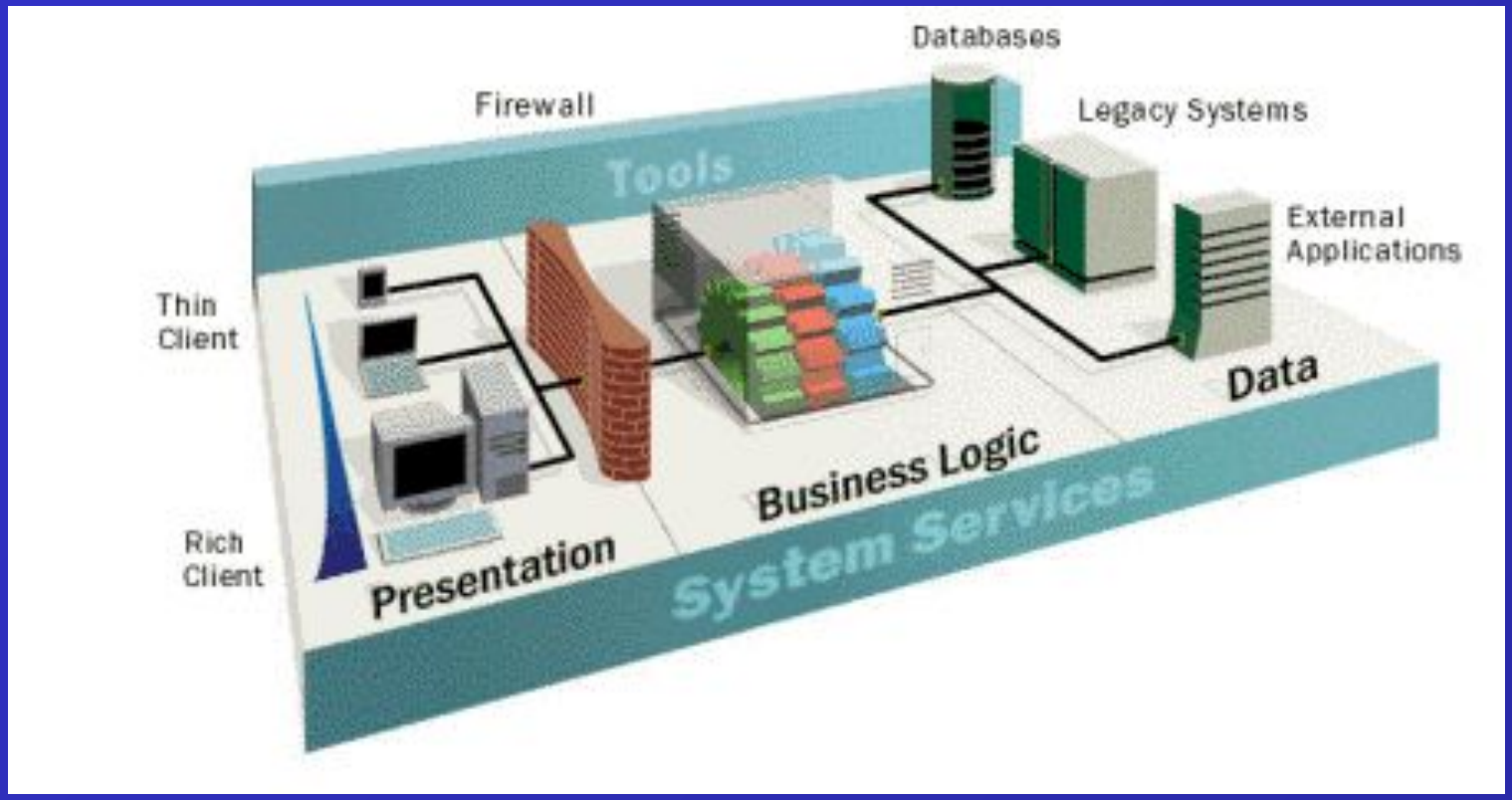
OLEOUT32.DLL

Automation Manager -
обеспечивает связь с сервером

AUTPRX32.DLL

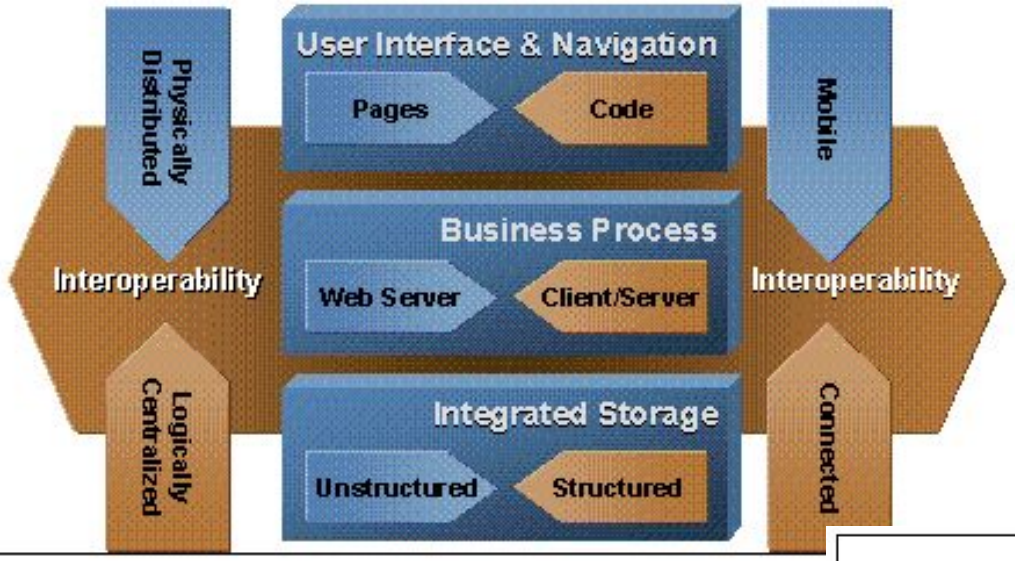
Automation Manager - обеспечивает
связь с клиентом

Microsoft® Windows® Distributed interNet Applications Architecture (Windows DNA)

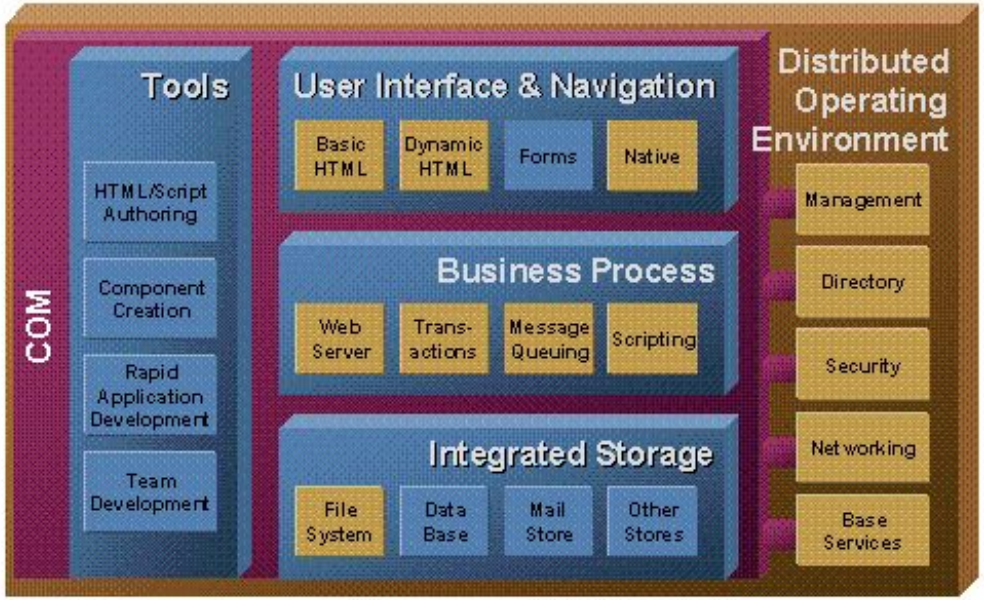


Client's Applications - COM component's (MTS) - SQL Database

Windows Platform Application Strategy



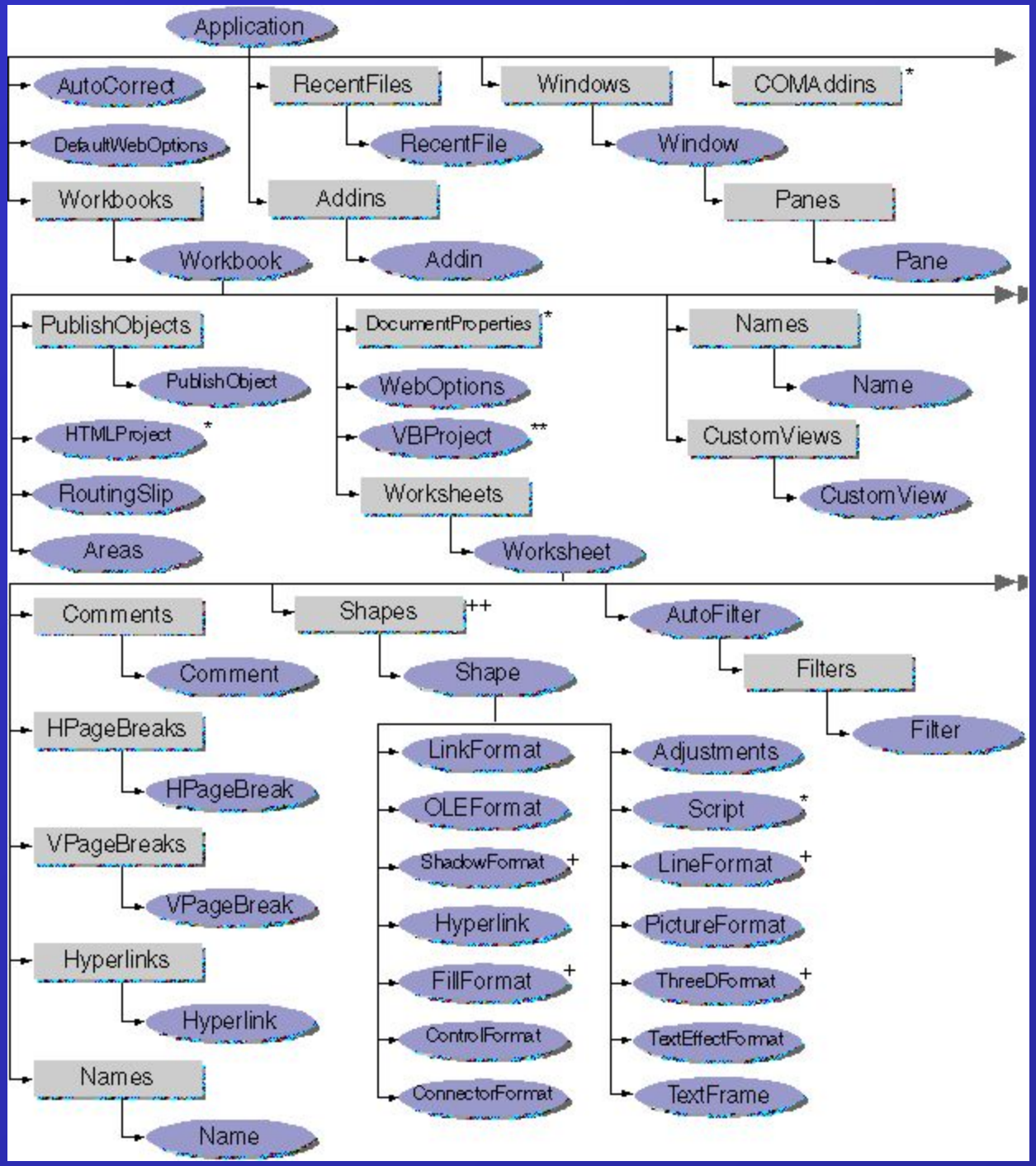
Windows DNA Services



Объектная модель Excel 2000

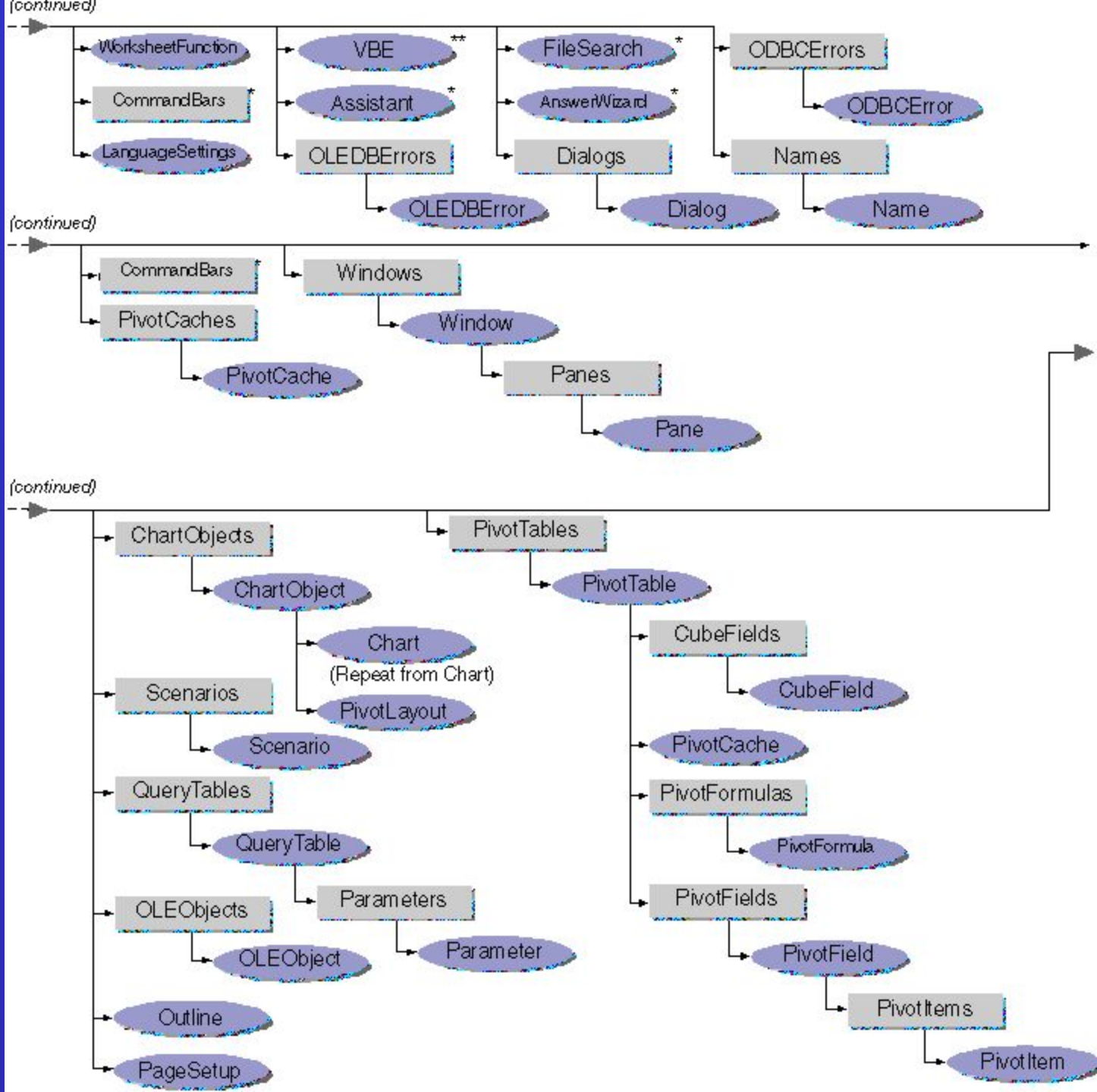
1(4)

Microsoft Excel 2000
в MSDN oct99
…/office/Excel9.olb
VBAXL9.CHM



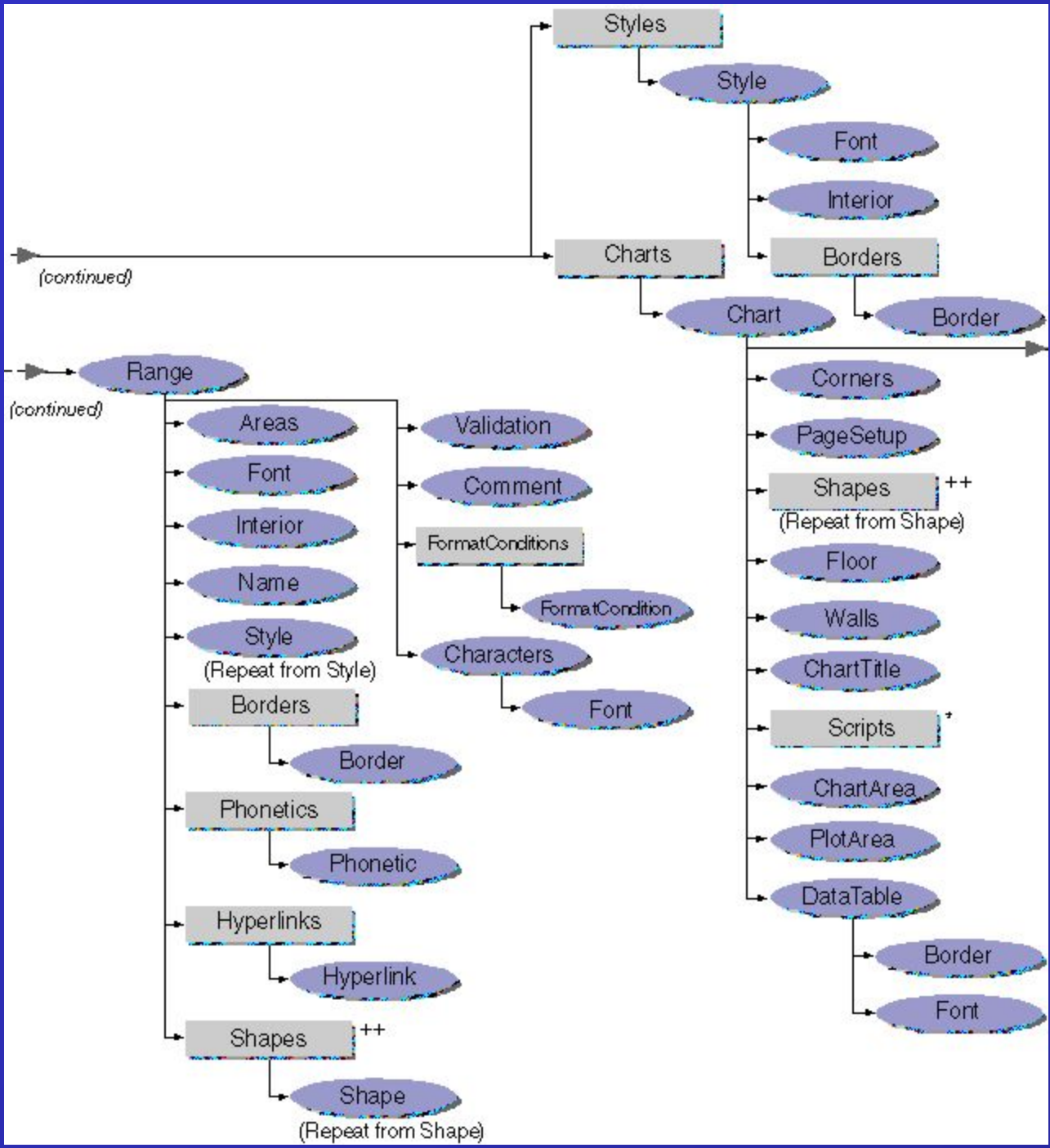
Excel 2000 Объектная модель

2(4)



Объектная модель Excel 2000

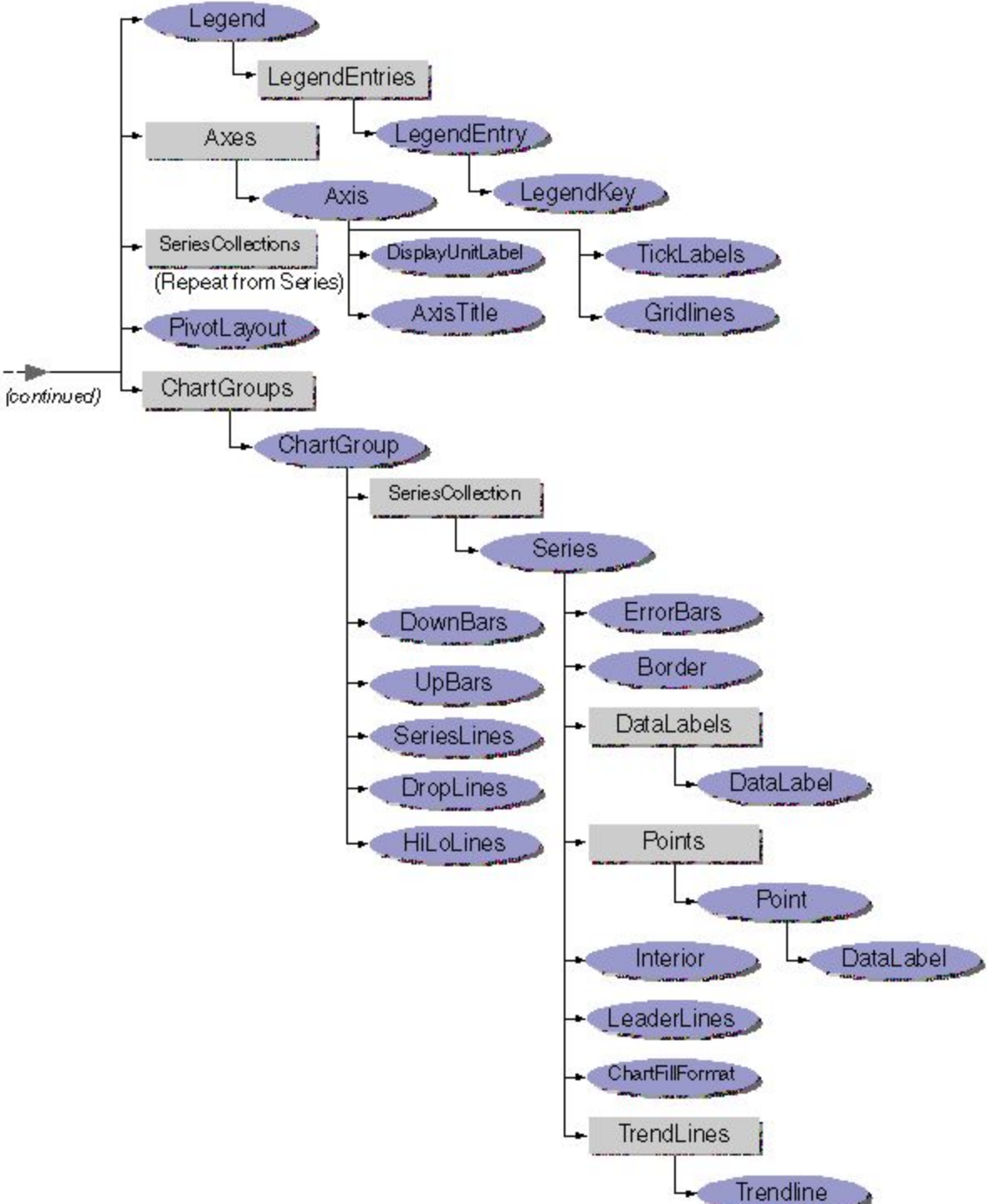
3(4)



Excel 2000

Объектная модель

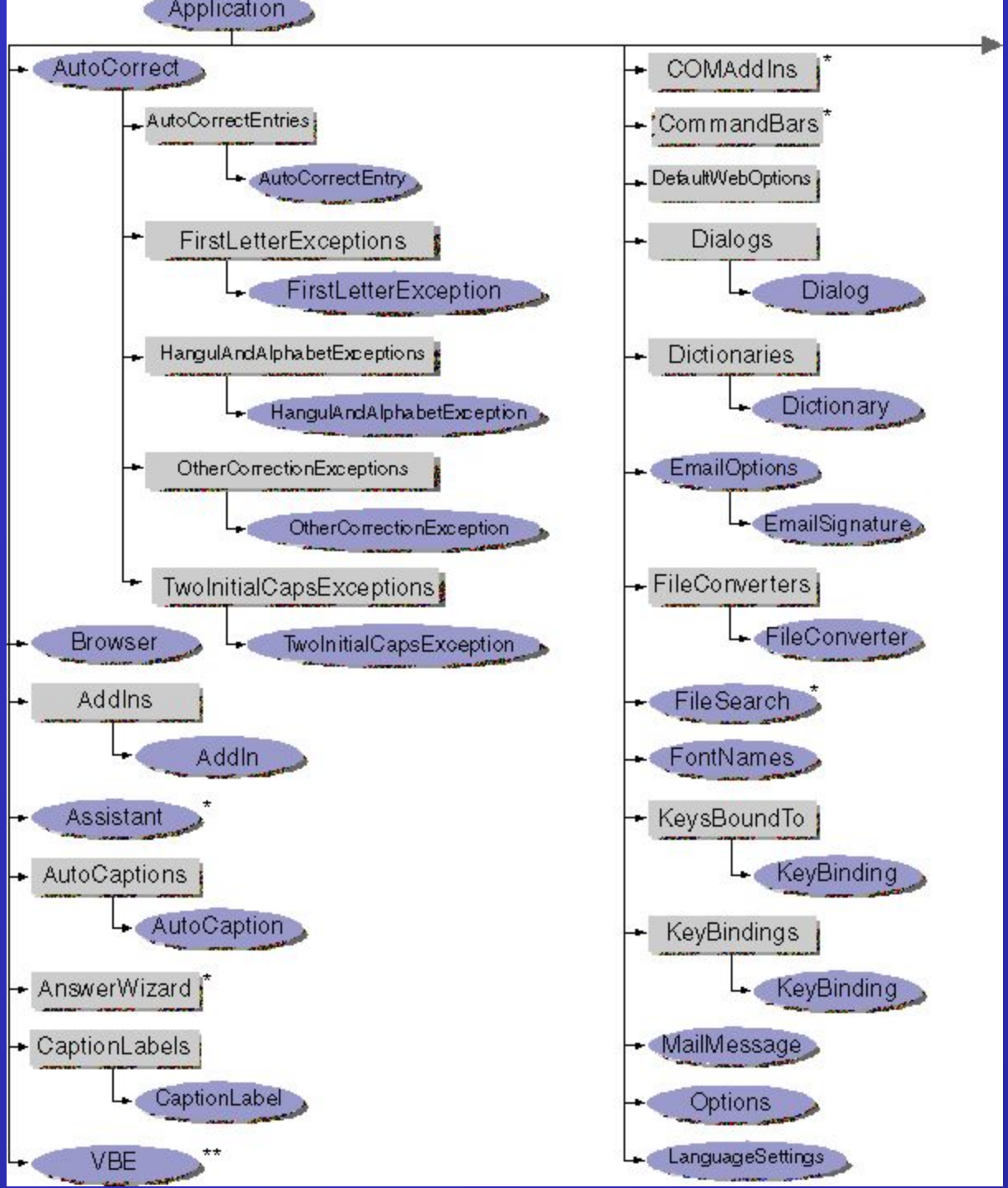
4(4)



Объектная модель Word 2000

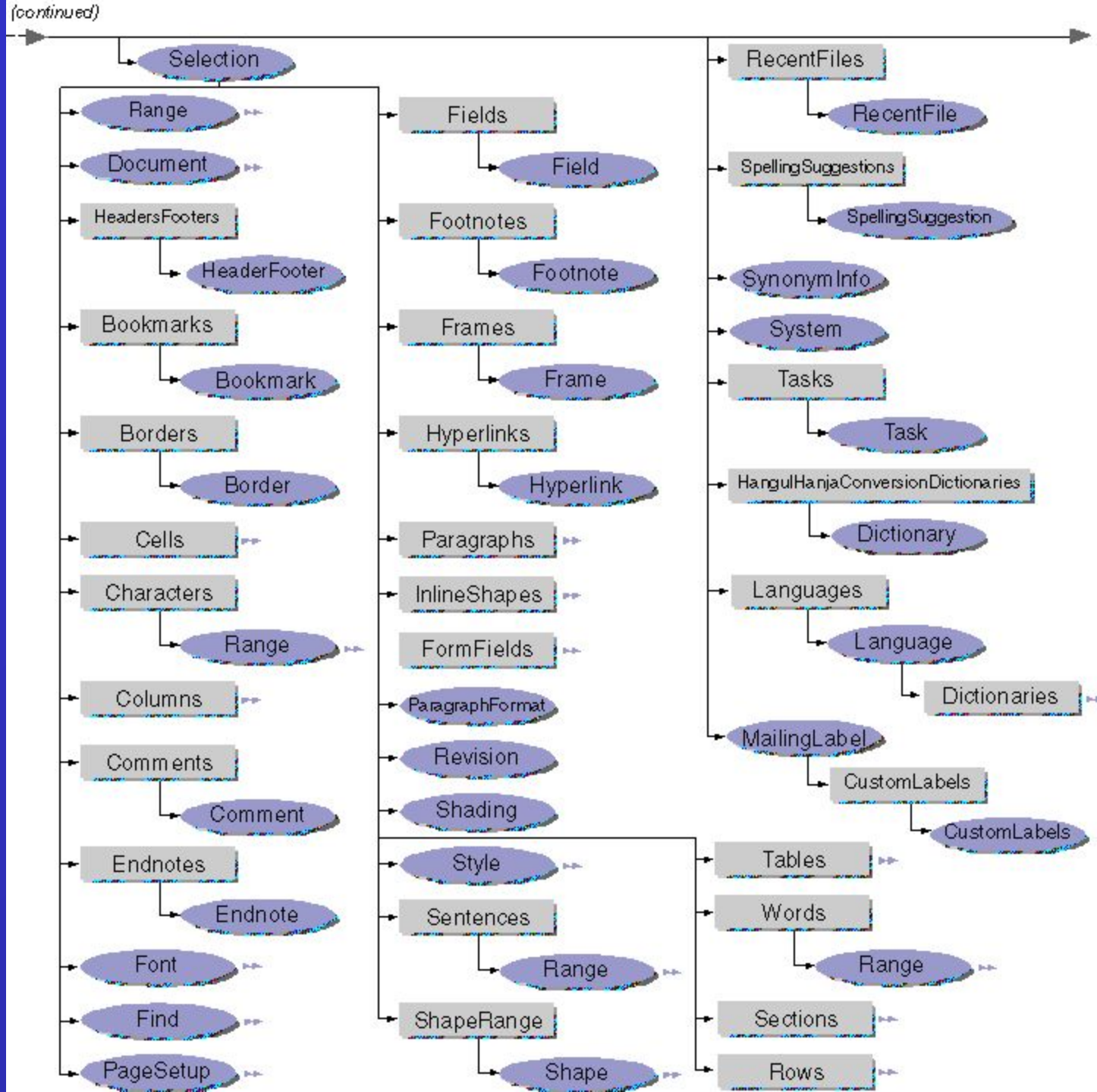
1(7)

Microsoft Word 2000 в MSDN oct99
…/office/MSWord9.olb
VBAWRD9.CHM



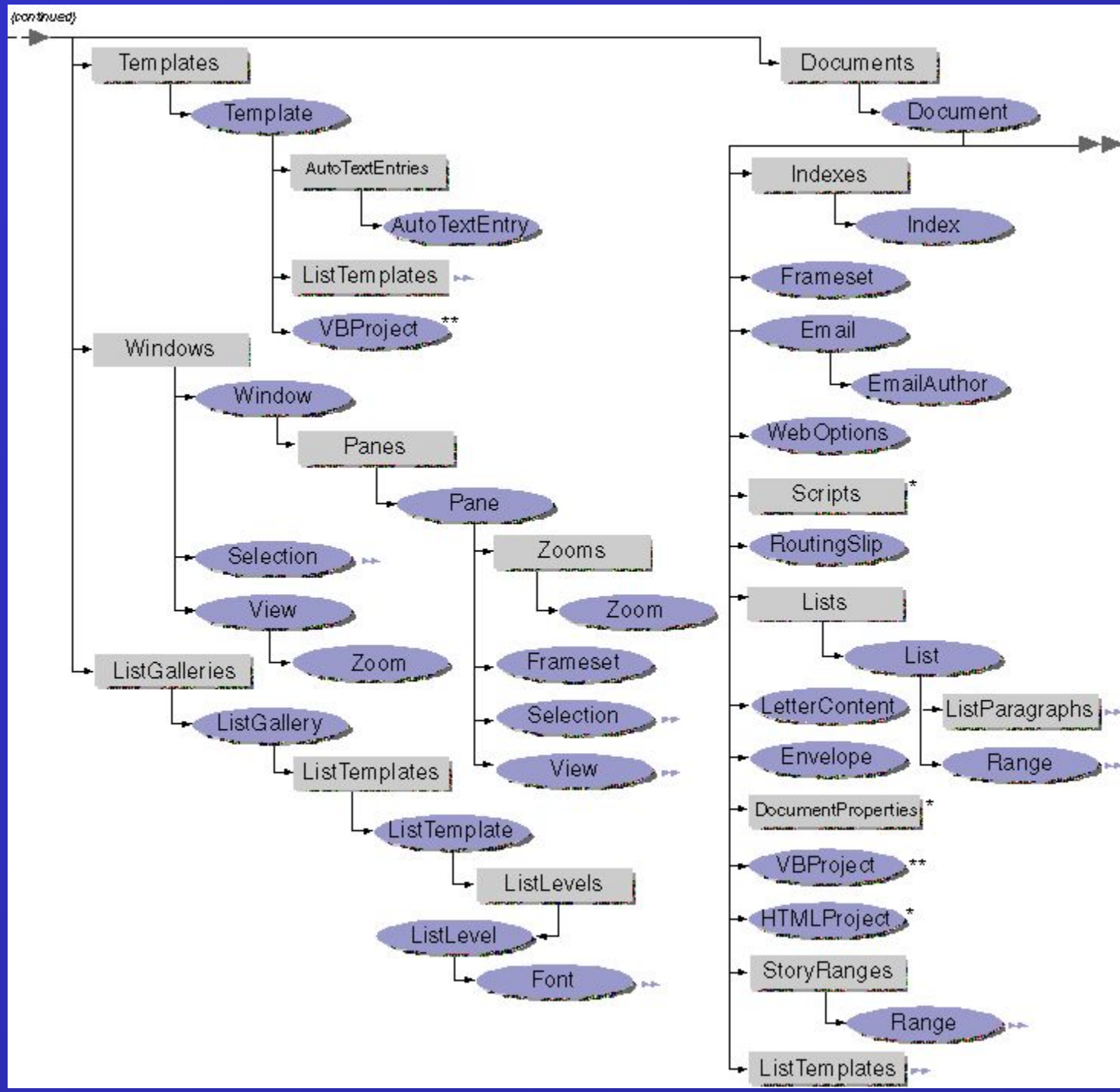
Объектная модель Word 2000

2(7)



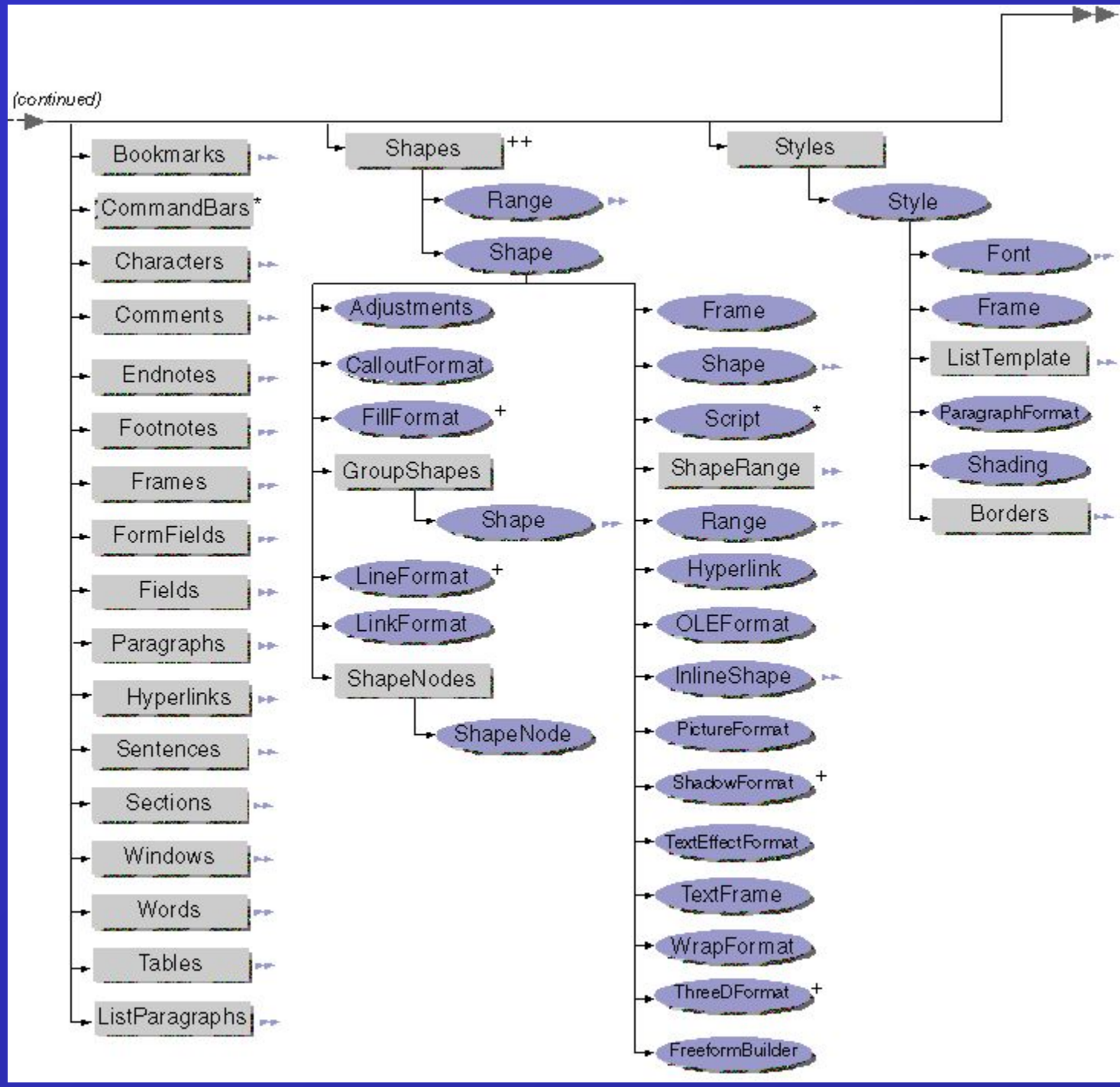
Объектная модель Word 2000

3(7)



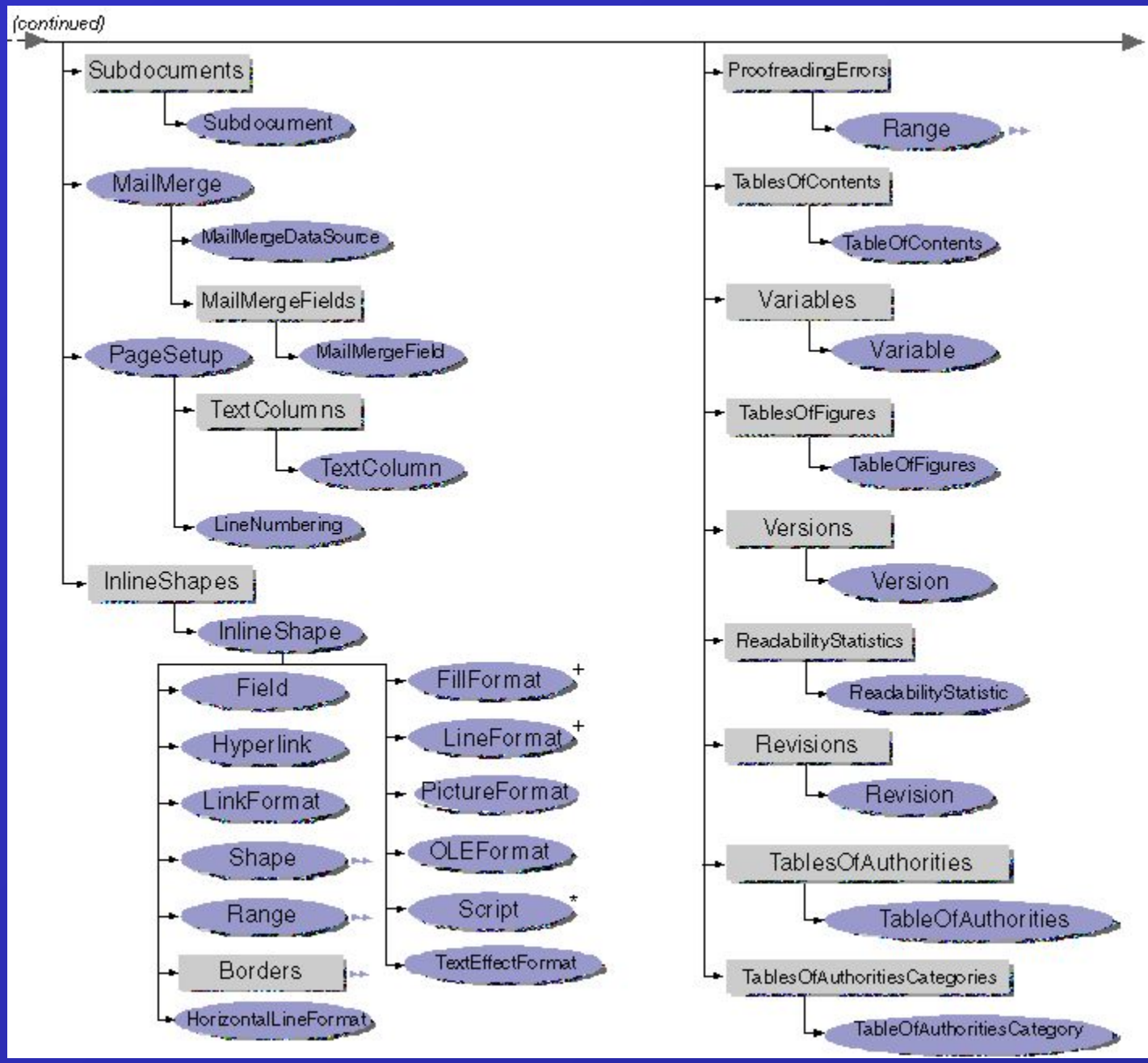
Объектная модель Word 2000

4(7)



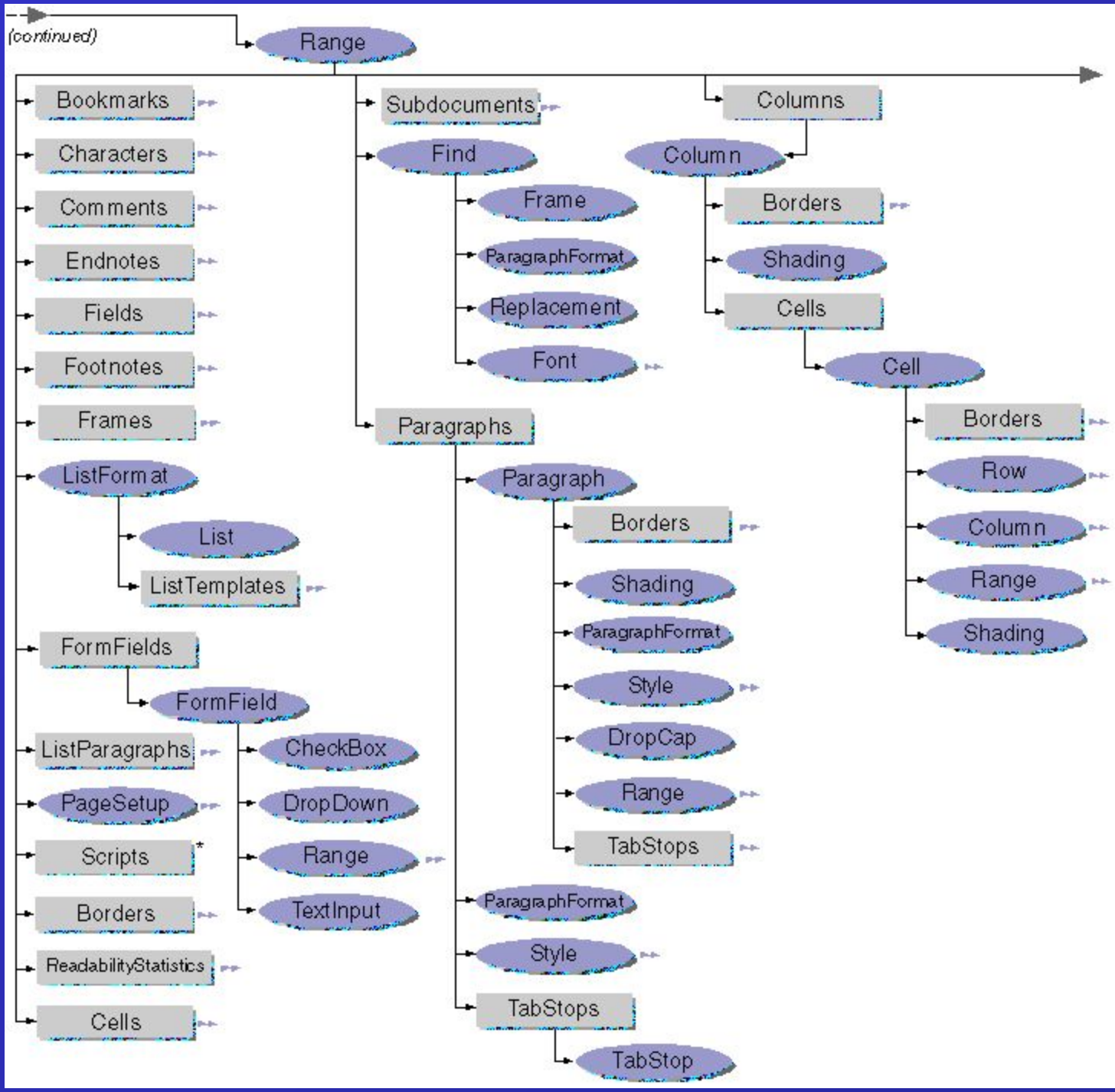
Объектная модель Word 2000

5(7)



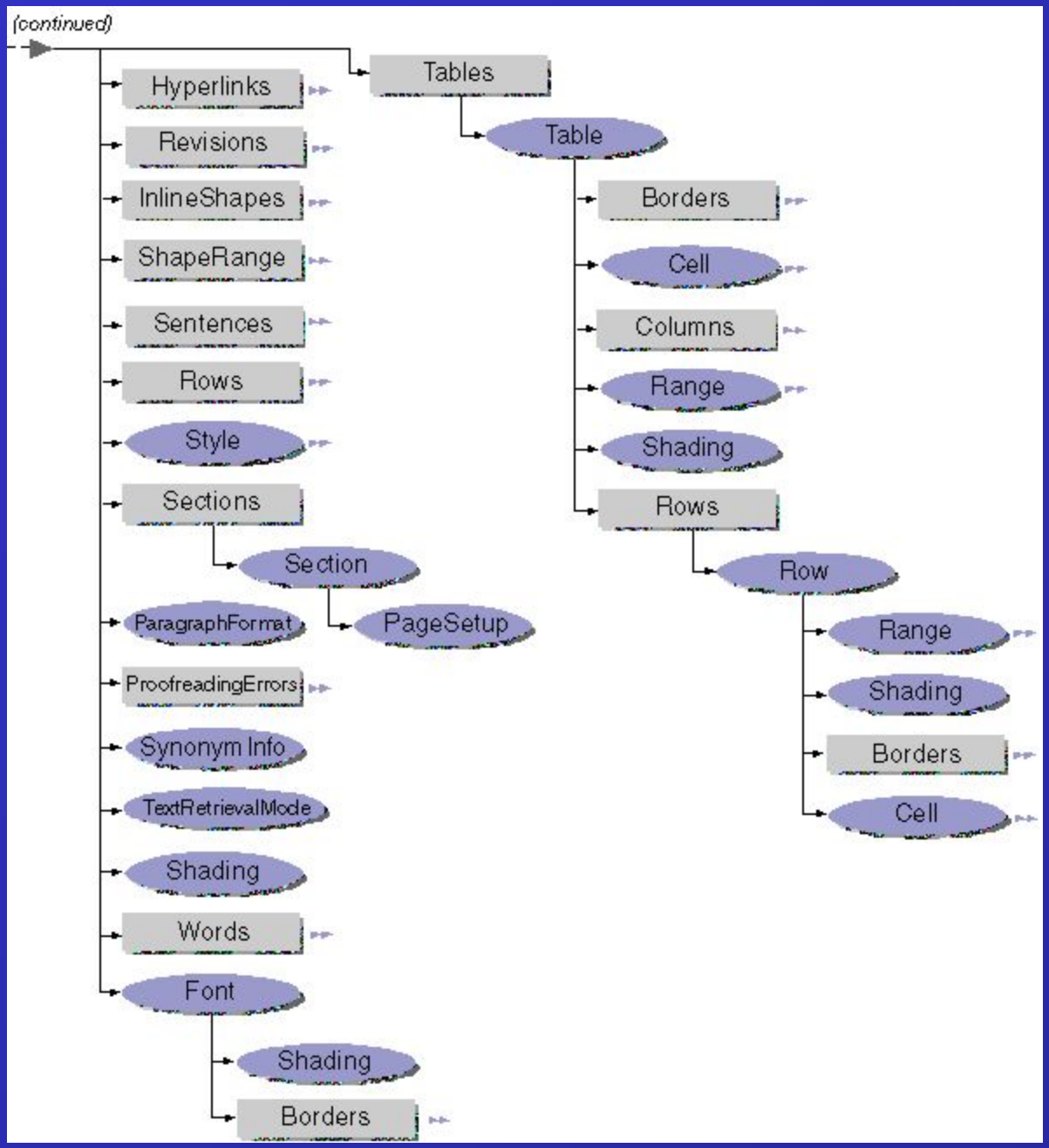
Объектная модель Word 2000

6(7)



Объектная модель Word 2000

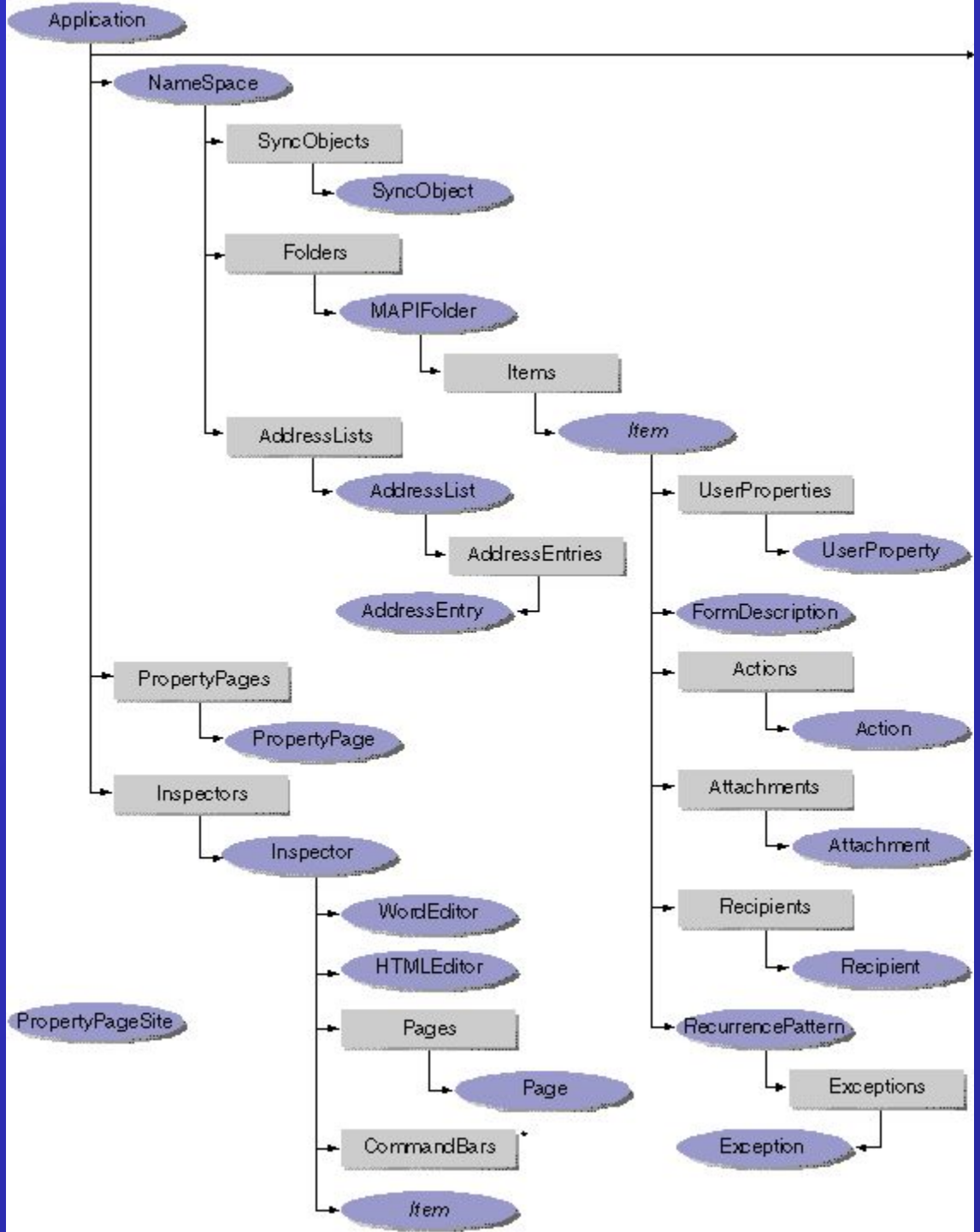
7(7)

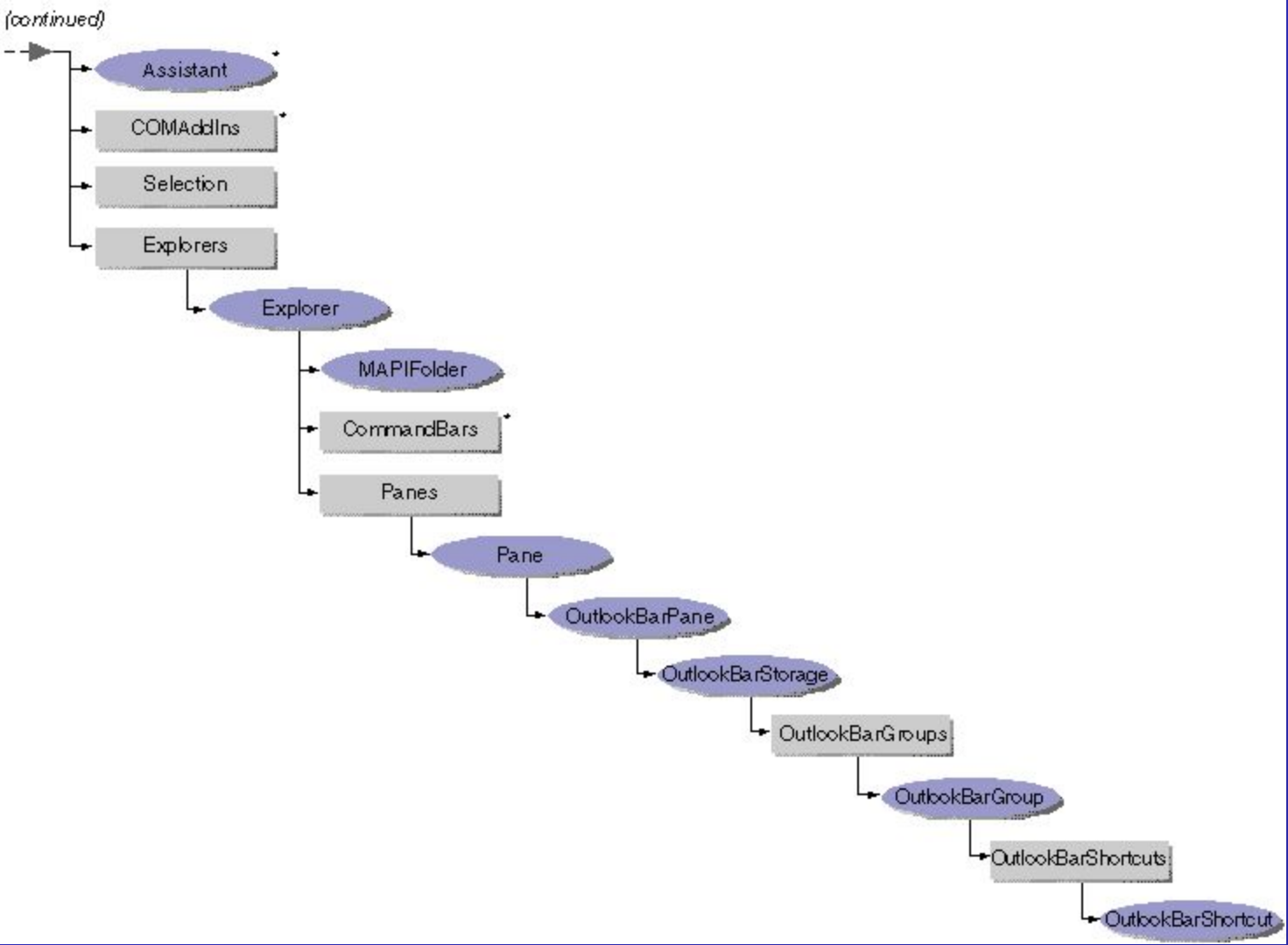


Объектная модель Outlook 2000

1(2)

Microsoft Outlook 2000 в MSDN oct99
…/office/MSOutl9.olb
VBAOUTL9.CHM



Объектная модель Outlook 2000

2(2)