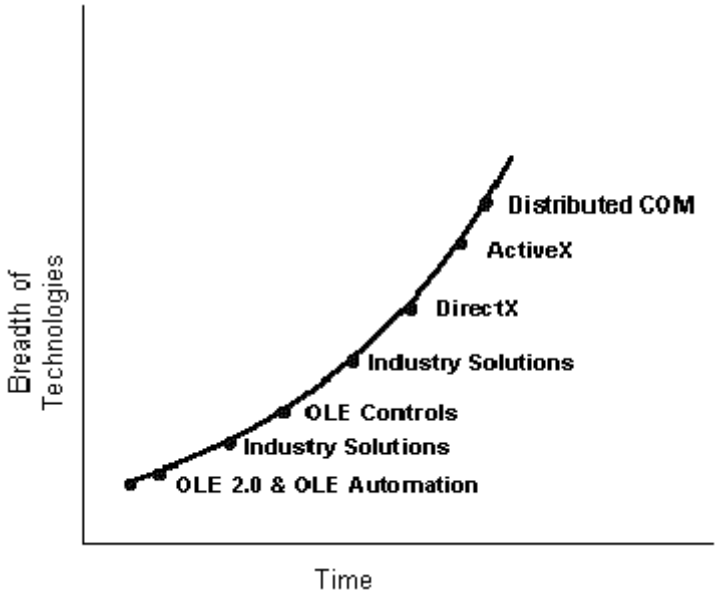
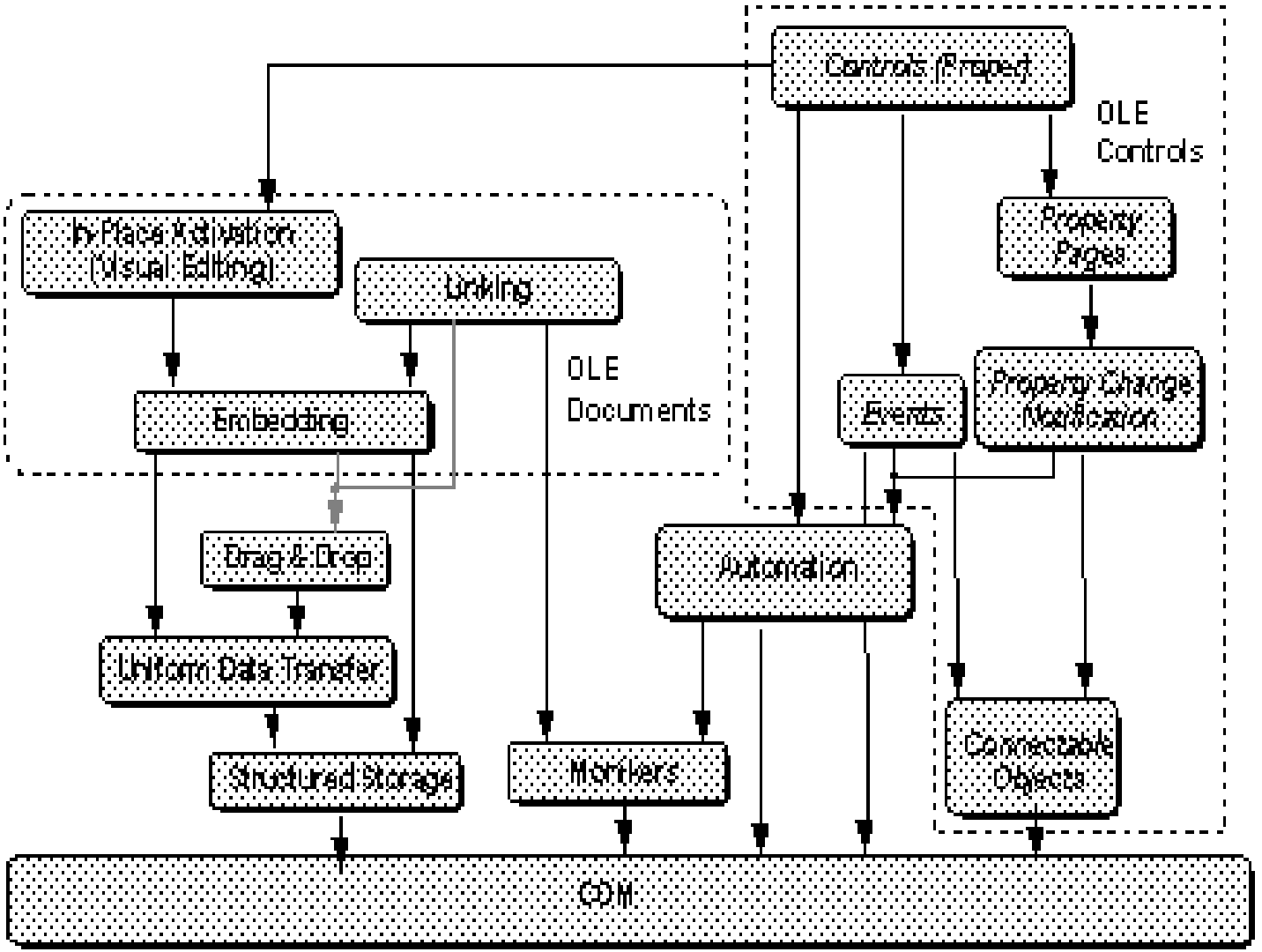
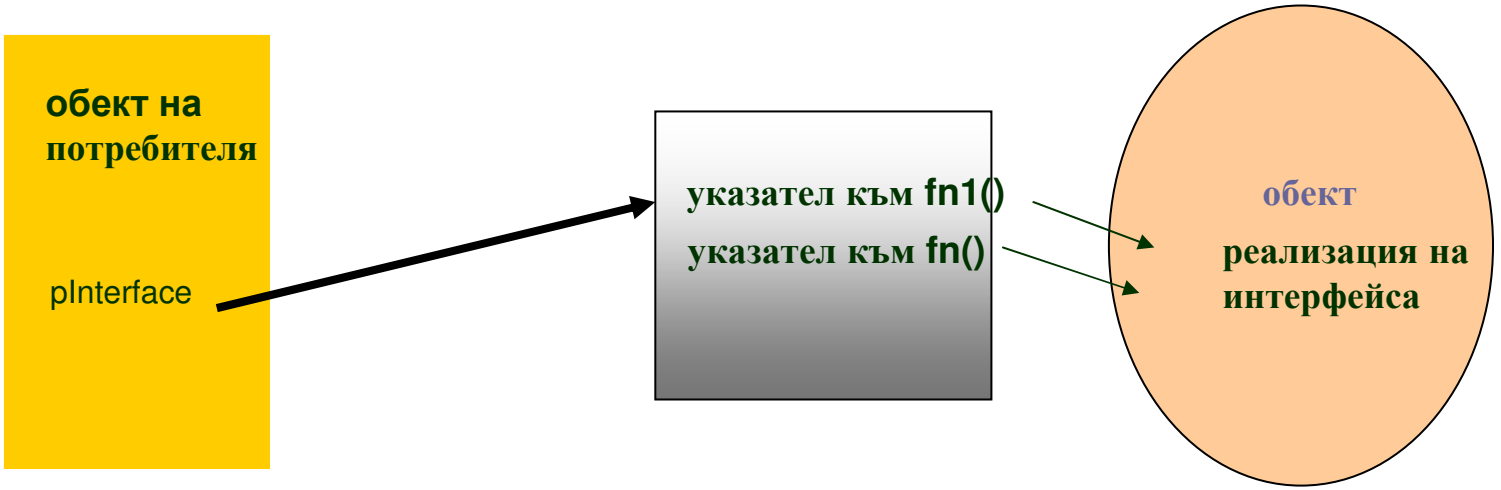
Основи на COM технологията за изграждане на съставни обекти




Controls (Properties)
OLE Controls
In-Place Activation (Visual Editing)
Linking
Property Pages
OLE Documents
Embedding
Events
Property Change Notification
Drag & Drop
Automation
Uniform Data Transfer
Connectable Objects
Structured Storage
Monikers
COM

Обекти и интерфейси

Интерфейс е набор от семантично свързани функции, реализирани в определен обект.
Това включва прототип или сигнатура.
Използването (instantiation) става, когато интефейсът е реализиран и е изработен указател към масив от указатели към функциите му:

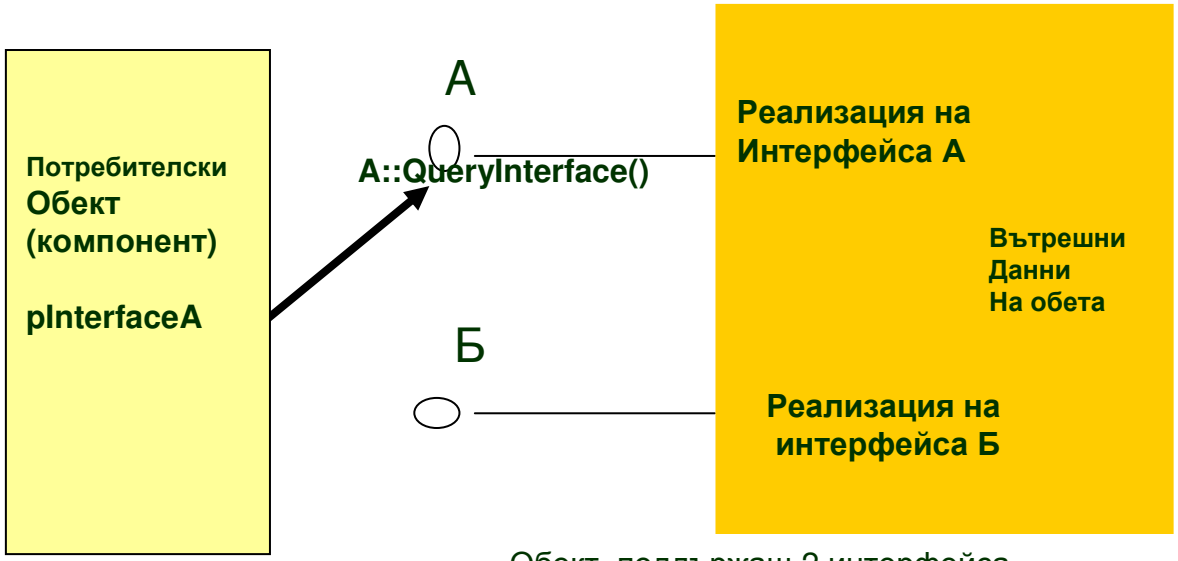


обекти и интерфейси

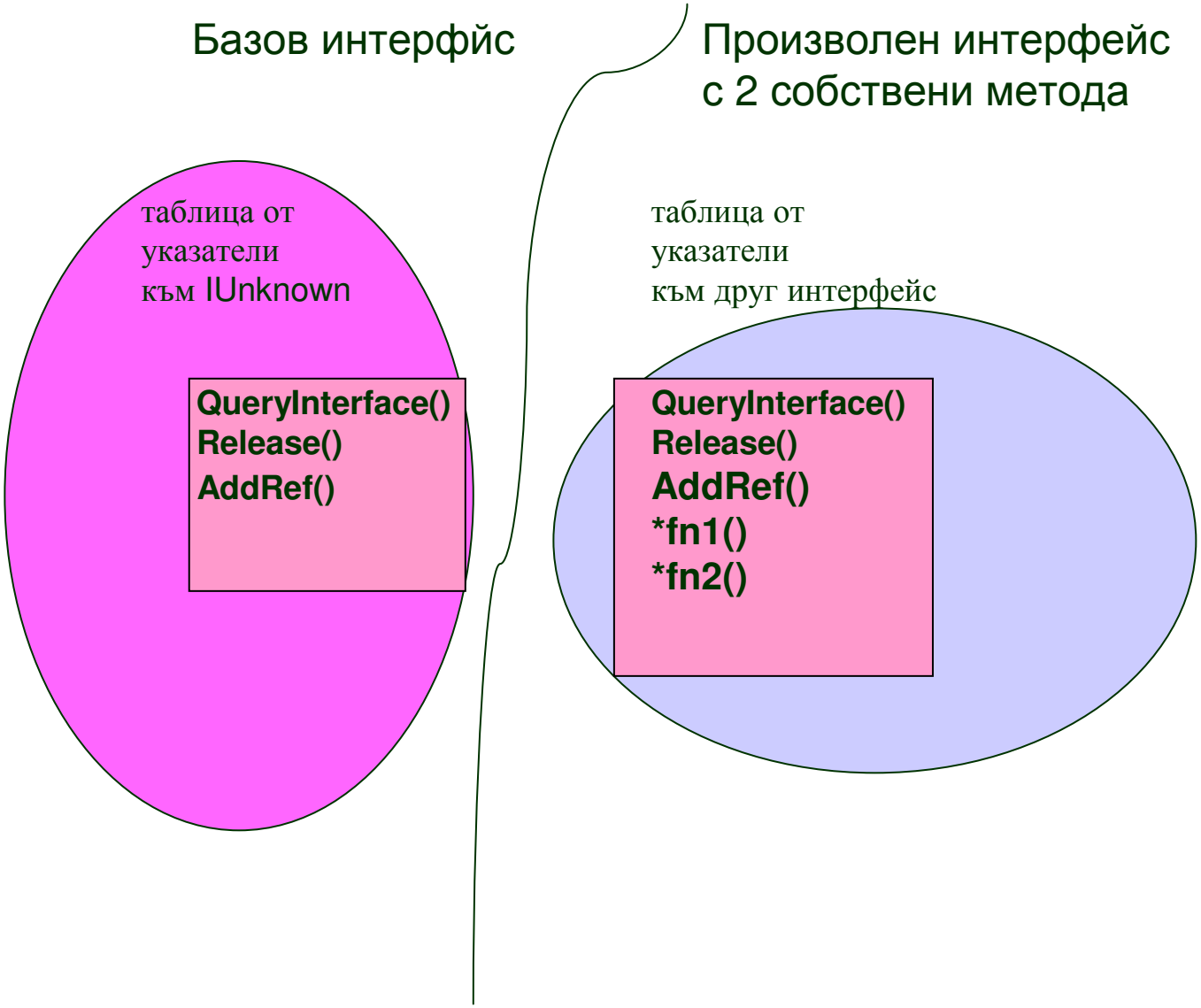
потребителски обект

достъп до функции на интерфейс А

обект с интерфейси



Обект, поддържащ 2 интерфейса


Базов интерфйс
Произволен интерфейс
с 2 собствени метода
таблица от
указатели
към IUnknown
QueryInterface()
Release()
AddRef()
таблица от
указатели
към друг интерфейс
QueryInterface()
Release()
AddRef()
*fn1()
*fn2()

Uniform Data Transfer (UDT)& Notification подсистеми

OLE има 2 въведения:
- **не ограничава формата до този на clipboard, а добавя указания за типа данни, устройство на обмена и препоръчителна среда на обмен.**
- **разделят се протокола от обмена на данните.**

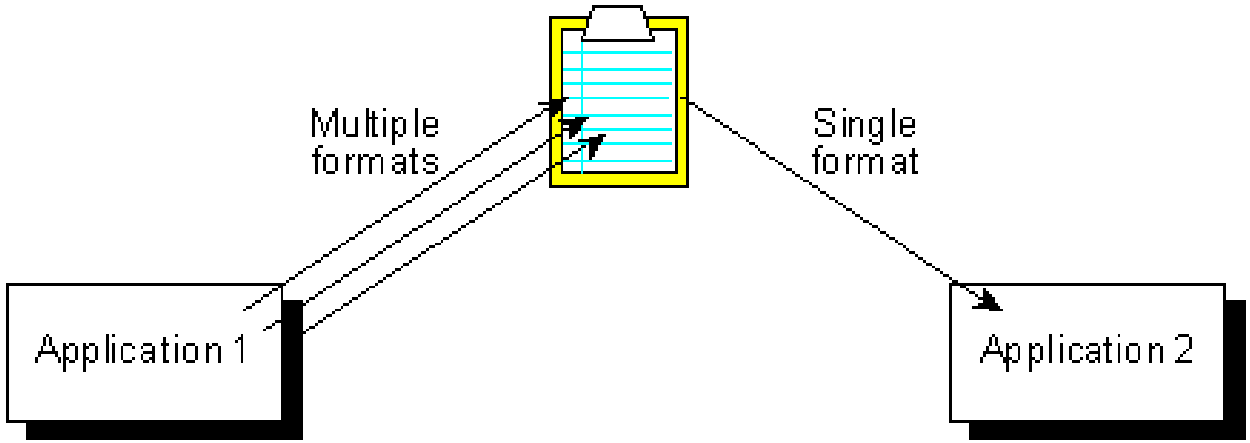
Данновият обект е COM обект и поддържа интерфейс: IDataObject с два съществени метода:
SetData се използва за поставяне на данните в данновия обект и
GetData – за извличане на данните.

Самият даннов обект с поставени в него данни стои в clipboard

поставянето става с
::OleSetClipboard())
а консуматорът на данните получава указател към IDataObject с
::OleGetClipboard.

OLE Clipboard и данновия обект (DO)

1.OLE clipboard е COM базиран. Работи се с методи, част от COM интерфейси, а не с API функции.
2. Данните се прехвърлят не единствено през глобалната памет, както в класическия едноименен механизъм.



Данновият обект и интерфейса IDataObect работят основно с 2 структури:

FORMATETC:

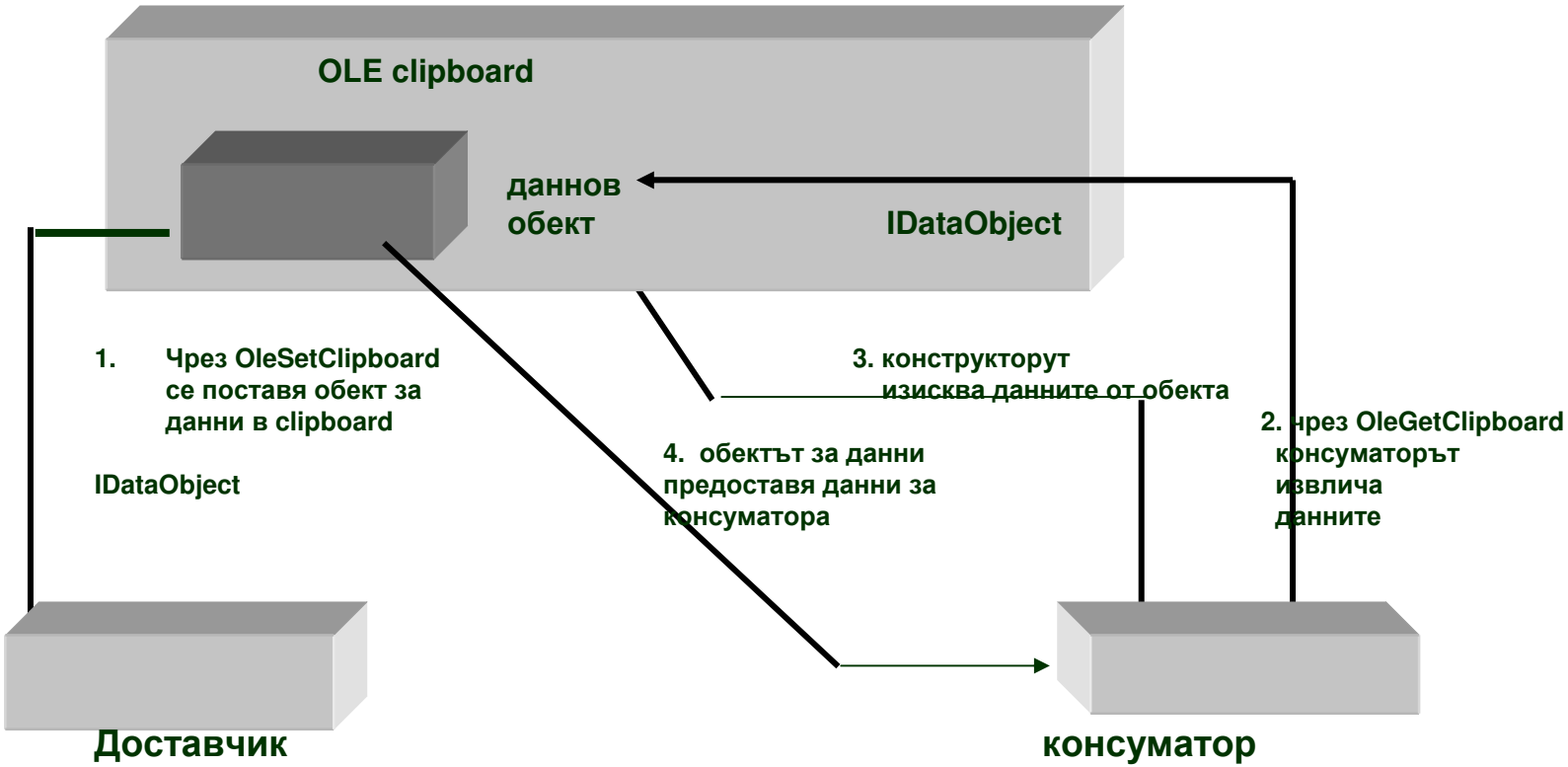
описва обекта, който ще се трансферира - обобщения Clipboard формат на данните,
вида и начина на изобразяване на данни,
възможни места за съхранение на данните, както и тип и х-ки на средата
към която е възможен обмен. Описват се данни за устройството, което
съхранява обекта, възможните начини за предаване на данни
(компресия, като икона, в пълен образ), какво става с образа ако е извън
изобразяваното поле и др.
Тази структура, след инициализация в източника се подава в данновия
обект с COleDataSource::CacheData().

STGMEDIUM

Описва текущото местоположение на данните за трансфер.
Например, при глобална памет, тук се поддържа HGLOBAL,
при файл – низ с името на файла и т.н.

Всеки Windows прозорец може да стане източник на данни за OLE clipboard. За целта е достатъчно в описващия го клас да се инстанциира обект от тип COleDataSource (той поддържа прехвърляните даннии реализира интерфйса IDataObject).

Всъщност в клипборда са не самите данни, а COM обект, капсулиращ ги (IDataObject).



Схемата по-горе, но представена като последователност от обръщения към функции изглежда така:

1. IDataObject::SetData() // данните са в обекта DataObject.
2. ::OleSetClipboard() //данновият обект е в clipboard.
3. ::OleGetClipboard() //конс. взима указател към IDataObject.
4. IDataObject::GetData() //извличат се данните.

MFC поддържа OLE clipboard с помощта на два мощни класа:
COleDataSource и COleDataObject (в консуматора).

Ето примерна последователност от действия, които следва да се реализират в *консуматора* при извличане на дановия обект и същинските данни:

1. Създава се COleDataObject обект;

2. С:
COleDataObject::AttachClipboard()
се свързва нашия COleDataObject с OLE clipboard.

Оттук нататък всички повиквания за данни и методи на COleDataObject се трансферират към методи на IDataObject
(указател за него е получен с ::OleGetClipboard()), чиято обвивка е AttachClibboard();)

3. С повиквания от вид: COleDataObject::GetGlobalData() се извличат данни.

4. Освобождава се блока заета глобална памет.

Действия, извършвани в източника:

1. създава се обект в heap (не в стек) от тип : COleDataSource

2. COleDataSource::CacheGlobalData() //данните се кешират в обекта

3. COleDataSource::SetClipboard() // обекта се поставя в клипборда

друга, полезна абстракция с OLE clipboard:

COleDataObject::GetFileData() асоциира клипборд с файл.

Това ни позволява от този момент да работим с функции на CFile (например метода: GetFileData(), който връща указател към CFile).

Тази абстракция работи за обекти в глобалната памет, файлове, CStream поточни обекти.

Drag&Drop механизмите и данновия обект

- Работи както с цял документ, така и с части от него;
- Прехвърля произволен тип данни.

1. източникът на операция-drag подава data object
(указател към IDataObject на ::DoDragDrop()), определя началото и края на операцията и управлява интерфейса (мишката).
2. Всеки прозорец, който може да бъде целеви , следва да се регистрира в системата (::RegisterDragDrop()).
3. Приемникът на операцията-drop получава data object , определя използвания в него формат и решава какво следва да се направи при drop в целевия прозорец

Drag&Drop интерфейс	Действие
интерфейс IDropSource	имплементиран от обекта – източник на данни.Има 2 полезни метода: за обновяване на курсора и за промени в хода на операцията, вследствие клавиш или мишка. Те се викат директно от COM средата.
интерфейс IDropTarget	имплементиран вобекта – приемник на данните (цел). Има 4 полезни метода: 3 свързани със събития на курсора, когато той е в целта и 1 се вика при пускане – Drop()
Интерфейс IDataObject	В източника

Операции в източника

1.детектира се drag
2. източникът подава с повикването на ::DoDragDrop():
   IDataObject, IDropSource, стойност показваща типа позволени операции над данните
   (напр. местене, копиране), адрес за попълване от DoDragDrop() как е привършила операцията.
   Така източникът знае какво се случва в другия край и напр. при успешен drop може да изтрие
   данните от документа-източник
3.DoDragDrop() държи управлението, до
   1. drop в целта;
   2. прекратяване (Esc).

   През това време COM подсистемата (към ОС) вика методи на IDropSource
   (напр. за обновяване вида на курсор при някакви условия, или източникът да прекрати
   операцията при други..)

Операции в drop целта

1. ::RegisterDragDrop(hwnd, указател към IDropTarget);

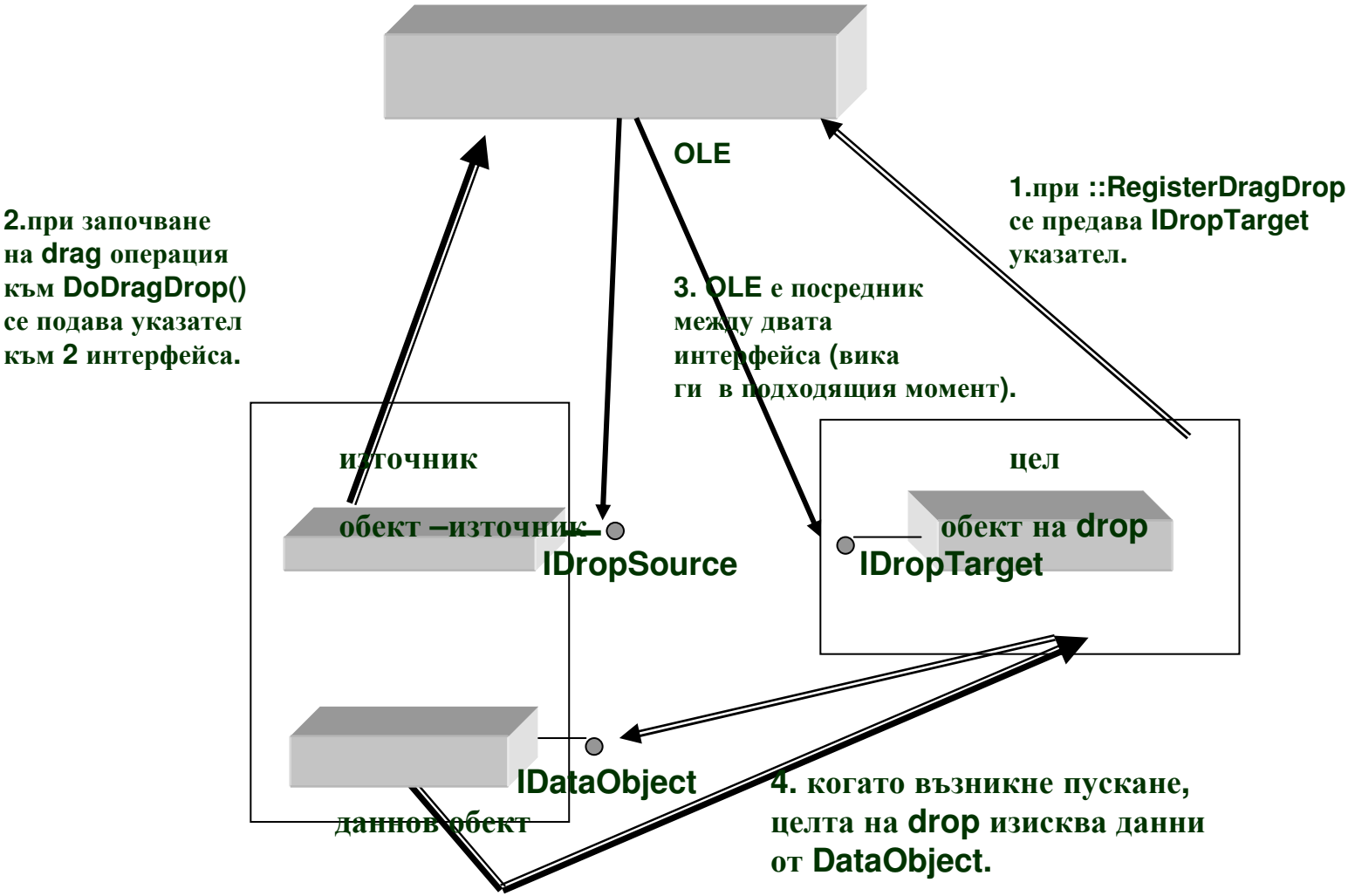
2. когато курсор попадне в целта, системата вика IDropTarget методи (вкл. и Drop() – описващ какво се случва при пускане, с подаден IDataObject и * към място за съобщения към източника). Те се пишат от програмиста!!!

3. Попълва се параметър, връщан към източника

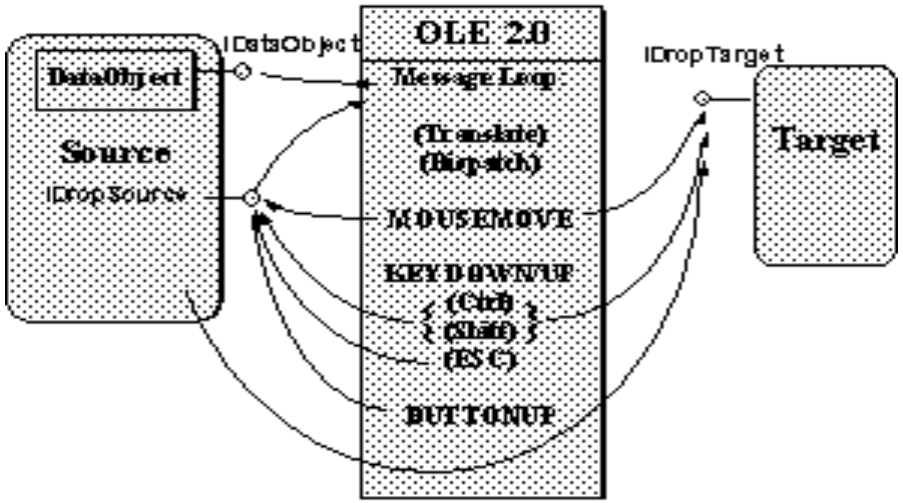
4. извличат се данните (напр. pDataObject --> GetData())

методи на COleDropTarget

припокриване	цел
OnDragEnter	Вика се когато курсор навлезе в прозореца –цел.
OnDragLeave	Определя поведение, когато drag операцията напусне прозореца без пускане.
OnDragOver	Курсорът се влачи в рамките на прозорец.
OnDrop	Тук следва да се поемат за обработване данните – предмет на операцията (това е в прозореца – цел).
OnScrollBy	Определя поведение в случайте, когато в целевия прозорец се налага скролиране.


OLE
1.при ::RegisterDragDrop
се предава IDropTarget
указател.
2.при започване
на drag операция
към DoDragDrop()
се подава указател
към 2 интерфейса.
3. OLE е посредник
между двата
интерфейса (вика
ги в подходящия момент).
източник
обект –източник
IDropSource
цел
обект на drop
IDropTarget
IDataObject
данннов обект
4. когато възникне пускане,
целта на drop изисква данни
от DataObject.
5.DataObject подава данни към
целта на Drop.

DoDragDrop функцията изпълнява цикъл, следящ събитията от мишка или клавиатура и реализиращ повиквания към методи на IDropSource или IDropTarget :



MFC и OLE drag & drop

MFC имплементира класове за всички COM интерфейси към обекти, необходими за целта:

COleDataSource – обект за данни;
COleDropSource – обект източник;
COleDropTarget – обект цел (негова инстанция се поставя в изгледа на приложението – цел).

Примерна извадка от код:

```
// в CMyView напр. добавяте променлива (в  приложение- drop цел.
//Тази даннова променлива е от тип COleDropTarget или производен.

COleDropTarget m_oledroptarget;

//  викате метод Register за създадения тип

m_oledroptarget.Register(this);

// следва предефиниране на методите на IDropTarget,
//особено на OnDrop().
```

а какво правим в източника:

```
HANDLE hdata = ::GlobalAlloc(……., указател към данните);
COleDataSource ods;
ods::CacheGlobalData( CF_TEXT, hdata);
DROPEFFECT de = ods.DoDragDrop(……);
/* методът DoDragDrop() освен че обвива едноименната функция на OLE, поставя
обекта в clipboard и подава указател към IDropSource на ::DoDragDrop(). */
```

Съставни документи: Object linking и обект - Moniker

Документите могат да съдържат части, създавани или обработвани от други.
Тогава говорим за съставни док. В този случай те се явяват източник на данни и трябва да съдържат информация за връзка с обработващото прил (. linking).

Данните за връзка се поддържат в отделен файл.
Така и що се касае до взаимното местоположение на обработващите приложения.
За решаване на този проблем се въвежда обект – moniker. Той съдържа информация как да се осъществи свързването и в какво обкръжение.
Moniker е също COM обект, капсулиращ както името на данновия файл, така и информация, дефинираща обработките над данни с това име. Тази информация е достъпна чрез интерфейс наричан IMoniker.
Потребителите използват обекта moniker, когато искат да работят с това име.
В интерфейса IMoniker се дефинират операциите, които могат да се изпълнят при свързване на обект с данни, носещи даденото име.

Има:

1. File moniker съдържа пътеката и името на файла с данни (абсолютна и отностелна пътека). Когато свързан обект се активира, данните му се взимат от файл по указания абсолютен път, активира продукта с който е бил създаден (по разширението в името), получава обкръжението в което се зарежда за обработка файла (от item moniker) и зарежда файла.
Ако не успее опитва с относителния път. Код за свързване върши всичко това.

2. Item moniker за формиране на обкръжение за обработка.
Съдържа име на обект с данни за обкръжението и код за свързване , който изисква приложението-сървър на тези данни да върне указатл към обекта Item moniker
Съставен moniker включва верига от горните съставки, както и код за свързване, който прохожда последователно всички monikersв списъка и ги активира за връзка със своите обекти и обработващи приложения.

Показана е структура на съставен moniker: