A Neptunusz felfedezésének érdekessége, hogy vele még nem sikerült teljes mértékben magyarázatot találni az Uránusz pályaháborgás aira.

Ennek következtében később egy újabb bolygót kezdtek keresni. Ezt a munkát az amerikai *Percival Lowell* (1855–1916) indította útjára 1905-ben. A fáradságos munka csak 1930-ban – jóval *Lowell* halála után – hozott eredményt. A megtaláló a *Lowell*rõl elnevezett csillagvizsgáló fiatal munkatársa *Clyde W. Tombaugh* (1906–1997) volt. Ezt a bolygót Plutónak nevezték el.

A 18. század közepére a tudósok már tudták, hogy az állócsillagok szférája nem mozdulatlan, vagyis egy több ezer éves hitet sikerült megcáfolniuk, sõt azt is megfigyelték, hogy az egyes csillagok is változnak, fejlõdnek. A Mira Ceti fényváltozásait már 1595-ben felfedezte *Johannes Fabricius* (1587–1615), de e jelenség egzaktabb leírása csak a távcsövek korában született meg. 1669-ben hasonlót tapasztaltak az Algol esetében is.

A 18. században a kutatások a Tejút szerkezetére is kiterjedtek, s így a sztellárasztronómia is önálló szakterület lett. Hogy a Tejút csillagok halmaza, azt már *Galilei* óta látták, de, hogy inkább egy csillagokkal benépesített korongról van szó, az *Thomas Wright* (1711–1786) elmélete. *Herschel*, e század egyik legnagyobb megfigyelõ csillagásza, ezt lényegében igazolja. Ez után már ki tud lépni Galaxisunkból is, felismervén annak nem egyedi voltát, s hogy maga a Galaxis is saját mozgást végez. A távoli objektumok vizsgálatát tette lehetõvé a kettõscsillagok létének herscheli felismerése (1802), s a közöttük levõ fizikai kapcsolat leírása.

Pierre Simon Laplace

Mintegy száz esztendõ alatt tehát az égi mozgások egyszerû kémlelésén, s a Naprendszer belsõ törvényeinek felismerésén túlmenõen nemcsak a Tejútrendszer szerkezetének feltárása, de más galaxisok létének és összetevõinek kutatása is megindult. A csillagászat hatókörét az új, nagy teljesítményû távcsövek félelmetesen kitágították, s mindezek a már kidolgozott dinamikus világkép szép igazolásai voltak. E kor, ha alapjaiban nem is változtatta meg a már elfogadottá vált csillagászati világképet, annak hatókörét mégis bõvítette. *Pierre Simon Laplace* (1749–1827) ugyanis már el merte hagyni a – még newtoni képben is meglevõ – elsõ mozgatót, s az általunk ismerten túli galaxisok létének felismerésével a zárt világegyetem képe egyre nyitottabbá vált.

A dinamikus világkép egyrészt megszabadult az arisztotelészi „elsõ mozgatótól”, másrészt elfogadta a brunói hipotézist a világok sokaságáról, világegyetemünk nem véges voltáról.

Szenkovits Ferenc

Komponensorientált paradigma

II. rész

Elõzõ számunkban nyomon követtük az objektumorientált paradigma átalakulását komponensorientált paradigmává. Most a komponensorientált paradigma objektummodelljeirõl lesz szó.

A SOM modell

Az **IBM SOM** (*System Object Model*) az OS/2 operációs rendszerben jelenik meg, annak érdekében, hogy lehetõvé tegye a rendszer új komponensekkel való kibõvítését.

A leíró és definíciós nyelve megengedi a többszörös öröklõdést, a pointer aritmetikát, de nem léteznek sablonok létrehozására szolgáló mechanizmusok, és nem létezik szemétgyûjtés.

Dinamikus típusellenõrzést hajt végre, verziószámokat tud ellenõrizni, de nem tudja kezelni az osztott objektumokat.

Az OS/2 mellett a SOM az AIX, OS/400 és Mac OS operációs rendszereken mûködhet.

A COM modell

A **COM** (*Component Object Model*) a Microsoft cég komponensorientált objektum modellje és a Windows típusú alkalmazások közötti információ- és objektumcserét szolgálja.

A COM modell interfész csomagját *típus-könyvtárnak* (*type-library*) nevezzük, és az implementációs csomag nem más, mint maga a *Windows Registry*.

A COM objektumok több interfésszel rendelkezhetnek, és mindegyik interfész tartalmaz egy *QueryInterface* metódust, amely lehetõvé teszi a különbözõ interfészek közötti navigálást. Ez tulajdonképpen a polimorfizmusnak egy speciális esete. A kliens a metódus segítségével lekérdezheti a szerver által implementált interfészeket és választhat közülük. A *QueryInterface* metódus ugyanakkor a verziószám lekérdezésében is fontos szerepet játszik. Mindegyik interfész rendelkezik egy globális azonosítóval (*GUID – Globally Unique Identifier*), egy 128 biten ábrázolt számmal, amely térben és idõben változatlan marad, és fontos információkat hordoz az illetõ interfészrõl.

A COM objektumok dinamikus memóriakezelése az *osztály- és objektumgyárak* segítségével valósul meg (*class factory*, *object factory*).

Egy speciális COM-objektum osztályt képeznek az *ActiveX kontroll*ok, amelyek automatikus regisztrálási képességgel rendelkeznek.

Minden COM modellre épülõ interfésznek van közös õs-interfésze, az *IUnknown* interfész.

A következõ példában egy COM modellre épülõ interfészt szemléltetünk:

```
type
  IMalloc = interface(IUnknown)                    //öröklõdés
  ['{00000002-0000-0000-0000-000000000046}']       //int. azonos.
    function Alloc(Size: integer); stdcall;        //hívási mód
    procedure Free(P: pointer); stdcall;
  end;
```

A metódusok kódjának a leírása bármilyen COM nyelvben megtörténhet, az interfész azonosító segítségével ezt be tudjuk importálni.

A nyelvfüggetlenség felvet egy komoly problémát. Tudjuk azt, hogy különbözõ nyelvekben másképp van megoldva a paraméterátadás, a *verem* (*stack*) kezelése, az eljárások függvények hívása. Ezért be kellett vezetni az úgynevezett *hívási konvenciót*. A hívási módokat és ezek tulajdonságait a következõ táblázat szemlélteti:

Direktíva hívási mód	paramétertárolási sorrend	stack leépítés	direkt regiszterhasználat
register	Balról jobbra	a rutin által	Igen
pascal	Balról jobbra	a rutin által	Nem
cdecl	Jobbról balra	a hívó által	Nem
stdcall	Jobbról balra	a rutin által	Nem
safecall	Jobbról balra	a rutin által	Nem

CORBA

A perzisztencia segítségével elértük azt, hogy az objektumok függetlenné váltak az õket létrehozó programtól, vagyis az objektumok címtartománya nem korlátozódik az operációs rendszer által a program számára kijelölt memóriatartományra.

A perzisztencia elvének egyik legismertebb megvalósítása a **CORBA** (*Common Object Request Broker Architecture*), melynek segítségével olyan szoftver-komponenseket definiálhatunk, amelyek különbözõ hálózati pontokon, eltérõ operációs rendszereket használva, egy közös protokollon keresztül képesek a kommunikációra és az együttmûködésre. Ez a protokoll az **ORB** (*Object Request Broker*) és az **IIOP** (*Internet Inter-ORB Protocoll*).

A CORBA operációs rendszer és platform független.

Az ORB felelõs az objektumok közötti kapcsolatok létrehozásáért és fenntartásáért. Fontos szerepe az is, hogy transzparenssé tegye a különbözõ címtartományok közötti kommunikációt. Az ORB felett az objektumok tehát úgy létesítenek kapcsolatot, mintha egyetlen program, egyetlen címtartomány szerves részei lennének.

Az ORB mûködési elve teljesen ráépül a kliens-szerver paradigmára. A kliens objektumokat, komponenseket kér. A szerver objektumokat, komponenseket szolgáltat. Az ORB tehát, feladata megvalósításának érdekében, több összetevõt tartalmaz kliens és szerver oldalon.

Kliens oldalon:

- **A kliens IDL (Interface Definition Language) kapcsolódási felület (Client IDL Stubs)**: tulajdonképpen egy statikus felület a szerver szolgáltatásainak eléréséhez és a szerverobjektumok aktivizálásának módjait tartalmazza. A távoli objektumokat képviseli helyileg – tulajdonképpen interfészek halmaza, amely az elérési, hívási standardokat írja le.
- **Dinamikus hívási felület (Dynamic Invocation Interface, DII)**: olyan dinamikus programok összessége, amelyek futás alatt választják ki a szerver oldali objektumokat és képesek meghívni azok metódusait.
- **Az interfész-szótár programozói felület (Interface Repository API):** futás idejû hozzáférést enged az interfész-szótárhoz. Az interfész-szótár az IDL definíciók feldolgozott formáját tartalmazza: az objektumok és metódusaik leírását, paramétereit. A tárolt adatok futás közben kicserélhetõk, törölhetõk stb.
- **AZ ORB felület (ORB interface)**: szolgáltatások halmaza.

Szerver oldalon:

- **A szerver IDL kapcsolódási felület (Server IDL Stub, skeleton)**: a szerverobjektumok által nyújtott szolgáltatásokat definiálja.
- **Dinamikus kapcsolódási felület (Dynamic Skeleton Interface, DSI)**: a DII párja, futási idõben képes információkat szolgáltatni az elérhetõ metódusokról.
- **Objektumadapter (Object Adapter)**: itt helyezkedik el az objektumok hívásához, létrehozásához, azonosításához szükséges kód.
- **Implementációs szótár (Implementation Repository)**: az osztályok leírását tartalmazza.
- **ORB felület**: a szerver oldalról is elérhetõ, megfelel a kliens oldalinak.

A CORBA osztályok definiálására az IDL (*Interface Definition Language*) nyelvet használjuk. Az IDL deklaratív nyelv. Támogatja a típusdeklarációt, támogatja a metódusok, konstansok, adatelemek, kivételek deklarációját, de nem tartalmaz procedurális elemeket, hisz a metódusokat nem itt kell implementálni, hanem valamilyen más, CORBA-ra támaszkodó nyelvben. Az is elõfordulhat, hogy a különbözõ osztályokat más-más nyelvben implementáljuk – ezek az osztályok könnyen hivatkozhatnak egymásra az IDL deklaráción keresztül. Egy IDL program vázlatosan a következõ:

```
module <azonosító>
{
  <típusdeklarációk>;
  <konstansdeklarációk>;
  <kivételdeklarációk>;

  interface <azonosító> [: öröklõdés]
  {
       <típusdeklarációk>;
       <konstansdeklarációk>;
       <kivételdeklarációk>;
       <attribútumdeklarációk>
       [<mód>] <azonosító> (<paraméterek>)
         [raises <kivétel>] [kontextus];
  }
}
```

Egy IDL struktúra olyan osztályra képzõdik, melynek minden attribútuma publikus. Az osztály két konstruktorral fog rendelkezni, az egyik argumentum nélküli, és minden argumentumot – a típusának megfelelõen – 0-ra vagy `null`-ra inicializál. A másik konstruktor az attribútumoknak megfelelõ paraméterlistával hívható és inicializálja azokat a paramétereknek megfelelõen.

Kovács Lehel

Látványos és érdekes csillagászati jelenségek 2020-ig

A jövõben bekövetkezõ csillagászati jelenségek közül gyûjtöttük össze a fontosabbakat, érdekesebbeket és látványosabbakat. A felsorolt jelenségek idõrendi sorrendben következnek.

A teljes napfogyatkozások közül azokat soroltuk fel, melyek teljességi sávja 5000 km-nél közelebb húzódik hazánkhoz. Ilyenkor elutazhatunk Európa, Észak-Afrika, a Közel-Kelet és Ázsia közelebb esõ helyeire. Megadjuk hazánk és a teljesség sávja közti távolságot, a legközelebbi országot, amely egy-egy napfogyatkozás-expedíció célja lehet. A gyûrûs napfogyatkozások már valamivel kevésbé látványosak. Közülük csak a 3000 km-nél közelebb elhaladó jelenségeket soroltuk fel. Expedíciók ilyenkor is indulhatnak.

A részleges napfogyatkozások látványáért nem érdemes utazni, ezek közül csak a hazánkból is láthatók kerültek jegyzékünkbe. A jelenség közepének idejét és a maximális fázist adtuk meg hazánk közepére. Az ország különféle részein ezek kismértékben változhatnak.

A holdfogyatkozások közül csak a teljes árnyékban lejátszódó teljes fogyatkozásokat említjük, amelyek hazánkból is megfigyelhetõk. Jeleztük a teljesség kezdetét, végét, tartamát és fázisát. A részleges vagy a félárnyékos holdfogyatkozásokat nem soroltuk fel, mivel ezek kevésbé látványosak.

Szerepelnek még a Merkúr-átvonulások a Nap elõtt és a még szebb és ritkább Vénusz-átvonulások.

A Földünkrõl látható valamennyi ilyen jelenséget felsoroljuk, megjegyezve, hogy hazánkból, ebbõl mennyi látszik. Bolygóészlelõk számára fontosak a nagy Mars oppozíciók, amikor több hónapig igen nagy a Mars látszó átmérõje (az azt megelõzõ és követõ oppozíciók is még jónak mondhatók). Ritka és érdekes az élérõl látszó Szaturnusz gyûrûrendszere is.