Monitorozási fogalmak és módszerek (előadásjegyzet)

Kocsis Imre és Tóth Dániel előadása alapján készítette

Micskei Zoltán és Salánki Ágnes

2015.04.13.

A monitorozás folyamata

A monitorozásra a következő, általános meghatározás adható:

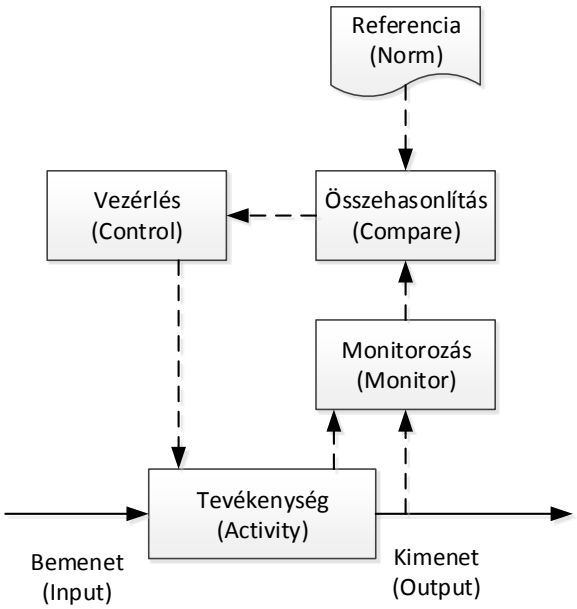
Monitorozás: *A monitorozás valamely „helyzet" megfigyelése, mely során az időbeni változásokat kívánjuk érzékelni.*

A vizsgált rendszer bármelyik, fontos elemét lehet értelme megfigyelni. A monitorozás egy folyamatos, aktuális állapotképet tart fel a rendszerünkről. A monitorozás azonban nem csak megfigyelésre korlátozódik, tipikusan következő lépésként valami beavatkozást vagy riasztást is végrehajtunk.

A monitorozás sokféle jellemzőre kiterjedhet, lehet ez egy adott komponens vagy a rendszer

- normális vagy attól eltérő tevékenysége,
- teljesítménye és kihasználtsága,
- nem engedélyezett változtatása, stb.

Monitorozási ciklus: Az alapvető monitorozási és beavatkozási ciklust az alábbi ábra szemlélteti.



1. ábra: Az úgynevezett 'Monitor Control Loop' (ITIL)

1. A monitorozás során valamilyen tevékenységet figyelünk meg. Megfigyelhetjük valamilyen módon a belső jellemzőit vagy pusztán szorítkozhatunk a kimenetein észlelhető jellemzők vizsgálatára (erre később még visszatérünk). Példaként, ha egy webkiszolgálót akarunk megfigyelni, akkor belső jellemző lehet a kiszolgáló által indított folyamatok vagy

szálak száma, a konfigurációs állomány alapján betöltött webhelyek száma, míg külső jellemző a kliensek által érzékelt válaszidő vagy a sikeres és sikertelen kérések aránya.

2. A monitorozó komponens magától nem tudja értelemszerűen eldönteni, hogy a megfigyelt állapot vagy teljesítményérték megfelelő-e, ehhez nekünk kell definiálni az elvárt referencia értékeket. Ezek alkalmazás és szituáció specifikus értékek lesznek mindig (még ha egy ugyanolyan típusú webszervert tekintünk is, mások lesznek a határértékek attól függően, hogy az a webszerver egy iskolában fut vagy egy bankban).
3. A megfigyelt és a referencia érték összehasonlítása alapján tudunk dönteni a monitorozás eredményéről.
4. Ezek alapján pedig életbe léphet a definiált vezérlés. Itt is széles skálát lehet elképzelni a monitorozott érték megjelenítésétől kezdve email riasztáson át a tényleges, automatizált beavatkozásig (pl. szolgáltatás újraindítása vagy erőforrások átkonfigurálása).

Ha egy nagyobb rendszert tekintünk, akkor a monitorozás erősen összefügg és néha nehéz is igazából szétválasztani a többi felügyeleti folyamattól. Például a monitorozandó komponensek listáját és adatait a konfigurációs-adatbázis szolgáltathatja, a monitorozásból származó adatok bemenetként szolgálhatnak a rövid- és középtávú kapacitástervezésnek, vagy pedig a rendszer különböző komponenseinél megfigyelt jellemzők és jelenségek összesítetett, rendszer szintű feldolgozása majd az eseménykezelés feladata lesz.

Monitorozási megközelítések: magas szinten a következő megközelítéseket érdemes elkülöníteni.

- Aktív vagy passzív:
    - Aktív esetén a monitorozó beavatkozik a rendszer működésébe, és lekérdezi annak állapotát.
    - Passzív esetén a monitorozó csak megfigyeli a rendszer normál működését és üzeneteit, és az alapján próbál meg következetéseket levonni.
- Reaktív vagy proaktív:
    - Reaktív esetén a monitorozó akkor avatkozik be, amikor már valami nem elvárt működését vagy hibát észlelt, tehát a beavatkozás elsődleges célja a hibakezelés.
    - Proaktív esetén megpróbálja a monitorozó megbecsüli az észlelt jellemző alapján a működés trendjét, és még azelőtt beavatkozni, mielőtt ténylegesen kialakul a hibás állapot. Ekkor a beavatkozás elsődleges célja a hiba megelőzése.
- Folyamatos vagy kivétel alapú:
    - Folyamatos monitorozás esetén az adatok gyűjtése és feldolgozása (bizonyos mintavételezési frekvenciát figyelembe véve persze) állandónak tekinthető. Ez a monitorozási mód alkalmas a rendszer időbeli állandóinak meghatározására (a rendelkezésre állás számításához például szükségesek adatok a teljes időtartományból).
    - Kivétel alapú: a monitorozó folyamat elindítását vagy a jelentéskészítést valamilyen különleges esemény váltja ki. Ez kevesebb erőforrást igényel, ugyanakkor fennáll a fontos események elmulasztásának vagy késői érzékelésének veszélye.

Például egy passzív megközelítés lehet, hogy a monitorozott rendszer kifelé irányuló teljes hálózati forgalmát megfigyeljük az alapértelmezett átjárón, és úgy számolunk áteresztőképességet. Proaktív megközelítés esetén amikor látjuk, hogy az elmúlt egy órában ez

a szám folyamatosan csökkent, és közelít a figyelmeztetési szinthez, akkor riasztást küldünk vagy automatikusan beállítunk egy új kiszolgáló egységet.

Látható, hogy az egyes megközelítések között erőforrásigény és ezzel együtt költség szempontból jelentős különbségek lehetnek: az aktív, folyamatos monitorozás jellemzően a hálózati és számítási erőforrásokat tekintve jelent nagy adminisztratív többletterhelést, a proaktív beavatkozás esetén speciális logika kidolgozása szükséges az előrejelzésekhez stb. Ezek persze nyilván csak szemléletek, később a megvalósítások ezeknek valamilyen keverékét alkalmazzák egy valós rendszer monitorozása során.

- Adaptív monitorozást alkalmazó rendszerekben például a mintavételezési frekvencia a kivételek alapján változik, azaz amíg a rendszerünk az elvártnak megfelelően működik, addig alacsony frekvenciával kérjük le az adatokat, amikor pedig valami „fontos” történik a rendszerben, akkor ezt a frekvenciát megemeljük, hogy minél pontosabb képet kaphassunk a rendszerben aktuálisan zajló folyamatokról.
- Hasonlóan előfordul, hogy a monitorozó rendszerünk egy csúszóablakban mindig csak a legutolsó eseményeket tárolja és amennyiben megfelelően működik, csak az aggregált értékeket mentjük el a historikus tárba, kivétel esetén viszont a csúszóablak teljes tartalmát.

Az adatgyűjtés megvalósítása

A monitorozás folyamatából nézzük meg a tényleges megfigyelés, az adatgyűjtés feladatát részletesebben.

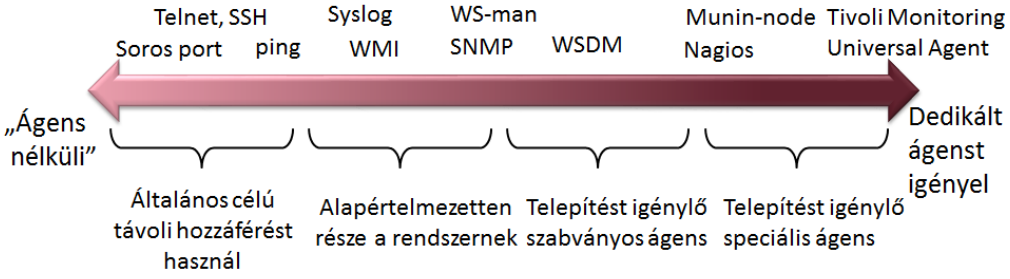
A következő típusú jellemzőket szokás általában távolról megfigyelni és lekérdezhetővé tenni:

- *Pillanatnyi értékek*
    - Skalár mennyiség: CPU-kihasználtság, RAM, tárhely telítettsége, …
        - A pillanatnyi érték nem feltétlenül abszolút értéke jelent, bizonyos esetekben az előző mintavételhez képest számolt különbözetet kérhetjük le (delta, esetleg „jiffy” értékek).
    - Diszkrét értékkészlet: Kiszolgáló-folyamat UP/DOWN/ERROR, …
- *Összegyűjtött mérési adatok*
    - Skalár mennyiség (pl. kumulatív hálózati forgalom)
        - Jellemzően egy meghatározott időablakban mért értékek teljes eloszlása vagy az eloszlás egy származtatott jellemzője. Például fölösleges minden bejövő kérés kiszolgálási idejét egyesével eltárolni, elég azt tudnunk, hogy az 0 és 10 ms között, 10 ms és 20 ms között, 20 ms és 30 ms között stb. van, ekkor csak a válaszidők eloszlását tároljuk. Vagy elég visszaadnunk a válaszidők átlagát, mediánját, maximumát stb. Ezek ekkor mind skalár mennyiségek, de nem pillanatnyi értékre, hanem adott időablakra vonatkoztatottak.
    - Napló bejegyzések
- *Értesítés eseményekről*
    - Diszkrét állapotváltozás (UP → DOWN)
    - Határérték túllépés (diszk telítettség > 90%)

Ágens. Az adatgyűjtés során sokféle, különálló elemet szeretnénk megfigyelni, akár egy adott számítógépen belül vagy akár hálózaton keresztül. A rendszer egy adott komponensének vagy jellemzőjének megfigyelését végző egységet nevezzük ágensnek (agent).

Az *ágens* tipikusan egy olyan kis beépülő komponens, ami képes adatszolgáltatásra valamilyen (hálózati) interfészen, értesítések küldésére különféle események bekövetkezéséről vagy akár egyszerű beavatkozások elvégzésére.

Az ágens és maga az adatgyűjtés kulcskérdése, hogy az ágens mennyire van integrálva a megfigyelt komponenssel. Itt szoktak teljesen beépített és „ágens nélküli” eseteket megkülönböztetni, de érdemesebb ezt inkább egy skálaként elképzelni, ahogy azt a következő ábra is jelzi.



2. ábra: Monitorozási ágens jellege a specializáltság alapján

Tehát a skála onnan indul, hogy például a komponenst kívülről, egy teljesen általános lekérdező eszköz segítségével vizsgáljuk, lehet a platform nyújtotta általános ágenst használni, vagy készíthetünk egy teljesen speciális, az adott komponensbe integrált ágenssel.

Példa 1 (Hardver). Legyen a megfigyelendő komponensünk egy hardver eszköz, egy Ethernet kapcsoló.

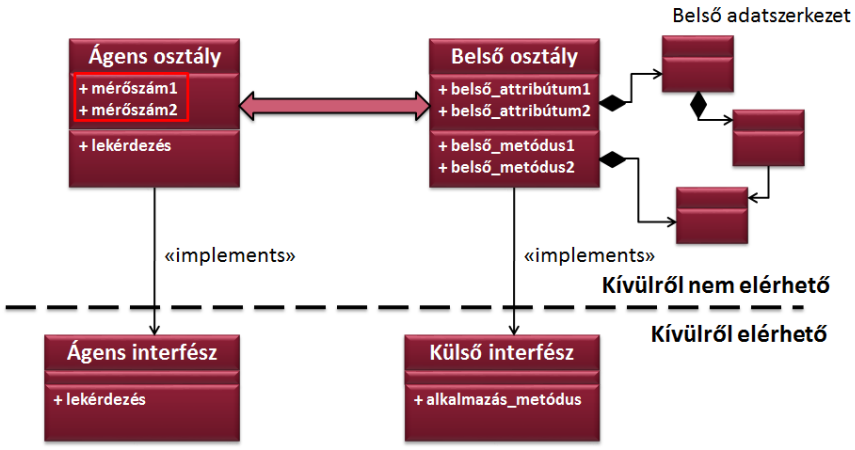
- Alapesetben ez az Ethernet rétegben működik, így kívülről a működését úgy tudjuk megfigyelni, hogy a portjaira kapcsolódva nézzük az oda küldött és onnan jövő Ethernet kereteket. Így azonban viszonylag korlátozott információ áll a rendelkezésünkre (például mert a kapcsoló ha már megtanult egyes hálózati címeket, akkor nem küld ki minden keretet minden portjára).
- Ha részletesebb információra van szükségünk, akkor el kell helyeznünk magában a kapcsolóban egy „ágenst”, ami gyűjti a minket érdeklő adatokat és azt távolról elérhetővé teszi. Így kapunk egy úgynevezett menedzselhető kapcsolót, ami tartalmaz egy menedzsment processzort. Ez közvetlenül hozzáfér a kapcsoló összes belső állapotához és TCP/IP protokollon keresztül azokat lekérdezhetővé teszi.

Példa 2 (Szoftver). Legyen most a megfigyelendő komponensünk egy szoftver, mondjuk egy adatbázis-kiszolgáló. Itt is elindulhatunk a teljesen általános esettől az egyre specializáltabb megoldásokig.

- Kívülről megfigyelve az adatbázis-kiszolgálónak küldhetünk távolról lekérdezéseket, majd mérhetjük a kiszolgáláshoz szükséges teljes időt, és ebből következethetünk a terheltségére. Látjuk azért, hogy ez még azért eléggé áttételes információ.
- Egy következő szint lehet, hogy ha már az adott számítógépen belül fut a monitorozó ágensünk, de még mindig nincs integrálva a megfigyelt komponenssel. Ilyen esetben például az operációs rendszer által nyújtott információk segítségével tudja lekérdezni a kiszolgáló folyamatát. Tipikus jellemzők amiket látunk ilyenkor: fut-e egyáltalán a

kiszolgáló folyamata, mennyi CPU-t vagy memóriát használ jelenleg a kiszolgáló, hány szálat indított el, mennyi és milyen állományokat tart nyitva a lemezről, mennyi I/O kérést hajt végre stb. Ebből már pontosabb képet kaphatunk a működésről és az esetleges hibákról.

- Végül pedig ha még részletesebb adatokra van szükség, akkor az ágenst elhelyezhetjük magán az adatbázis-kiszolgálón belül. Ilyenkor az ágens hozzáfér a belső adatszerkezetekhez és tud akár közvetlen függvényhívásokat végezni. A kiszolgáló forráskódját ilyenkor módosítjuk, adatgyűjtő hívásokat helyezünk el benne, amiken keresztül az ágens értesül a jellemzők pillanatnyi értékéről vagy a fontos eseményekről. Ezt a módosítást szokás felműszerezésnek (instrumentálásnak) nevezni. Egy ilyen elrendezésre mutat példát az alábbi ábra.



3. ábra: Monitorozandó alkalmazás instrumentálása belső ágens elhelyezésével

Fontos megjegyezni, hogy azt, hogy melyik megközelítést érdemes választani, azt minden egyes szituációban mérlegelni kell, hisz mindegyiknek megvan az előnye és hátránya. Igaz, hogy egy specializált ágenssel sokkal több adathoz férünk hozzá, de ennek megvalósítása drágább és bonyolultabb is (hardver esetén bele kell még építeni a felügyelő komponenst, szoftver esetén ki kell fejleszteni vagy konfigurálni az ágenst). Ugyanakkor lehet, hogy nincs is szükség az adott szituációban erre a többlet információra, vagy a specializált ágens használatára nincs is lehetőség (pl. nem áll rendelkezésre az alkalmazás forráskódja).

Lekérdezési interfész. A monitorozó ágensektől le kell tudni valahogy, legtöbbször távolról az összegyűjtött adatokat. Természetesen erre dolgozhatunk ki saját protokollokat, de érdemes lehet inkább valami szabványos, gyártó-független megoldást választani.

A lekérdezés megvalósításánál tipikusan két működési elvet szokás megkülönböztetni.

- *Pull* – a központi adatgyűjtő kezdeményezi az ágensek lekérdezést;
- *Push* – az ágens kezdeményezi az adatok elküldését a feliratkozott adatgyűjtő központnak.

(Figyelem: a különböző megvalósítások ezeket a fogalmakat eltérően használják, mindig érdemes megnézni, hogy az ágens vagy a monitorozó rendszer szemszögéből nézzük a kommunikációt).

A következő lista néhány gyakrabban használt lekérdezési protokollt gyűjt össze csoportosítva.

- *Általános távoli hozzáférési protokollok*: például telnet, SSH, de akár ide érthető legegyszerűbb esetben az ICMP ping üzenet is.

- *Hálózatmenedzsment protokollok*: alapvetően hálózati szintű jellemzők megfigyelésére kifejlesztett protokollok, de használhatóak akár alkalmazás szintű jellemzők gyűjtésére is, pl. Simple Network Management Protocol (SNMP), Netflow.
- *Konfigurációmenedzsment protokollok*: alapvetően konfigurációs jellemzők, hardver és komponens leltár kezeléséhez kifejlesztett protokollok, de természetesen ezek segítségével is lehet akár teljesítményjellemzőket lekérdezni, pl. CIM-XML vagy WS-Management.
- *Platform-specifikus protokollok*: egy-egy adott platform vagy keretrendszerhez kifejlesztett, de gyártó-független protokoll, pl. a Java platformhoz specifikált Java Management Extensions (JMX).
- *Gyártó-specifikus megvalósítások*: egy-egy adott nyílt vagy zárt forráskódú monitorozó keretrendszer által használt saját megvalósítás, pl. a Nagios keretrendszer egyszerű szöveges protokollja vagy az IBM cég Tivoli termékcsaládjának monitorozó ágense.

Ezekből a protokollokból az adott szituációt mérlegelve érdemes választani.

A távoli lekérdezés egy speciális fajtáját érdemes még megjegyezni, ez pedig a *szondázás* (probing). Ilyenkor kifejezetten távolról, a szolgáltatás normál hozzáférési pontját használva szeretnénk képet kapni a rendszer állapotáról. Ez egy egyszerűen megvalósítható megoldás, cserébe hiba esetén nehéz a hiba helyét pontosan meghatározni. Például ha egy többrétegű webes alkalmazást monitorozunk, akkor ha a szonda megpróbálja lekérni a webes alkalmazás kezdőoldalát, de az nem sikerül, akkor e mögött rengeteg hibalehetőség lehet a háttérben. Lehet, hogy a külső internetkapcsolat hibásodott meg vagy a webszerver hardvere állt le vagy a webszerver folyamata omlott össze vagy a háttérben lévő adatbázis állt le stb. Ilyen esetben akkor tudunk pontosabb képet kapni, ha többféle szondát használunk, amik különböző hibalehetőségekre érzékenyek, és azok összesített állapotából próbáljuk detektálni majd behatárolni az egyes hibákat. Ezt a folyamatot nevezik diagnosztikának.

Egy rendszerben általában az alábbi diagnosztikai részfeladatokat különböztetjük meg:

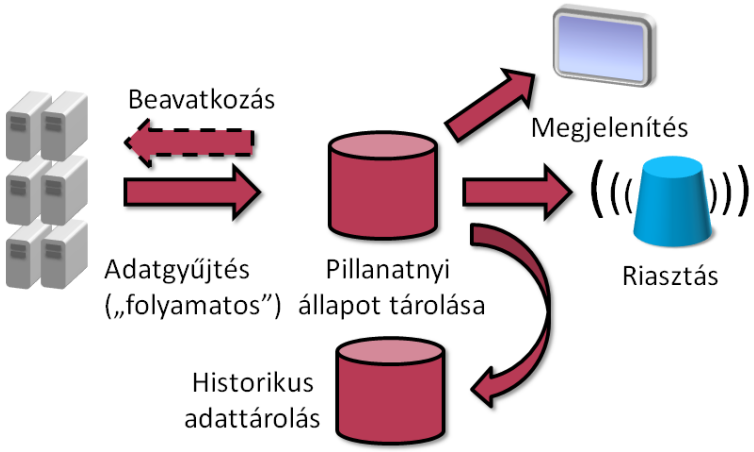
- Detektálás – azt próbáljuk meg kitalálni, van-e hiba a rendszerünkben;
- Lokalizálás – meg is határozzuk a hiba helyét. Számos esetben a hibahely meghatározása jóval komplexebb feladat, ez függ attól is, pontosan milyen mélységig szeretnénk a hibát behatárolni (pl. elég-e az, hogy az adatbázis kiszolgálóval van a baj, vagy ennél finomabb felbontással dolgozunk).

Monitorozási keretrendszerek

Egy általános monitorozási keretrendszert az alábbi ábrán szemléltetett módon lehet elképzelni (4. ábra).

Az adatgyűjtésről és az esetleges beavatkozási lehetőségekről az előző fejezetben volt szó. Ezt egészítik ki a rendszerek riasztási lehetőségekkel (email, SMS, chat), valamint egy megjelenítő felülettel.

A felület megtervezése esetén a kihívás az, hogy a nagy mennyiségű komponenst és megfigyelt jellemzőt valami könnyen áttekinthető formában adjuk meg, amiből könnyen észreveszi az adminisztrátor az esetleges problémákat és a fontos összefüggéseket. Ehhez általában a számítógépek és komponensek topológiáját megjelenítő térképeket, a jellemzők és állapotok színkódolását vagy hőtérképen való megjelenítését és hierarchikus elrendezést szoktak használni.



4. ábra: Általános monitorozási keretrendszer felépítése

	ESX Management Console	OK	2012-03-27 02:50:16	27d 10h 40m 46s	1/4	TCP OK - 0.000 second response time on p
	HTTP	OK	2012-03-27 02:46:30	55d 17h 27m 29s	1/4	HTTP OK: HTTP/1.1 301 Moved Permanently
	HTTPS	OK	2012-03-27 02:48:00	55d 17h 26m 35s	1/4	HTTP OK: HTTP/1.1 200 OK - 5293 bytes in
	PING	OK	2012-03-27 02:48:17	55d 17h 23m 12s	1/4	PING OK - Packet loss = 0%, RTA = 0.66 ms
	ESX Management Console	CRITICAL	2012-03-27 02:49:17	128d 2h 29m 24s	1/4	No route to host
	HTTP	CRITICAL	2012-03-27 02:50:19	128d 2h 29m 24s	1/4	No route to host
	HTTPS	CRITICAL	2012-03-27 02:46:30	128d 2h 29m 24s	1/4	No route to host
	PING	CRITICAL	2012-03-27 02:48:00	128d 2h 29m 24s	1/4	CRITICAL - Host Unreachable (152.66.253.1
	ITIM	OK	2012-03-27 02:48:19	53d 10h 46m 30s	1/4	HTTP OK: Status line output matched "HTTP/
	PING	OK	2012-03-27 02:49:24	51d 4h 40m 35s	1/4	PING OK - Packet loss = 0%, RTA = 0.27 ms
	SSH	OK	2012-03-27 02:46:30	23d 21h 19m 41s	1/4	SSH OK - OpenSSH_3.9p1 (protocol 1.99)

5. ábra: Példa monitorozási adatok megjelenítésére (Nagios)

A fenti ábrán a Nagios nyílt forráskódú, kisebb rendszerekhez (értsd kevesebb, mint 100 számítógép) szánt monitorozási keretrendszer felülete látszik. Az egyes csoportok számítógépeket jeleznek, azokon belül pedig a rajtuk futott ágensek és ellenőrzések látszanak. Például az első gép egy VMware ESX virtualizációs kiszolgáló, amit többféle módon figyelünk, kezdve az egyszerű PING ágenstől a speciális, az ESX menedzsment konzolját lekérdező ágensig. Ezek jelenleg mindegyike helyes állapotot (OK) detektál, és a jobb szélen lévő oszlopban látszik, hogy további, részletesebb adatokat is lekérdeztek.

Végül, a kinyert (akár részleges, akár aggregált) értékek historikus adattárba töltésének célja, hogy a rendszert később részletes offline analízisnek vessük alá (a valós idejű feldolgozásnál és jelentéskészítésnél gyakran hiányzik az idő vagy a számítási erőforrás teljes körű online analízishez) és pontosabb képet kapjunk rendszerünk működéséről – a rendszerfelügyeletnek ezen aspektusát az adatelemzéssel foglalkozó blokk foglalja majd össze.