A rendszerszintű gondolkodás fejlesztésének oktatásmódszertani lehetőségei az informatika oktatásban

Korom Szilárd[1], Illés Zoltán[2]

{[1]korom.szilard, [2]illes}@inf.elte.hu
ELTE IK

Absztrakt. Az informatika oktatása során a fejlesztendő kompetenciák között legtöbbször az algoritmikus gondolkodást, a problémamegoldást vagy az alkalmazói képességet említik. A rendszerszintű gondolkodás, mint informatikai kompetencia fejlesztésének lehetőségei ritkán kerülnek elő, pedig annak fontosságáról, jelenlétéről és elkerülhetetlenségéről már számos tanulmány készült. Az alábbi cikkben azt kívánjuk bemutatni, hogy a gyakorlati oktatás során egy tananyagot hogyan lehet úgy felépíteni, hogy az alkalmas legyen a kompetencia fejlesztésére.

Kulcsszavak: rendszerszintű gondolkodás, rendszergondolkodás, informatikai kompetenciák, kompetenciafejlesztés

1. Bevezetés

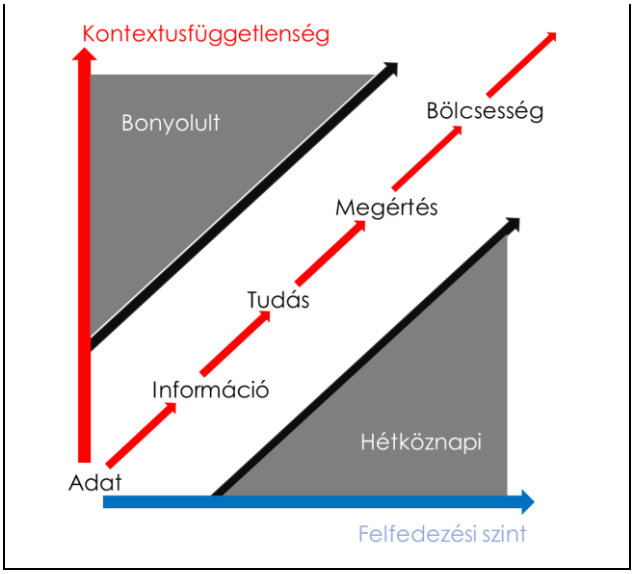
A rendszerszintű gondolkodás egy olyan konkrét kompetencia, mely az tanításmódszertani kutatásokban ritkán jelenik meg, bár már régóta tudjuk, hogy az informatikai kompetenciák egyike [1]. Mindemellett professzionális környezetben egyre részletesebben foglalkoznak vele. A szoftvertechnológiai, szoftverfejlesztés jellegű kutatásokban egyre gyakrabban említik nem csak, mint kompetenciát, de mint képességet, szemléletmódot is. Úgy tűnik ez elengedhetetlen nem csak az informatikával foglalkozó szakember számára, de szinte minden területen, ahol alapvetően komplex rendszerek működtetése, fenntartása a cél, márpedig egyre bonyolultabb rendszerek vannak az élet szinte minden területén, így a rendszergondolkodókra is egyre nagyobb szükség van [2].

A rendszerszintű gondolkodást már számos tanulmányban definiálták valamilyen módon, melyeket Arnold és Wade egyesített [2]. A definíciót lefordítottuk egy korábbi tanulmányunkban [3]:

> *„A rendszerszintű gondolkodás olyan szinergikus analitikus készségek csoportja, melyek javítják egy rendszer azonosításának és megértésének képességét, megjósolják viselkedésüket és módosítják azokat, a kívánt hatások elérése érdekében. Ezek a készségek rendszerként működnek együtt."*

A definíció önmagában kevés a számunkra. Ahhoz, hogy a rendszerszintű gondolkodás fejlesztéséhez alkalmas tananyagokat tudjunk készíteni, szükség lesz egy modellre, mely a rendszerszintű gondolkodást keretbe foglalja, mely útvonalat ad, így ellenőrizni tudjuk, mikor éri el a tanuló a kívánt szintet. A rendszerszintű gondolkodás fejlesztéséhez a DIKUW modellt érdemes párosítani [4]. Ez egy betűszó, ahol a betűk jelentései:

- *Data*, vagyis adat
- *Information*, vagyis információ
- *Knowledge*, vagyis tudás
- *Understanding*, vagyis megértés
- *Wisdom*, vagyis bölcsesség



1. ábra: A DIKUW (Adat; Információ; Tudás; Megértés; Bölcsesség) hierarchia [5]

A 1. ábra egy reprezentációja, hogy a tanuló egy új tananyagelemmel milyen utat járhat be. Például az adatbázis kezelésnél az „adat" azt jelenti, hogy a diák már hallott az Access programról, hallott kulcsszavakat (pl.: tábla, adatbázis), de ezek mögött semmiféle valós megértés, vagy jelentés nincsen. A végcél pedig nyilvánvalóan az, hogy a hallgató képes legyen felismerni problémák, rendszerek esetében pontosan hogyan, milyen adatbázis érdemes használni, tehát létezik egyfajta kompozíciós és dekompozíciós képesség is például. Természetesen a konkrét rendszer definiálja, pontosan milyen képességre lehet szükség. A képességek teljes listáját Arnold és Wade meghatározta a következő fő kérdések mentén:

- Hogyan közelítjük meg a rendszereket és a rendszerszintű problémákat?
- Mi a rendszer, mi van benne, és mi van rajta kívül?
- Hogyan szerveződik a rendszer tartalma?
- Hogyan hatnak egymásra a szervezet, az elemek, azok tulajdonságai és más tényezők, hogy létrehozzák a viselkedést? Mit tehetünk, hogy megváltoztassuk ezt a viselkedést?

Összességében tehát az oktató képes felmérni a rendszerszintű gondolkodás szintjét, ám ami még fontosabb, az 1. ábra, valamint Arnold és Wade listájának segítségével képesek vagyunk tananyagot készíteni, mely alkalmas a képesség fejlesztésére különböző korosztályokban, különböző témakörökön keresztül. A továbbiakban ezeket a lehetőségeket kívánjuk bemutatni részletes példák segítségével.

2. Kutatásmódszertan

A rendszerszintű gondolkodás fejlesztésére alkalmas tananyagokat dolgoztunk majd próbáltunk ki különböző tanulói csoportokkal. A kutatás lényege elsősorban a megvalósíthatóság és a tapasztalatok begyűjtése, rendszerezése volt. Természetesen akkor lenne teljes a kutatás, ha mérni tudnánk, hogy a módszertanunkkal ténylegesen tudtuk fejleszteni a rendszerszintű gondolkodást, illetve azt is, hogy más módszertannal nem tudnánk ilyen hatékonyan fejleszteni azt. Sajnos ezt még nem tudjuk

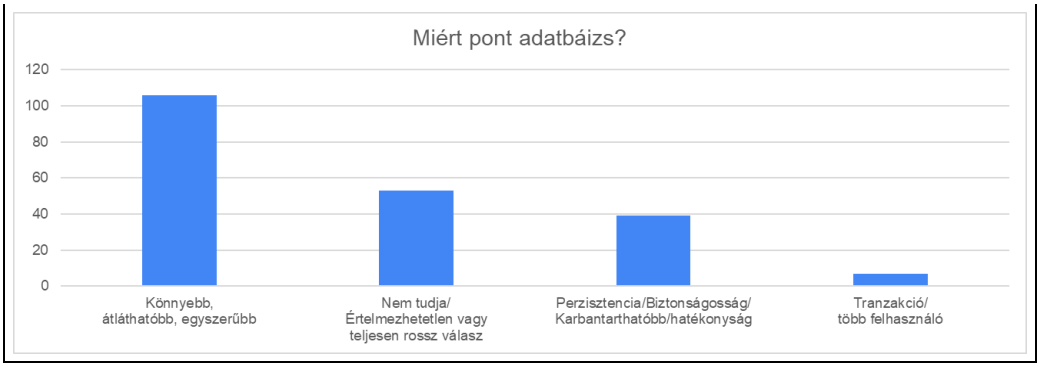
megtenni, mert ezen kompetencia szintjének mérésére nem létezik alkalmas, kvantitatív módszer. Miután elkészülünk egy ilyen rendszerrel, ezeket az igen fontos méréseket el kívánjuk majd végezni.

Az előzőeken felül mélyinterjúkat készítettünk szenior, tapasztalt programozókkal és szoftvertervezőkkel annak érdekében, hogy feltérképezzük, pontosan mire van szüksége egy szoftverfejlesztőnek a munkája során, vagyis az egyetemeknek mire kell felkészíteniük a hallgatókat a programtervező informatikus képzésen.

3. A kompetencia szerepe

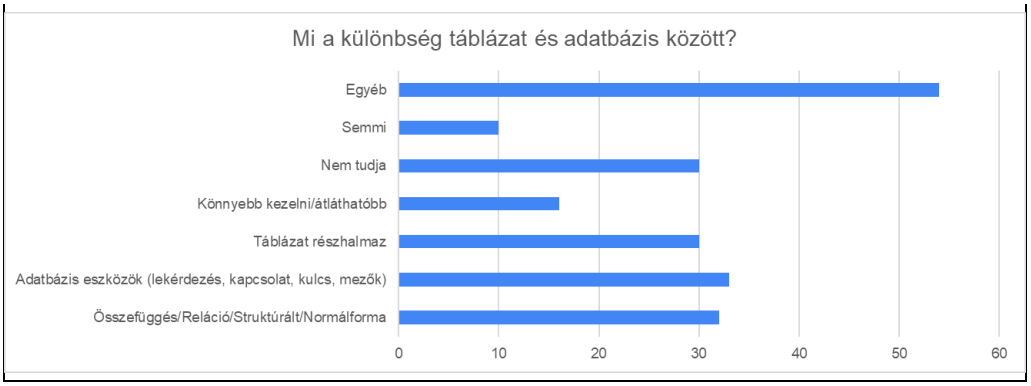
Jelenleg az informatika oktatása leginkább az alkalmazói rendszerekről, az algoritmikus gondolkodásról, illetve a digitális kultúra tantárgy bevezetése óta különböző digitális kompetenciák kialakításáról szól (mint az online kommunikáció, publikálás stb.) [6, 7]. Általánosságban elmondható, hogy a programozás oktatása és az algoritmikus képesség gyakran egyenlőnek van tekintve, vagyis a programozás oktatása az algoritmizálás kompetenciájának fejlesztésére szolgál, vagy azt tekinti fő feladatának. Jól mutatja ezt, hogy a kerettantervben is „Algoritmizálás, formális programozási nyelv használata" szerepel [7]. Egyre több esetben a módszertan a problémamegoldásra helyezi a fő fókuszt [8], azaz a programozást konkrét problémák megoldásán keresztül értelmezi. A fő probléma az, hogy a komponensek működése, a részproblémák megoldása általában nem okozza egy rendszer működését, hiszen sokszor a rendszer struktúrája és nem pedig az egyedi összetevők állapotai határozzák meg a működést [1]. Hiába tanul meg a diák vagy hallgató egyedi komponenseket, ha azokat nem tudja különböző informatikai kontextusokban értelmezni. A tanulás folyamata során kiemelkedő jelentősége van, hogy az új elem illeszkedjen a meglévő tudásbázishoz, beilleszkedjen az informatikáról kialakított tudásgráfba. A kontextus és a rendszerek elhagyásával a hallgatónak nincsen esélye erre, mert ekkor az új tudáselem egy izolált pont lesz a gráfban. A leggyakrabban ekkor a hallgató arra panaszkodik, hogy nem tudja mire jó ez az új tudás, hol és hogyan lehet alkalmazni vagy egyáltalán mi az értelme. Ha azonban már hozzá tudja kötni valamihez, akkor a motivációja is emelkedik, hiszen az új tudáselem egy korábbi problémára vagy egyszerűen egy lyukas foltra kínál megoldást.

Így fordulhat elő például az, hogy bár a diákok tanulják az Access-t középiskolában, mégsem értik mi a különbség egy relációs adatbázis és egy táblázat között [9]. Elsőéves hallgatók a 2. ábrán látható válaszokat adták arra a kérdésre, hogy általában miért adatbázisban tároljuk az adatokat és nem fájlokban. Az ábrán jól látszik, hogy a hallgatók több mint háromnegyedének gyakorlatilag fogalma sincs a helyes válaszról.



2. ábra: A „miért adatbázist használunk táblázat helyett" kérdésre adott válaszok arányai [9]

A 3. ábrán jól látható, hogy az Access oktatása középiskolában olyannyira sikeresen épít a táblázatkezelésre, hogy a hallgatók kétharmadának nem igazán sikerül értően megkülönböztetni a kettőt.



3 ábra: A „mi a különbség a táblázat és adatbázis között" kérdésre adott válaszok arányai [9]

Ebben a helyzetben a diákok valószínűleg meg tudnak oldani Access feladatokat, képesek felismerni egy adatbázist, sőt, összetett feladatokat is meg tudnak oldani, mégsem illesztették bele az informatikáról alkotott nagy képbe. A következő fejezetekben konkrét tananyagelemekkel kapcsolatban kívánjuk bemutatni ezt a problémát, valamint példákat kívánunk adni arra, hogyan lehet a rendszerszemléletet átadni.

4. Módszertani lehetőségek és példák a kompetencia fejlesztésére

4.1 Szövegszerkesztés – Word

A rendszerszintű gondolkodás fejlesztése a Word oktatása során azt jelenti, hogy nem az az egyetlen cél, hogy szövegszerkesztési problémákat oldjunk meg (pl.: hogyan lehet megoldani, hogy egyik oldal fektetett legyen, míg a másik állított, hogyan lehet önéletrajzot készíteni, meglévő sablonokat felhasználni), sem pedig, hogy növeljük az alkalmazói képességet, hanem hogy tanuljunk valami általánosat a szövegekről, az információ ábrázolásáról, formázásáról. Vegyék észre a diákok a kapcsolatot a Word és egy csevegő alkalmazás beviteli mezője, vagy egy fejlesztői környezet (IDE) között. A fontos fogalmak ezzel a megközelítéssel tehát sokkal inkább a gyűjtőfogalmak, mint az alakzatok, karakterszintű formázások, bekezdésszintű formázások, szakaszszintű formázások stb. A lényeg tehát, hogy amikor a diák elkészít például egy önéletrajzot, akkor ne a Word sajátosságiból induljon ki, hanem általános elképzelésekből, melyekhez egy, de nem az egyetlen eszköz a Word.

4.1.1 Óravázlat

A következőkben egy konkrét órapéldát kívánunk mutatni arra, hogy hogyan lehet a rendszerszintű gondolkodás kompetenciáját fejleszteni a szövegszerkesztésen keresztül.

Az óra a 8. évfolyam diákjai számára készült, mint a szövegszerkesztést bevezető óra. Ehhez mérten épít a következő alapismeretekre:

- Alapvető karakterformázások.
- Alapvető bekezdésformázások.
- Képek beágyazása a dokumentumba
- Bekezdés/oldal szegélyezés

Az 1. táblázat annak az órának a tervét mutatja, melyet élesben is megvalósítottunk, s mely véleményünk szerint a rendszerszintű gondolkodás fejlesztését tűzi ki célul.

Idő	Tananyag	A tevékenység célja	Munkaformák és módszerek	Megjegyzés
0–2 perc	A terem elfoglalása	A diákok bejönnek, elfoglalják a helyüket, felkészülnek az órára.	-	
2-5 perc	Köszönés, névsor, bevezetés	Bemutatkozás, jelenlévők ellenőrzése. Az előttünk álló téma megnevezése, rövid bemutatása	Oktató által irányított	
5-13 perc	Rövid bemutató	A téma felvezetése, gondolatébresztés. Mi mindenre használjuk a szövegszerkesztést?	Frontális	A mellékelt PowerPoint bemutató alapján.
13-15 perc	Feladat kiadása	A szöveg szerkezetének megértése	Frontális	A feladat: Az ismert műveleteket próbáljuk meg valamilyen szempont alapján kategorizálni. Mik azok, amik egy csoportba tartoznak?
15-20 perc	Feladatmegoldás	A téma önálló feldolgozása	Páros munka	
20-26 perc	A diákok válaszainak feldolgozása	Az alapvető fogalmak bemutatása, a szerkezet megismerése.	Ellenőrzés. Használandó eszköz: Word-SmartArt	Amennyiben a diákok által felállított kategóriák nem felelnek meg a valóságnak, a tanár finoman korrigáljon. Cél, hogy a szekció végére elkészüljön egy olyan struktúra, mely jól bemutatja az alapvető fogalmakat.
26-27 perc	Az ábra lerajzolása a füzetbe	Az elmélet rögzítése, hogy később visszakereshető legyen	Egyéni	A táblázat után látható a várható ábra
27-29 perc	Az önálló feladat kiadása, közben gépek bekapcsolása	Felkészülés az önálló feladatmegoldásra	Frontális	A feladat: Mindenki készítsen el egy dokumentumot a mellékelt nyers szöveg alapján[1]. Formázza úgy, ahogyan az szerinte a lehető legigényesebb, olvashatóbb. Az elkészült művekből a legigényesebbeket, a következő óra elején elemezni fogjuk.

[1] Maga a konkrét szöveg nem annyira lényeges, bár nem árt, ha témakör érdekli a diákokat.

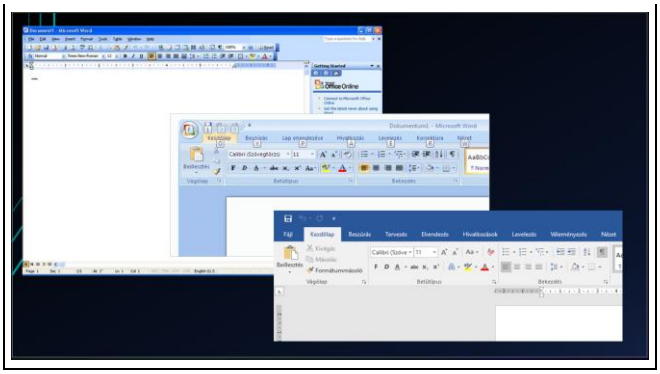
29-44 perc	Önálló feladat megoldása	Az eddig tanultak gyakorlati felhasználása, azaz egy olyan szemléletre való kitekintés, ahol a szövegszerkesztést nem instrukciók alapján csináljuk, hanem felhasználjuk a megszerzett tudást.	A diákok egyénileg dolgoznak. Az oktató járkál, ellenőrzi a munkavégzés folyamatát és segít, ha kell.	Az elkészített munkáknak a következő órán lesz nagy szerepe, hiszen ezekből néhányat közösen fogunk elemezni.

1. táblázat: Szövegszerkesztéses óravázlat a rendszerszintű gondolkodás fejlesztésére

A bevezető órának a lényege, hogy a meglévő alapvető információkra építve a program általános felhasználhatóságára és céljára helyezze a hangsúlyt. A diákok fő feladata a funkciók csoportosítása, s nem pedig pl. konkrét funkciók megtanulása különböző példákon, problémákon keresztül. Az oktató feladata a kontextus bemutatása és az összegző munka terelgetése és kiegészítése. Ehhez mérten az első óra legelején lévő igen rövid bevezetőhöz készült bemutató is összehasonlító, összegző diákat tartalmaz általánosságban a szövegszerkesztésről. A cikkben két példát kívánunk bemutatni a hatból.

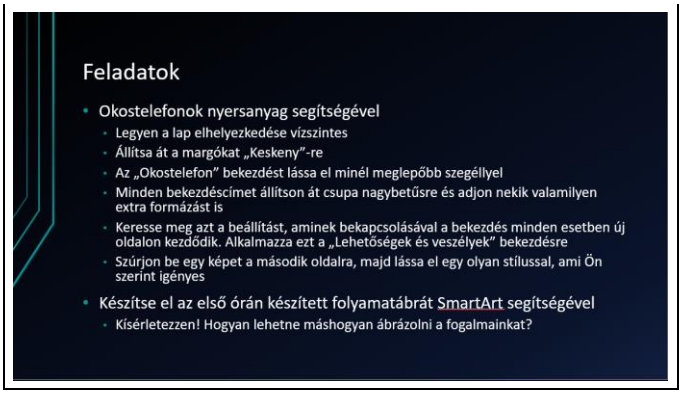
4. ábra: Az a dia a szövegszerkesztés órájának bevezető bemutatójából, mely a szövegek általános felhasználását mutatja be.

A 4. ábrán látható diának az a lényege, hogy rámutasson arra, hogy a szövegszerkesztés messze túlmutat a Wordon, de még a standard szöveges dokumentumokon is (pl.: videókban, logókban, forráskódban).



5. ábra: A szövegszerkesztő felületének mellékességét szemléltető dia

A 5. ábra a Word felhasználói felületének evolúcióját mutatja be, mellyel az oktató érzékeltetheti mennyire nem az a fontos, hogy a diákok azt tanulják meg, mi milyen beállítás hol van, hogyan hívják.



6. ábra: A szövegszerkesztő óra önálló feladatai igen vegyesek

Az óratervezetben, így a 6. ábrán az is megjelenik, hogy a szövegszerkesztésnél (rendszerszemlélettel) nincsen különbség a karakterszintű formázások és a szakaszszintű formázások között, hiszen ezek csupán 1-1 lehetőségek egy konkrét alkalmazásban. Ez a gyakorlatban azt jelenti, hogy ezeket a lehetőségeket egyáltalán nem szükséges külön-külön oktatni. Az óra tartása közben is azt tapasztaltuk, hogy a diákoknak egyáltalán nem okozott gondot a 6. ábrán látható feladatok vegyessége, ahogyan az sem, hogy eddig ismeretlen eszközöket kellett használniuk.

4.2 Táblázatkezelés – Excel

A rendszerszintű gondolkodás fejlesztése az Excel tanítása során azt jelenti, hogy valami általános szemléletet igyekszünk kialakítani a diákokban az adatok rendszerezéséről, értelmezéséről, feldolgozásáról és ábrázolásáról. Ha például a diákok kapnak egy mintafájlt válogatott és rendszerezett adatokról, amit be kell olvasniuk a táblázatkezelőbe (ahogyan az például az érettségin is lenni szokott [10], s ehhez mérten a gyakorló órákon is), akkor ezt a szemléletet nem kapják meg, hiszen a rendszerszintű gondolkodás szerinti legfontosabb feladatot nem ők végzik el. Az adatok összegyűjtése és rendszerezése a végeredmény szempontjából úgy, hogy a feldolgozáshoz szükséges függvényeket, lekérdezéseket hatékonyan meg lehessen csinálni maga a rendszerszintű gondolkodás (amit például egy tanár csinál egy dolgozat összeállításánál).

4.2.1 Óravázlat

A következő rendszerszintű gondolkodás fejlesztésére alkalmas példaóra kiválóan megvilágítja az imént említett problémát. Az óra a 9. évfolyam diákjai számára készült, mint a táblázatkezelést bevezető óra. Ehhez mérten épít a következő alapismeretekre:

- Alapvető formázások, cellaformázás
- Cellahivatkozások
- Alapvető, egyszerű függvények (pl.: Átlag, Darabteli, Min, Max)
- Egyszerű diagrammok

A 2. táblázat az óra vázlatát tartalmazza.

Idő	Tananyag	A tevékenység célja	Munkaformák és módszerek	Megjegyzés
0–2 perc	A terem elfoglalása.	A diákok bejönnek, elfoglalják a helyüket, felkészülnek az órára.	-	A gépet már most bekapcsolhatják
2-20 perc	Bevezetés	Bevezetés. A témakör szerepének áttekintése, ismétlés.	Oktató által irányított beszélgetés.	A következő kérdések alapján: - Mit tanulunk éppen? - Mi ennek a szerepe az informatikában? Miért fontos? - Miben más ez, mint egy normál táblázat? - Mik a táblázatkezelés sajátosságai? - Mik azok a függvények, elágazások?
20-30 perc	Egy témakör táblázatos absztrahálása.	A „táblázatos logika” elmélyítése, megközelítések összehasonlítása, esetleges félreértések tisztázása.	Páros munka.	5 percet kapnak a diákok, hogy a következő témakörben a szerintük leglogikusabb táblázatot összerakják mintaadatokkal: Iskola, osztály, diákok
30-40 perc	Milyen funkciókat várunk el.	Az elkövetkezendő függvények megtanulására ő maguknak legyen igénye.	Oktató által irányított beszélgetés.	Egy konkrét adathalmazból kiindulva/általánosítva fogalmazzuk meg, milyen (további)függvényeket várunk el a programtól. A konkrét adathalmaz mellékelve.
40-45 perc	Órazárás, gépek kikapcsolása, összefoglalás.	Az átbeszéltek alapján összefoglalás, az említett fogalmak elmélyítése.	Oktató által irányított beszélgetés.	

2. táblázat: Táblázatkezelés óravázlat a rendszerszintű gondolkodás fejlesztésére

Hasonlóan a szövegszerkesztős példához, ez az óra is 3 nagy részre osztható:

- Az eddig megtanult alapok rendszerezése, kontextusba helyezése
- Egy rendszer jellegű probléma megoldása
- Reflexió, melynek a végére a diákok a DIKUW modell szerinti „megértés" szintű kérdésekre válaszolnak

A táblázatkezelés esetében a rendszerprobléma az adatok „jól" szervezése volt. A 3. táblázat egy, a diákok által készített táblázatokból készült (a neveket kicseréltük, hiszen a diákok a sajátjaikkal töltötték fel). Ez a strukturális alapelv az 5 kiscsoportból két helyen megjelent.

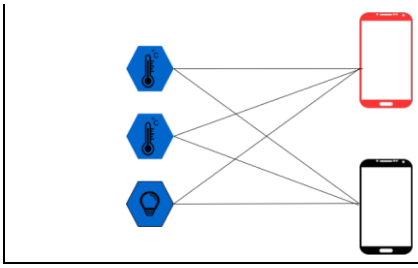
Névsor	**9.a**	**9.b**	**10.a**	**10.b**	**11.a**	**11.b**	**12.a**	**12.b**	**Összes**	**Átlag**
	Horváth Anna	Gábor Anna	Kiss Kincső	Tóth Ádám	Nagy Bence	Belső Donát	Orbán Etel	Pintér Emese		
	Kovács Mária	Pécsi Gábor	Varga József			Farkas Szonja	Vass Napsugár	Bakos Barna		
	Napos Ádám		Szabó Emese							
	Nagy Míra									
Létszám	4	2	3	1	1	2	2	2	20	2.125

3. táblázat: A diákok által készített táblázat

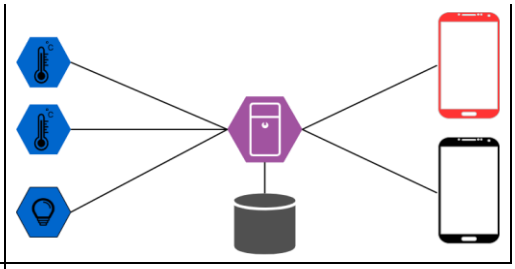
A táblázat felosztása nem teljesen rossz, hiszen csupán annyi instrukciót kaptak, hogy „Iskola, osztály, diákok" (bár élőben említésre került, hogy az nincsen rögzítve, pontosan milyen adatokat akarunk tárolni). A táblázat képes befogadni névsor adatokat osztályonként, de gyakorlatilag semmi másra nem képes. A legnagyobb probléma a táblázattal az, hogy nem bővíthető, további diák-, osztály-, iskolaszintű tulajdonságokkal nem kiegészíthető. Persze mivel nem relációs adatbázisban vagyunk, nem feltétlenül szükséges ilyen megkötéseket tennünk. A lényeg itt sokkal inkább az, hogy a diákok számára nem magától értetődő, hogy a feladatokban miért szinte mindig a normálformált adatstruktúra látható. Mivel az óra közben természetesen vetődött fel ez a helyzet, ráadásul „jó" megoldás is született, ezért értelmesen tudtuk összehasonlítani, sőt, a diákok tudták levonni a következtetés mikor melyik jó.

4.3 Informatikai rendszerek

Egy korábbi cikkünkben arra mutattunk példát, hogy komplex informatikai- vagy szoftverrendszerek eljátszhatók a diákokkal tantermi környezetben gyakorlatilag eszköz és előismeret nélkül [11]. Az ilyen játékok lényege, hogy a diákok valós tapasztalatot szerezzenek, játékosan fedezzenek fel megoldásokat, szembesüljenek buktatókkal [12]. A játék során a diákoknak 1-1 algoritmus, alkalmazás hardver vagy felhasználó szerepét kell felvenniük és közreműködve kell szimulálniuk egy informatikai rendszer működését. Ehhez csupán annyi szükséges, hogy az oktató kitalálja a csoport számára legérdekesebb, de az eszközökhöz és helyhez mérten leginkább megvalósítható informatikai rendszert. A fentebb említett cikkben mi egy okosotthon projektet választottunk, melynek lehetséges architektúrái a 7. és 8. ábrán láthatók.

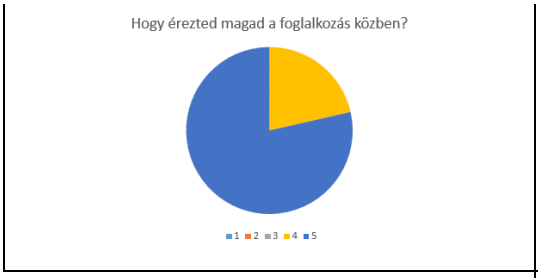


7. ábra: Okosotthon architektúra szerver nélküli változata



8. ábra: Okosotthon architektúra szerveres változata

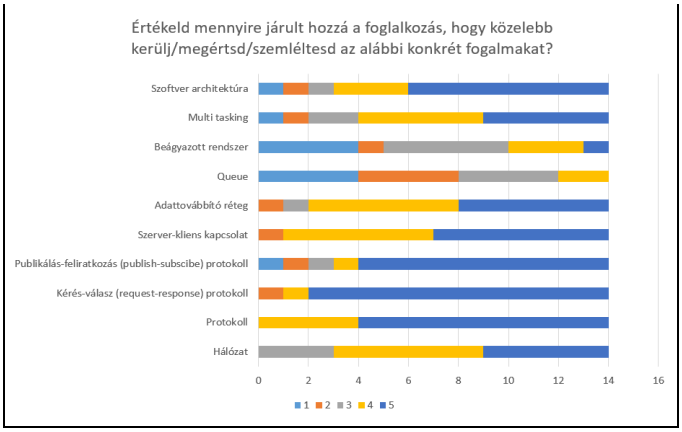
A játékot kipróbáltuk első féléves informatikatanár szakos hallgatók között. A tapasztalatok nagyon pozitívak voltak. A hallgatóknak tetszett a játék, mely a 9. és 10. ábrán látható visszajelzésekből készült diagrammokon is látható. Aktívan részt vettek benne és tényleges szimulációját tudták adni egy valós informatikai rendszer működésének. Nekünk, mint oktatóknak csupán a terelgetés volt a szerepünk valamint az informatikai analógiák hangsúlyozása.



9. ábra: A „hogy érezted magad” kérdésre adott válaszok arányai

10. ábra: A hallgatók által adott értékelések a foglalkozásra

A 11. ábrán látható a visszajelzésekről készített diagram. Ez alapján elmondható, hogy ezen konkrét informatikai rendszer által érintett legfontosabb fogalmakhoz tartozó megértést sikerült elmélyíteni a hallgatókban



11. ábra: A hallgatók által adott értékelés az egyes informatikai fogalmak

szemléletének sikerességéről

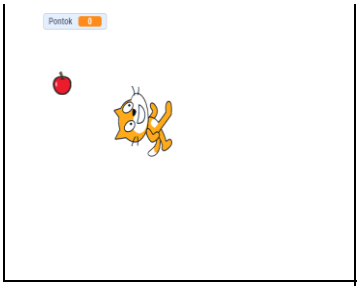
4.4 Programozás

A programozás esetében a rendszerszintű gondolkodás fejlesztése nagyon tág határok között mozog, hiszen a programok, szoftverrendszerek nagyon sokféle rendszert alkothatnak és gyorsan bonyolódnak. Oktatási környezetben bizonyos rendszerek szimulációja igen nehéz. Például azt bemutatni, hogy a shell script programozás hogyan illeszkedik az informatika nagy egész rendszerébe, pontosan milyen helyzetekben hasznos vagy elengedhetetlen nagyon nehéz. Tantermi körülmények között sem a konténerizált világ, sem a szoftverrendszerek funkcionális, automatikus tesztelése, sem pedig az operációs rendszerek működtetése nem igazán megvalósítható. Teljesen más rendszerről van szó, ha a beágyazott vagy alacsonyszintű programozásról, a webfejlesztésről, vagy magasszintű nyelvekről van szó. A rendszerszintű gondolkodás fejlesztése szoftverrendszereken, fejlesztésen keresztül tehát nagyban függ az oktató céljaitól, a csoport összetételétől, érdeklődési körétől és életkori sajátosságaitól. Ez persze inkább „feature" mint sem „bug", hiszen az oktató által kellően skálázható, modularizálható, spirális tananyag készíthető.

Mivel a szoftverrendszerek gyorsan bonyolódnak, ráadásul a technológiai „stack" (verem) is hamar nagyra duzzadhat, az oktatónak alaposan meg kell válogatnia, milyen rendszert visz be a diákok számára. Ahogyan azt a korábbi fejezetekben is említettük, a rendszerszintű gondolkodás fejlesztése nem azt jelenti, hogy bármilyen bonyolultságú, nehézségű rendszer bevihető a diákok számára. A nehézség pont abban rejlik, hogy a feladat maga csak annyira legyen bonyolult, amit a diákok javarészt saját erőből, minimális oktatói segítséggel/terelgetéssel képesek részeire bontani, a meglévő elemeket felismerni, vagy a rendszer működését módosítani, kiegészíteni. A szoftverrendszerekkel tehát az a probléma, hogy nehéz olyan kézzel fogható, könnyen érthető példát találni, mely könnyen skálázható, alakítható az oktató aktuális igényeinek megfelelően.

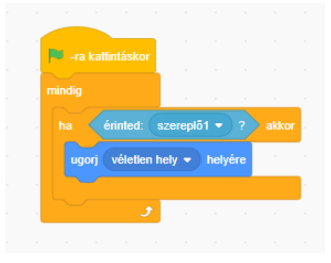
4.4.1 Scratch példa

Meg kell jegyezni azonban, hogy nem csak szoftverrendszerekkel lehet a rendszerszintű gondolkodást fejleszteni. A Scratch[2] környezetben is lehet készíteni olyan játékokat, mely rendszerproblémákat hoznak elő, pedig maga a felület nehezen nevezhető komplex rendszernek. Ilyenek azok a játékok, ahol több szereplő van interakcióban egymással, reagálnak valahogyan egymás viselkedésére, hiszen könnyen „race condition", vagyis versenyhelyzet állhat fenn, ami az egyik leggyakoribb rendszerprobléma a szoftverfejlesztés világában. Ezt könnyen demonstrálja a 12 - 14.ábrán látható példa, ahol akkor kapunk pontot, amikor az egyik szereplőnk elkapja a másikat, ami az elkapásra úgy reagál, hogy egy véletlen helyen megjelenik.



12. ábra: A Scratch játék kinézete.

13. ábra: A macska kódja (irányító



14. ábra: Az alma kódja

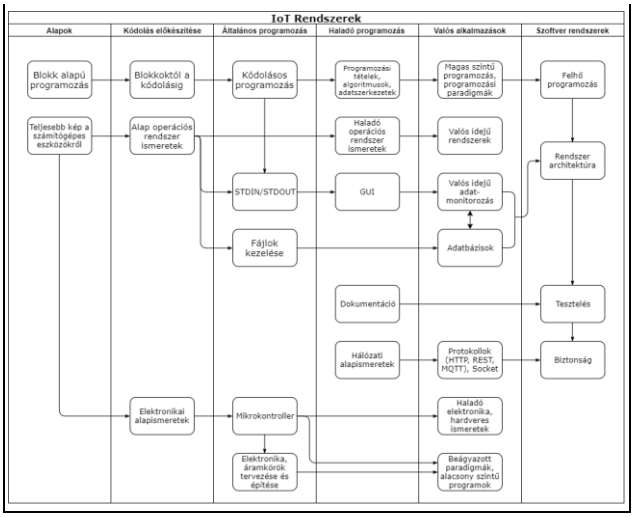
[2] https://scratch.mit.edu

gombokat kezelő kódjain kívüli)

Ez a megoldás problémás, hiszen nem determinisztikus a viselkedés, habár külön-külön a komponensek algoritmikusan és szemantikailag is helyesek. A kimenet attól függ, hogy egymáshoz képest a két szereplő mikor ér el a feltételben szereplő utasítás végrehajtására. Ha a macska kódja nagyon gyors, akkor akár több pontot is kaphatunk, mire az alma másik helyre ugrik, de ha az alma kódja nagyon gyors, akkor hamarabb elugrik, minthogy a macska érzékelte volna az érintkezést, vagyis egyetlen pontot sem kapunk. Végső soron tehát az ilyen helyzetek eredménye attól függ, hogy a párhuzamos folyamatok mikor és mennyi processzoridőt kapnak, vagyis az ütemezőtől és a rendszer aktuális állapotától

4.4.2 Szoftverrendszerek

Ahogyan korábbi cikkeinkben is javasoltuk, a beágyazott és valós idejű, vagyis „IoT" rendszerek kiválóan alkalmasak a kompetencia fejlesztésére [13]. A 15. ábra jól bemutatja mennyire könnyen alakíthatóak ezek a példák.



15. ábra: IoT rendszerekkel oktatható informatikai modulok.

A 15. ábra azt mutatja be, hogy az IoT rendszereken keresztül folytatott oktatás milyen komponenseket érinthet, mikre helyezheti az oktató a hangsúlyt.

Mi az alábbi példát próbáltuk ki mesterképzéses hallgatókkal az Eötvös Lóránd Tudományegyetemen. A csoport létszáma 8 fő volt. Okosotthon projekteket szerettünk volna építeni a hallgatókkal Raspberry Pi eszközökkel. Itt a hangsúly a valós idejű rendszereken volt.

Óra	Leírás
1.	**IoT bevezetés** Témakör bevezetése, elméleti háttér áttekintése. Egy minta program bemutatása, mely egy valós életbeli IoT projektet demonstrál **Hardverelemek összeépítése** A hardverek összeépítése, összekötése, élőben kipróbálása mintakódokkal. • Hogyan kommunikáljunk Sense Hat modullal (szenzor adatok begyűjtése, LED mátrixra kirajzolás) • Hogyan vezéreljük a kamera modult? tkinter Python GUI csomag alap funkcióinak kipróbálása
2.	**MQTT protokoll** MQTT broker installálása, konfigurálása és helyi hálózati kommunikációra használása. A publish-subscribe filozófia megismerése, események és eljárások használata az előző órai példák felhasználásával.
3.	**Felhő szolgáltatások** Firebase nem-relációs valós idejű adatbázis konfigurálása és integrálása egy okos otthon szimulációs projektbe.
4-6.	**IoT Lab** Önálló, valóság közeli rendszerek tervezése és implementálása két fős csapatokban.

4. táblázat: Mesterképzéses tananyag: IoT rendszerek oktatása

A 4. táblázat bemutatja a 6 órás blokk általános tervezetét, melyből jól látszik, hogy a modul egy lehetséges végleges projekt demonstrációjával kezdődik. A következő órákon pedig 1-1 önmagában értéket képviselő komponenssel foglalkozunk, melyeket a hallgatók rögtön tudják azonosítani, hogy hova tartoznak. Az utolsó órák pedig biztosítják, hogy biztosan képesek legyenek a megtanultakat együtt alkalmazni.

4.5 Egyéb módszertani fogások

Idáig konkrét példákon keresztül próbáltuk bemutatni a rendszerszintű gondolkodás fejlesztésének lehetőségeit. Vannak azonban általános lehetőségeink is, amit szinte mindenhol lehet alkalmazni. Erre már készültek tanulmányok, amik a gondolattérkép, visszajelző hurkok, ok-okozati térképek használatainak hasznosságát hangsúlyozzák ki [14]. A lényeg tehát, hogy gyakorlatilag minden tananyagelemről elmondható, hogy beilleszthető valamiféle kontextusba, hozzáfűzhető egy más meglévő tudáshoz vagy egymáshoz. Ha ezt a munkát a diákok végzik el az oktató segítségével, akkor a tudás megértéssé válik, hiszen annak értelmezését az ábrák elkészítésével a diákok elvégzik.

5. Visszajelzés az iparból

Készítettünk mélyinterjút 7 tapasztalt programfejlesztővel. A beszélgetések előtt direkt nem említettük, hogy a rendszerszintű gondolkodás lesz a téma, hogy ne befolyásoljuk őket. Az interjú a következő pontok mentén zajlott:

- Milyen olyan komplex/emlékezetes problémák jutnak eszedbe, amelyeket meg kellett oldanod, mint fejlesztő?
- Mire volt szükséges, hogy megold ezeket a problémákat?
  - Kompetencia
  - Szakmai tudás
  - Egyetemi tantárgy
  - Egyéb
- Fontosnak tartod az egyetem elvégzését?
- Hogyan lehetne az egyetemi képzést jobbá tenni?

A beszélgetések teljes kifejtése nem fér ezen cikk kereteibe (és szeretnénk is még bővíteni az interjúalanyok számát), azonban ami számunka jelen esetben releváns, hogy a hétből minden alany említette a rendszerben való gondolkodás fontosságát. Volt olyan fejlesztő, aki az algoritmikus gondolkodást nem is említette (habár annak fontossága megkérdőjelezhetetlen), tehát elmondható, hogy a rendszerszintű gondolkodás a programozói életben a legfontosabb kompetenciák közé tartozik. A megkérdezettek hangsúlyozták az egyetem fontosságát, visszatérő kritika viszont az volt, hogy csak később derült ki számukra az, hogy 1-1 tantárgy milyen fontos is valójában. Ahogyan ezt korábban is említettük a kontextus, a rendszerek elhagyására, nem hatékony bemutatására utalhat.

6. Összefoglalás

A cikkben megpróbáltuk a szövegszerkesztésen, táblázatkezelésen és programozásoktatáson, valamint általánosan használható eszközökön keresztül bemutatni, milyen lehetőségeink vannak a különböző korosztályok esetében a rendszerszintű gondolkodás fejlesztésére. Hogy az általános elveket alátámasszuk, konkrét tananyagpéldákat készítettünk és próbáltunk ki élesben a nyolcadik, kilencedik évfolyam diákjaival, valamint az egyetemi mesterképzés hallgatóival. A teszteléseink a megvalósíthatóságot és a szükségességet bizonyították. Annak igazolása, hogy a módszertan tényleg hatékony még hátravan, hiszen nem áll rendelkezésre alkalmas mérési módszer. Az azonban kijelenthető, hogy a rendszerszintű gondolkodás kompetenciáját lehet fejleszteni általános iskolától az egyetemig, az informatikán belül minden témakörön keresztül.

A rendszerszintű gondolkodás fejlesztését előtérbe helyező tananyagoknak szükségük van némi alapvető előismeretre, amiből lehet általánosítani. A nagy előnyük viszont az, hogy amennyiben a diákoknak sikerül a rendszerszemléletet átadni, sokkal sikeresebben fognak új funkcionalitásokat, addig nem ismert eszközöket felhasználni (ahogyan például a Word feladatokat sikeresen megoldották azok oktatása nélkül), az új anyagot régebbiekkel összekötni, így egy teljesebb képet tudnak alkotni az informatika tudomány egészéről, a konkrét eszköz felhasználhatóságáról, előnyeiről és hátrányairól.

Alapvető előismereteken túl szükség van egy problémára. A probléma vagy egy meglévő rendszerben jelentkező rendszerprobléma, vagy egy elkészítendő komplex megoldás. A komplex alapvetően egymásra ható, egymással kommunikáló, többkomponensű rendszert jelent (ahogy a példákban említettük, ez lehet akár egy Scatch játék is), mely dinamikusan választható, alakítható az aktuális korosztálynak és célnak megfelelően.

Programozók, programtervezők visszajelzéseiből, valamint korábbi kutatásokból tudjuk, hogy a rendszerszintű gondolkodás a legfontosabb kompetenciák közé tartozik, melynek fejlesztése jelenleg nem történik meg kellően hatékonyan. Mivel a kompetencia meglévő tananyagokkal is fejleszthető, pusztán módszertani fogásokkal (vagyis nem feltétlenül szükséges külön modult szánnunk rá), ezért

érdemes a különböző témaköröknél meggondolni, hogy az adott elem alkalmas-e, azon keresztül hatékonyan tudjuk-e fejleszteni a kompetenciát.

Köszönetnyilvánítás

A KULTURÁLIS ÉS INNOVÁCIÓS MINISZTÉRIUM ÚNKP-22-3 KÓDSZÁMÚ ÚJ NEMZETI KIVÁLÓSÁG PROGRAMJÁNAK A NEMZETI KUTATÁSI, FEJLESZTÉSI ÉS INNOVÁCIÓS ALAPBÓL FINANSZÍROZOTT SZAKMAI TÁMOGATÁSÁVAL KÉSZÜLT.

Irodalom

1. L. Zsakó és P. Szlávi: *Informatikai kompetenciák: Algoritmikus gondolkodás* In InfoDidact 2010, Szombathely (2010).
2. R. D. Arnold and J. P. Wade: *A Definition of Systems Thinking: A Systems Approach,* Procedia Computer Science, (2015) 669-678
3. S. Korom: *Architektúrális gondolkodás fejlesztése valós idejű rendszerekkel* In. InfoDidact 2018, Zamárdi (2018)
4. S. Korom., Z. Illés*: Systemic Thinking in Programming Education*. In Recent Innovations in Computing. Lecture Notes in Electrical Engineering, vol 855. Springer, Singapore (2022)
5. B. Castellani, F. Hafferty, C. Schimpf:. *E-Social Science from a Systems Perspective: Applying the SACS Toolkit*. Special Issue: Proceedings from the Urbino-Conference. 7, (2009) 89-106.
6. *Kerettanterv az általános iskola 5–8. évfolyama számára* https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_F.docx (utoljára megtekintve: 2022.11.19)
7. *Kerettanterv a gimnáziumok 9–12. évfolyama számára* https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_K.docx (utoljára megtekintve: 2022.11.19)
8. M. Newman: *A Pilot Systematic Review and Meta-Analysis on the Effectiveness of Problem-Based Learning* (2003)
9. S. Korom and Z. Illés: *Competence Improvements with IoT systems*, Central-European Journal of New Technologies in Research, Education and Practice, vol. 3, no. 1, (2020)
10. *Központi írásbeli feladatsorok, javítási-értékelési útmutatók*, Oktatási hivatal https://www.oktatas.hu/kozneveles/erettsegi/feladatsorok (utoljára megtekintve: 2022.11.19)
11. S. Korom és Z. Illés: *Offline situational game for teaching IT systems*, Central-European Journal of New Technologies in Research, Education and Practice, vol. 4, no. 1, (2022) 25-37
12. T. C. Bell, I. H. Witten, M. Fellows: *Computer Science Unplugged . . . off-line activities and games for all ages* (1998)
13. S. Korom: *IOT Systems in Education*, In Marek, Smid; René, Bílik; Veronika, Stoffová (eds) XXXII. Didmattech 2019, Trnava, Szlovákia : Trnava University in Trnava Faculty of Education, (2019)
14. Leyla Acaroglu: *Tools for Systems Thinkers: Systems Mapping,* Online: https://medium.com/disruptive-design/tools-for-systems-thinkers-systems-mapping-2db5cf30ab3a (utoljára megtekintve: 2022.11.19)