# Szalayné Tahy ZsuzsannaBemutató portfólió

Budapest, 2019. november 13.

## Az Informatika (programozás) oktatásának módszertana

Az első mesterpedagógusi ciklusom jelentős részében a szakmai munkát a doktori iskola és kutatás jelentette. A választott téma informatikaoktatás, amelyben az informatika és az oktatás nagyjából egyenlő súllyal szerepel. A doktori iskola elvégzése mellett óralátogatásokkal, tananyagelemzésekkel, tananyagfejlesztéssel, egyetemi órákon demonstrálással és oktatással is bővítettem ismereteimet, gyűjtöttem tapasztalatokat. A tanulmányok 3,5 éve után elkezdtem megírni a disszertációt. Ez a legfeljebb 120 oldalas kutatási beszámoló pillanatnyilag 237 oldal, így nem mondhatom, hogy szinte kész, mert bőség zavárban szenvedek, ami a tézisek megfogalmazására is igaz. A disszertációm előzetesének a tartalomjegyzékét mellékeltem. Ebből jól látszik a kutatásom horizontális mérete. A nagy látószögből következik, hogy vertikálisan nem tudtam túlságosan elmerülni a részletekben. Egy-egy témában a legjellemzőbb gyakorlati tapasztalatokat elemeztem, első sorban a gyakorlati felhasználhatóságot szem előtt tartva. Tézisek megfogalmazása és igazolása szempontjából ez a felépítés nem túl szerencsés, de úgy gondolom, hogy pillanatnyilag az informatikaoktatás számára hasznosabb a kutatandó témák felvetése.

A kutatást a tantárgy és a tantárgyban tanított tudomány definiálásával kezdtem. A magyar közoktatásban az informatikatanárokat az ELTE-IK képezi, amely a matematikus, programtervező matematikus tudományágakból átalakuló programtervező informatikusokat is képezi. Emiatt az informatikának a matematikával való kapcsolata domináns az oktatásban. Az első kutatási eredményeim közé tartozik annak bemutatása, hogy az informatika nem matematika. Ebből következik, hogy az informatikaoktatást nem lehet matematikaórán jól végezni. A kutatásom további részében azt is bizonyítottam, hogy az informatika nem fizika (még a mérnökinformatika sem). Tudománytörténeti vonatkozásokat és új értelmezéseket is figyelembe véve definiáltam az Informatika tudományt, aminek oktatását kutatom.

## Informatika, programozás

Az Informatika tudomány meghatározásában jelentős szerepe van az informatikában, illetve programozásban megnyilvánuló paradigmáknak. Az informatikai gondolkodás sokféle lehet, ebből két domináns irányzatot hasonlítottam össze: a programtervező (matematika alapú) és a mérnökinformatikus (fizika alapú) gondolkodás módszereket. A két irányzat egymással ellentétes állításokat fogalmaz meg, ugyanakkor jellemző, hogy egy adott alkalmazás esetén a kettő együtt lesz igazán hatékony.

A gondolkodásmódbéli különbségek az oktatás során is jól megfigyelhetők, rengeteg fejtörést okoznak az informatikát tanítóknak. Az egyik legtipikusabb probléma a break; használata: engedjük, helyettesítsük logikai kifejezéssel vagy követeljük meg a módszeres programozást. A problémát a keresés (eldöntés) algoritmusának tanítása kapcsán vizsgáltam gimnáziumi tanulók, illetve az ELTE-IK és BME-VIK hallgatói, oktatói körében.

A Lineáris Keresés Buktatói cikkemet mellékeltem. Ezt az InfoDidact konferencián mutattam be, emellett angol nyelvű változatát a DidMatTech konferencián adtuk elő, valamint az oktatók programozási gondolkodásmódjával kiegészített prezentációt a Szent István Gimnázium tudományos napján láthatták az érdeklődő diákok.

Az eredmény egy diagram, amelyen megmutatom a tapasztalt megoldási utakat és látszik, hol és miért okoz nehézséget a feladattípus.



Az informatika szakmódszertan több oktatási módszert is megnevez és elemez a tapasztalt hatékonyság és az alkalmazás gyakorisága alapján. Az egyes módszerek jellemző informatikusi gondolkodásmódot követnek, kiemelnek egyes kompetenciákat és háttérbe szorítanak (esetenként informális tanulásra bízva) másokat. Nem csak a break; kapcsán van vita. Egy feladat algoritmikus és OOP (adat) szemléletű megoldása teljesen más gondolkodási folyamat eredménye. A módszerek alkalmazásánál – én úgy látom – a legfontosabb szempont a tanár gondolkodásmódja, de a tanórákon rendszeresen gondot okoz (angliai iskolában is) a másképp gondolkodó diák. Ezért az oktatásmódszertant a többféle paradigma összehangolt oktatása felé kellene elvinni. Ez egyben a tömegesen egyéni oktatás módszereinek kutatását és alkalmazását is szükségessé teszi. Úgy vélem, a kutatásokat ebbe az irányba folytatva, kiindulásként a már nevesített tanulói típusokhoz kellene meghatározni a valószínűsíthetően eredményes módszereket, eszközöket. Így került az elemzésembe a napjainkban alkalmazott oktatási módszerek hatásaként nehézséggel küzdő dyslexiás, dysgraphiás, dyscalculiás, ADHD-s és ASD-s diákokkal kapcsolatos megfigyeléseim, illetve elemeztem a fiúk-lányok eltérő informatika tanulási motivációját, módszerét, ahogy azt is, hogy van-e olyan ember, aki nem képes megtanulni programozni, vagy ez az állítás a csak a számára rosszul megválasztott módszer következménye.

A gimnáziumokban tanító kollégáim programozásoktatási gyakorlata is ebbe az irányba mozdult. A vezetéssel programozásról áttérnek az egyéni munkára és gondolkodás tanítására. Ebből viszont az következik, hogy egy-egy órán akár 6 különböző dologgal foglalkoznak a diákok, amit nagyon nehéz követni. A tanári munka megkönnyítése szintén módszertani kutatást igényel, amiben a képernyő átvétele és a társértékelés, mentorálás helyes megszervezése megoldást jelenthet.

## Oktatásmódszertan - Learning Activity Unit

A kutatás során a tanulási folyamat jellemzésére egyre gyakoribb igény volt. Ezért alkottam egy tanulási modellt, amelyben egy tanulási egység a Learning Activity Unit, azaz LAU; a tudás megszerzését a LAU-modell írja le.

A LAU alapja több tudás és tanuláselméleti modell: a Bloom taxonómia, a Learning Pyramid, a SOLO taxonómia, a DIKW hierarchia, melyek adott szempontrendszerben állapotokat nevez meg. Ezek közös fogalmi vonatkozásait kombináltam az időben zajló átmenetekkel, figyelembe véve az Ebbinghaus-féle felejtési görbét.

**Ismétlő**

**Módosított**

**Kreatív**

**1. Ismeretszerzés**

**2. Kipróbálás**

**3. Alkalmazás**

**4. Szünet**

**5. Felhasználás**

**Aktív Moderált Passzív**

Elsőként a BME-VIK mérnökinformatikus képzés programozás alapjai 1 tantárgya lett a LAU-modellnek megfelelően átdolgozva. De mondhatnám fordítva is: a tantárgy tanulmányozása során alakult ki bennem a modell megalkotásának szükségessége, ezzel magyaráztam az oktatónak, hogy miért nem tudnak a hallgatók egyes dolgokat. Ezt követően, a fejlődő tudás önértékelésére „Haladási napló”-t is készítettünk. Valójában ez a kérdőívesített változata annak, amit az adott témákból a szubjektív értékelésnél figyelnék. Ezért az – oktatásmódszertanból általában képzetlen – oktatók is jól használhatnák önkéntes visszajelzési felületnek, de erre csak elvétve volt példa. A Haladási naplót a hallgatók szerették, a leggyengébbeknek is segített felmérni, hogy hogyan állnak, miben kell fejlődniük. A naplónak volt egyfajta gamifikációs hatása is azáltal, hogy folyamatosan jelentek meg újabb elemek és mindegyik elemben lehetett előre haladni az 0–5 ös szinteken.

Megpróbáltam a naplót átalakítani az ELTE-IK programozási alapismeretek tantárgyához, de ott nem volt értelme használni. A tárgy felépítésébe az egyre magasabb szintű ismeret megszerzése be van építve, a legfontosabb témákat (matematikai specifikáció, adat, algoritmus, struktogram, kódolás, C++ nyelv, tesztelés) LAU1 szinten már az első három héten veszik a hallgatók és attól kezdve párhuzamos a fejlesztés. Így a Haladási naplóból elveszett a hetente új téma varázsa, ráadásul az egyes szintekhez tartozó kérdések lecserélhetők a kiadott feladatok megoldására. Ami az ELTE-IK-n problémát jelent, az LAU-ok párhuzamos feldolgozása miatt az egyes LAU-okban bekövetkező hosszabb szünet. Például, ha kimarad a struktogram rajzolásának gyakorlása, ezzel halványul az abban használandó szintaktika ismerete.

A tapasztalatok mentén a tantervet, egyes tanfolyamok tematikáját is LAU szerint értékeltem. Jól látható a modell alapján, hogy mi hiányzik már a tervezésnél – jellemzően a gyakorlás és kreatív felhasználás – ahogy az is látható (fel lehet rá lelkileg készülni), hogy a heti 1 órának, az elmaradt tanóráknak milyen hatása lesz az egyes tananyagrészek tanulási állapotára, a tanítás hatékonyságára.

A programozás kezdőszintű oktatása 8–10 készség fejlesztését jelenti, azaz 8–10 LAU futna közel párhuzamosan (kódolás, logika, adattípusok kezelése, algoritmikus gondolkodás, tesztelés, hibakeresés, fájlkezelés, feladatok témájának az ismerete, absztrakciós készség) Az új és újabb elemek beépítése a tananyagba a régiek ritkább előfordulását eredményezik. Heti 1 óra esetén jellemzően a 8. órán derül ki, hogy a téma elején tanult LAU-okra már nem emlékszik a diák. Hasonló eredményre vezet az egy nap alatt elsajátított alapok (gyorstalpaló tanfolyam) hasznossága, ha utána 2 hónapig nem szükséges a tanult ismeret. A modell használata megkönnyíti a tanítás tervezését. Másképp: a modell leírja azt, ahogy a mindennapokban a diákok tudásával számolhatunk.

A tananyag LAU mentén történő elemzése a mérés-értékelés minősítésére is alkalmas, kimutatható például, hogy a tesztek az adott témából (LAU-ból) csak az 5a, 5b szintet mérik, a kreatív felhasználás (5c) csak a tesztelésre (a kérdéstípusoknak megfelelő trükkök alkalmazására) vonatkozik, a kérdéses tananyagra valószínűleg nem. Kivételt képez a kötetlen témájú mérések problémafelvetései. A LAU 5c szint mérése azt jelenti, hogy az adott tudás használatára nem készült a diák, mégis eszébe jutott és sikeresen alkalmazta egy adott probléma megoldásához.

A Research Gate-en az erről írt cikkünket (“Why Can’t I Learn Programming?” The Learning and Teaching Environment of Programming) eddig körülbelül 90 érdeklődő olvasta. Gyanítom, hogy az olvasottság egy része abból adódik, hogy a tanárképzésben ajánlott cikk (talán a nagyszombati egyetemen), de láttam dél-afrikai, sry lankai olvasót is. A portfólióba nem ezt a cikket tettem bele, mert angol nyelvű, helyette a mérés-értékelésről szóló InfoEra előadás prezentációját mellékelem.

## Informatikaoktatás

Az informatikaoktatás nagy kérdése a programozásoktatás és – napjainkban – digitális írástudásnak nevezett alkalmazói ismeretek oktatásának viszonya. Ezt cizellálja az egyik, illetve másik terület oktatásának szükségességéről folyó alkudozás, a tantervi elvárások tanórához rendelése körüli anomáliák, az egyéni kudarcok és sikerek kivetítése a teljes populációra.

A kutatásomban áttekintettem a NAT-okat, tanterveket, elemeztem azokat a tanmeneteket, amelyekkel sikeresen készítettem fel diákokat továbbtanulásra, versenyre és azokat is, amelyeknek a tanítása során problémák merültek fel. Ezek és az eddig leírtak alapján határoztam meg az informatika tudomány, az egyetemi informatikaoktatás és a közoktatási informatikaoktatás egymáshoz való viszonyát. (Most jön egy nagyon összetett bekezdés, de szerintem nagyon fontos.)

Az informatika tudomány – bár tartalmában folyamatosan változik, bővül – gondolkodási módszereiben és eszközeiben egységes rendszert alkot, amelynek oktatásához a tudományra specifikált oktatási módszertan és szakember szükséges. Az informatikai eszközöknek vannak felhasználói és alkotói; az alkalmazásoknak (szoftvereknek) vannak alkalmazói és programozói, a tevékenységeknek vannak – jellemzően gépi – végrehajtói és – jellemzően humán – szervezői. Azonban az informatikai kompetenciák oktatása során figyelembe kell venni, hogy az alkalmazói programok felhasználói jellemzően alkotók is egy másik tudomány területén; egyúttal az adott – magas szintű – alkalmazást alkotó programozók egy alacsonyabb szintű absztrakcióhoz készített alkalmazás felhasználói.

Példákkal illusztrálva: A dokumentum – legyen az prezentáció, web, nyomtatott, szöveges vagy multimédiás – készítő alkalmazások használatakor a gondolatainkat fejezzük ki, kódoljuk az adott alkalmazási környezetben, létrehozunk egy terméket. Másik oldalról, a programozó a kódolás, programozás során használ a gondolatai kifejezésére egy programozási nyelvet és egy fejlesztői alkalmazást, ezek segítségével hozza létre a terméket. A létrehozó-alkalmazó lánc eleje a villamosmérnökök kezében van, a vége pedig egy gyakorlati felhasználás, többnyire nem informatikai területen. (Ismét nehéz bekezdés következik)

A humán értelemre – a tudatosságra – jellemző, hogy a felhasználói, alkalmazói, végrehajtói kompetenciák fejlettsége az alkotói, programozói, szervezői kompetenciák fejlettségével összefügg. Az informatikaoktatás során e kétféle kompetenciacsoportot dialektikus egységben kell fejleszteni.

Konkrétabban: Az alkalmazói programok esetén a „hogyan használom”, „mire használom” tudások mellett a „hogyan működik” tudása is szükséges, ez biztosítja az alkalmazás LAU5c szintű, hosszú távon hasznosítható ismeretét. A programozás témakörben a programkészítés elmélete mellett a programozói környezet, mint alkalmazás és a programozási nyelv alkalmazott ismerete is szükséges. (Utóbbira ellenpélda a papíron programozás, a matematikai algoritmusok elméleti megfogalmazása.)

Következmények: A programozás fogalmainak, paradigmáinak tanítását a programozás tanítását jóval megelőzve, az alkalmazások tanítása során el kell kezdeni. Minden digitális írástudást fejlesztő óra egyben programozást előkészítő óra is. A programozás oktatását nem elméleti alapozással kell kezdeni, hanem a programkészítés gyakorlati ismereteivel: nyelvi ismeretek, kódolás, futtatás, tesztelés, fejlesztőkörnyezet ismerete. A programozás elméleti tárgyalását meg kell előzze egy program önálló megírásának képessége.

A „indirekt programozásoktatás” előadás- és cikksorozatban alkalmazási típusonként mutattam be a programozás beépítését az alkalmazások oktatásába. Ezek közül a táblázatkezelés oktatásba beépíthető digitális adat, algoritmus, eljárás, függvény fogalmakkal kapcsolatos ismereteket tárgyalja a marosvásárhelyi MathInfo konferencián bemutatott, majd a kolozsvári Acta Didacta Napocensia folyóiratban megjelent cikkem, amit mellékeltem. A sorozat összes előadása az alábbi felvetéssel indul:

|  |  |  |
| --- | --- | --- |
| A képen szöveg látható  Automatikusan generált leíráselképzelt cél:fejlett digitális írástudás és programozási ismeretek, fejlett informatikai gondolkodás | tapasztalt eredménycsekély digitális írástudás és programozási ismeret, informatikai gondolkodás hiányzik | megoldás:az informatikai gondolkodásmód fejlesztése a digitális írástudás és programozási ismeretek együtt tanításával.  |

Az informatikaoktatás ilyen értelmezését nagyon nehéz bevinni a közoktatásba, mert a tanár informatikáról, informatikaoktatásról alkotott képét kell módosítani, ami nekem eddig csak személyes beszélgetéssel sikerült. Az, hogy a Paint oktatása nem tanórai rajzolgatást jelent, hanem például a 3 bájtos RGB színábrázolás vizsgálatát, sem a tantervből, sem a tanmenetből nem látszik, óravázlatban unalmas és nagyon egyedi ahhoz, hogy kiderüljön, hogyan lehet hasonló szemlélettel tanítani a többi órán. Bár a legalapvetőbb elemeit a programozás indirekt tanításának az egyes cikkekben leírtam, ez tudományosan megfelelő, de az oktatási gyakorlatban elterjesztésre – ahol az úgy tanítom, ahogy tanultam módszer a leggyakoribb – lényegében alkalmatlan.

A programozás indirekt oktatásának gyakorlati oldalát próbálom bemutatni a Guess the Code sorozattal. Ebben a digitális írástudás oktatásában felhasználható feladatokba építek be valami extrát, amitől a programozási ismeret szükségessé válik. A példák kicsit hasonlítanak az E-hód verseny, mindennapokban felfedezhető informatikai, programozási ismereteket népszerűsítő feladataihoz, de a tanított alkalmazásokra vonatkoznak. Mellékeltem a 2018-as Science On Stage szegedi rendezvényről készült, beszámolót, ami az ISZE kiadásában megjelenő Inspirációban olvasható. Itt egy olyan példát szeretnék hozni, ami a poszteren nem szerepelt, mert úgy semmitmondó: Rajzolj 400 px széles, 300 px magas magyar zászlót (eddig egy uncsi feladat) a lehető legkevesebb lépésben! Írd le sorszámozással az egymásutáni lépéseket! Egy lépésnek számít… A feladat izgalmas, mert többféle eszközt érdemes kipróbálni a használatához. A diákoknak algoritmust (szekvenciát) kell írniuk. A lépések definiálásával lehet módosítani a nyertes megoldást, az rámutat arra, hogy az előregyártott eljárások hogyan segítik a kód írását, egyben lehetővé teszi, hogy a végén szinte mindenki győztes legyen. Emellett már egy ilyen pici feladat is felhívja a figyelmet a hatékonyságra, minőségi informatikai tudásra: maximális elérhető pontossággal, minimális idő alatt kell megoldani a feladatokat.

A Guess the Code feladatokat sokfelé hozom példaként, a gimnáziumi kollégáimnak is emlegetem néha, de még így is csak „csepegtetem” a lehetséges alkalmazásokat. Nagyon nehéz a megszokott feladatoktól elszakadni, a megírt prezentációk bemutatása helyett felfedeztető módra váltani. Márpedig az informatikai gondolkodás nem minta követéssel tanulható, hanem problémahelyzetek egyéni megoldásával. Itt ismét visszajutunk a LAU modellhez. Az informatikai gondolkodás az egyes ismeretek LAU1A és LAU5c szintjén a legjellemzőbb. Azonban mind a digitális írástudás tanulásának, mind a programozás tanulásának része kell legyen a LAU köztes szintje. Nem elég megérteni az új tananyagot, ki kell próbálni, saját tudássá kell tenni, rutint kell szerezni az alkalmazásból ahhoz, hogy kreatívan, készségszinten használható ismeret legyen.