Doktori (PhD) értekezés tézisei

SZÁMÍTÓGÉPES GONDOLKODÁS FEJLESZTÉSE NEM-TRADICIONÁLIS PROGRAMOZÁSI KÖRNYEZETEKBEN

Csapó Gábor

Témavezető: Dr. Csernoch Mária

DEBRECENI EGYETEM

Informatikai Tudományok Doktori Iskola

Debrecen, 2019

1. Elméleti megalapozás

1.1. Bevezetés

A modern információs társadalom és a mindennapi élet alapvető elvárása az adatok kezelésével kapcsolatos szöveg- és táblázatkezelői környezetek értő szintű használata. Ebben a gyakorlatban az algoritmikus készség és a számítógépes gondolkodás kulcsfontosságú szerepet töltenek be (Wing, 2006), melyek kialakítása és fejlesztése a témakörökhöz kapcsolódó tudáselemekkel együtt elsősorban az informatikaoktatás feladata. Azonban tapasztalatok és korábbi kutatások alapján a közoktatást elvégző tanulók nem rendelkeznek olyan szintű számítógépes gondolkodással, amelyre támaszkodhatnának a későbbi tanulmányaik vagy mindennapjaik során, illetve, amely az értő használatát ezeknek a szoftverkörnyezeteknek lehetővé tenné (Ben-Ari, 1999; Ben-Ari & Yeshno, 2006; Biró et al., 2015; Csernoch et al., 2015; EuSpRIG, 2019; Csernoch, 1997, 2009, 2010; Garrett, 2015; ICAEW, 2016; Kruck et al., 2003; Panko, 2013). Ennek eredményeképp a felhasználók nagy mennyiségben hibás dokumentumokat állítanak elő, amelyek emberi és anyagi erőforrások pazarlását eredményezik. A jelenség okát az informatikaoktatás során használt megközelítésekben és módszertanokban kell keresni (Csernoch & Biró, 2016). Az alkalmazói ismeretek oktatása során gyakran találkozhatunk az irodai szoftverek funkcióira kihegyezett, tényleges gondolkodást nem fejlesztő felületi módszerekkel. Ez a számítógépes problémamegoldás tipológiájában (Csernoch & Biró, 2015; Csernoch, 2017) a legalacsonyabb szintnek, a próbálkozás és varázsló alapú (Trial-And-Error Wizard) megközelítésnek feleltethető meg. Illetve, az ACM & IEEE (2013) jelentésben megfogalmazott tudásszintek közül kizárólagosan a második szintre fókuszál (eszközhasználat konkrét eseteken).

Az algoritmikus készség és a számítógépes gondolkodás fejlesztésére az egyik leghangsúlyosabb témakör a programozás oktatása. Figyelembe véve a tanulók tudás- és készségszintjét (Biró et al., 2015; Csernoch et al., 2015), kijelenthetjük, hogy a programozásoktatás a közoktatásban nem éri el célját. Az ezen problémákból eredő algoritmikus készség és számítógépes gondolkodás hiányában nem lehetséges tudatosan megtervezett, hibamentes és jólformázott dokumentumok elkészítése. A gyakorlat azt mutatja, hogy csupán „barkácsolni” tudnak a felhasználók (Ben-Ari, 1999; Csernoch, 2009, 2010; Csernoch & Biró, 2016; Panko, 2016; Panko & Port, 2013).

Célom olyan módszerek és módszertanok vizsgálata az informatikaoktatásban, amelyek alkalmasak a tanulók algoritmikus készségének és számítógépes gondolkodásának fejlesztésére nem tradicionális programozási környezeteket használva. A munkám során a táblázatkezelés, szövegkezelés és programozás témakörök oktatására kifejlesztett alternatív megközelítések és eszközök elemzésére és ezek hatékonyságvizsgálatára fókuszálok.

1.2. Spreadsheet Lego (Sprego)

A táblázatkezelés témakör algoritmikus szemléletű, sémaközpontú (Kahneman, 2011; Merriënboer & Sweller, 2005; Pólya, 1954) oktatását valósítja meg a Sprego módszertan (Csernoch, 2014; Csernoch & Biró, 2014). A Sprego a Pólya (1954) által megfogalmazott négylépéses problémamegoldási megközelítést alkalmazza a táblázatkezelői feladatok megoldása során:

1. A probléma megértése: Az adatok és a feladat elemzése. Az elérni kívánt output megfogalmazása.
2. A terv megépítése: A megoldáshoz szükséges algoritmus megtervezése.
3. A terv megvalósítása: Az algoritmus kódolása táblázatkezelői függvények használatával.
4. Visszatekintés: A kapott eredmények elemzése, tesztelése, diszkusszió.

A tradicionális módszerek által alkalmazott probléma-specifikus táblázatkezelői függvények használata helyett, a Sprego általános célú függvények kis csoportjára támaszkodik. A tanulók ezeket a függvényeket egymásba ágyazva, összetett tömbképletek létrehozásán keresztül fedik le a probléma-specifikus függvények funkcionalitását, miközben alapvető algoritmusokat sajátítanak el. A feladatmegoldás során a problémákat lépésekre bontva, az egyes lépések tervezésével, illetve azok input és output értékeinek kiemelésével, a módszertan sémákat alakít ki a tanulókban.

A módszertannal végzett munkát számos unplugged és semi-unplugged eszköz támogatja (Bell & Newton, 2013). Az unplugged eszközök csoportjába tartoznak az algoritmusok megépítését és a többszintű függvények felépítését segítő 3D nyomtatott matrjoska babák, origami csónakok, illetve jelölő mellények. A semi-unplugged eszközök repertoárjába az általunk fejlesztett algoritmusvizualizációs szoftverek tartoznak (Csapó & Sebestyén, 2017; Gulácsi & Dienes, 2018; Sebestyén et al., 2018), melyek célja, hogy vizuális reprezentációként szolgáljanak a leggyakrabban előforduló Sprego táblázatkezelői problémák algoritmusaira. Ugyancsak a semi-unplugged kategóriába tartoznak a módszertan által alkalmazott autentikus források. Ezek a források internetről letöltött és konvertált adattáblák, melyek olyan, valós

életben is előforduló adatokkal dolgoznak, amelyekkel a tanulók a mindennapjaik során is találkozhatnak. A Sprego alkalmas a táblázatkezelés témakör mellett az adatbázis-kezelés és programozás témakörök előkészítésére és oktatására is. A módszertan szoftverkörnyezettől, platformtól és korosztálytól függetlenül is alkalmazható, a tanulói csoportok sajátosságait figyelembe véve.

1.3. Error Recognition Model (ERM)

A szövegszerkesztés témakör oktatására az ERM módszertant (Csernoch, 1997, 2009, 2010) alkalmazzuk, amely a programozásoktatásban is használt debugging módszert követi (Gould, 1975; Gould & Drongowski, 1974; Jerinic, 2014). A módszertan megépítésének a TAaAS (Testing Algorithmic and Application Skills) projekt szolgált előmérésként (Biró et al., 2015; Csernoch et al., 2015). A tradicionális megközelítések felületorientált fókuszával szemben az ERM hibás dokumentumok elemzésén és javításán keresztül alakítja ki a tanulókban a jól formázott dokumentumok készítésének gyakorlatát (Csernoch, 2017). A módszer különválasztja a nyomtatásban és digitális formában felismerhető hibákat, melyeket hat hibakategóriába sorol: szintaktikai, szemantikai, tipográfiai, tördelési, formázási és stílus. A feladatmegoldás során a tanulók semi-unplugged eszközként alkalmazzák az interneten nagy számban előforduló autentikus hibás dokumentumokat. Ezek a dokumentumok illeszkednek a tanítási-tanulási folyamat adott szintjéhez, kijavításukon keresztül fokozatosan bevezetve a szövegkezelői ismereteket. Továbbá, olyan valós életből vett forrásokként szolgálnak, amelyek illeszkednek a tanulói csoportok érdeklődési köréhez, korosztályához és háttérismereteihez. A munkavégzést unplugged eszközként a dokumentumok nyomtatott verziója segíti, melyeket a tanulók a hibák megjelölésére, magyarázatára, a hibajavítás folyamatának nyomon követésére és változó feladatkontextusokban referenciákként használnak. Az unplugged eszközön a tanulók eltérő színekkel elkülönítik a nyomtatott formában azonosítható szintaktikai, szemantikai, tipográfiai és egyes formázási hibákat, valamint a digitális formában felismerhető tördelési, stílus és további formázási hibák megjelölését. Az ERM módszer korosztálytól és platformtól függetlenül alkalmazható az oktatásban, ismételten szem előtt tartva a tanulók és csoportok jellemzőit.

1.4. Vizuális programozás

A programozás témakör oktatására során elfogadott gyakorlat a különféle oktatási céllal készült programnyelvek alkalmazása, melyek között a vizuális programozást használó környezetek is jelen vannak (Fincher et al., 2010;

Fowler et al., 2012; Klassner & Anderson, 2003; Papadakis et al., 2014; Resnick et al., 2009). Munkánk során a vizuális programozás esemény-utasítás alapú megközelítését implementáló Construct 3 környezettel dolgoztunk (Csapó, 2019). A Construct 3 segítségével a tanulók multimédiás alkalmazásokat és játékokat készíthetnek, melyek vizuális kódját eseményblokkok definiálásával állítják elő. Az eseményblokkok a bennük definiált események (feltételek) bekövetkeztét vizsgálják, melyek teljesülésekor a blokkba foglalt utasítások végrehajtódnak. A környezet elsődlegesen ipari szoftverfejlesztési céllal készült, olyan felhasználókat megcélozva, akik nem rendelkeznek programozói tapasztalattal. Ebből kifolyólag vizuális nyelve általános célú megközelítést követ, minimális korlátozásokat gördítve a fejlesztők útjába. A feladatmegoldás során a tanulók az algoritmusépítést helyezik előtérbe. A feladatok részfeladatokra bontását, a célkitűzés, majd a részfeladat megoldására szolgáló algoritmus felépítése követi annak input és output értékeinek kiemelésével. Ezt követően a diákok természetes nyelven megfogalmazzák a vizuális programkódot, majd azt programkörnyezetben kódolják, ellenőrzik és szükség szerint debugolják. A környezettel végzett munka fontos velejárója, hogy a tanulók azonnali visszajelzést kapnak a munkájukról. Fontos kiemelni, hogy annak ellenére, hogy a Construct 3 elsődlegesen vizuális alkalmazások készítésére szolgál, tradicionális programozási feladatok megoldása is lehetséges a segítségével. A környezet platformtól függetlenül alkalmazható minden olyan eszközön, amely képes modern böngészőprogramok futtatására.

2. Tézisek

2.1. [T1] A Spreadsheet Lego módszertan szignifikánsan hatékonyabb a táblázatkezelés oktatására, mint a hagyományos, felületorientált megközelítések.

A táblázatkezelés témakör oktatására elvégeztünk egy vizsgálatot, melynek során a Sprego módszertan hatékonyságát elemeztük a hagyományos felületi megközelítésekkel szemben. A vizsgálatba egy helyi középiskola tanulócsoportjait vontunk be. Célunk az volt, hogy statisztikai eszközökkel is rávilágítsunk a Sprego módszertan algoritmus szemléletű megközelítésének hatékonyságára középiskolai környezetben. Ennek megfelelően a kutatási folyamat során keletkezett adatokat több szempontból is elemeztük. A tézis vizsgálatát három egységre bontottuk:

1. A Sprego módszertan hatékonyságának elemzése szemben a tradicionális módszerekkel, azonos előzetes tudással rendelkező és azonos korcsoportba tartozó tanulók esetén.
2. Az előzetes, hagyományos megközelítésekkel szerzett ismeretek hatása a Sprego módszertannal kialakított tudásra.
3. A választott problémamegoldási stratégia elemzése azon tanulók esetében, akik a Sprego módszertannal történő munkavégzést megelőzően tradicionális megközelítésekkel tanulták a témakört.

2.1.1. A vizsgált minta és a tesztelési időszak bemutatása

A vizsgálatot egy helyi középiskola három csoportjában végeztük el: két 7. évfolyamos és egy 10. évfolyamos csoportban. A tanulók az intézmény képzési rendjében 6 évfolyamos képzésben vettek részt. A vizsgálati csoportok Sprego módszertannal tanulták a témakört: egy 7. évfolyamos, N = 15 és a 10. évfolyamos, N = 18 csoport. A kontroll csoport pedig a hagyományos felületi megközelítésen keresztül haladt előre az elérhető hivatalos tankönyvre támaszkodva: a második 7. évfolyamos csoport, N = 13. Fontos kiemelni, hogy a két 7. évfolyamos csoport nem rendelkezett előzetes táblázatkezelő ismeretekkel, míg a 10. évfolyamos tanulók korábban két évfolyamon keresztül tanulták a témakört. Jelen vizsgálat keretein belül a 7. évfolyamos tanulók heti egy alkalommal 13 (vizsgálati) és 12 (kontroll), a 10. évfolyamos tanulók pedig heti két alkalommal 6 és fél héten keresztül tanultak táblázatkezelést.

2.1.2. Kutatási módszerek, stratégiák

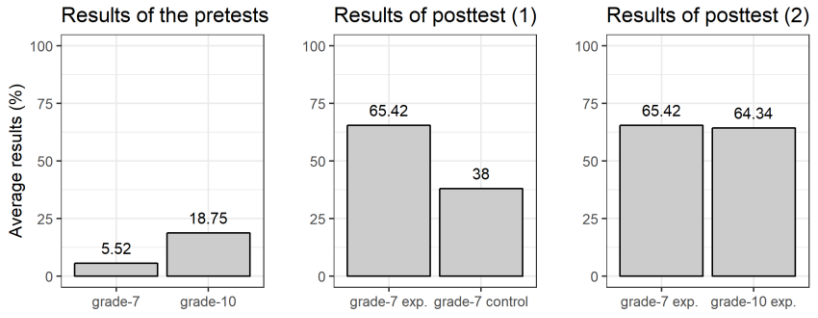
Az adatgyűjtést nyomtatott tesztek formájában végeztük minden csoportban. A tanulók a témakört megelőző tanórán előteszteket töltöttek ki, illetve a témakör utolsó tanóráján utóteszteket. Az adatgyűjtés mindkét alkalmával azonos tesztlapot használtunk. A teszt kialakításánál kiemelten figyeltünk arra, hogy az a feladatokba foglalt tudáselemekkel lefedje a táblázatkezelés témakörben tanult tudásegységeket. A teszt első felében alapvető képletkezelői és végrehajtási sorrend feladatokat oldottak meg a tanulók. Ezeket a teszt második felében valódi életből vett táblázatokon megfogalmazott problémák megértése, algoritmizálása, képletalkotási és függvényhasználati problémák követték. Az adatok tárolására saját tervezésű adatbázist hoztunk létre, amelybe a feladatokba foglalt tudáselemek alapján minden tanulói választ a lehető legrészletesebben elemekre bontottunk. A tézisbe foglalt három egység alapján az elemzés a következő felosztást követve valósult meg:

1. A 7. évfolyamos vizsgálati és kontroll csoportok utótesztjeinek összehasonlítása.
2. A 7. és 10. évfolyamos vizsgálati csoportok utótesztjeinek vizsgálata.
3. A tanulók elő- és utótesztekben alkalmazott függvényépítési megközelítéseinek analizálása a tesztlapok azonos programozási tételen alapuló feladatain keresztül.

Az adatok elemzését az R szoftver segítségével végeztük. A szignifikanciavizsgálat az 1. és 2. egységben Mann–Whitney próbák alkalmazásával, a 3. egységben pedig binomiális próbákkal történt.

2.1.3. Tézis vizsgálata

A vizsgálat első egységében az adatok elemzése alapján a Sprego módszertannal tanult csoport egy kivétellel minden esetben sikeresebben teljesítette a feladatokat, mint a kontroll csoport. A hagyományos módszertannal tanuló diákok magasabb pontszámot szereztek a HA() függvény konstans outputokkal történő alkalmazását vizsgáló feladatban. Ennek ellenére a későbbi feladatokban, melyek szintén igényelték a HA() függvény használatát, jelentősen alacsonyabb pontszámokat szereztek. A vizsgálati csoport a tesztbe foglalt feladatok 72,73%-ában ért el szignifikánsan jobb eredményt. Hasonló eredmény figyelhető meg a teljes teszt eredményeit egyben vizsgálva is, mely alapján a vizsgálati csoport 65,42%-os eredménye, szemben a kontroll csoport 38,00%-ával szintén szignifikáns eltérést mutatott ($p = 0{,}0013$) (1. ábra, középső rész).



1. ábra: A vizsgálati (grade-7/grade-7 exp. és grade-10/grade-10 exp.) és kontroll (grade-7 control) csoportok összesített százalékos eredményei az elő- és utóteszteken.

A vizsgálat második egységének eredményei rávilágítottak, hogy az előzetes felszínes tudás hátráltatja a Sprego módszertanban a problémamegoldó

készségek kialakulását. A 10. évfolyamos tanulók korosztályukból eredő előnyeik ellenére – kor, tapasztalat, előzetes tanórák stb. – nem tudtak jobb eredményeket elérni az utóteszteken, mint az előzetes ismeretekkel nem rendelkező 7. évfolyamos tanulók. A teszt feladatainak 72,73%-ában a 7. évfolyamos csoport teljesített jobban, azonban szignifikáns különbség nem mutatható ki a két csoport között. A teszt összesített eredményeit tekintve a 7. évfolyamos tanulók 65,42%-os, a 10. évfolyamos csoport pedig 64,34%-os szinten teljesítette azt, szignifikáns különbség nélkül ($p = 0,7388$) (1. ábra, jobb oldal), amelyek összehangban vannak más tantárgynak hasonló méréseinek eredményeivel (Champagne et al., 1983).

A tézisbe foglalt harmadik vizsgálati egység adatai egyértelműen mutatják, hogy az előzetes ismeretekkel nem rendelkező 7. évfolyamos csoportok az általuk tanult módszertan eszköztárát alkalmazzák. Ezzel szemben a 10. évfolyamos tanulók utótesztjének esetében (mikor már megismerkedtek a korábbi hagyományos módszerek mellett a Sprego módszertannal is) jelentősebb eltérések mutathatók ki. Amennyiben az üres kitöltéseket a tradicionális megközelítések megoldási stratégiájához csoportosítjuk, a vizsgált feladatok 33,32%-ában mutatható ki szignifikáns eltérés a választott módszereket vizsgálva ($p = 0,0225$). Azonban, ha az üres kitöltéseket figyelmen kívül hagyjuk, a feladatok 66,67%-ában választják a tanulók szignifikáns különbséggel a Sprego megoldási megközelítését ($p = 0,0064$ és $p = 0,0117$).

A tézisbe foglalt egységek eredményei alapján kijelenthetjük, hogy a Sprego módszertan az azonos korosztályba tartozó, azonos előzetes ismerettel rendelkező tanulók esetében szignifikánsan hatékonyabb a táblázatkezelés témakör oktatására, mint a tradicionális felületi megközelítések. Továbbá, a hagyományos megközelítésekkel kialakított előzetes tudás negatívan befolyásolja a Sprego módszertannal végzett munkát és sémaépítést. Azok a tanulók, akik korábban tradicionális megközelítésekkel tanulták a témakört a feladatok 66,67%-ában szignifikáns különbséggel támaszkodtak a Sprego módszertan megoldási stratégiájára. A fenti eredményeket figyelembe véve a [T1] tézis igazolt (Csapó et al., 2019).

2.2. [T2] A séma-alapú algoritmusépítésre alapozott táblázatkezelői módszertan oktatása hosszú távú tudást alakít ki.

A táblázatkezelés témakörben az algoritmusokra épülő sémaközpontú tudás kialakításának hatékonysága mellett kiemelten fontos a megszerzett tudás tartósságának vizsgálata is. A tézisbe foglalt kutatásunk során a Sprego módszertannal kialakított hosszú távú tudás hatékonyságát elemeztük szemben a tradicionális megközelítésekkel szerzett tudással. A vizsgálatot egy évvel a

táblázatkezelés témakör oktatását követően végeztük el, a tanulók képletalkotási és -elemző készségeinek tesztelésével. A kutatásaink során gyűjtött adatokat az alábbi két egységre bontva elemeztük:

1. Az egy évvel korábban szerzett táblázatkezelői tudás hatékonyságának vizsgálata Sprego, illetve tradicionális megközelítésekkel tanult csoportokban.
2. A [T1] tézis vizsgálatába foglalt, korábban hagyományos módszerrel tanult diákok Sprego tanórákat megelőző tudásszintjének összevetése a táblázatkezelésben előzetes ismeretekkel nem rendelkező csoportok eredményeivel.

2.2.1. A vizsgált minta és a tesztelési időszak bemutatása

A vizsgálatot egy helyi középiskola hét csoportjában végeztük el. A csoportok mindegyike a kutatási folyamatot megelőzően egy évvel korábban tanulta utoljára a táblázatkezelés témakört, a vizsgálati csoportok a Sprego módszertan, a kontroll csoportok pedig hagyományos megközelítések segítségével. A vizsgálati csoportok halmazát két 8. évfolyamos csoport (N = 18 és 18), illetve két 10. évfolyamos csoport (N = 17 és 15) alkotta. A kontroll csoportok összetételét illetően három 10. évfolyamos csoport (N = 7; 13 és 7) tartozott ebbe a kategóriába. A tanulók összesített létszámát tekintve a teljes minta 95 főt számlált. A két 8. évfolyamos csoport, valamint a kontroll csoportok közül két 10. évfolyamos csoport az intézményben 6 évfolyamos képzésben vett részt, amely befolyásolta a rendelkezésre álló tanórák számát. A vizsgálati csoport 8. évfolyamos tanulói az előző tanévben heti egy alkalommal 8, illetve a 10. évfolyamos diákok heti egy alkalommal 6 héten keresztül tanulták a témakört. Ezzel szemben a kontroll csoportok közül két 10. évfolyamos csoport heti két alkalommal 6, valamint azt megelőzően 7. évfolyamban heti egy alkalommal 8 héten keresztül, illetve egy 10. évfolyamos csoport heti egy alkalommal 6 héten át tanult táblázatkezelést.

2.2.2. Kutatási módszerek, stratégiák

A tanulók hosszú távú tudását nyomtatott formában, késleltetett utótesztek segítségével vizsgáltuk. A teszteket a diákok egy évvel a táblázatkezelés témakör befejezése után, a vizsgálat tanévének utolsó két hónapjában töltötték ki a csoportok időbeosztásától függően. A késleltetett utóteszt a TAaAS felmérésben is alkalmazott táblázatkezelői kérdésekre épült. A tanulók egy mintatáblázat alapján képletalkotási, illetve képletelemzési feladatokat oldottak meg, melyek a táblázatkezelés témakörben tanult alapvető adatkezelési algoritmusokra épültek. Az adatok tárolására ismételten létrehoztunk egy magas

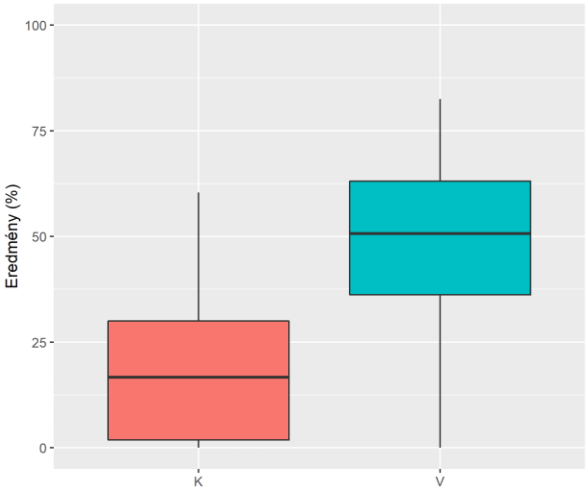
elemszámmal rendelkező adatbázist. A vizsgálat két egysége alapján a következő elemzéseket végeztük el:

1. A vizsgálati és kontroll csoportok késleltetett utótesztjeinek elemzése és a közöttük lévő eltérések feltárása.
2. A [T1] tézis vizsgálatában szereplő előtesztek alapján a 7. évfolyamos csoportok (vizsgálati és kontroll egybefogva) eredményeinek összevetése a 10. évfolyamos tanulók előtesztjeivel.

Az adatok elemzése az R szoftverrel történt. Az 1. egységben az adatok normalitásának vizsgálatára Shapiro-Wilk féle normalitás tesztet alkalmaztunk, illetve az 1. és 2. egységekben a szignifikanciavizsgálat Mann–Whitney próbák segítségével valósult meg.

2.2.3. Tézis vizsgálata

A tézis vizsgálatának első egységébe foglalt eredmények alapján a Sprego módszertannal tanult csoportok minden feladatban jobb eredményeket értek el, mint a kontroll csoportok. Fontosnak tartjuk kiemelni, hogy a 8. évfolyamos vizsgálati csoportok összteljesítménye (38,73% és 49,38%) nem mutat jelentős különbséget a két évfolyammal idősebb, 10. évfolyamos vizsgálati csoportok eredményeihez képest (55,08% és 50,37%). Ezzel szemben a kontroll csoportok 10. évfolyamos tanulói (14,52%; 28,54% és 7,20%) a két évfolyamnyi előny ellenére sem tudták elérni a 8. évfolyamos vizsgálati csoportok szintjét. A csoportok összesített eredményeit tekintve a vizsgálati csoportok 48,20%-os, a kontroll csoportok pedig 19,37%-os teljesítménnyel oldották meg a késleltetett utóteszteket. A százalékos eredményekben látható különbség a Mann-Whitney próba alapján szignifikáns eltérést mutatott a két csoportkategória között ($p = 0,0000$) (2. ábra).



2. ábra: A kontroll (K) és vizsgálati (V) csoportok százalékos eredményeinek eloszlása a késletetett utóteszt eredményei alapján.

A vizsgálat második egységében a [T1] tézis előtesztjeinek elemzése alapján a 7. évfolyamos tanulók – az előzetes ismereteik hiányát tükrözve – minimális pontszámmal oldották meg a feladatlapokat, 5,52%-os összeredményt elérve. Ezzel szemben a 10. évfolyamos tanulók eredményei négy feladat kivételével minden esetben szignifikánsan jobbnak bizonyultak. A teljes tesztet vizsgálva, a 10. évfolyamos tanulók 18,75%-kal teljesítették azt, szintén szignifikáns eltéréssel ($p = 0,0000$) (1. ábra, bal oldal). Fontos azonban kiemelni, hogy a 10. évfolyamos tanulók ezt a különbséget nagyon alacsony pontszámokkal érték el a teszt képletalkotási részében (8,00% alatt), szemben a 7. évfolyamos tanulók 0,00%-os értékeivel.

Az eredmények alapján a késletetett utóteszteken a Sprego módszertannal tanult diákok szignifikánsan jobban teljesítettek, mint a tradicionális módszerekkel tanult csoportok. Továbbá, a táblázatkezelés témakör elején írt tesztek rávilágítottak, hogy a táblázatkezelést korábban két évfolyamon keresztül hagyományos módszerrel tanuló 10. évfolyamos diákok csak minimális pontszámmal tudták felülmúlni a témakört még nem tanult csoportokat. Az ismertetett eredmények alapján a [T2] tézis igazolt. Az eredmények publikálása a közeljövőben várható.

2.3. [T3] Az első éves informatika szakos hallgatók nem tudnak támaszkodni a közoktatásban elsajátított szövegkezelői ismereteikre.

Azok a tanulók, akik a közoktatást sikeresen elvégzik és sikeres közép- vagy emelt szintű informatika érettségi vizsgát tesznek – a követelmények teljesítése alapján – olyan szövegkezelői ismeretekkel rendelkeznek, amelyre magabiztosan és hatékonyan támaszkodhatnak (Biró et al., 2015; Csapó, 2017; Csernoch et al., 2015). Ennek vizsgálatára elemeztük elsőéves informatika szakos hallgatók dokumentumelemző és hibafelismerő készségeit. A vizsgálatban a tanulók a hibás dokumentumokban gyakran előforduló hibaesetek közötti összefüggések meglátásának képességét teszteltük. Más megfogalmazásban, munkánkkal arra kerestük a választ, hogy a tanulók rendelkeztek-e a dokumentumok azonos típusú szerkezeti elemei közötti összefüggések azonosításához szükséges algoritmikus készséggel és számítógépes gondolkodással. A tézisben foglalt vizsgálat további lépéseként a kapott eredményeket összevetettük a tanulók informatika érettségin elért százalékos eredményeivel is. Ennek megfelelően a tézis elemzését két egységre bontottuk:

1. Annak vizsgálata, hogy a tanulók milyen arányban látták meg a kapcsolatot a megadott dokumentum azonos típusú szerkezeti egységei között.
2. A tanulók előző lépésben kapott eredményeinek és a középszintű informatika érettségi eredményeik összevetése.

2.3.1. A vizsgált minta és a tesztelési időszak bemutatása

A tanulók dokumentumelemző és hibafelismerő készségének vizsgálatában a mintát a Debreceni Egyetem Informatikai Karára 2014-ben felvett elsőéves Gazdaságinformatikus (GI) és Programtervező informatikus (PTI) hallgatók alkották. A tanulók többsége közvetlenül a középiskolai tanulmányaikat követően nyert felvételt az egyetemre, tehát a vizsgálatot megelőzően pár hónappal közép- és emelt szintű informatika érettségi bizonyítványokat szereztek. A tanulók létszámát illetően a GI csoport 86 főt, a PTI csoport 117 főt foglalt magába. A teljes minta nagysága N = 203 volt. A vizsgálat során a tanulók hozott tudásának szintjére voltunk kíváncsiak, így a kutatás nem foglalt magába oktatási tevékenységet.

2.3.2. Kutatási módszerek, stratégiák

A tézis vizsgálata folyamán az adatgyűjtést a TAaAS projekt keretein belül végeztük el. A hallgatók a felsőfokú tanulmányaik kezdetekor, az első tanítási

hét folyamán vettek részt a felmérésben, melynek szövegkezelés részének hibakategorizáló egysége szolgált jelen kutatásunk alapjául. A teszt kitöltésekor a tanulóknak egy mintadokumentum megjelölt hibás egységeit kellett a megfelelő hibakategóriákba sorolni. Az adatok feldolgozása saját tervezésű adatbázisban történt meg, azonban jelen vizsgálatban a feladatlap sajátosságaiból következve minimális elemszámmal dolgoztunk. A tanulók terminológiai tudáshiányosságait feltételezve az adatok elemzése során elsődlegesen arra helyeztük a fókuszt, hogy az azonos hibakategóriába tartozó hibapárokat azonos kategóriába sorolták-e, függetlenül annak helyességétől. A vizsgálat két egysége alapján az elemzés során a következő lépéseket követtük:

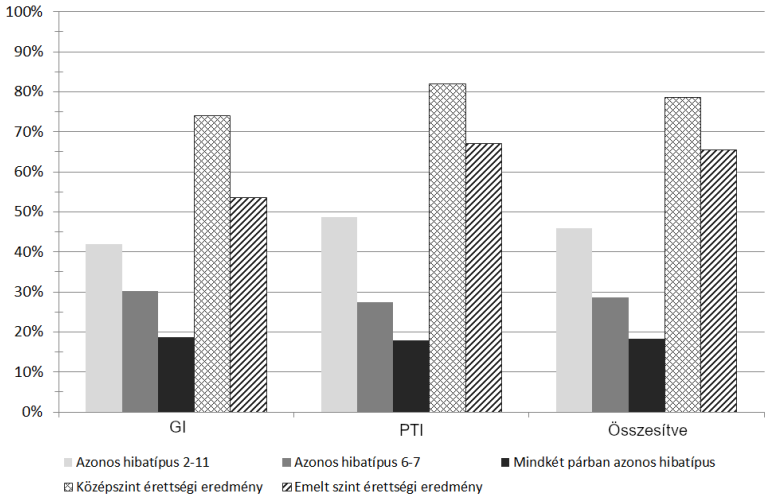
1. Az azonos hibakategóriába tartozó tördelési hibapárok tagjai közötti összefüggések felismerése.
2. A teszten elért eredmények összehasonlítása a hallgatók közép szintű informatika érettségi eredményeivel.

Az adatok elemzését táblázatkezelői környezetben végeztük, a Microsoft Excel szoftver segítségével.

2.3.3. Tézis vizsgálata

A vizsgálat első egységében kapott eredmények alapján a hibakategóriák közül az egyik párban az összefüggéseket a GI hallgatók 34,88%-a, valamint a PTI kitöltők 38,46%-a látta meg (összesítve: 36,95%). Az eredmények a mindkét hibapárban meglátott összefüggéseket egyszerre vizsgálva a következőképp alakulnak: GI: 18,60%; PTI: 17,95%; összesítve: 18,23%. A százalékos eredményeket vizsgálva látható, hogy a két csoport tagjai közel azonos arányban ismerték fel a dokumentum azonos típusú szerkezeti egységei között az összefüggéseket. Amennyiben az elemzésben az azonosságok felismerése mellett a helyes hibakategóriák megjelölését is vizsgáljuk (ahogyan a tesztlapon a feladat is kérte), úgy mindkét hibapár esetében csupán a hallgatók 1,97%-a tudta teljesíteni a feladatot.

A második egységben méréseink azt mutatták, hogy a GI csoport tagjai a mindkét hibapárban felismert azonosságokat illetően a hozott informatika érettségi eredményekhez hasonlítva 55,40%-os különbséget értek el. Ez az eltérés a PTI csoport esetében 64,05%. A teljes csoport átlagos eredményét vizsgálva (18,23%) az átlagos középszintű informatika érettségi eredményekhez képest 60,50%-os különbség mutatható ki. Kiemelnénk, hogy az azonosságok felismerése csupán a hibapárok tagjainak azonos hibakategóriába sorolását jelenti, azok helyességét nem vizsgálja (3. ábra).



3. ábra: A csoportok (GI: Gazdaságinformatikus és PTI: Programtervező informatikus) százalékos eredményei az azonosságok felismerésében, valamint a közép- és emelt szintű informatika érettségin.

Az eredmények alapján a hallgatóknak kevesebb mint a fele ismerte fel az azonosságokat egy hibapár esetében is. Az eltérő érettségi eredmények (3. ábra) ellenére a két csoport közel azonos szinten teljesítette a teszteket. Kijelenthetjük, hogy a hallgatók nem voltak képesek a dokumentum azonos szerkezeti egységei közötti összefüggések meglátására, valamint, hogy az érettségi eredmények nem tükrözték a teszten elért alacsony pontszámokat. Mivel ez a hiányosság lehetetlenné teszi a jól formázott dokumentumok létrehozását, a [T3] tézis igazolt (Csapó, 2017).

2.4. [T4] Az esemény-utasítás alapú vizuális programozás oktatása hatékonyabban fejleszti a tanulók algoritmikus készségét, mint a blokk alapú oktatási célú programnyelvek alkalmazása.

A programozás oktatásának gyakorlatában eszközök széles választékával találkozhatunk. Ezen eszközök közül kiemelt figyelmet kapnak az oktatási céllal kifejlesztett EPL-ek (Educational Programming Language). Az EPL-ek alkalmazásának és a vizuális programozás oktatásban megjelenésének gyakorlata nem új elképzelés, azonban korábbi mérések alapján (Biró et al., 2015; Csernoch et al., 2015) hatékonyságuk megkérdőjelezhető. A tézisbe foglalt kutatásunk során egy helyi középiskolában két vizuális programozásra alapuló környezet (Construct 3 és Scratch) oktatását és hatékonyságának

vizsgálatát végeztük el. Munkánkkal arra a kérdésre kerestük a választ, hogy a különféle programozási környezetek alkalmazása milyen hatékonysággal képes fejleszteni a tanulók algoritmikus készségét és számítógépes gondolkodását (Hubwieser, 2004).

2.4.1. A vizsgált minta és a tesztelési időszak bemutatása

Az eltérő programozási eszközök algoritmikus készségfejlesztői hatékonyságának vizsgálata egy helyi középiskola két 8. évfolyamos csoportjában történt. A tanulók az intézményben 6 évfolyamos képzésben vettek részt, melynek során a középiskolai éveik alatt korábban még nem tanulták a témakört. A vizsgálati csoport (N = 14) a Construct 3, a kontroll csoport (N = 14) pedig a Scratch környezet segítségével oldotta meg a feladatokat. A csoportok oktatása során kiemelt figyelmet fordítottunk a tanulók azonos előrehaladási tempójának biztosítására, ahogyan arra is, hogy a két eltérő környezet sajátosságai ellenére a csoportok azonos, elsősorban programozásorientált feladatok megoldásán keresztül tanuljanak. A vizsgálati csoport heti egy alkalommal 18, a kontroll csoport heti egy alkalommal 17 héten keresztül tanulta a programozást.

2.4.2. Kutatási módszerek, stratégiák

Az adatgyűjtésre a témakört megelőzően (előtesztek) és lezárását követően (utótesztek) nyomtatott tesztek segítségével került sor, mindkét alkalommal megegyező tesztlapokra támaszkodva. A feladatlapok összeállítását olyan, széleskörűen alkalmazott feladatok felhasználásával végeztük (ELTE IK T@T Labor, 2019; Pohl & Dagienė, 2019), amelyek a tanulók algoritmikus készségeinek vizsgálatára szolgálnak. A feladatok elemzése és kiválasztása során törekedtünk arra, hogy azok a tanulók algoritmikus készségét minél több szempont szerint megvizsgálják. Az adatok tárolására és elemzésére – a korábbi gyakorlatunkat követve – saját tervezésű adatbázist hoztunk létre. A feladatok sajátosságaiból adódóan több esetben is felmerült, hogy a lehetséges válaszok szűk tartományban mozogtak. Ebből kifolyólag a feladatok elemekre bontása korlátozott volt. Az eszközök algoritmizálási készségfejlesztő hatékonyságának vizsgálatához a vizsgálati és kontroll csoportok utótesztjei közötti eltéréseket analizáltuk. Az adatok elemzésére az R szoftvert alkalmaztuk. A vizsgálati és kontroll csoportok eredményei között tapasztalt eltérések szignifikanciavizsgálatát – figyelembe véve a minta méretét – Fisher-féle egzakt teszttel végeztük.

2.4.3. Tézis vizsgálata

Az előtesztek elemzése alapján kijelenthetjük, hogy a vizsgálati és a kontroll csoportok között nem volt eltérés a tanulók tudásszintjét illetően a tanulási folyamatot megelőzően. A témakör végén írt utótesztek vizsgálata kimutatta, hogy a vizsgálati és a kontroll csoport is hasonló eredményekkel oldotta meg a teszteket, nagy arányban helyes válaszokat adva. A feladatokra érkezett megoldások alapján szignifikáns különbséget egy esetben sem mutattunk ki a vizsgálati és kontroll csoportok között. Ez alól a legnagyobb eltérést mutató, gráfhasználati esetet vizsgáló feladat sem volt kivétel (a vizsgálati csoportból öt fő, a kontroll csoportból egy fő oldotta meg helyesen, $p = 0{,}0730$). Kiemelnénk, hogy az erre a feladatra érkezett válaszok tekintetében az előteszthez képest az utóteszten több tanuló választotta a disztraktorként szereplő, látszólagos helyes megoldást amellett, hogy a két teszten kapott helyes válaszok aránya változatlan maradt (Csernoch et al., 2015). Fontos megemlíteni, hogy két feladat esetében nem tudtuk elvégezni a szignifikancia vizsgálatot. Ennek okai, hogy az egyik feladatot mindkét csoport 100%-os aránnyal helyesen oldotta meg, illetve, hogy a másik feladat megoldásait annak variációs lehetőségeiből adódóan külön kezeltük. Ezekben az esetekben a Fisher-féle egzakt teszt nem volt alkalmazható.

Az eredményeket figyelembe véve, mivel nem találtunk szignifikáns különbséget az eltérő programozási eszközöket használó csoportok között az algoritmikus készséget vizsgáló feladatok megoldásaiban, a [T4] tézis elvetett (Hubwieser, 2004). Az eredmények publikálása a közeljövőben várható.

3. Jövőbeli elképzelések

Terveink között szerepel a Spreadsheet Lego módszertan hatékonyságvizsgálatának – a jelen kutatások során kapott pozitív eredményekből kiindulva – elvégzése nagyobb tanulói mintán. További vizsgálatok indokoltak a módszertanhoz fejlesztett oktatószoftverek hatékonyságának elemzésére az oktatási folyamatok támogatását illetően. Célunk még az Error Recognition Model módszertan hatékonyságvizsgálatának elvégzése a szövegkezelés témakör oktatására a középfokú informatikaoktatás keretein belül. A programozásoktatás során használt eszközök algoritmikus készségfejlesztő hatékonyságának vizsgálatát szeretnénk nagyobb tanulói mintán ismét elvégezni, eszközök szélesebb választékát elemezve. Emellett indokoltnak látjuk az alkalmazott feladatlap újra tervezését is a tanulói válaszok szélesebb eloszlásának biztosítása és további vizsgált készségek bevezetése érdekében.

4. Irodalomjegyzék

ACM & IEEE (2013). Computer Science Curricula: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM Press, and IEEE Computer Society Press, New York. DOI=http://dx.doi.org/10.1145/2534860.

Bell, T. & Newton, H. (2013). Unplugging Computer Science. D. M. Kadijevich, C. Angeli, & C. Schulte, Improving Computer Science Education, pp. 75–90. Routledge.

Ben-Ari, M. (1999). Bricolage Forever!. PPIG 1999, 11th Annual Workshop, University of Leeds: Computer-Based Learning Unit, UK, 5–7 January, 1999.. http://www.ppig.org/papers/11th-benari.pdf. Letöltés dátuma: 2016.12.01.

Ben-Ari, M. & Yeshno, T. (2006). Conceptual models of software artifacts. Interacting with Computers, 2006, 18(6), pp. 1336–1350. DOI=http://doi.org/10.1016/j.intcom.2006.03.005.

Biró, P., Csernoch, M., Máth, J. & Abari, K. (2015). Measuring the level of algorithmic skills at the end of secondary education in Hungary. Procedia - Social And Behavioral Sciences, 2015, 176, pp. 876–883.

Champagne, A. B., Gunstone, R. F. & Klopfer, L. E. (1983). Naive knowledge and science learning. Research in Science and Technological Education

Csapó, G. (2017). Az informatika érettségi és az informatikushallgatók szövegkezelési kultúrája. Válogatott tanulmányok a pedagógiai elmélet és szakmódszertanok köréből. http://www.irisro.org/pedagogia2017januar/85CsapoGabor.pdf Letöltés dátuma: 2019.09.11.

Csapó, G. (2019). Placing event-action based visual programming in the process of computer science education. Acta Polytechnica Hungarica, 2019, 16(2), Cognitive Infocommunications, pp. 35–57. http://www.uni-obuda.hu/journal/Csapo_89.pdf. Letöltés dátuma: 2019.08.14. DOI=http://doi.org/10.12700/APH.16.2.2019.2.3.

Csapó, G. & Sebestyén, K. (2017). Educational Software for the Sprego Method. The Turkish Online Journal of Educational Technology, INTE 2017 October, pp. 986–999. http://www.tojet.net/special/2017_10_1.pdf. Letöltés dátuma: 2019.08.14.

Csapó G., Csernoch M. & Abari K. (2019). Sprego: Case Study on the Effectiveness of Teaching Spreadsheet Management with Schema Construction. Education and Information Technologies. Accepted.

Csernoch, M. (1997). Methodological Questions of Teaching Word Processing. 3rd International Conference on Applied Informatics, Eger-Noszvaj, Hungary, pp. 375–382.

Csernoch, M. (2009). Teaching word processing – the theory behind. Teaching Mathematics and Computer Science, 2009, 1, pp. 119–137.

Csernoch, M. (2010). Teaching word processing – the practice. Teaching Mathematics and Computer Science, 2010, 8(2), pp. 247–262.

Csernoch, M. (2014). Programozás táblázatkezelő függvényekkel – Sprego. Műszaki Könyvkiadó, Budapest.

Csernoch, M. (2017). Thinking Fast and Slow in Computer Problem Solving. Journal of Software Engineering and Applications, 2017, 10(1). http://file.scirp.org/pdf/JSEA_2017012315324696.pdf. Letöltés dátuma: 2017.07.08.

Csernoch, M. & Biró, P. (2014). Sprego programming. Spreadsheets in Education (eJSiE), 2014, 8(1). http://epublications.bond.edu.au/cgi/viewcontent.cgi?article=1175&context=ejsie. Letöltés dátuma: 2015.04.03

Csernoch, M. & Biró, P. (2015). Számítógépes problémamegoldás. TMT, Tudományos és Műszaki Tájékoztatás, Könyvtár- és információtudományi szakfolyóirat, 2015, 62(3), pp. 86–94.

Csernoch, M. & Biró, P. (2016). Teaching methods are erroneous: approaches which lead to erroneous end-user computing. European Spreadsheet Risk Interest Group Conference, EuSpRIG2016, 2016, London. http://www.eusprig.org/mcsernoch-2016.pdf. Letöltés dátuma: 2016.07.21.

Csernoch, M., Biró, P., Máth, J. & Abari, K. (2015). Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments. Informatics in Education, 2015, 14(2), pp. 175–197. DOI=http://doi.org/10.15388/infedu.2015.11.

ELTE IK T@T Labor (2019). Archivum | e-Hód. http://e-hod.elte.hu/archivum/. Letöltés dátuma: 2019.08.04.

EuSpRIG, European Spreadsheet Risk Interest Group (2019). EuSpRIG Horror Stories. http://www.eusprig.org/horror-stories.htm. Letöltés dátuma: 2019.06.28.

Fincher, S., Cooper, S., Kölling, M. & Maloney, J. (2010). Comparing alice, greenfoot & scratch. 41st ACM technical symposium on Computer science education, ACM, 2010, pp. 192–193.

Fowler, A., Fristce, T. & MacLauren, M. (2012). Kodu Game Lab: a programming environment. The Computer Games Journal, 2012, 1(1), pp. 17–28.

Garrett, N. (2015). Textbooks for Responsible Data Analysis in Excel. Journal of Education for Business, 2015, 90(4), pp. 169-174. DOI=http://doi.org/10.1080/08832323.2015.1007908.

Gould, J. (1975). Some psychological evidence on how people debug computer programs. International Journal of Man-Machine Studies, 1975, (7)1, pp. 151–182.

Gould, J. & Drongowski, P. (1974). An exploratory study of computer program debugging. Human Factors, 1974, 16(3), pp. 258–277.

Hubwieser, P. (2004). Functional Modelling in Secondary Schools Using Spreadsheets. Education and Information Technologies, 2004, 9(2), pp. 175–183. DOI=http://doi.org/10.1023/B:EAIT.0000027929.91773.ab.

Jerinic, L. (2014). Teaching Introductory Programming Agent-based Approach with Pedagogical Patterns for Learning by Mistake. International Journal of Advanced Computer Science and Applications, 2014, (5)6.

Gulácsi, Á. & Dienes, N. (2018). 3D Software Environment for Educational Sprego Programming. The Turkish Online Journal of Educational Technology, INTE 2018 November (2), pp. 837–844. http://www.tojet.net/special/2018_12_3.pdf. Letöltés dátuma: 2019.01.21.

Klassner, F. & Anderson, S. D. (2003). Lego MindStorms: Not just for K-12 anymore, IEEE Robotics & Automation Magazine, 2003, 10(2), pp. 12–18.

ICAEW (2016). Spreadsheet competency framework: A structure for classifying spreadsheet ability in finance professionals. ICAEW, London. http://www.icaew.com/-/media/corporate/files/technical/information-technology/it-faculty/spreadsheet-competency-framework.ashx. Letöltés dátuma: 2019.08.14.

Kahneman, D. (2011). Thinking, Fast and Slow. Farrar, Straus and Giroux, New York.

Kruck, S. E., Maher, J. J. & Barkhi R. (2003). Framework for Cognitive Skill Acquisition and Spreadsheet Training. Journal of Organizational and End User Computing, 2003, 15(1), pp. 20–37.

Merriënboer, J.J.G. van & Sweller, J. (2005). Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions. Educational Psychology Review, 2005, 17(2), pp. 147–177. DOI=http://doi.org/10.1007/s10648-005-3951-0.

Panko, R. (2013). The Cognitive Science of Spreadsheet Errors: Why Thinking is Bad. 46th Hawaii International Conference on System Sciences, Maui, 2013, pp. 4013–4022. IEEE.

Panko, R. (2016). What We Don’t Know About Spreadsheet Errors Today: The Facts, Why We Don't Believe Them, and What We Need to Do. arXiv preprint. http://arxiv.org/abs/1602.02601. Letöltés dátuma: 2016.07.21.

Panko, R. & Port, D. (2013). End User Computing: The Dark Matter (and Dark Energy) of Corporate It. Journal of Organizational and End User Computing, 2013, 25(3), pp. 1–19.

Papadakis, S., Kalogiannakis, M., Orfanakis, V. & Zaranis, N. (2014). Novice programming environments. Scratch & app inventor: a first comparison. Workshop on Interaction Design in Educational Environments, ACM, 2014.

Pohl, W. & Dagienė, V. (2019). What is Bebras | www.bebras.org. https://www.bebras.org/. Letöltés dátuma: 2019.08.04.

Pólya, G. (1954). How To Solve It. A New Aspect of Mathematical Method. 2nd edition, 1957. Princeton University Press, Princeton, New Jersey.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: programming for all, Communications of the ACM, 2009, 52(11), pp. 60–67.

Sebestyén, K., Csapó, G. & Csernoch, M. (2018). Visualising Sprego Inequality Problems With 2D Representations. The Turkish Online Journal of Educational Technology, INTE 2018 November (2), pp. 888–898. http://www.tojet.net/special/2018_12_3.pdf. Letöltés dátuma: 2019.08.14.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 2006, 49(3), pp. 33–35.

DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

Nyilvántartási szám: DEENK/323/2019.PL
Tárgy: PhD Publikációs Lista

Jelölt: Csapó Gábor
Neptun kód: QVGE6Y
Doktori Iskola: Informatikai Tudományok Doktori Iskola
MTMT azonosító: 10052249

A PhD értekezés alapjául szolgáló közlemények

Magyar nyelvű könyvrészletek (1)

1. **Csapó, G.**: Az informatika érettségi és az informatikushallgatók szövegkezelési kultúrája. In: Válogatott tanulmányok a pedagógiai elmélet és szakmódszertanok köréből. Szerk.: Karlovitz János Tibor, International Research Institute s.r.o., Komárno, 387-394, 2017. ISBN: 9788089691401

Idegen nyelvű tudományos közlemények hazai folyóiratban (1)

2. **Csapó, G.**: Placing Event-Action-based Visual Programming in the Process of Computer Science Education.
APH. 16 (2), 35-57, 2019. EISSN: 1785-8860.
DOI: http://dx.doi.org/10.12700/APH.16.2.2019.2.3
IF: 1.286 (2018)

Idegen nyelvű tudományos közlemények külföldi folyóiratban (1)

3. **Csapó, G.**, Csernoch, M., Abari, K.: Sprego: Case Study on the Effectiveness of Teaching Spreadsheet Management with Schema Construction.
Educ. Inf. Technol. "Accepted by Publisher", 1-16, 2019. ISSN: 1360-2357.

Magyar nyelvű konferencia közlemények (2)

4. **Csapó, G.**, Sebestyén, K.: Oktatóprogram a Sprego táblázatkezelő módszerhez. In: INFODIDACT 2015. Szerk.: Szlávi Péter, Zsakó László, Webdidaktika Alapítvány, Zamárdi, 1-18, 2015. ISBN: 9789631238921
5. **Csapó, G.**: Vizuális programozás oktatása középiskolákban. In: INFODIDACT 2014. Szerk.: Szlávi Péter, Zsakó László, Webdidaktika Alapítvány, Zamárdi, 1-10, 2014. ISBN: 9789631206272

DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

Idegen nyelvű konferencia közlemények (2)

6. **Csapó, G.**, Sebestyén, K.: Educational software for the Sprego method.
Turk. Online J. Educ. Technol. Special Issue for INTE 2017, October, 986-999, 2017. EISSN: 2146-7242.

7. **Csapó, G.**: Sprego virtual collaboration space.
In: 8th IEEE International Conference on Cognitive Infocommunications: CogInfoCom 2017 : Proceedings : September 11-14, 2017 Debrecen, Hungary, IEEE Computer Society, Piscataway, 137-142, 2017. ISBN: 9781538612644

Idegen nyelvű absztrakt kiadványok (1)

8. **Csapó, G.**: Sprego virtual collaboration space: improvement guidelines for the MaxWhere seminar system.
In: 8th IEEE International Conference on Cognitive Infocommunications : CogInfoCom 2017 : Proceedings : September 11-14, 2017 Debrecen, Hungary, IEEE Computer Society, Piscataway, 143-144, 2017. ISBN: 9781538612644

További közlemények

Idegen nyelvű konferencia közlemények (2)

9. Sebestyén, K., **Csapó, G.**, Csernoch, M.: Introduction to Algorithmic Based Data Management in Spreadsheet Environment.
Turk. Online J. Educ. Technol. "Accepted by Publisher", 1-8, 2019. EISSN: 1303-6521.

10. Sebestyén, K., **Csapó, G.**, Csernoch, M.: Visualising Sprego Inequality Problems With 2D Representations.
Turk. Online J. Educ. Technol. Special Issue for INTE-ITICAM-IDEC 2018, 888-898, 2018. EISSN: 1303-6521.

A közlő folyóiratok összesített impakt faktora: 1,286
A közlő folyóiratok összesített impakt faktora (az értekezés alapjául szolgáló közleményekre): 1,286

A DEENK a Jelölt által az iDEa Tudóstérbe feltöltött adatok bibliográfiai és tudománymetriai ellenőrzését a tudományos adatbázisok és a Journal Citation Reports Impact Factor lista alapján elvégezte.

Debrecen, 2019.09.18.

1. Theoretical background

1.1. Introduction

The conscious application of data handling in spreadsheet and text management environments is a requirement in today's informational society and in our everyday life. In this practice computational thinking and algorithmic skills play a critical role (Wing, 2006). Developing these skills along with the accompanying knowledge items for these topics primarily relies on ICT (Information and Communications Technology) education. However, according to prior research, students who complete secondary education do not possess the required level of computational thinking on which they could build during their everyday lives or their further studies (Ben-Ari, 1999; Ben-Ari & Yeshno, 2006; Biró et al., 2015; Csernoch et al., 2015; EuSpRIG, 2019; Csernoch, 1997, 2009, 2010; Garrett, 2015; ICAEW, 2016; Kruck et al., 2003; Panko, 2013). Furthermore, the low levels of these skills make it impossible to use the spreadsheet and text management environments consciously. This results in a large number of erroneous user productions, which causes a loss of both human and financial resources. The reason behind this phenomenon originates from the approaches and methodologies used in ICT education (Csernoch & Biró, 2016). Teaching end-user applications usually resolves around the use of birotical software interfaces without developing any actual thinking. In the typology of computer problem solving (Csernoch, 2017) this approach is in compliance with the lowest, Trial-and-Error Wizard based, level. Furthermore, regarding the ACM & IEEE (2013) report it focuses only on the second level of knowledge levels (using a tool in a concrete context).

The development of algorithmic and computational thinking skills most often emerges in the topic of programming. Taking the students' level of knowledge and skills into consideration (Biró et al., 2015; Csernoch et al., 2015) we can conclude that public programming education has failed to reach its goals. The low level of algorithmic and computational thinking skills originating from these problems makes the design and creation of consciously structured, error-free and well-formatted documents impossible. Practice shows that it only allows users to bricolage (Ben-Ari, 1999; Csernoch, 2009, 2010; Csernoch & Biró, 2016; Panko, 2016; Panko & Port, 2013).

My goal is to analyze methodologies and methods in ICT education which are suitable to develop students' algorithmic and computational thinking

skills using non-traditional programming environments. During my work I focus on analyzing and testing the effectiveness of alternative approaches and tools developed for teaching spreadsheet management, text management and programming.

1.2. Spreadsheet Lego (Sprego)

The Sprego methodology (Csernoch, 2014; Csernoch & Biró, 2014) teaches the spreadsheet management topic with an algorithmic and schemata centric focus (Kahneman, 2011; Merriënboer & Sweller, 2005; Pólya, 1954). Sprego applies the four-step problem solving process of Pólya (1954) for solving spreadsheet tasks:

1. Understanding the problem: Analyzing the data and the task. Defining the desired output.
2. Devising a plan: Designing the algorithm required for solving the problem.
3. Carrying out the plan: Coding the algorithm using spreadsheet functions.
4. Looking back: Analyzing and testing the results. Discussion.

Instead of using the large number of problem specific functions present in traditional approaches, Sprego focuses on a small number of general-purpose functions. Using and embedding these functions into each other the students construct composite array formulas covering the functionalities of the problem-specific functions, while also learning fundamental algorithms in the process. During the problem-solving practice, Sprego decomposes the problems into steps, and develops schemata through the planning of these steps while highlighting their input and output values.

Sprego is supported with various unplugged and semi-unplugged tools (Bell & Newton, 2013). As unplugged tools, the students work with 3D printed Matrjoska dolls, origami boats and visibility vests to facilitate the construction of the algorithms and the building of multilevel functions. Regarding the semi-unplugged tools, the methodology has two algorithm visualization programs of our own development (Csapó & Sebestyén, 2017; Gulácsi & Dienes, 2018; Sebestyén et al., 2018). These tools serve as visual representations for the algorithms of the most common Sprego problems. Further examples of the group of semi-unplugged tools are the authentic sources used in problem-solving. These sources are webtables downloaded and converted from the internet in which the students work with real-life data that they encounter in their everyday lives as well. Besides teaching spreadsheet management, the Sprego methodology is suitable for preparing and teaching database management and programming, as well. The method can be used without software

and platform restrictions and is suitable for all age groups, taking their particularities into consideration.

1.3. Error Recognition Model (ERM)

For teaching text management, we use the ERM methodology (Csernoch, 1997, 2009, 2010) which follows the debugging method used in programming education (Gould, 1975; Gould & Drongowski, 1974; Jerinic, 2014). For the creation of the methodology, the TAaAS (Testing Algorithmic and Application Skills) project served as a pre-measurement (Biró et al., 2015; Csernoch et al., 2015). Instead of focusing on the interface of the text management environments – as traditional approaches do – the ERM develops the practice of creating well-formatted documents through analyzing and correcting erroneous texts (Csernoch, 2017). The method separates the errors recognizable in printed and in digital form, grouping them into six categories: syntactic, semantic, typographic, layout, formatting, and style. During the problem-solving the students use authentic erroneous documents as a semi-unplugged tool; these can be found on the internet in large quantities. These documents connect to the given level of the teaching-learning process, gradually introducing text management knowledge items through their correction. Furthermore, they present real-life examples of texts that match the students' interest, age groups and background knowledge. The work with ERM is aided by the printed form of the documents used as unplugged tools. The students use these printouts to mark and detail the errors in them, to guide them through the correction process, and also for future reference in different problem contexts. The students use different colors to mark the syntactic, semantic, typographic and some formatting errors recognizable in printed form, and the layout, style and further formatting errors identifiable in digital form. The methodology can be used in education, on any platform, and for teaching all age groups, again, taking the particularities of students and teachers into consideration.

1.4. Visual programming

It is an accepted practice to use various educational programming languages for teaching programming, among which visual programming environments are also present (Fincher et al., 2010; Fowler et al., 2012; Klassner & Anderson, 2003; Papadakis et al., 2014; Resnick et al., 2009). In our work we used Construct 3, which implements an event-action based visual programming form (Csapó, 2019). Using this environment, students create multimedia applications and games whose visual codes are built up by defining event

blocks. These blocks test the events (conditions) encompassed in them and run the connected actions upon their fulfillment. The environment was developed with software production in mind, targeting users who have no prior programming experience. It follows that its visual programming language uses a general-purpose approach, with minimal limitations in terms of project complexity. During the problem-solving process students focus on the construction of algorithms. In the first step, they decompose the tasks into subtasks, which is followed by the definition of the goal, and the building of the algorithm required for solving the selected subtask with its input and output values highlighted. During the following steps students define the visual code, using natural language which is then translated into the event-action based form of the environment. Then they check the results and if necessary, debug the code with teacher guidance. One of the most important aspects of working with this environment is that students get immediate feedback on their work. Note that despite the visual design of Construct 3, traditional programming tasks are also solvable with its help. The environment can be used without platform restrictions on any device that can run modern web browsers.

2. Theses

2.1. [T1] The Spreadsheet Lego methodology is significantly more effective for teaching spreadsheet management compared to the traditional surface approaches.

We carried out an analysis on the effectiveness of the Sprego methodology compared to traditional surface approaches for teaching spreadsheet management. The experiment focused on student groups from a local high school. Our goal was to statistically study the effectiveness of the algorithm focused approach of Sprego in secondary education. Based on this, we analyzed the data gathered in the research from multiple perspectives. We divided the examination of the thesis into three parts:

1. The analysis of the effectiveness of Sprego compared to traditional methods in groups with identical prior knowledge within the same age group.
2. The effect of prior knowledge developed with traditional approaches on the learning outcomes of Sprego.
3. The analysis of selected problem-solving approaches with students who, before learning with the Sprego methodology, studied spreadsheet management using traditional approaches.

2.1.1. The samples investigated and the phases of the research

The analysis was carried out with groups from a local high school: two grade-7 groups and one grade-10 group. Regarding the schedule of the school, the students took part in a 6-year training program. The experimental groups studied the topic with the Sprego methodology: one grade-7 (N = 15), and the grade-10 groups (N = 18). The control group relied on the traditional approach using the officially available textbook: this was the second grade-7 group (N = 13). It is worth noting that the two grade-7 groups possessed no prior knowledge of spreadsheet management, while the grade-10 students had learned it beforehand over two school years. In the current research, the grade-7 students studied the topic for 13 (experimental) and 12 (control) weeks with one lesson per week, and the grade-10 students for 6 and half weeks with two lessons per week.

2.1.2. Research methods, strategies

For the data collection we used printed out test sheets in every group. The students completed the pre-tests right before the spreadsheet management topic and the post-tests during the last lesson of the topic. For both occurrences of the data collection we used the same test sheets. During the design of the test we focused on covering the knowledge items of the topic with the items present in the tasks. In the first part of the test the students solved basic formula management and execution order tasks. In the second part of the test this was followed by understanding problems – using tables from real-life contexts –, and creating algorithms and formulas using functions. For storing the data, we created a database in which we stored the students' answers disassembled into items. Following the three parts of the current thesis we analyzed the data as follows:

1. Comparing the post-tests of the grade-7 experimental and control groups.
2. Analyzing the post-tests of the grade-7 and grade-10 experimental groups.
3. Examining the formula creation approaches used by students in the pre- and post-tests, focusing on the tasks based on similar programming items.

The analysis of the data was carried out using the R software package. In the first and second part we conducted the significance analysis with Mann–Whitney tests and in the third part with the use of binominal tests.

2.1.3. Analyzing the thesis

Based on the analysis in the first part of our research, students who studied the topic with the Sprego methodology completed the tests more successfully

in every task than the control group, with one exception. The students who studied with traditional approaches gained a higher score in the task which involved using the IF() function with constant outputs. Notwithstanding this, we could observe a notable reduction in points in the following tasks, which also required the use of the IF() function. The experimental group gained significantly better scores in 72.73% of the tasks. A similar result was found considering the overall score of the tests: the experimental group's result of 65.42% compared to the control group's 38.00%, showing a significant difference ($p = 0.0013$) (Figure 1, middle part).

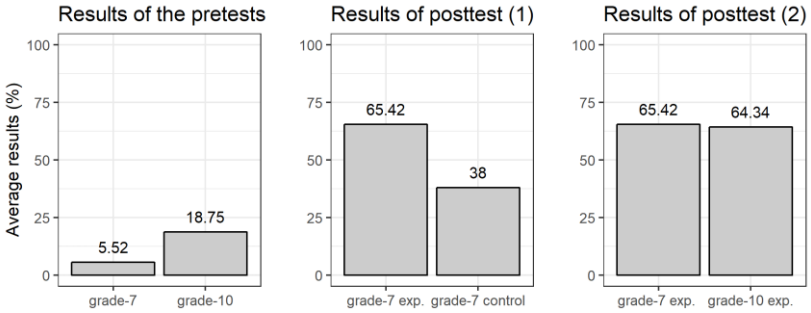


Figure 1: The overall percentage results of the experimental (grade-7/grade-7 exp. and grade-10/grade-10 exp.) and control (grade-7 control) groups in the pre- and post-tests.

The results of the second part of the analysis pointed out that prior, surface knowledge hinders the acquisition of the problem-solving skills developed with the Sprego methodology. The grade-10 students, despite their advantages – age, experience, prior classes, etc. – could not achieve better results on the post-test than the grade-7 students who had no prior knowledge of the topic. In 72.73% of the tasks the grade-7 students performed better, although without significant differences. Regarding the overall score of the test, the grade-7 students scored 65.42%, compared to the grade-10 students' 64.34%, with no significant difference between the groups ($p = 0.7388$) (Figure 1, right part), which are in accordance with the results of similar measurements conducted with different subjects (Champagne et al., 1983).

In the third part of this thesis, the data clearly show that the grade-7 students who had no prior knowledge use the toolset of the methodology which they have learned the topic with. In contrast, the grade-10 students' post-test – after they learned Sprego as well – shows observable differences. If we count the empty solutions as part of the problem-solving strategy of the traditional

approaches, in 33.32% of the analyzed tasks a significant difference can be observed regarding the deviation between the chosen methods ($p = 0.0225$). However, if we discard the empty solutions, the students choose to work with Sprego with a significant difference in 66.67% of the analyzed tasks ($p = 0.0064$ and $p = 0.0117$).

Based on the results of the parts included in this thesis, we can conclude that the Sprego methodology is significantly more effective in teaching spreadsheet management with students of the same age and of the same prior knowledge, compared to the traditional surface approaches. Furthermore, the knowledge acquired with traditional approaches negatively affects the work and schemata developed with the Sprego methodology. Those students who learned the topic with traditional methods beforehand, chose to work with the problem-solving strategy of Sprego in 66.67% of the analyzed tasks with a significant difference. Considering the results above, the [T1] thesis is proved (Csapó et al., 2019).

2.2. [T2] The teaching of schemata-based, algorithm focused spreadsheet management methodology develops long-term knowledge.

Besides the effectiveness of the schemata centric, algorithm-based knowledge in the spreadsheet management topic, it is also especially important to analyze the permanence of this developed knowledge. Our research in this thesis focuses on examining the effectiveness of the long-term knowledge developed with the Sprego methodology compared to the knowledge acquired with traditional approaches. The analysis was carried out one year after the students finished learning the spreadsheet management topic, testing the students' formula creation and analytical abilities. We examined the data gathered during our research, divided into two parts:

1. The effectiveness of the knowledge acquired one year prior to our research in groups who studied the topic with Sprego and with traditional approaches.
2. Following on the data acquired in the [T1] thesis, the comparison of the knowledge level of students who studied spreadsheet management with traditional methods prior to their Sprego classes, with the results of the groups who had no prior knowledge in the topic.

2.2.1. The samples investigated and the phases of the research

We examined seven groups from a local high school. The students in each group studied spreadsheet management one year prior to our research. The experimental groups studied the topic with the Sprego methodology, while

the control groups used traditional approaches. Two grade-8 (N = 18 and 18), and two grade-10 groups (N = 17 and 15) formed the experimental groups. Regarding the number of students in the control groups, there were three grade-10 groups (N = 7; 13 and 7). The whole sample included 95 students in total. The two grade-8 and two of the grade-10 control groups took part in the school's 6-year course, which affected their available number of lessons. The grade-8 students in the experimental groups studied the topic last year for 8 weeks with one lesson per week, and the grade-10 students for 6 weeks with the same timetable. In contrast, two grade-10 classes of the control groups studied spreadsheet management for 6 weeks with two lessons per week, and before that for 8 weeks with one lesson per week in grade-7. The remaining grade-10 students in the last control group studied the topic for 6 weeks with one lesson per week.

2.2.2. Research methods, strategies

We analyzed the long-term knowledge of students in printed form using delayed post-tests. The students completed the tests during the last two months – depending on the schedule of the groups – of the school year, one year after they had studied spreadsheet management. The delayed post-test was based on the spreadsheet management questions present in the TAaAS measurement. The students had to solve formula creation and analysis tasks using a sample table provided, focusing on the fundamental data handling algorithms they had learned during the course. For storing the data, we created a database with a large number of items, taking into consideration the multiple possible solutions to the tasks. Based on the two parts of the analysis, we completed the following examinations:

1. The analysis and the differences between the experimental and control groups' delayed post-tests.
2. Using the results of the pre-tests from the research detailed in the [T1] thesis, the examination of the differences between the grade-7 (experimental and control groups combined) and grade-10 students.

We used the R software package for the analysis of the data. In the first part we analyzed the normality of the data using Shapiro-Wilk normality tests. For examining the significance of the differences between the groups, we applied Mann–Whitney tests in both parts of the research.

2.2.3. Analyzing the thesis

Based on the results in the first part of this thesis the groups who studied the topic with Sprego completed the tests more successfully than the control

groups in every task. It is worth noting that the grade-8 experimental groups' overall scores (38.73% and 49.38%) show no noteworthy differences from the results of the two years older grade-10 experimental groups (55.08% and 50.37%). In contrast, the grade-10 control groups (14.52%; 28.54% and 7.20%) despite their two years advantage, could not reach the grade-8 experimental groups' level. Considering the summarized results of the groups, the experimental groups completed the test with a 48.20% score, while the control groups with 19.37%, making the difference between the two group categories significant using Mann-Whitney tests (p = 0,0000) (Figure 2).

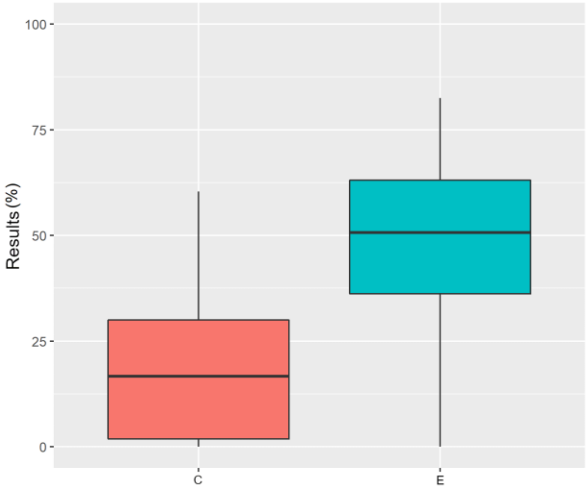


Figure 2: The distribution of the control (C) and experimental (E) groups' percentage results based on the results of the delayed post-test.

In the second part of the analysis, based on the examination of the [T1] thesis pre-tests, the grade-7 students – reflecting the lack of prior knowledge – solved the tasks with minimal points, resulting in an overall score of 5.52%. Compared to this, the grade-10 students achieved significantly better results in all parts of the tests, except for four tasks. Analyzing the overall test, the grade-10 students completed it with a score of 18.75%, which also shows a significant difference (p = 0.0000) (Figure 1, left part). However, it is important to point out that the grade-10 students achieved this difference with minimal scores in the section involving formula creation tasks (below 8.00%), compared to the grade-7 students' 0.00% scores.

Based on the results of the thesis, the students who studied with the Sprego methodology were significantly better on the delayed post-test than the groups who followed traditional approaches. Furthermore, the pre-tests written before the spreadsheet management topic indicate that grade-10 students who studied the topic for two years beforehand with traditional methods could only achieve better scores with minimal differences compared to the grade-7 students who had no prior experience. Based on the results presented in this section, the [T2] thesis is proved. The publication of the results is expected in the near future.

2.3. [T3] First-year undergraduates in computer science tertiary education cannot build on their text-management knowledge acquired in secondary education.

The students who complete secondary education and take computer science graduation exams either at an intermediate or advanced level are considered – based on their fulfillment of the requirements – to have a firm knowledge of text management upon which they can build effectively (Biró et al., 2015; Csapó, 2017; Csernoch et al., 2015). To test this knowledge, we examined the document analysis and error recognition skills of first year undergraduates in computer science tertiary education. We studied the students' abilities to recognize connections between error occurrences commonly found in erroneous text documents. In other words, we were interested in whether students possessed the required algorithmic and computational thinking skills to recognize the connections between similar structural parts of a document. As the next step of our research we compared the outcomes with the students' computer science graduation exam results. Therefore, we separated the analysis of the thesis into two parts:

1. The analysis of the rate at which students recognized the connection between similar structural parts of the document provided.
2. Comparing the results of the first step with the students' results on their computer science graduation exam at intermediate level.

2.3.1. The samples investigated and the phases of the research

We analyzed the document analysis and error recognition abilities of undergraduates who enrolled at the University of Debrecen, Faculty of Informatics in 2014 on Business Informatics BSc (BI) and Computer Science BSc (CS) majors. The majority of students applied to this institution right after they had finished secondary education and successfully completed their computer science graduation exams at intermediate or advanced level. Regarding

the size of the sample, the BI group consisted of 86 undergraduates, while the CS group was made up of 117 students. Considering both groups the overall size of the sample was N = 203. During our research we were interested in the knowledge the students had developed prior to tertiary education; therefore, our work did not include educational activity.

2.3.2. Research methods, strategies

During the analysis of the thesis, we relied on the TAaAS project for data collection. The undergraduates completed the tests during the first week of their tertiary education, of which we focused on the part involving error recognition and categorization in text management. During the testing phrase, the undergraduates had to sort marked erroneous parts of a sample document into the correct error categories. For processing the data, we created a database, but in this case – based on the particularities of the tests – we had to work with a minimal number of items. We assumed that the students had insufficient knowledge of the terminology; therefore, we focused on whether students could sort similar error occurrences into the same categories, regardless of their correctness. Based on the two parts of the research, we followed the steps below during the analysis:

1. The recognition of the connections between the members of error pairs belonging in the same – layout – error category.
2. Comparing the results of the tests with the overall results of the students' computer science graduation exams at intermediate level.

During the analysis of the data we worked in a spreadsheet management environment, using Microsoft Excel.

2.3.3. Analyzing the thesis

Based on the results of the first part of the analysis, 34.88% of the BI, and 38.46% of the CS group (summarized: 36.95%) saw the connection between members of one error-pair. Considering both error pairs, 18.60% of the BI, and 17.95% of the CS group (summarized: 18.23%) saw the connections between its members. The examination of the percentage results shows that a similar proportion of the two groups recognized the connections between related structural elements of a document. If we also include the marking of the correct error categories (as the task on the test sheets required), then only 1.97% of the students could complete the task for both error-pairs.

Regarding the results in the second part of this thesis, the BI group's score shows a 55.40% difference between recognizing the connections in both error pairs and their average results on the computer science graduation exams at

intermediate level. Considering the CS group, this difference is 64.05%. Analyzing the results of the whole sample (18.23%) a 60.50% difference is found compared to the overall score of the computer science graduation exams at intermediate level. Note that recognizing the connections only required the students to mark both members in the same category regardless of them being correct or not (Figure 3).

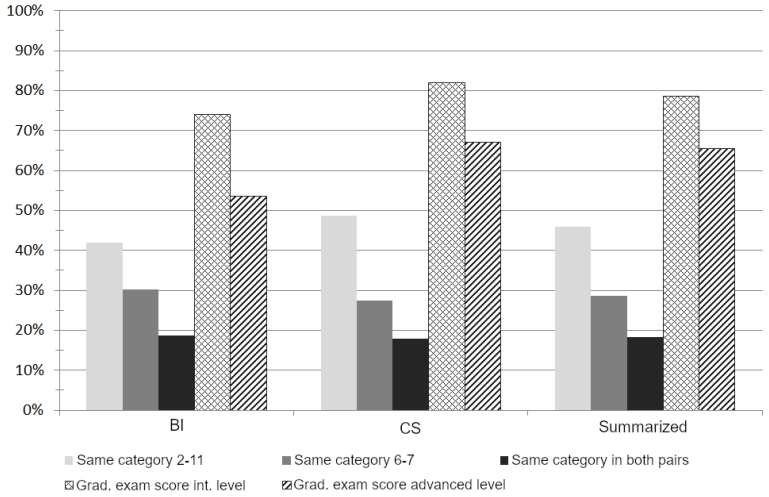


Figure 3: The results of the groups (BI: Business Informatics and CS: Computer Science) in recognizing the connections between the members of error pairs and on their computer science graduation exams at intermediate and advanced levels.

Based on the results, less than half of the undergraduates recognized the connections even in one error pair. Even though the groups achieved a differing score on their computer science graduation exams (Figure 3), both groups completed the tests at a similar level. We can conclude that the students were not able to recognize the connections between similar structural parts of a document, and that their scores on the computer science graduation exams did not reflect the low scores on the tests. As the lack of this skill makes the creation of well-formatted documents impossible, the [T3] thesis is proved (Csapó, 2017).

2.4. [T4] Teaching programming using event-action based visual programming develops the students' algorithmic skills more effectively than using block-based languages developed for education.

The practice of programming education applies a wide range of tools for teaching the topic, of which the EPLs (Educational Programming Languages) developed exclusively for educational use receive special attention. The use of EPLs and visual programming methods in education is not a novel approach; however, based on prior measurements (Biró et al., 2015; Csernoch et al., 2015), their efficacy is questionable. During our work we have taught and tested the effectiveness of two visual programming environments (Construct 3 and Scratch) with student groups in a local high school. We were interested in how the use of differing programming environments help develop students' algorithmic and computational thinking skills (Hubwieser, 2004).

2.4.1. The samples investigated and the phases of the research

The analysis of the effectiveness of using different programming tools for developing students' algorithmic and computational thinking skills was carried out in two grade-8 groups in a local high school. The students took part in a 6-year training program during which they did not encounter the topic in high school. The experimental group (N = 14) solved the problems in the topic using the Construct 3 environment, while the control group (N = 14) used the Scratch program. During the teaching of both groups we ensured they progressed at a similar pace in the topic. We also focused on the fact that despite the particularities of the differing environments, the students solved the same, programming-oriented tasks adapted for their work environments. The experimental group studied the topic for 18 weeks with one lesson per week, and the control group for 17 weeks with the same schedule.

2.4.2. Research methods, strategies

The data collection was carried out using pre-tests right before, and post-tests after, the topic, in printed form and using the same test sheets in both test cases. In the composition of the test, we used tasks developed for measuring the students' algorithmic and computational thinking skills (ELTE IK T@T Labor, 2019; Pohl & Dagienė, 2019). During the examination and selection of these tasks we focused on having them collect data on the algorithmic skills from multiple perspectives. For storing and analyzing the data we – continuing our previous practice – created a database. The particularities of

the tasks often resulted in a small array of student answers; therefore, the disassembly of answers into items was limited. To analyze the effectiveness of different tools for developing the students' algorithmic skills we examined the differences between the post-tests of the experimental and control groups. We analyzed the data using the R software package, and the significance of differences between the groups – considering the size of the sample – using Fisher's exact test.

2.4.3. Analyzing the thesis

Based on the examination of the pre-test the experimental and the control groups showed no difference in terms of student knowledge level prior to the learning process. The analysis of the post-test conducted at the end of the programming topic shows that the experimental and control groups completed the tests with similar results, providing correct answers in a high proportion. Based on the solutions we received to the problems, no significant difference could be found between the two groups in either of the tasks. This is also true for the task about graph usage showing the biggest difference in the test (five students from the experimental group and one from the control group solved it correctly, $p = 0.0730$). We would like to point out that with this task more students chose the seemingly right, distractor answer in the post-test than in the pre-test without changing the proportion of the correct answers (Csernoch et al., 2015). Note that we could not complete the significance analysis in two cases. One of the tasks was solved correctly by 100% of the students, and in another task we separated the answers from the rest of the data due its variation possibilities. In these cases, we could not apply Fisher's exact test.

Considering the results of the tasks designed to measure the algorithmic skills of students, since we could not find any significant difference between the groups who used differing programming tools, the [T4] thesis is rejected (Hubwieser, 2004). The publication of the results is expected in the near future.

3. Future perspectives of the research

Based on the positive results we received during our current research, we plan to conduct further analysis on the effectiveness of the Sprego methodology on a larger student sample. Following on, future research is required to examine the efficacy of the Sprego applications in supporting the learning processes. We also plan to analyze the effectiveness of the Error Recognition

Model methodology for teaching text management in secondary ICT education. Furthermore, we aim to expand our research regarding the effectiveness of the tools used in programming education considering the students' algorithmic skills. This analysis requires a more populated student sample and the redesign of the test sheets to introduce a larger scale of more varied student answers, a larger number of analyzed programming tools and further tested skills.

4. References

ACM & IEEE (2013). Computer Science Curricula: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM Press, and IEEE Computer Society Press, New York. DOI=http://dx.doi.org/10.1145/2534860.

Bell, T. & Newton, H. (2013). Unplugging Computer Science. D. M. Kadijevich, C. Angeli, & C. Schulte, Improving Computer Science Education, pp. 75–90. Routledge.

Ben-Ari, M. (1999). Bricolage Forever!. PPIG 1999, 11th Annual Workshop, University of Leeds: Computer-Based Learning Unit, UK, 5–7 January, 1999.. http://www.ppig.org/papers/11th-benari.pdf. Retrieved on: 2016.12.01.

Ben-Ari, M. & Yeshno, T. (2006). Conceptual models of software artifacts. Interacting with Computers, 2006, 18(6), pp. 1336–1350. DOI=http://doi.org/10.1016/j.intcom.2006.03.005.

Biró, P., Csernoch, M., Máth, J. & Abari, K. (2015). Measuring the level of algorithmic skills at the end of secondary education in Hungary. Procedia - Social And Behavioral Sciences, 2015, 176, pp. 876–883.

Champagne, A. B., Gunstone, R. F. & Klopfer, L. E. (1983). Naive knowledge and science learning. Research in Science and Technological Education

Csapó, G. (2017). The computer science graduation exam and the text-management culture of computer science undergraduates. In Hungarian: Az informatika érettségi és az informatikushallgatók szövegkezelési kultúrája. Válogatott tanulmányok a pedagógiai elmélet és szakmódszertanok köréből. http://www.irisro.org/pedagogia2017januar/85CsapoGabor.pdf Retrieved on: 2019.09.11.

Csapó, G. (2019). Placing event-action based visual programming in the process of computer science education. Acta Polytechnica Hungarica, 2019, 16(2), Cognitive Infocommunications, pp. 35–57. http://www.uni-obuda.hu/journal/Csapo_89.pdf. Retrieved on: 2019.08.14. DOI=http://doi.org/10.12700/APH.16.2.2019.2.3.

Csapó, G. & Sebestyén, K. (2017). Educational Software for the Sprego Method. The Turkish Online Journal of Educational Technology, INTE 2017 October, pp. 986–999. http://www.tojet.net/special/2017_10_1.pdf. Retrieved on: 2019.08.14.

Csapó G., Csernoch M. & Abari K. (2019). Sprego: Case Study on the Effectiveness of Teaching Spreadsheet Management with Schema Construction. Education and Information Technologies. Accepted.

Csernoch, M. (1997). Methodological Questions of Teaching Word Processing. 3rd International Conference on Applied Informatics, Eger-Noszvaj, Hungary, pp. 375–382.

Csernoch, M. (2009). Teaching word processing – the theory behind. Teaching Mathematics and Computer Science, 2009, 1, pp. 119–137.

Csernoch, M. (2010). Teaching word processing – the practice. Teaching Mathematics and Computer Science, 2010, 8(2), pp. 247–262.

Csernoch, M. (2014). Programming with Spreadsheet Functions: Sprego. In Hungarian: Programozás táblázatkezelő függvényekkel – Sprego. Műszaki Könyvkiadó, Budapest.

Csernoch, M. (2017). Thinking Fast and Slow in Computer Problem Solving. Journal of Software Engineering and Applications, 2017, 10(1). http://file.scirp.org/pdf/JSEA_2017012315324696.pdf. Retrieved on: 2017.07.08.

Csernoch, M. & Biró, P. (2014). Sprego programming. Spreadsheets in Education (eJSiE), 2014, 8(1). http://epublications.bond.edu.au/cgi/viewcontent.cgi?article=1175&context=ejsie. Retrieved on: 2015.04.03

Csernoch, M. & Biró, P. (2015). Computer problem-solving. In Hungarian: Számítógépes problémamegoldás. TMT, Tudományos és Műszaki Tájékoztatás, Könyvtár- és információtudományi szakfolyóirat, 2015, 62(3), pp. 86–94.

Csernoch, M. & Biró, P. (2016). Teaching methods are erroneous: approaches which lead to erroneous end-user computing. European Spreadsheet Risk Interest Group Conference, EuSpRIG2016, 2016, London. http://www.eusprig.org/mcsernoch-2016.pdf. Retrieved on: 2016.07.21.

Csernoch, M., Biró, P., Máth, J. & Abari, K. (2015). Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments. Informatics in Education, 2015, 14(2), pp. 175–197. DOI=http://doi.org/10.15388/infedu.2015.11.

ELTE IK T@T Labor (2019). Archivum | e-Hód. http://e-hod.elte.hu/archivum/. Retrieved on: 2019.08.04.

EuSpRIG, European Spreadsheet Risk Interest Group (2019). EuSpRIG Horror Stories. http://www.eusprig.org/horror-stories.htm. Retrieved on: 2019.06.28.

Fincher, S., Cooper, S., Kölling, M. & Maloney, J. (2010). Comparing alice, greenfoot & scratch. 41st ACM technical symposium on Computer science education, ACM, 2010, pp. 192–193.

Fowler, A., Fristce, T. & MacLauren, M. (2012). Kodu Game Lab: a programming environment. The Computer Games Journal, 2012, 1(1), pp. 17–28.

Garrett, N. (2015). Textbooks for Responsible Data Analysis in Excel. Journal of Education for Business, 2015, 90(4), pp. 169-174. DOI=http://doi.org/10.1080/08832323.2015.1007908.

Gould, J. (1975). Some psychological evidence on how people debug computer programs. International Journal of Man-Machine Studies, 1975, (7)1, pp. 151–182.

Gould, J. & Drongowski, P. (1974). An exploratory study of computer program debugging. Human Factors, 1974, 16(3), pp. 258–277.

Hubwieser, P. (2004). Functional Modelling in Secondary Schools Using Spreadsheets. Education and Information Technologies, 2004, 9(2), pp. 175–183. DOI=http://doi.org/10.1023/B:EAIT.0000027929.91773.ab.

Jerinic, L. (2014). Teaching Introductory Programming Agent-based Approach with Pedagogical Patterns for Learning by Mistake. International Journal of Advanced Computer Science and Applications, 2014, (5)6.

Gulácsi, Á. & Dienes, N. (2018). 3D Software Environment for Educational Sprego Programming. The Turkish Online Journal of Educational Technology, INTE 2018 November (2), pp. 837–844. http://www.tojet.net/special/2018_12_3.pdf. Retrieved on: 2019.01.21.

Klassner, F. & Anderson, S. D. (2003). Lego MindStorms: Not just for K-12 anymore, IEEE Robotics & Automation Magazine, 2003, 10(2), pp. 12–18.

ICAEW (2016). Spreadsheet competency framework: A structure for classifying spreadsheet ability in finance professionals. ICAEW, London. http://www.icaew.com/-/media/corporate/files/technical/information-technology/it-faculty/spreadsheet-competency-framework.ashx. Retrieved on: 2019.08.14.

Kahneman, D. (2011). Thinking, Fast and Slow. Farrar, Straus and Giroux, New York.

Kruck, S. E., Maher, J. J. & Barkhi R. (2003). Framework for Cognitive Skill Acquisition and Spreadsheet Training. Journal of Organizational and End User Computing, 2003, 15(1), pp. 20–37.

Merriënboer, J.J.G. van & Sweller, J. (2005). Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions. Educational Psychology Review, 2005, 17(2), pp. 147–177. DOI=http://doi.org/10.1007/s10648-005-3951-0.

Panko, R. (2013). The Cognitive Science of Spreadsheet Errors: Why Thinking is Bad. 46th Hawaii International Conference on System Sciences, Maui, 2013, pp. 4013–4022. IEEE.

Panko, R. (2016). What We Don't Know About Spreadsheet Errors Today: The Facts, Why We Don't Believe Them, and What We Need to Do. arXiv preprint. http://arxiv.org/abs/1602.02601. Retrieved on: 2016.07.21.

Panko, R. & Port, D. (2013). End User Computing: The Dark Matter (and Dark Energy) of Corporate It. Journal of Organizational and End User Computing, 2013, 25(3), pp. 1–19.

Papadakis, S., Kalogiannakis, M., Orfanakis, V. & Zaranis, N. (2014). Novice programming environments. Scratch & app inventor: a first comparison. Workshop on Interaction Design in Educational Environments, ACM, 2014.

Pohl, W. & Dagienė, V. (2019). What is Bebras | www.bebras.org. https://www.bebras.org/. Retrieved on: 2019.08.04.

Pólya, G. (1954). How To Solve It. A New Aspect of Mathematical Method. 2nd edition, 1957. Princeton University Press, Princeton, New Jersey.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: programming for all, Communications of the ACM, 2009, 52(11), pp. 60–67.

Sebestyén, K., Csapó, G. & Csernoch, M. (2018). Visualising Sprego Inequality Problems With 2D Representations. The Turkish Online Journal of Educational Technology, INTE 2018 November (2), pp. 888–898. http://www.tojet.net/special/2018_12_3.pdf. Retrieved on: 2019.08.14.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 2006, 49(3), pp. 33–35.

UNIVERSITY of DEBRECEN

UNIVERSITY AND NATIONAL LIBRARY
UNIVERSITY OF DEBRECEN
H-4002 Egyetem tér 1, Debrecen
Phone: +3652/410-443, email: publikaciok@lib.unideb.hu

Registry number: DEENK/323/2019.PL
Subject: PhD Publikációs Lista

Candidate: Gábor Csapó
Neptun ID: QVGE6Y
Doctoral School: Doctoral School of Informatics
MTMT ID: 10052249

List of publications related to the dissertation

Hungarian book chapters (1)

1. **Csapó, G.**: Az informatika érettségi és az informatikushallgatók szövegkezelési kultúrája. In: Válogatott tanulmányok a pedagógiai elmélet és szakmódszertanok köréből. Szerk.: Karlovitz János Tibor, International Research Institute s.r.o., Komárno, 387-394, 2017. ISBN: 9788089691401

Foreign language scientific articles in Hungarian journals (1)

2. **Csapó, G.**: Placing Event-Action-based Visual Programming in the Process of Computer Science Education.
APH. 16 (2), 35-57, 2019. EISSN: 1785-8860.
DOI: http://dx.doi.org/10.12700/APH.16.2.2019.2.3
IF: 1.286 (2018)

Foreign language scientific articles in international journals (1)

3. **Csapó, G.**, Csernoch, M., Abari, K.: Sprego: Case Study on the Effectiveness of Teaching Spreadsheet Management with Schema Construction.
Educ. Inf. Technol. "Accepted by Publisher", 1-16, 2019. ISSN: 1360-2357.

Hungarian conference proceedings (2)

4. **Csapó, G.**, Sebestyén, K.: Oktatóprogram a Sprego táblázatkezelő módszerhez. In: INFODIDACT 2015. Szerk.: Szlávi Péter, Zsakó László, Webdidaktika Alapítvány, Zamárdi, 1-18, 2015. ISBN: 9789631238921
5. **Csapó, G.**: Vizuális programozás oktatása középiskolákban. In: INFODIDACT 2014. Szerk.: Szlávi Péter, Zsakó László, Webdidaktika Alapítvány, Zamárdi, 1-10, 2014. ISBN: 9789631206272

UNIVERSITY of DEBRECEN

UNIVERSITY AND NATIONAL LIBRARY
UNIVERSITY OF DEBRECEN
H-4002 Egyetem tér 1, Debrecen
Phone: +3652/410-443, email: publikaciok@lib.unideb.hu

Foreign language conference proceedings (2)

6. **Csapó, G.**, Sebestyén, K.: Educational software for the Sprego method.
   Turk. Online J. Educ. Technol. Special Issue for INTE 2017, October, 986-999, 2017. EISSN: 2146-7242.

7. **Csapó, G.**: Sprego virtual collaboration space.
   In: 8th IEEE International Conference on Cognitive Infocommunications: CogInfoCom 2017 : Proceedings : September 11-14, 2017 Debrecen, Hungary, IEEE Computer Society, Piscataway, 137-142, 2017. ISBN: 9781538612644

Foreign language abstracts (1)

8. **Csapó, G.**: Sprego virtual collaboration space: improvement guidelines for the MaxWhere seminar system.
   In: 8th IEEE International Conference on Cognitive Infocommunications : CogInfoCom 2017 : Proceedings : September 11-14, 2017 Debrecen, Hungary, IEEE Computer Society, Piscataway, 143-144, 2017. ISBN: 9781538612644

List of other publications

Foreign language conference proceedings (2)

9. Sebestyén, K., **Csapó, G.**, Csernoch, M.: Introduction to Algorithmic Based Data Management in Spreadsheet Environment.
   Turk. Online J. Educ. Technol. "Accepted by Publisher", 1-8, 2019. EISSN: 1303-6521.

10. Sebestyén, K., **Csapó, G.**, Csernoch, M.: Visualising Sprego Inequality Problems With 2D Representations.
    Turk. Online J. Educ. Technol. Special Issue for INTE-ITICAM-IDEC 2018, 888-898, 2018. EISSN: 1303-6521.

Total IF of journals (all publications): 1,286
Total IF of journals (publications related to the dissertation): 1,286

The Candidate's publication data submitted to the iDEa Tudóstér have been validated by DEENK on the basis of the Journal Citation Report (Impact Factor) database.

18 September, 2019