

Metóda pre sociálne programovanie a posudzovanie kódu

Bc. Michal Tomlein

Vedúci práce: Mgr. Jozef Tvarožek, PhD.

Problémová oblasť

- ✦ Výučba programovania
 - ✦ Učiteľ nemôže poskytovať spätnú väzbu všetkým študentom naraz
- ✦ Posudzovanie kódu rovesníkmi (PCR)
 - ✦ Ako integrovať do procesu výučby
 - ✦ Ako vyberať posudzovateľa
- ✦ Sociálne prístupy a kolaborácia

Ciele

- ✦ Preskúmať možnosti aplikovania posudzovania kódu rovesníkmi ako formy spätnej väzby
- ✦ Vyberať vhodného posudzovateľa
- ✦ Analyzovať korelácie medzi osobnostnými črtami a schopnosťou posudzovať kód

Súvisiace práce

- ✦ Komerčné riešenia
 - ✦ *SmartBear CodeCollaborator*
 - ✦ *Atlassian Crucible*
 - ✦ *GitHub*
 - ✦ Asynchrónne, založené na VCS

Dashboard
WordFinder.WordFinder()
Similar code..

Collapse all comments
Show instructions

41 { super("Word Finder"); // call ...
automatically generated by Checkstyle
'{' should be on the previous line.
+ 0 / -0

45 - 48 setDefaultCloseOperation(EXIT_...
23 hours ago by elena_11 (2)
some comment
+ 1 / -0

37 /**
38 * Make a
39 */
40 public Word
41 {
42 super(
43
44 // cal
45 setDef
46
47 // cre
48 initCo
49
50 // cre
51 cont =

Section #1, Team #1
@ Kyle Ryan
CodeFile.cpp
kim Carter
@ Anu Agrawal

1 // ConvertIntToChar.cpp : Defines the entry point for the console application.
2 //
3
4 #include "stdafx.h"
5 using namespace std;
6 #include <string>
7 #include <set>
8
9 char* convertIntToChar(int num);
10 int convertCharToInt(char* ch);
11
12 int _tmain(int argc, _TCHAR* argv[])
13 {
14 string s1("Pawan");
15 set<int> s;
16 s.insert(1);
17 s.insert(1);
18 int num = 1234567;
19 char *ch = convertIntToChar(num);
20 cout<<ch;
21 return 0;
22 }
23
24 char* convertIntToChar(int num)
25 {
26 char ch[10];
27 int i = 0;
28 while(num > 0)
29 {
30
31 int digit = num %10;
32 num = num/10;
33 ch[i] = (char)digit;
34 i++;
35 }
36 ch[i] = num;
37 ch[++i] = '\0';
38 // reverse ch
39 return ch;
40 }
41

Post Comment
Location
Severity [choose one]
Type Choose a type...
Post Comment Clear

Kyle Ryan
2 posts
? Line #14-17: These variables are never used
0 comments
Anu Agrawal
2 posts
? Line #24: Needs comments for the method
0 comments
Kyle Ryan
2 posts
? Line #24: Comments from the method are missing
0 comments
Anu Agrawal
2 posts

Caesar

OSBLE

Example 3:

Identify the bugs in this code.

```
public class Buggy {
    static String name;

    public static void main(String[] args) {
        int n = name.length();

        System.out.println(n);

        name += "; The end.";
        System.out.println(name);
    }
}
```

Example 4:

This code is supposed to find the word "target" in

```
String A[n], *name; // initialized to some wo
int i=0;
while (!A[i].equals("target"))
    i++;
```

Comments

In response to Louis Savitz's comment:
the local variable "i" will contain the index of the word at which the loop

Tag: General

In response to Rassan Walker's comment:
Oh, yes, I overlooked that. Thanks for pointing out my mistake!

Tag: General

In response to Rassan Walker's comment:
although this loop does through

Tag: General

Classroom Salon

- ✦ Konverzácia
- ✦ Asynchrónne
- ✦ Kód sa musí vložiť
- ✦ Slabá integrácia s IDE

Posudzovanie kódu

- ✦ Overenie kvality kódu
- ✦ Forma spätnej väzby – zvyčajne asynchrónna (oneskorená)
- ✦ Vyžadujú sa schopnosti:
 - ✦ Porozumieť kódu druhého
 - ✦ Identifikovať problémy
 - ✦ Formulovať pripomienky

Posudzovanie kódu je forma
pomáhania inému s kódom

Výber posudzovateľa

- ✦ Odporúčanie ľudí ľuďom
- ✦ Súčasné práce:
 - ✦ podľa štýlu učenia a znalostí tém
 - ✦ rôzne prístupy neaplikovateľné na kontext výučby

Hodnotenie osobností

- ✦ Štýly učenia (ILS)
- ✦ Osobnostný model Big Five
 - ✦ aplikovateľnosť získaných poznatkov
 - ✦ najčastejšie sa vyskytujúci
 - ✦ rozdielne hodnoty poukazujú na rozdielne spôsoby spracovávania informácií
 - ✦ na fakulte širšie využitie

Myšlienka

- ✦ Programovanie v online vývojovom prostredí
- ✦ Živé synchrónne posudzovanie
 - ✦ Vhodné pre kontext výučby
 - ✦ Netradičné a zaujímavé
- ✦ Živá diskusia o kóde
 - ✦ Možnosť pomôcť posudkom k riešeniu

Prvý prototyp

Firefox

127.0.0.1:7000/fiit/spp

Google

Peoplia

5 Histogram

2 Najčastejší znak

0 Cézarova šifra

0 Dešifrovanie

0 Tajná správa

Bc. Michal Tomlein

Napište funkciu `char most_frequent_char(const char *str)`, ktorá vráti najčastejšie sa vyskytujúce písmeno malej anglickej abecedy v zadanom reťazci. Predpokladajte, že môžete využiť funkciu `histogram`

sifry-2.c

Štandardný vstup

Štandardný výstup

```
1 // sifry-2.c -- Sifry - Najcastejsi znak
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int *histogram(const char *str);
7
8 char most_frequent_char(const char *str)
9 {
10     int count[256], i, max;
11
12     // Tu napis svoj kod
13     for (i = 0; i < 256; i++)
14         count[i] = 0;
15     for (i = 0; str[i]; i++)
16         if (str[i] >= 'a' && str[i] <= 'z')
17             count[str[i]]++;
18     for (max = 0, i = 1; i < 256; i++)
19         if (count[max] < count[i])
20             max = i;
21     return max > 0 ? (char)max : '?';
22 }
23
24 int main()
25 {
26     printf("%c\n", most_frequent_char("")); // ?
27     printf("%c\n", most_frequent_char("hell0000000")); // l
28     printf("%c\n", most_frequent_char("aardvark")); // a
29     return 0;
30 }
```

Kompilácia ... OK

sifry-2.c: In function 'most_frequent_char':
sifry-2.c:17:7: warning: array subscript has type 'char'

Kompilovať

Spustiť program

Odoslať riešenie

Otázky / komentáre:

Michal: Čo je to subscript?

Jožko: To je výraz medzi [], ktorým prístupuješ k prvku poľa.

Jožko: Stačí, keď ho pretypuješ.

Michal: Na char?

Jožko: Áno.

Tvoja otázka / komentár:

Pošli

Napište funkciu `int *histogram(const char *str)`, ktorá vytvorí histogram výskytov malých písmen anglickej abecedy v reťazci `str`.

sifry-1.c

Štandardný vstup

Štandardný výstup

```
1 // sifry-1.c -- Sifry - Histogram
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int *histogram(const char *str)
7 {
8     // hist je pole 26 intov s hodnotou 0
9     int *hist = (int *)calloc(26, sizeof(int));
10
11     int i;
12     for (i = 0; str[i]; ++i) {
13         if (str[i] >= 'a' && str[i] <= 'z')
14             hist[str[i] - 'a'] ++ 1;
15     }
16
17     return hist;
18 }
19
20 int main()
21 {
22     int *hist = histogram("hello");
23     free(hist);
24     return 0;
25 }
```

Kompilácia ... CHYBA (kliknite na chyby nižšie)

sifry-1.c: In function 'histogram':
sifry-1.c:14:29: error: expected ';', before numeric constant

Pošli

Správy pre mňa:

Michal: ++ je unáry operátor, nemôžeš zaň napísať číslo.

Nová správa:

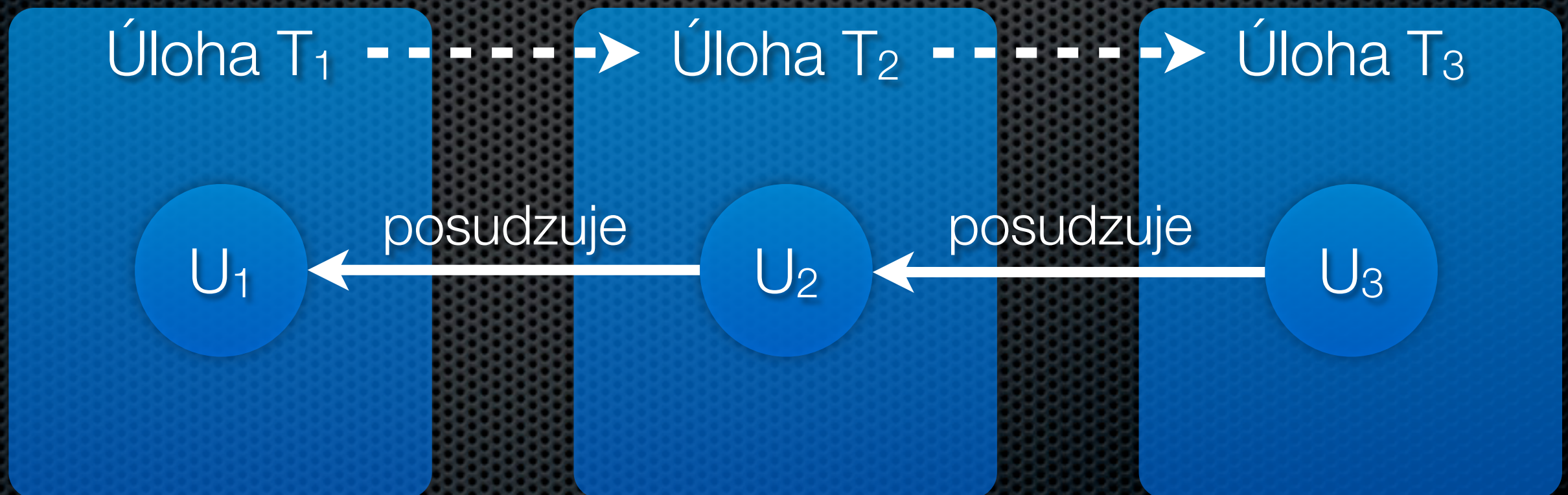
Pošli správu

Metóda

Živé posudzovanie prináša
problém opisovania

Obmedzenie opisovania

- ✦ **Posudzujú sa len vyriešené úlohy** — len ak posudzovateľ už danú úlohu vyriešil



Model používateľa

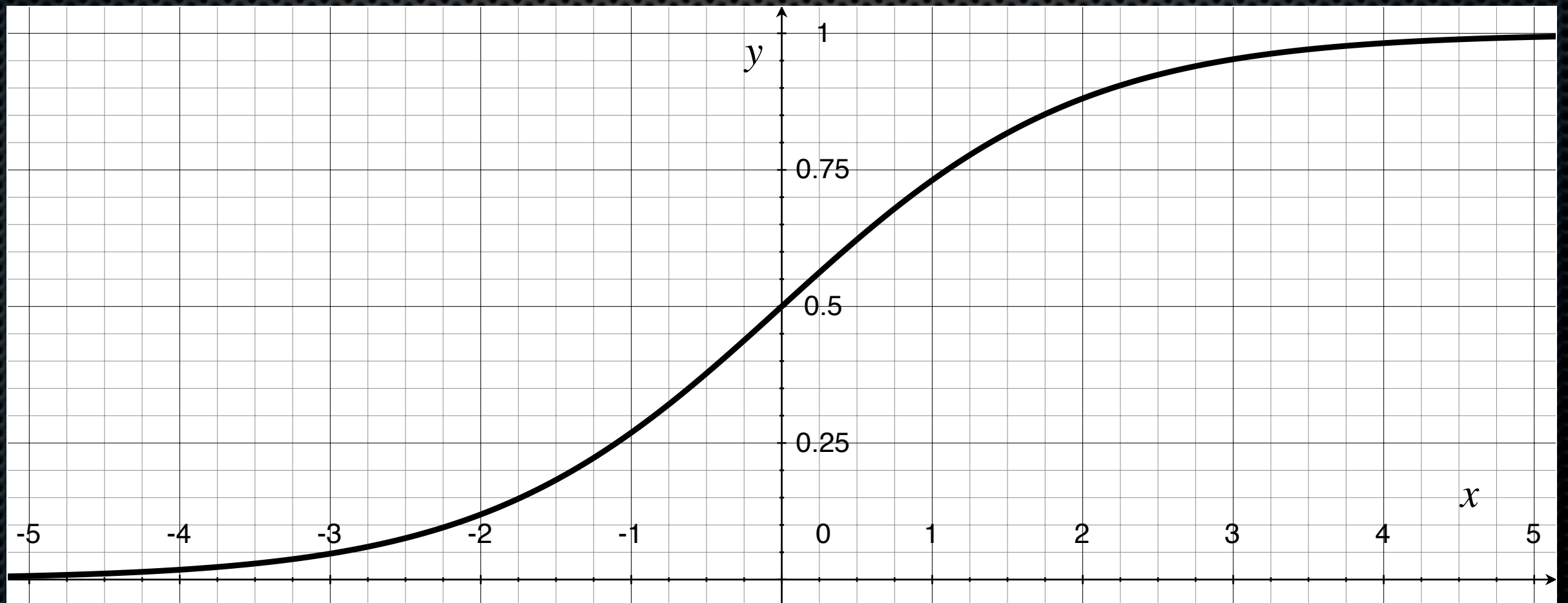
- ✦ Osobnostný model Big Five (extroverzia, ochota, svedomitosť, neuroticizmus, otvorenosť)
- ✦ Určujeme posudzovateľské schopnosti:
 - ✦ Poskytnúť pomoc/radu — *deliver*
 - ✦ Prijatť pomoc/radu — *receive*

Metóda vyberá
posudzovateľa s najväčšou
pravdepodobnosťou
úspechu spolupráce

Pravdepodobnosť úspechu spolupráce i a j

- Raschov model $e^{deliver_i - receive_j}$

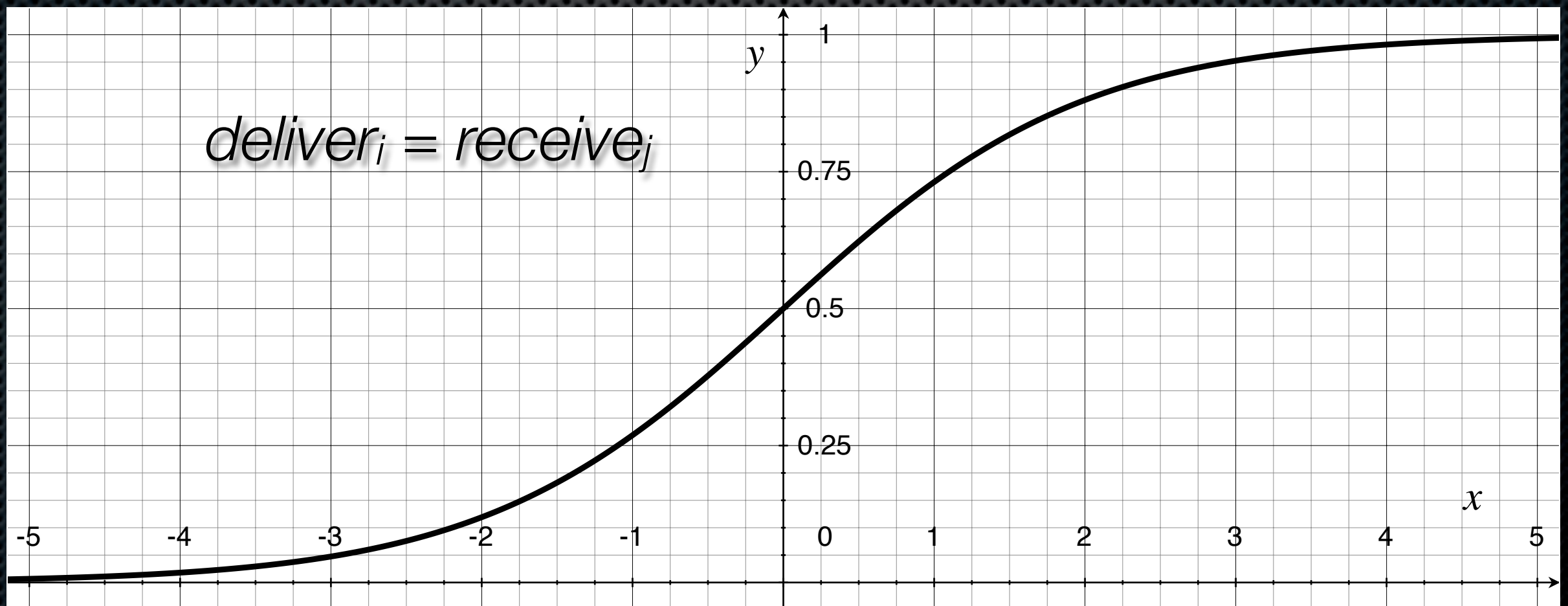
$$P(i, j) = \frac{e^{deliver_i - receive_j}}{1 + e^{deliver_i - receive_j}}$$



Pravdepodobnosť úspechu spolupráce i a j

- Raschov model $e^{deliver_i - receive_j}$

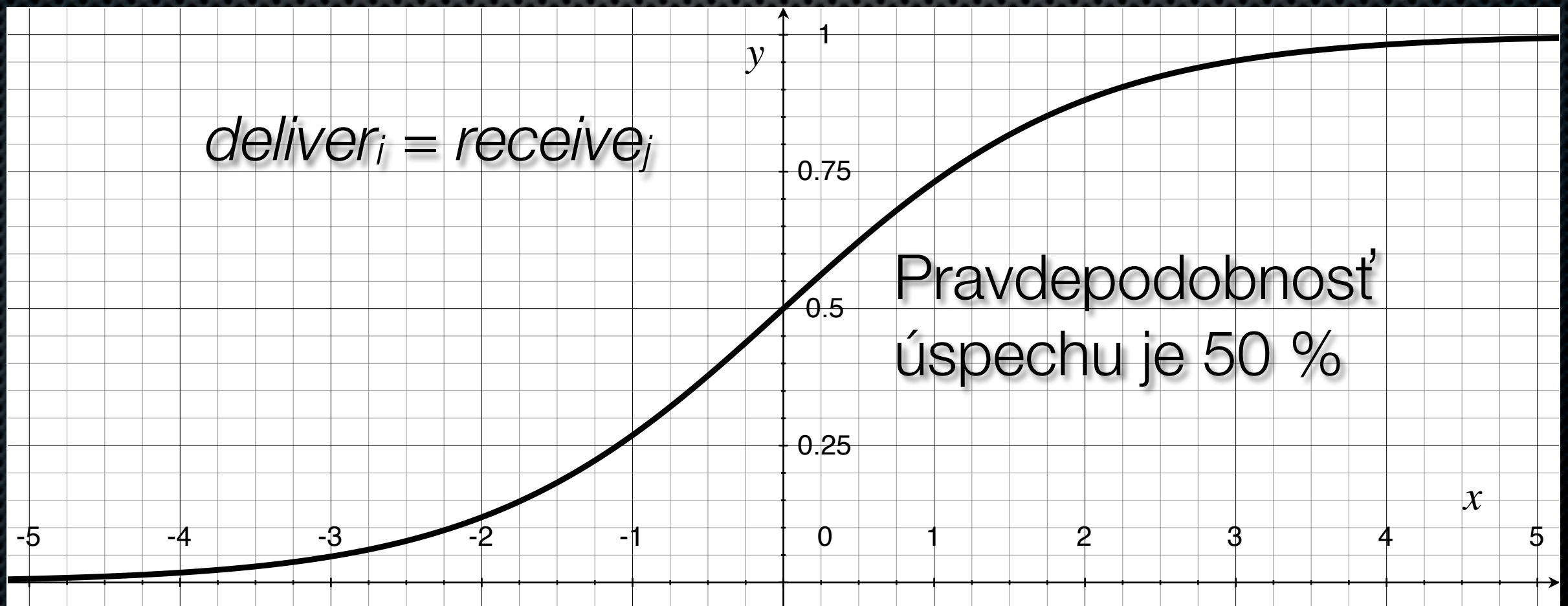
$$P(i, j) = \frac{e^{deliver_i - receive_j}}{1 + e^{deliver_i - receive_j}}$$



Pravdepodobnosť úspechu spolupráce i a j

- Raschov model $e^{deliver_i - receive_j}$

$$P(i, j) = \frac{e^{deliver_i - receive_j}}{1 + e^{deliver_i - receive_j}}$$



Metóda pracuje v 2 fázach

1. Kalibrácia modelu

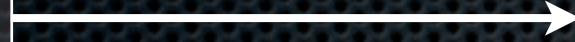
- Náhodný výber posudzovateľov

2. Aplikovanie modelu

- Výber na základe modelu

Model
používatele

Dotazník
Big Five



Model
uživatele

Dotazník
Big Five



Model
používatele'a

Náhodný výber posudzovateľa

Dotazník
Big Five



Model
používateľa

Náhodný výber posudzovateľa

Hodnotenie
úspešnosti
párov

Dotazník
Big Five



Model
používatele'a

Posudz. \ Použ.	U_1	U_2	U_3	U_4
U_1		1		0
U_2	0		1	1
U_3	1			0
U_4		1		

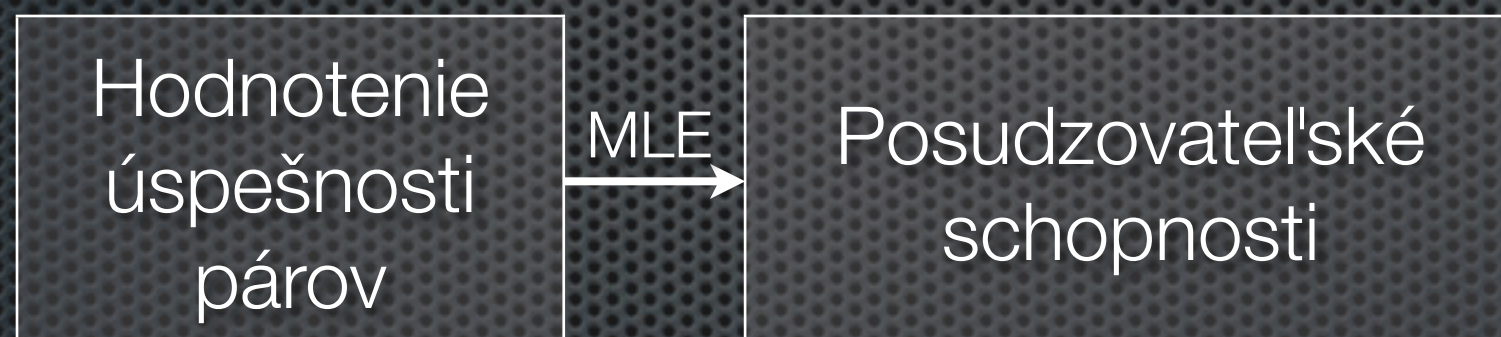
Náhodný výber posudzovateľa

Hodnotenie
úspešnosti
párov



Posudz. \ Použ.	U_1	U_2	U_3	U_4
U_1		1		0
U_2	0		1	1
U_3	1			0
U_4		1		

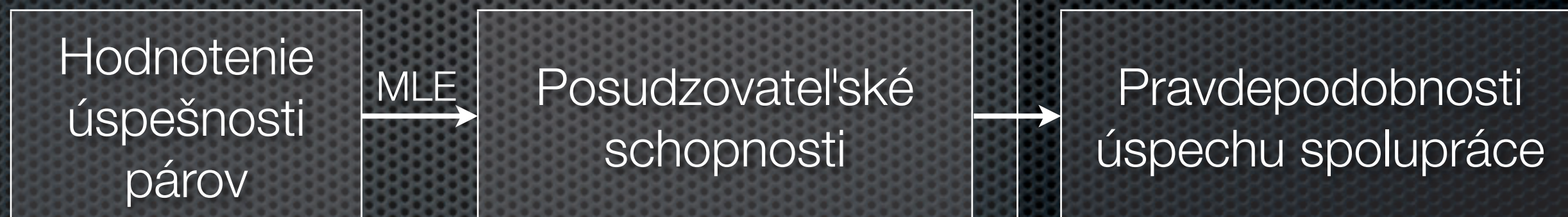
Náhodný výber posudzovateľa





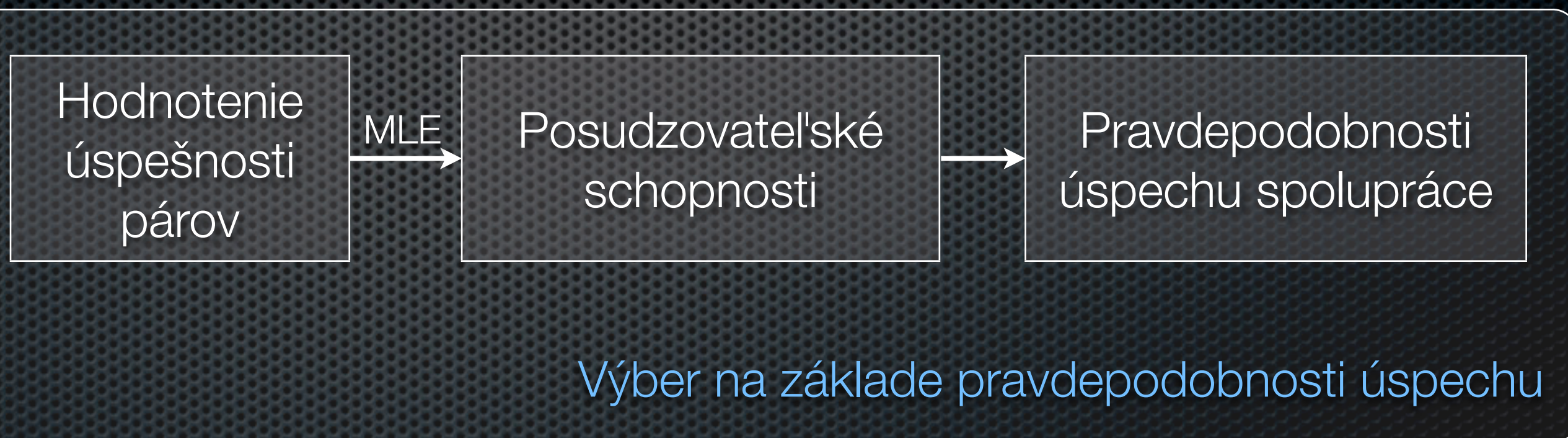
Posudz. \ Použ.	U_1	U_2	U_3	U_4
U_1		1		0
U_2	0		1	1
U_3	1			0
U_4		1		

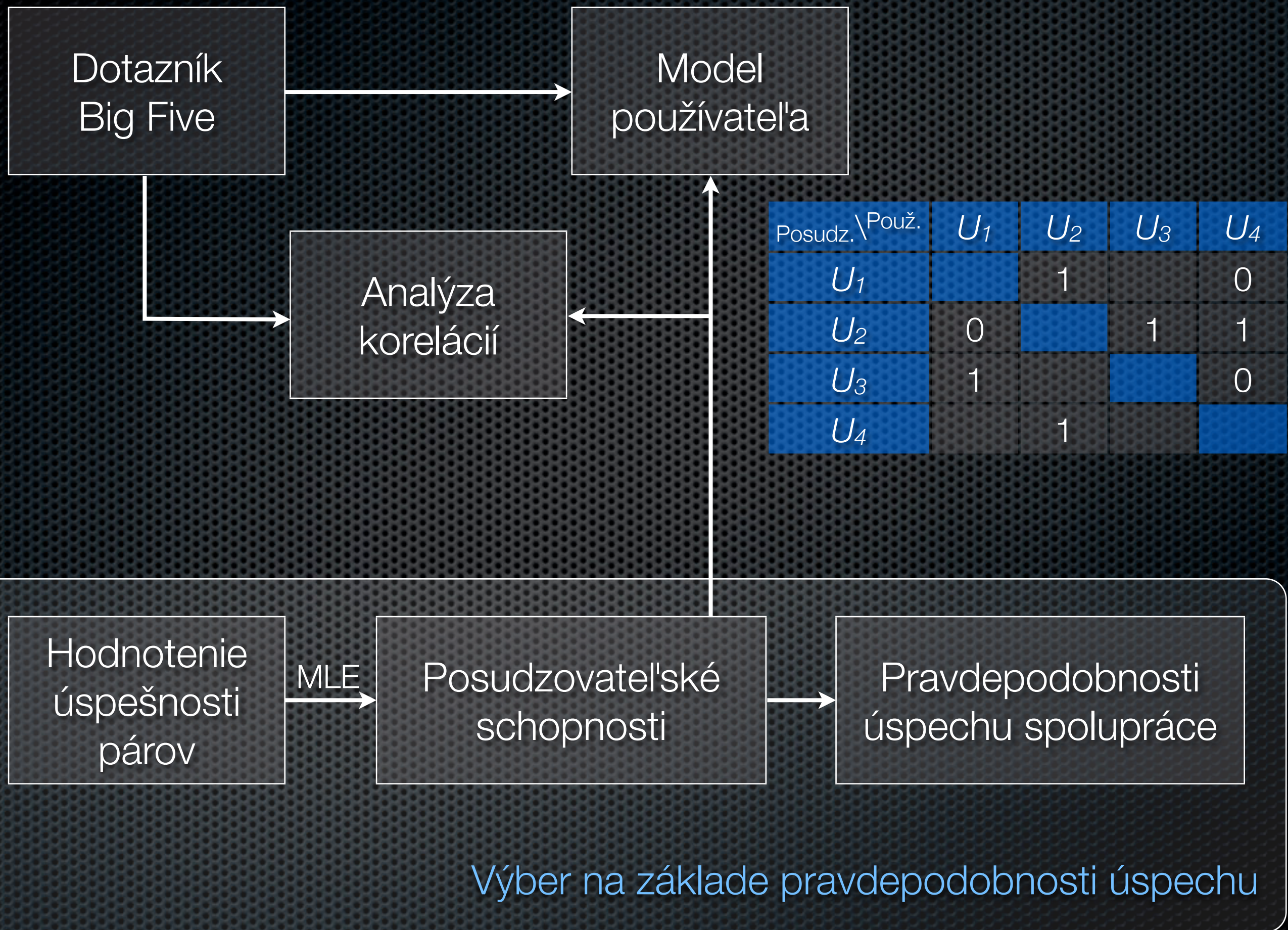
Náhodný výber posudzovateľa





Posudz. \ Použ.	U_1	U_2	U_3	U_4
U_1		1		0
U_2	0		1	1
U_3	1			0
U_4		1		





Dotazník
Big Five

Hodnotenie
úspešnosti
párov

MLE

Posudzovateľské
schopnosti

Pravdepodobnosti
úspechu spolupráce

Výber na základe pravdepodobnosti úspechu

Dotazník
Big Five

Posudzovateľské
schopnosti
z korelácií

Hodnotenie
úspešnosti
párov

MLE

Posudzovateľské
schopnosti

Pravdepodobnosti
úspechu spolupráce

Výber na základe pravdepodobnosti úspechu

Dotazník
Big Five

Posudzovateľské
schopnosti
z korelácií

90 %

Hodnotenie
úspešnosti
párov

MLE

Posudzovateľské
schopnosti

Pravdepodobnosti
úspechu spolupráce

10 %

Výber na základe pravdepodobnosti úspechu

Experimenty...

...alebo ako naplniť maticu

1. experiment

- ✦ 172 študentov doma v určenom čase
- ✦ 5 úloh
- ✦ Hypotéza: Existuje korelácia medzi osobnostnými črtami a schopnosťou poradiť alebo prijať radu.

Výsledok 1. experimentu

- ✦ 391 vzniknutých párov autor–posudzovateľ
- ✦ Vzniklo 11 dialógov (185 správ)
- ✦ Študenti zväčša pridelený kód ignorovali
- ✦ Problém pri menšom rozlíšení

Po 1. experimente

- ✦ Jednoúlohové rozhranie
- ✦ Žiadosti o posudok/radu
- ✦ Hypotéza: Ak študenti budú mať možnosť vedome požiadať o radu a ponúknuť radu, budú aktívnejší.

Druhý prototyp

FIIT STU – Procedurálne programovanie 2012/2013 (leto) | Peoplia

https://www.peoplia.eu/fiit/pp

Reader

Peoplia

1 Úloha 7-10 Úloha 7-2

Bc. Matus Tomlein

Napište program, ktorý zistí počet riadkov `N` v súbore `neusporiadane.txt`, alokuje v pamäti dva bloky po `N` položiek reálnych čísel a do prvého bloku načíta čísla zo súboru. Predpokladajte, že súbor obsahuje v každom riadku práve jedno reálne číslo. Predpokladajte, že súbor neobsahuje rovnaké číslo viackrát. Potom prekopírujte obsah z prvého bloku do druhého tak, aby čísla v druhom bloku boli vzostupne usporiadané. Nakoniec program zapíše obsah usporiadaného bloku do súboru `usporiadane.txt`. Program negeneruje žiaden výstup na štandardný výstup. Využite ukazateľovú aritmetiku.

Ukážka súboru `neusporiadane.txt`:

```
4.8
9.26
```

Tvoja otázka:
Ako načítam čísla zo vstupu?

Už netreba

Kamoši sa pýtajú:

uloha-7-1.cneusporiadane.txtŠtandardný výstup

```
1 // uloha-7-1.c -- Tyzden 7 - Uloha 1
2
3 #include <stdio.h>
4
5 int main()
6 {
7     // sem napis svoje riesenie
8
9     return 0;
10 }
```

KompilovaťSpustiť programOdoslať riešenie

Druhý prototyp

FIIT STU – Procedurálne programovanie 2012/2013 (leto) | Peoplia

https://www.peoplia.eu/fiit/pp

Reader

Peoplia

1 Úloha 8-10 Úloha 8-20 Úloha 8-3

Bc. Michal Tomlein

Napište funkciu `int strinsert(char *dst, int len, const char *src, int offset)`, ktorá do reťazca `dst` od pozície `offset` vloží kópiu reťazca `src`. Argument `len` určuje počet znakov vyhradených pre pole `dst` (vrátane ukončovacieho znaku `\0`). Ak nie je možné do vyhradeného miesta reťazec vložiť, funkcia nič nevykoná a vráti 1. Inak (ak je možné do vyhradeného miesta reťazec vložiť), vráti 0. Napr. pre `dst:totojeretazec`, `offset:6`, `src:druhy` volanie `strinsert(dst, 50, src, 6)` vráti 0 a v reťazci `dst` bude `totojeretazecdruhy`.

uloha-8-1.c

Štandardný vstup

Štandardný výstup

```
1 // uloha-8-1.c -- Tyzden 8 - Uloha 1
2
3 #include <stdio.h>
4
5 int strinsert(char *dst, int len, const char *src, int offset)
6 {
7     // sem napis svoje riesenie
8
9     return 1;
10 }
11
12 int main()
13 {
14     char buf[100];
15
16     sprintf(buf, "totojeretazec");
17     if (strinsert(buf, 100, "druhy", 6))
18         printf("Nepodarilo sa vlozit retazec.\n");
19     else
20         printf("%s", buf);
21     return 0;
22 }
```

Tvoja otázka:

Spýtaj sa kamoša

Kamoši sa pýtajú:

Matus (Úloha 7-1):
Ako načítam čísla zo vstupu?
Klikni a poraď

Kompilovať

Spustiť program

Odoslať riešenie

Druhý prototyp

FIIT STU – Procedurálne programovanie 2012/2013 (leto) | Peoplia

https://www.peoplia.eu/fiit/pp

Reader

Peoplia

1 Úloha 8-10 Úloha 8-20 Úloha 8-3

Bc. Michal Tomlein

Napište program, ktorý zistí počet riadkov `N` v súbore `neusporiadane.txt`, alokuje v pamäti dva bloky po `N` položiek reálnych čísel a do prvého bloku načíta čísla zo súboru. Predpokladajte, že súbor obsahuje v každom riadku práve jedno reálne číslo. Predpokladajte, že súbor neobsahuje rovnaké číslo viackrát. Potom prekopírujte obsah z prvého bloku do druhého tak, aby čísla v druhom bloku boli vzostupne usporiadané. Nakoniec program zapíše obsah usporiadaného bloku do súboru `usporiadane.txt`. Program negeneruje žiaden výstup na štandardný výstup. Využite ukazateľovú aritmetiku.

Ukážka súboru `neusporiadane.txt`:

```
4.8
9.26
```

uloha-7-1.cneusporiadane.txtŠtandardný výstup

```
1 // uloha-7-1.c -- Tyzden 7 - Uloha 1
2
3 #include <stdio.h>
4
5 int main()
6 {
7     // sem napis svoje riesenie
8
9     return 0;
10 }
```

Otázky / komentáre:

Michal: Ahoj.
Matus: Ahoj.
Michal: Bude ti stačiť fscanf().

Tvoja otázka / komentár:

Vráť sa k mojej úlohePošli

2. experiment

- ✦ 79 študentov v učebni
- ✦ Opakovaný predmet

Výsledok 2. experimentu

- Študenti ignorovali možnosť požiadať o radu
- Počet odoslaných správ: 3
- Dotazník pre 73 z nich

Výsledok 2. experimentu

- Dotazník pre 73 z nich – najčastejšie vyjadrenia:

Prečo sa nezapojili	Počet
nepotreboval	12
nevšim. / nevedel	10
nebola príležitosť	2
nikto sa neozval	1
zaneprázdnený	1
nebolo koho	1
nikto sa nepýtal	1
bojím sa	1

Výsledok 2. experimentu

- Dotazník pre 73 z nich – najčastejšie hodnotenie:

Hodnotenie	Počet
fajn	43
dobré, ale	12
nefungovalo	5
netreba	2

„veľmi dobrá funkcia ak by na otázky neodpovedali študenti“

Po 2. experimente

- ✦ Zvýraznenie príchodu žiadosti o radu
- ✦ Automatické žiadosti
 - ✦ po 3 chybných kompiláciách
- ✦ Hypotéza: Študenti budú ochotnejší poradiť ako požiadať o radu.

3. experiment

- ✦ 69 študentov v učebni
- ✦ Opakovaný predmet

Výsledok 3. experimentu

- ✦ Automatické žiadosti priniesli zlepšenie
 - ✦ 31 % zodpovedaných žiadostí (21 párov)
- ✦ Vzniklo 5 dialógov, ale len 8 správ
- ✦ Nestačilo na naplnenie matice

Zhrnutie

- ✦ Identifikovali sme nedostatky súčasných prístupov
- ✦ Navrhli sme metódu založenú na:
 - ✦ živom, synchrónnom posudzovaní kódu
 - ✦ výbere posudzovateľa na základe pravdepodobnosti úspechu spolupráce
- ✦ Zostrojili dva prototypy
- ✦ Realizovali 3 experimenty

Závery

- ✦ Živé posudzovanie vhodné pre kontext výučby
- ✦ Potrebné viazať spätnú väzbu na kód
- ✦ Väčší dôraz na pochopenie procesu riešenia úloh
- ✦ Motivácia študentov je kľúčová
- ✦ Zaujímavý ďalší výskum: uvažovanie odpovedí na položku – úlohu (klasický IRT prístup)

Porovnanie experimentov

	Miesto	Predmet	Študentov	Dĺžka	Pridel'ovanie
1	Doma	SPP	172	8 h	Náhodné, automatické
2	Učebňa	PP	79	3 h	Žiadosti (manuálne)
3		(opakov.) TEAP	69	3 h	Žiadosti (aj automatické)

Výsledky experimentov

	Študentov	Žiadostí	Vzniknutých párov	Dialógov	Správ
1	172	—	391	11 (3 %)	185
2	79	14	4 (29 %)	2 (50 %)	3
3	69	24 + 44 auto.	5 + 16 (31 %)	5 (24 %)	8

Riešenie v kontexte iných scenárov posudzovania

- ✦ Nie je závislé na posudzovaní vo význame pomoci
- ✦ Toto je prispôsobenie PCR pre potreby výučby
- ✦ Riešenie je viac závislé na kontexte výučby ako na našej interpretácii posudzovania