Metóda pre sociálne programovanie a posudzovanie kódu

Bc. Michal Tomlein

Vedúci práce: Mgr. Jozef Tvarožek, PhD.

Problémová oblasť

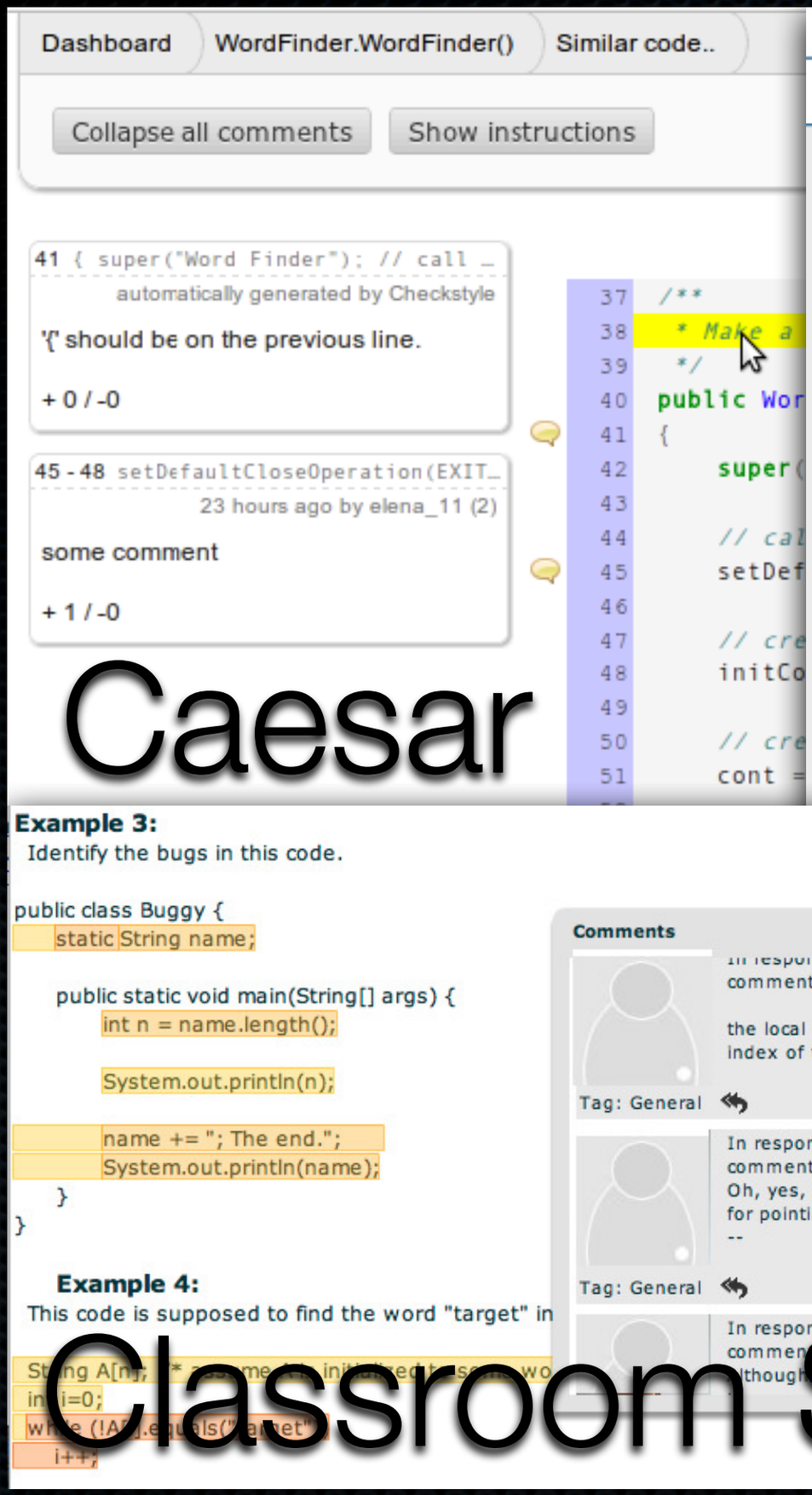
- Výučba programovania
  - Učiteľ nemôže poskytovať spätnú väzbu všetkým študentom naraz
- Posudzovanie kódu rovesníkmi (PCR)
  - Ako integrovať do procesu výučby
  - Ako vyberať posudzovateľa
- Sociálne prístupy a kolaborácia

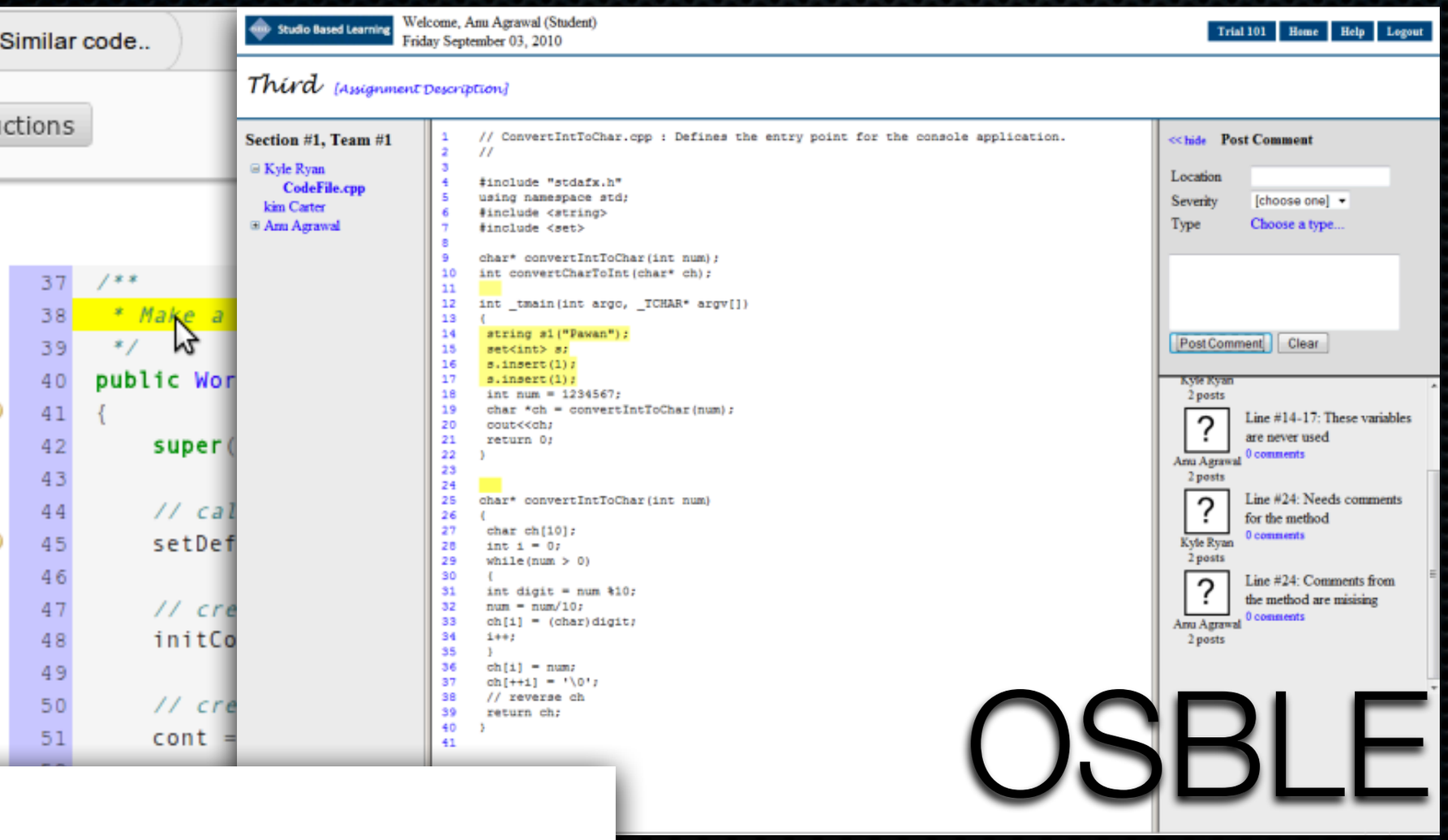
Ciele

- 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





- 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 Najčastejši znak | 0 Cézarova šifra | 0 Dešifrovanie | 0 Tajná správa | Bc. Michal Tomlein

Napíšte 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

```c
// sifry-2.c -- Sifry - Najcastejsi znak

#include <stdio.h>
#include <stdlib.h>

int *histogram(const char *str);

char most_frequent_char(const char *str)
{
  int count[256], i, max;

  // Tu napis svoj kod
  for (i = 0; i < 256; i++)
    count[i] = 0;
  for (i = 0; str[i]; i++)
    if (str[i] >= 'a' && str[i] <= 'z')
      count[str[i]]++;
  for (max = 0, i = 1; i < 256; i++)
    if (count[max] < count[i])
      max = i;
  return max > 0 ? (char)max : '?';
}

int main()
{
  printf("%c\n", most_frequent_char("")); // ?
  printf("%c\n", most_frequent_char("hell0000000")); // l
  printf("%c\n", most_frequent_char("aardvark")); // a
  return 0;
}
```

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 pristupuješ k prvku poľa.

Jožko: Stačí, keď ho pretypuješ.

Michal: Na char?

Jožko: Áno.

Tvoja otázka / komentár:

Pošli

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

sifry-1.c | Štandardný vstup | Štandardný výstup

```c
// sifry-1.c -- Sifry - Histogram

#include <stdio.h>
#include <stdlib.h>

int *histogram(const char *str)
{
  // hist je pole 26 intov s hodnotou 0
  int *hist = (int *)calloc(26, sizeof(int));

  int i;
  for (i = 0; str[i]; ++i) {
    if (str[i] >= 'a' && str[i] <= 'z')
      hist[str[i] - 'a'] ++ 1;
  }

  return hist;
}

int main()
{
  int *hist = histogram("hello");
  free(hist);
  return 0;
}
```

Kompilácia ... CHYBA (kliknite na chyby nižšie)

sifry-1.c: In function 'histogram':

sifry-1.c:14:29: error: expected ';' before numeric constant

Správy pre mňa:

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

Nová správa:

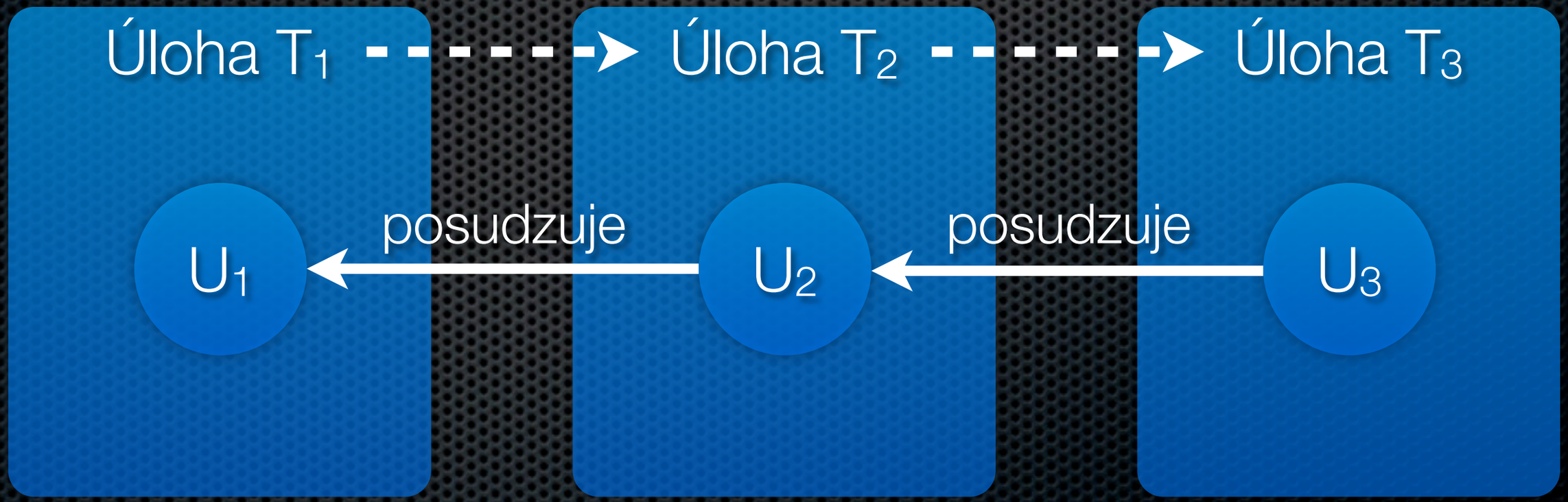
Pošli správu

Metóda

Živé posudzovanie prináša problém opisovania

Obmedzenie opisovania

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



Model používateľa

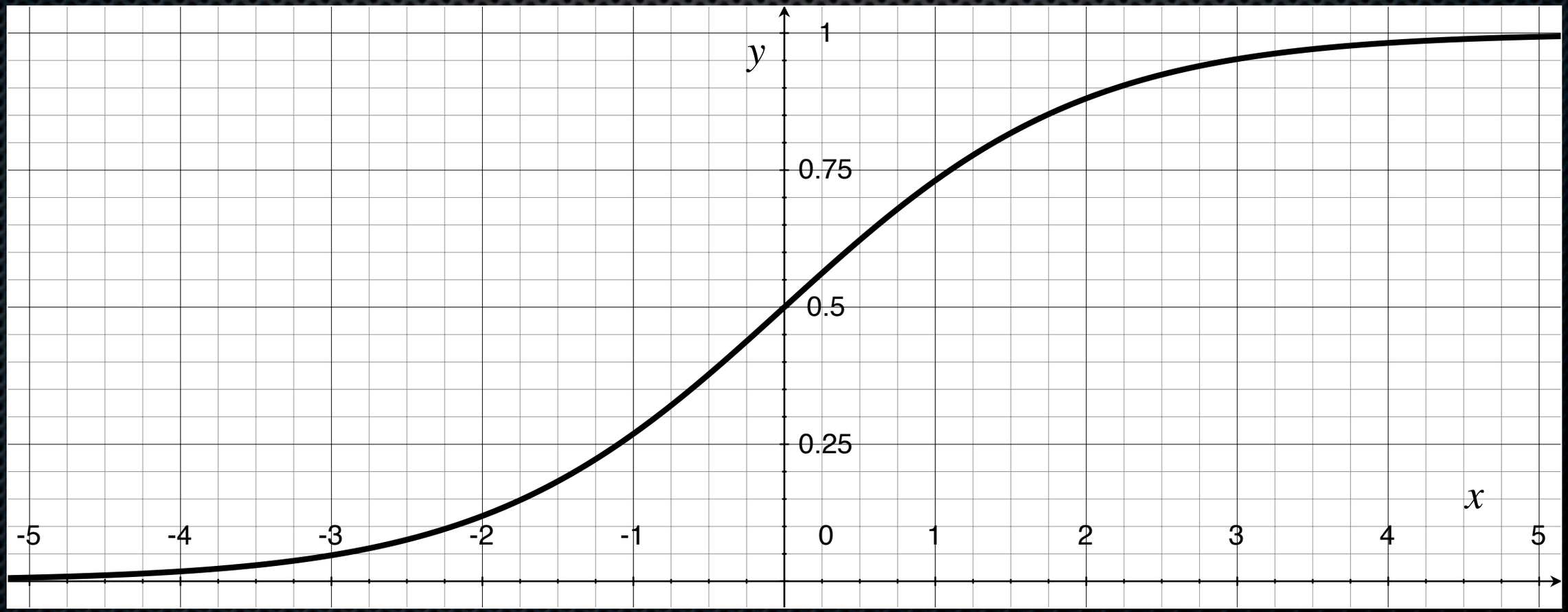
- Osobnostný model Big Five (extroverzia, ochota, svedomitosť, neuroticizmus, otvorenosť)
- Určujeme posudzovateľské schopnosti:
  - Poskytnúť pomoc/radu — *deliver*
  - Prijať pomoc/radu — *receive*

Metóda vyberá posudzovateľa s najväčšou pravdepodobnosťou úspechu spolupráce

Pravdepodobnosť úspechu spolupráce *i* a *j*

- Raschov model

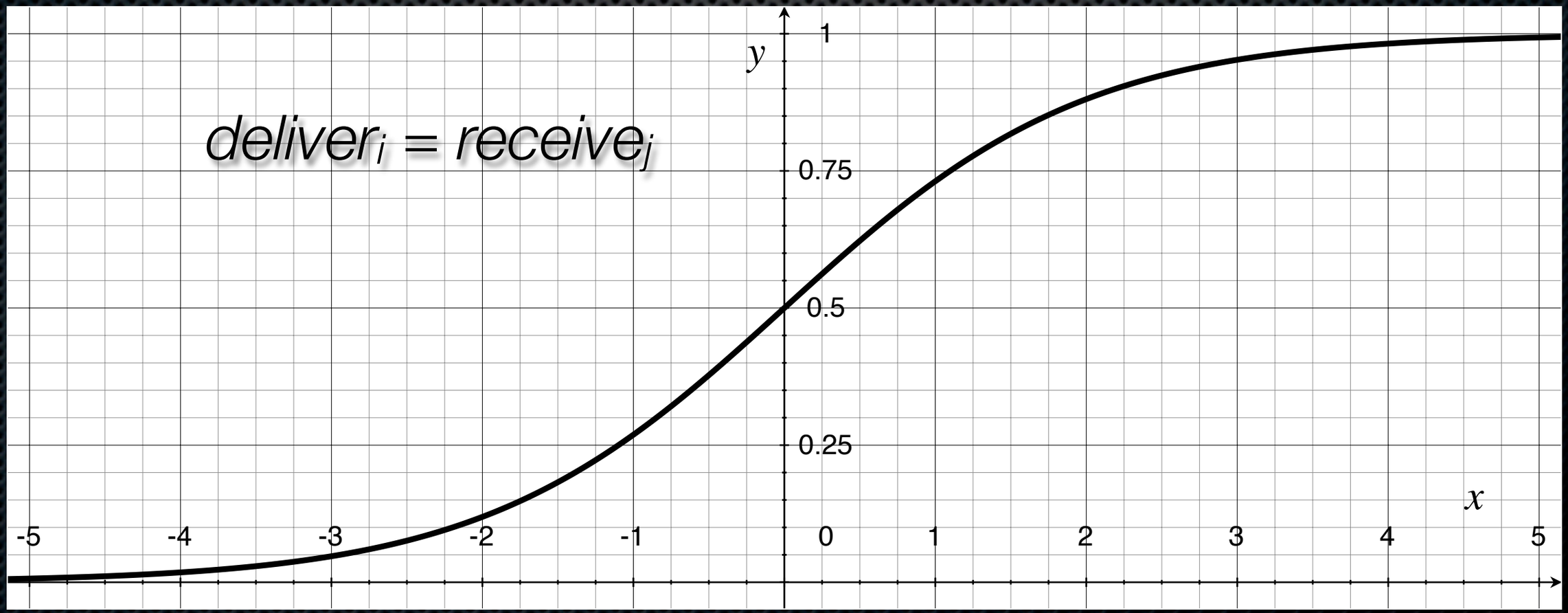
$$P(i,j) = \frac{e^{deliver_i - receive_j}}{1 + e^{deliver_i - receive_j}}$$



Pravdepodobnosť úspechu spolupráce *i* a *j*

- Raschov model

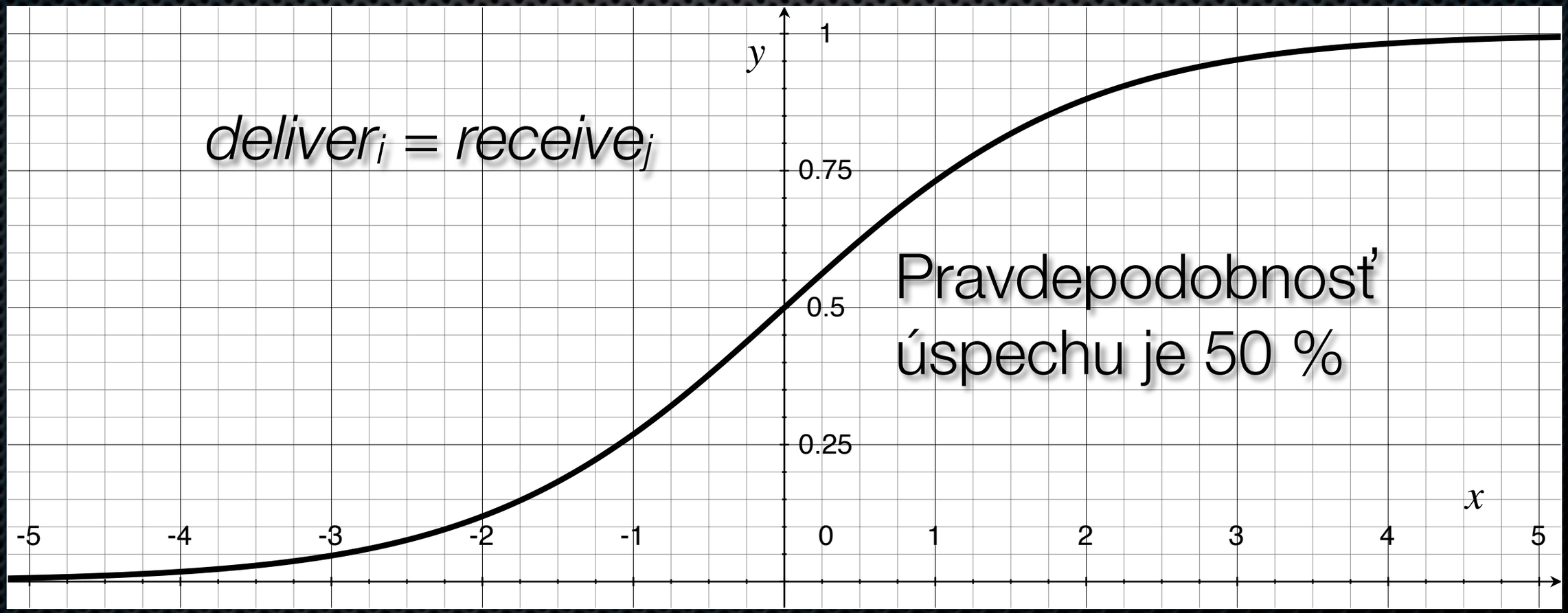
$$P(i,j) = \frac{e^{deliver_i - receive_j}}{1 + e^{deliver_i - receive_j}}$$



Pravdepodobnosť úspechu spolupráce *i* a *j*

- Raschov model

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



Metóda pracuje v 2 fázach

1. Kalibrácia modelu
   - Náhodný výber posudzovateľov
2. Aplikovanie modelu
   - Výber na základe modelu

Model používateľa


Dotazník
Big Five
Model
používateľa

Náhodný výber posudzovateľa

Dotazník Big Five

Model používateľa

Náhodný výber posudzovateľa

Hodnotenie úspešnosti párov

Dotazník Big Five

→

Model používateľa

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

Dotazník Big Five → Model používateľa

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 → (MLE) → Posudzovateľské schopnosti

Dotazník Big Five → Model používateľa

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 → (MLE) Posudzovateľské schopnosti → Pravdepodobnosti úspechu spolupráce

Dotazník Big Five → Model používateľa

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		

Hodnotenie úspešnosti párov → (MLE) Posudzovateľské schopnosti → Pravdepodobnosti úspechu spolupráce

Výber na základe pravdepodobnosti úspechu

Dotazník Big Five

Model používateľa

Analýza korelácií

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		

Hodnotenie úspešnosti párov

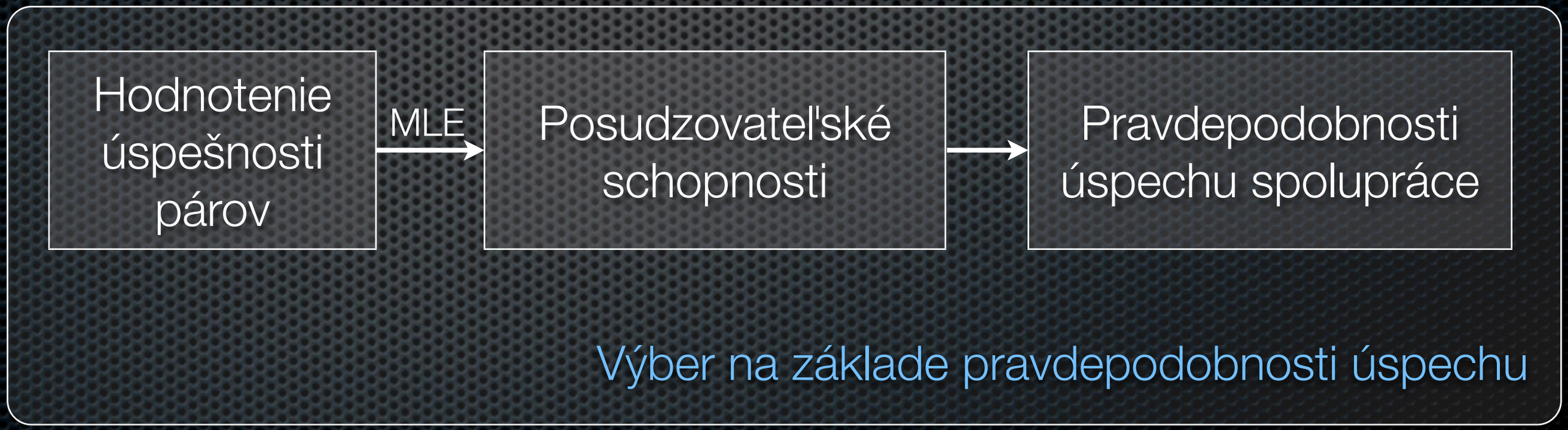
MLE

Posudzovateľské schopnosti

Pravdepodobnosti úspechu spolupráce

Výber na základe pravdepodobnosti úspechu

Dotazník
Big Five



Dotazník
Big Five

Posudzovatel'ské
schopnosti
z korelácií

Hodnotenie
úspešnosti
párov

MLE

Posudzovatel'ské
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 7-1 | 0 Úloha 7-2 | Bc. Matus Tomlein

Napíšte 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 suboru `usporiadane.txt`. Program negeneruje žiaden výstup na štandardný výstup. Využite ukazovateľovú aritmetiku.
Ukážka súboru `neusporiadane.txt`:

```
4.8
9.26
```

uloha-7-1.c | neusporiadane.txt | Štandardný 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 }
```

Tvoja otázka:

Ako načítam čísla zo vstupu?

Už netreba

Kamoši sa pýtajú:

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 8-1 | 0 Úloha 8-2 | 0 Úloha 8-3 | Bc. Michal Tomlein

Napíšte 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) vrati 0 a v reťazci dst bude to

uloha-8-1.c | Štandardný vstup | Štandardný výstup

```c
// uloha-8-1.c -- Tyzden 8 - Uloha 1

#include <stdio.h>

int strinsert(char *dst, int len, const char *src, int offset)
{
  // sem napis svoje riesenie

  return 1;
}

int main()
{
  char buf[100];

  sprintf(buf, "totojeretazec");
  if (strinsert(buf, 100, "druhy", 6))
    printf("Nepodarilo sa vlozit retazec.\n");
   else
    printf("%s", buf);
  return 0;
}
```

Tvoja otázka:

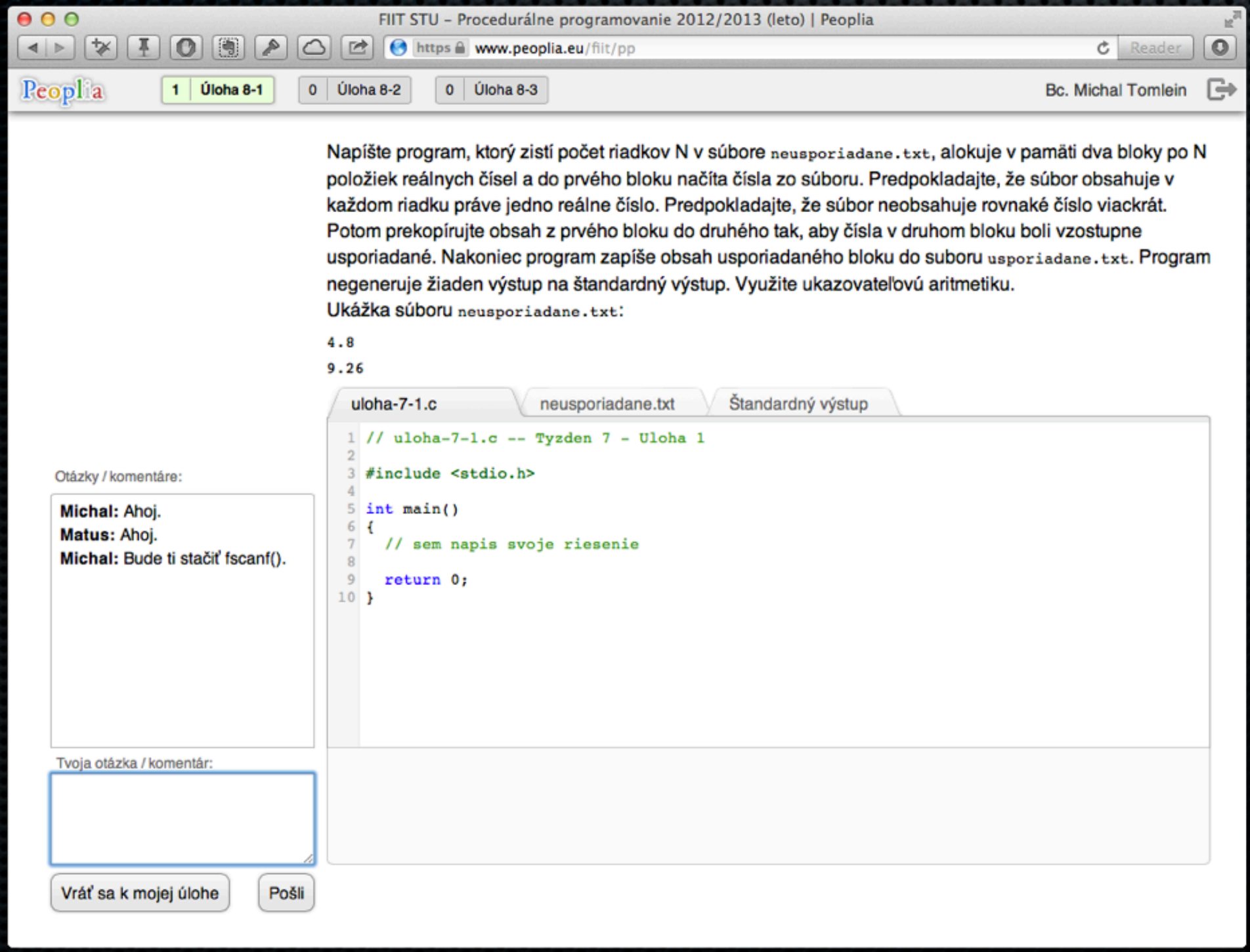
Spýtaj sa kamoša

Kamoši sa pýtajú:

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

Kompilovať | Spustiť program | Odoslať riešenie

Druhý prototyp



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 (opakov.) TEAP	79	3 h	Žiadosti (manuálne)
3			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