Webový systém pro testování znalostí studentů

Výzkumný úkol

Bc. František Navrkal

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

2017-09-14

O čem je výzkumný úkol?

Cíle práce (ze zadání):

- prostudovat teorii adaptivního testování znalostí a bayesovských sítí,
- navrhnout a realizovat webový systém pro realizaci adaptivního testování (systém má obsahovat i GUI, nástroje pro analýzu výsledků a sběr dat).

Cíle práce (de facto):

- nastínit základní koncepty nutné pro pochopení fungování aplikace,
- ukázat postup pro návrh adaptivního testu,
- stručně zdokumentovat aplikaci.

Počítačové adaptivní testování

Přizpůsobování testu na základě informací o studentovi

- z průběhu testu a
- z ostatních údajů o studentovi.

Motivace:

- nutno pokrýt všechny úrovně dovedností,
- odstranění nudy a nepozornosti u zdatných studenty,
- odstranění zmatku, frustrace, náhodně uhodnutých odpovědí u těch méně zdatných.

Průběh testování

Opakování:

1. výběru otázky na základě výběrového kritéria,
2. zobrazení otázky studentovi, získání odpovědi a vyhodnocení,
3. aktualizace znalostního modelu.

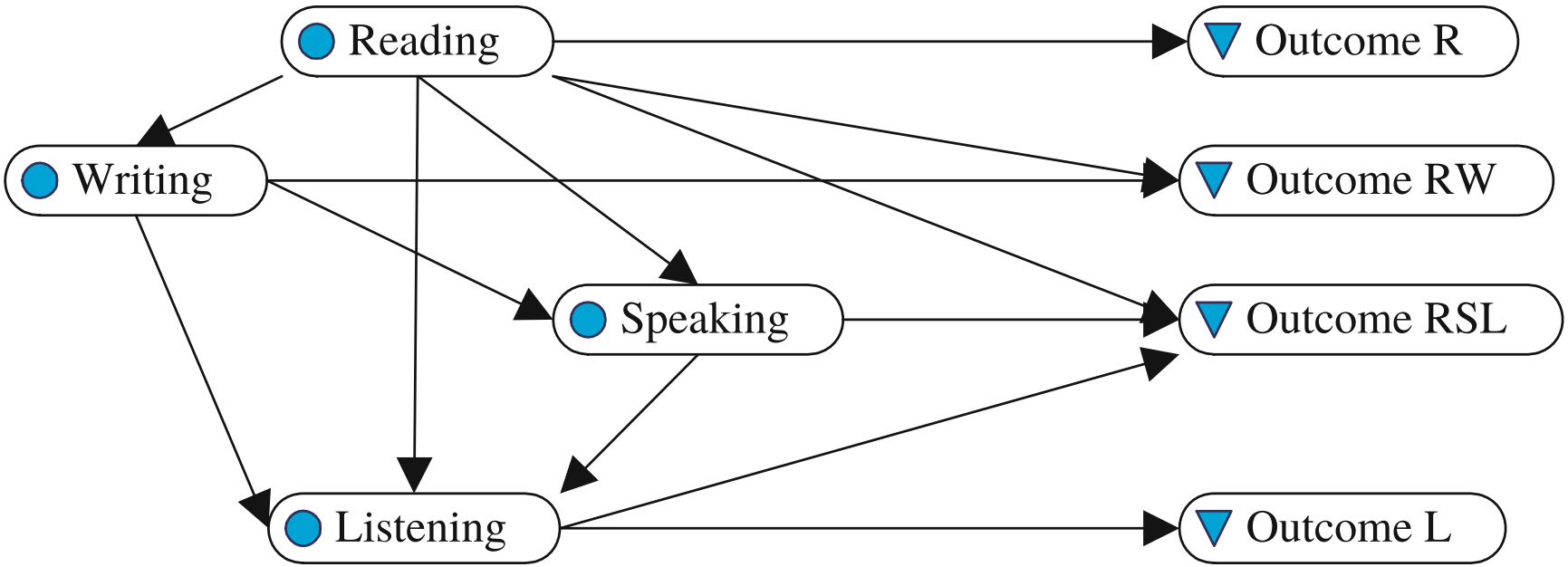
Dohromady jeden krok testu.

Bayesovské sítě

Pravděpodobnostní datový model popisující kauzální vztahy mezi náhodnými veličinami.

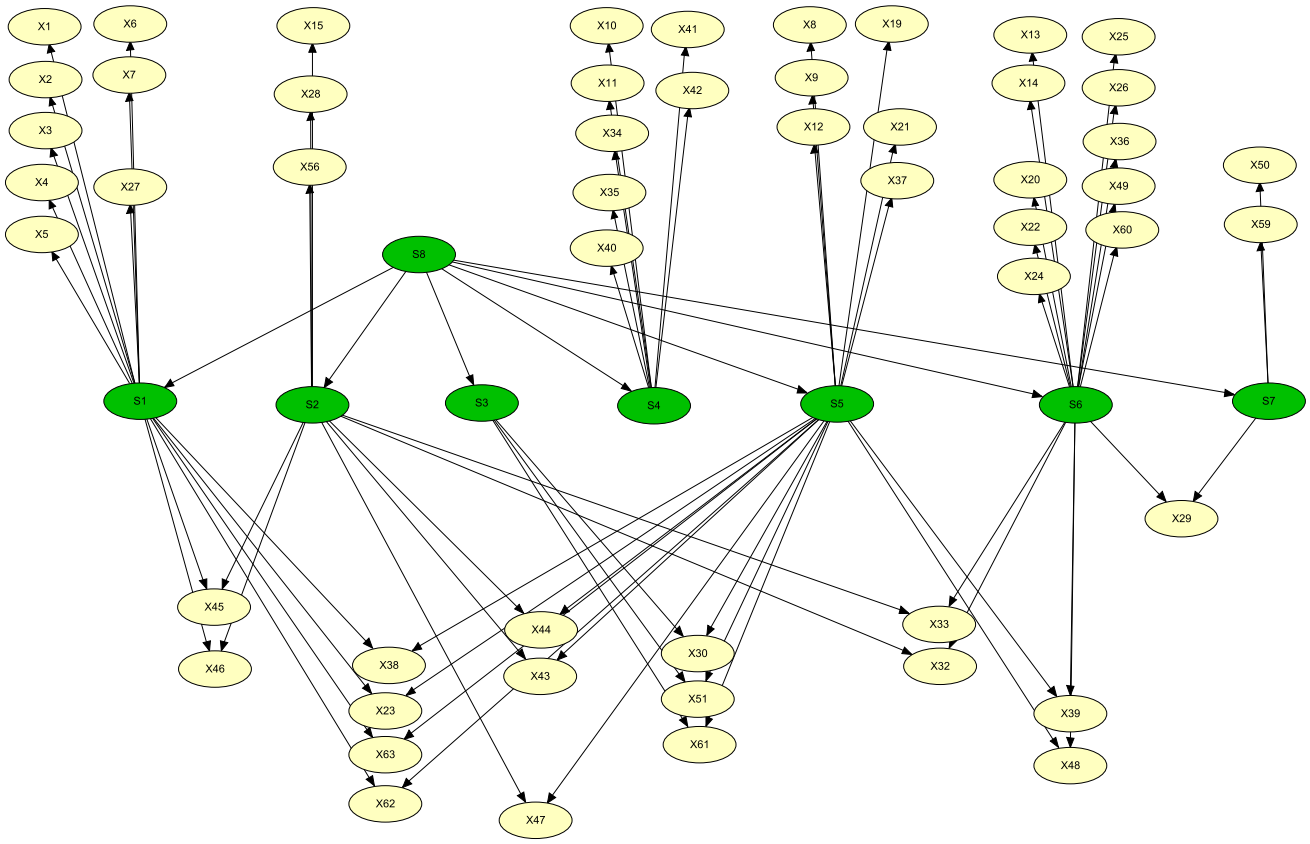
- Struktura:
  - orientovaný acyklický graf.
- Podmíněné pravděpodobnosti:
  - funkce pravděpodobnostních rozdělení náhodných veličin
  - nebo speciálně tabulky podmíněných pravděpodobností.

Struktura pro testování znalostí z angličtiny



Zdroj: ALMOND, R.G., MISLEVY, R.J., STEINBERG, L., YAN, D., WILLIAMSON, D. Bayesian Networks in Educational Assessment. Springer, 2015.

Struktura pro testování znalostí z matematiky

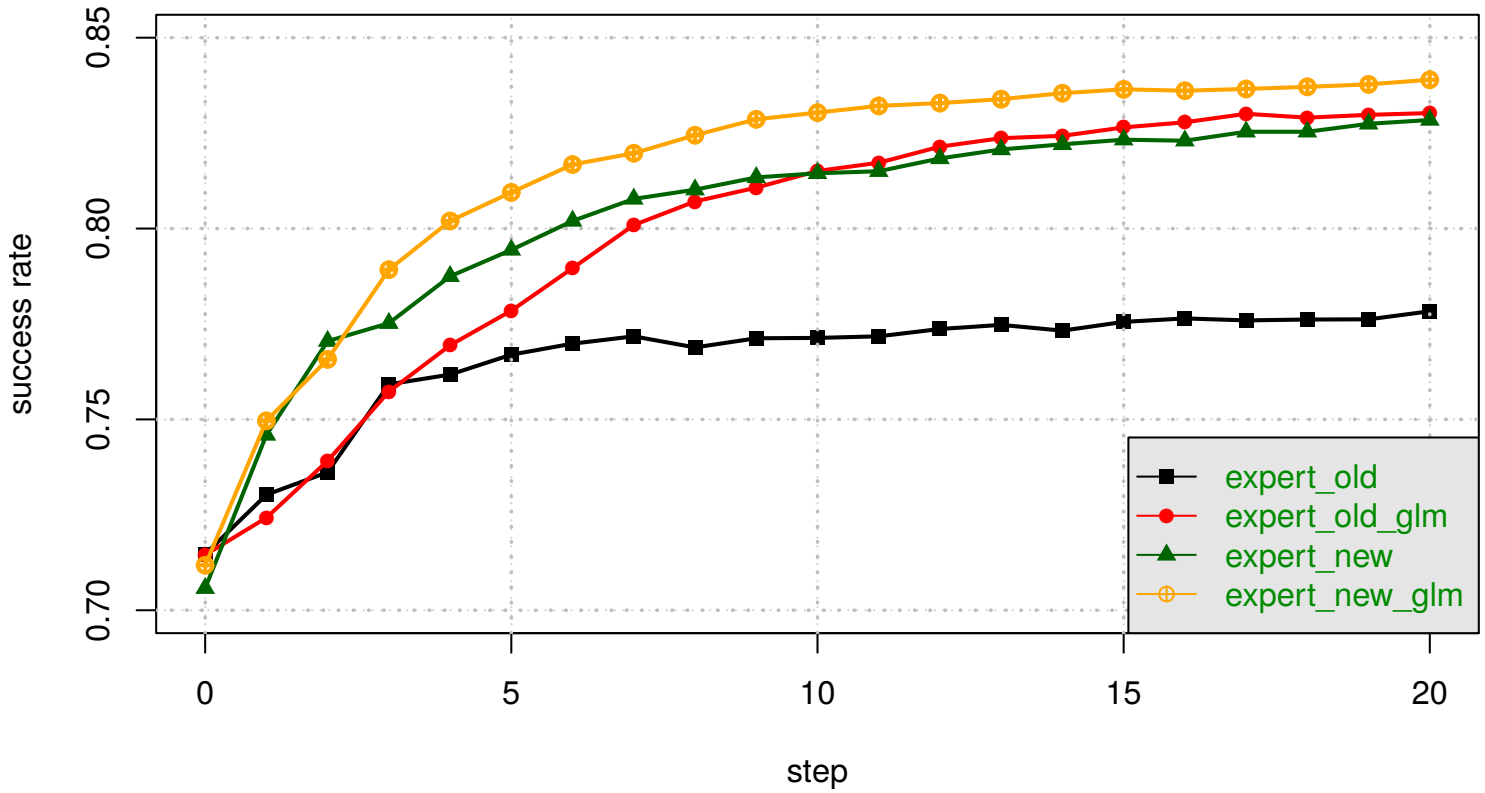


Zdroj: PLAJNER, M., J. VOMLEL. Student Skill Models in Adaptive Testing. Proceedings of the Eighth International Conference on Probabilistic Graphical Models, pp. 403–414, 2016.

Možný postup návrhu adaptivního testu

1. Návrh otázek a kandidátních struktur bayesovské sítě,
2. získání dat pomocí statických testů,
3. naučení tabulek podmíněných pravděpodobností,
4. porovnání modelů.

Srovnání výkonu modelů

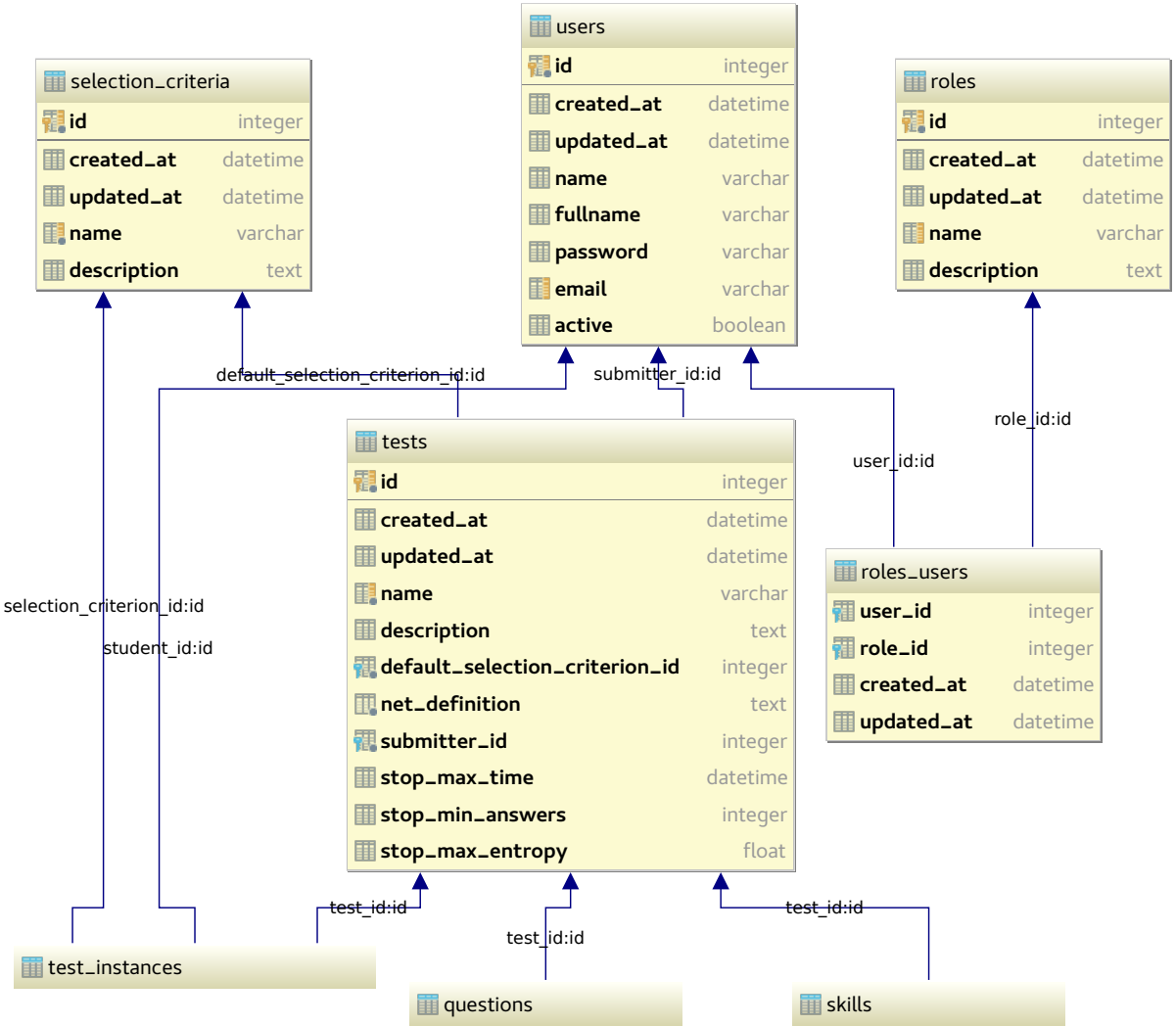


Zdroj: PLAJNER, M., J. VOMLEL. Student Skill Models in Adaptive Testing. Proceedings of the Eighth International Conference on Probabilistic Graphical Models, pp. 403–414, 2016.

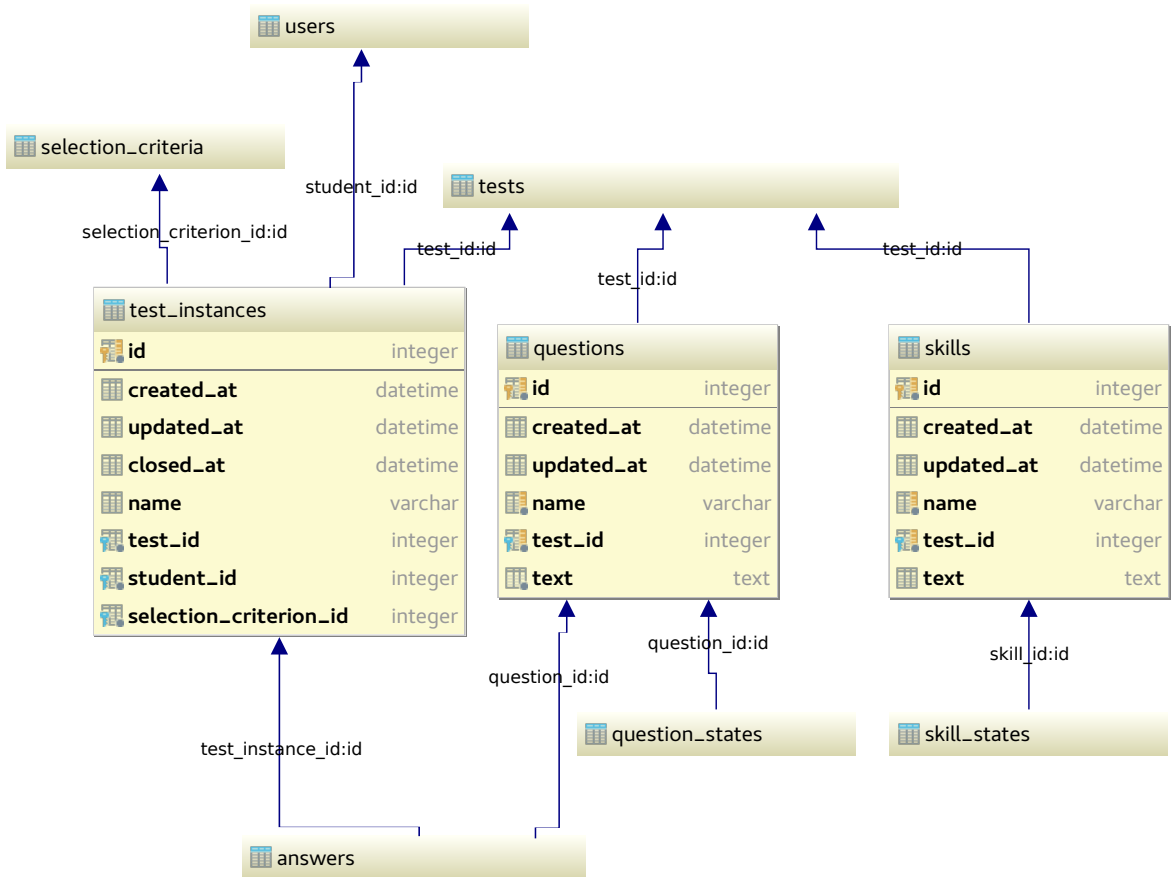
Struktura systému

1. Aplikace samotná, napsaná v *Pythonu* – v podstatě mojí vlastní knihovna CArisTotle –,
2. SQL databáze, přistupovaná přes *SQLAlchemy*, – nyní pro vývoj *SQLite* –,
3. knihovna catest v *R*, spojená přes *RPy2* se zbytkem systému a
4. *Jinja* šablony pro tvorbu výstupního HTML kódu.

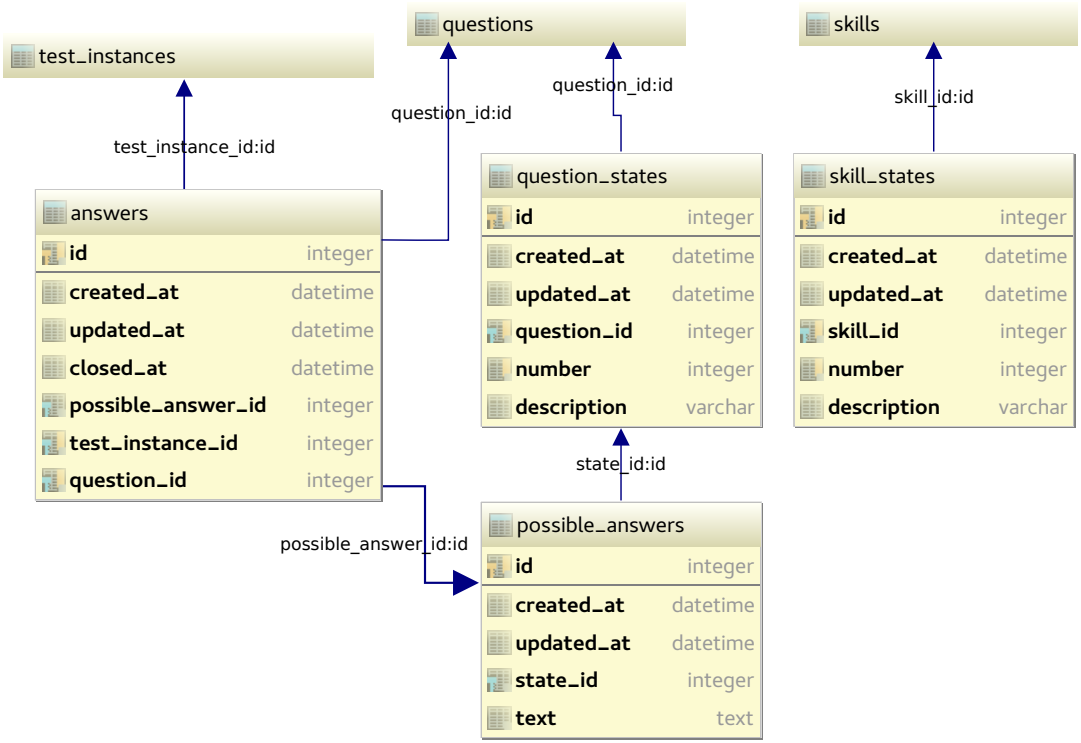
Datový model (1/3)



Datový model (2/3)



Datový model (3/3)



Fragmenty kódu R v Pythonu

```
ro.r('''get.marginals <- function(model, node_vector) {
                model <- one.dimensional.marginals(model,
                                node.index(model, node_vector))
                x <- list()
                for(i in 1:length(node_vector)){
                    x[[i]] <- model@marginals[[i]]@a
                }
                x
            }''')
r_get_marginals = ro.r['get.marginals']
```

Ukázka volání R z Pythonu

```
r_pick_question = ro.r['pick.question']


def pick_question(model, all_questions: List[str],
                  candidate_questions: List[str],
                  selection_criterion: int = 1):
    r_all_questions = ro.StrVector(all_questions)
    r_candidate_questions = ro.StrVector(candidate_questions)
    pick_obj = r_pick_question(model, r_all_questions,
                               selection_criterion,
                               r_candidate_questions)
    question_indices = list(pick_obj[2])
    picked_questions = [candidate_questions[index - 1] for index in
                        question_indices]
    return picked_questions
```

Ukázka pohledové funkce

```
@app.route('/test/<int:test_id>')
@login_required
def test_overview(test_id):
    test: Test = get_entity_by_type_and_id(Test, test_id)
    check_test_existence_and_redirect_if_not_exists(test)
    test_instances = list_test_instances_by_test_and_student(test,
                                                             current_user)
    possible_criteria = list_selection_criteria()
    test_instance_options_form = \
        TestInstanceOptionsForm(possible_criteria=[(sc.id, sc.name) for sc
                                                   in possible_criteria],
                                default_criterion=
                                    test.default_selection_criterion.id)
    return render_template("test_overview.html", test=test,
                           test_instances=test_instances,
                           test_instance_options_form=
                               test_instance_options_form,
                           possible_criteria=possible_criteria)
```

Ukázka z Jinja šablony

```
{% for test_instance in test_instances %}
<article>
    <div class="row">
        <div class="six columns">
            <p>Název případu: {{ test_instance.name }}</p>
            <p>Případ zahájen: {{ test_instance.created_at }}</p>
        </div>
        <div class="six columns">
            <a class="button"
              href="{{ url_for('test_instance_overview',
                               test_id=test.id,
                               test_instance_id=test_instance.id) }}">
                Otevřít přehled případu</a>
        </div>
    </div>
</article>
{% endfor %}
```

Postup testování

1. Student je odkázán na web nebo konkrétní test,
2. student si vybere ručně, nebo je mu vybrána systémem otázka,
3. student zodpoví otázku,
4. pokud je dosaženo ukončovacího kritéria (čas, počet otázek, entropie), test končí, jinak následuje krok 2.

Výsledky testu jsou vidět průběžně i po jeho ukončení.

Problémy s časovou náročností

Problémy:

- Jen když je potřeba použít `catest`,
- *R* hodně kopíruje data (předávání argumentů funkcím),
- `catest` není příliš optimalizována.

Možná řešení:

- Asynchronní předpočítávání možných scénářů vývoje testu,
- rychlejší implementace `catest` (asi v něčem jiném než *R*),
- provedení optimalizace `catest`.

Možnosti budoucího vývoje

Chybí:

- Nástroje pro analýzu výsledků (dat je ovšem dost),
- webové rozhrání pro zadávání testů a
- webové rozhraní pro administraci.

Další postup:

- Navrhnout strojově čitelný formát pro zadávání testu,
- navrhnout analytické nástroje,
- implementovat příslušná rozhraní a nástroje.

Děkuji za pozornost a věnovaný čas.