Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

DISERTAČNÍ PRÁCE

2012

Ing. Petr Voborník

Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Univerzální testovací prostředí

Disertační práce

Autor: Ing. Petr Voborník
Školitel: prof. RNDr. Eva Milková, Ph.D.

Hradec Králové březen 2012

Prohlášení

Prohlašuji, že jsem tuto disertační práci na téma *Univerzální testovací prostředí* zpracoval zcela samostatně s využitím uvedené literatury.

V Hradci Králové, dne 5.3.2012 ……

Petr Voborník

Poděkování

Rád bych tímto poděkoval své školitelce prof. RNDr. Evě Milkové, PhD. za trpělivost, cenné rady, připomínky, metodické vedení a veškerou pomoc při zpracování této práce.

Díky za podporu patří i mé skvělé manželce a rodině.

ANOTACE

Disertační práce „Univerzální testovací prostředí" se aktivně zabývá oblastí elektronického testování z hlediska jejího aktuálního stavu, vývoje nového originálního produktu a řešení řady dílčích úloh z této oblasti.

První část analyzuje současný stav v oblasti práce, popisuje a kriticky hodnotí několik stávajících systémů pro různé kategorie elektronického testování. Výsledky dotazníkového šetření upozorňující na nedostatky těchto systémů se spolu s jejich některými nepopiratelně kvalitními vlastnostmi staly podkladem pro návrh a postupnou realizaci univerzálního testovacího prostředí. To se snaží oprostit od nedostatků již existujících systémů, rozšířit a obohatit funkcionalitu o nové žádané funkce a zároveň reagovat na současné moderní trendy v oblasti vývoje aplikací.

Druhá část práce prezentuje výsledky hlavního cíle (vytvoření univerzálního testovacího prostředí) a zároveň seznamuje i s výsledky cílů dílčích (originální řešení úloh důležitých pro realizaci hlavního cíle).

Představen je vlastní jazyk pro vytváření testových otázek, umožňující jak jejich bohaté grafické a interaktivní zpracování, tak i širokou variabilitu pro vnesení náhodné složky přímo do jednotlivých otázek. Ty mohou být díky tomu používány opakovaně bez rizika automatizované interpretace řešení zkoušeným, aniž by rozuměl látce v otázce zahrnuté. Použití těchto funkcí nijak neomezuje automatické hodnocení testů, právě naopak, jsou k dispozici daleko širší možnosti celkového i částečného hodnocení výsledků zkoušení, včetně podpory určení míry podobnosti dvou textových řetězců s překlepem.

Dílčí problémy, jež vývoj univerzálního testovacího prostředí přinesl, jsou prezentovány v následujících podkapitolách. Ty byly řešeny převážně originálními postupy, založenými na soudobých poznatcích a výzkumech. Principy těchto subsystémů byly zároveň navrženy a popsány tak, aby poznatky z nich plynoucí mohly být aplikovatelné i mimo cílové nasazení v tomto projektu a sloužily jako výchozí bod či rámcová metodika pro vývoj nebo výzkum podobně zaměřených systémů či oblastí.

Byl například vytvořen nový způsob výběru otázek do testu, jež bere v potaz předchozí výsledky a časový vliv na zapomínání. Charakteristické řetězce zase umožňují pokročilou práci s náhodnými prvky uvnitř otázek a jejich ovlivnění pro generování více různých, než pouze čistě náhodných hodnot. Popsán byl i vlastní způsob bezpečné komunikace klientské aplikace se serverem prostřednictvím internetu z hlediska autentizace uživatele, šifrování dat, ale i automatizovaného efektivního předávání a objektově relačního mapování dat. Pro administrační rozhraní byl navržen, realizován a popsán přístup konfiguračního frameworku, jež je alternativou ke klasickým strukturálně či objektově programovaným frameworkům a přináší množství výhod především pro vývoj dektopových či RIA informačních systémů. V řadě případů bylo optimálního řešení dosaženo prostřednictvím nově vytvořených mini-jazyků, které mohou být zároveň inspirací nejen pro jejich přímé použití v dalších projektech, ale i jako směr, jímž lze vést řešení některých nových problémů.

V posledních kapitolách jsou prezentovány příklady stávajícího nasazení univerzálního testovacího prostředí v praxi a jeho integrace do aplikací třetích stran prostřednictvím aplikačního rozhraní.

ANNOTATION

The thesis "Universal Testing Environment" focuses on the electronic testing, e-testing, its current possibilities and the issues of new product development, same as solutions for numerous specific tasks in this field.

The first part analyzes the present state, describes and evaluates several existing systems for various categories of e-testing. The shortcomings of these systems were identified through user questionnaires evaluation and thus have become the basis, together with undoubtedly their positive qualities, for the design and gradual creation of a universal testing environment. It attempts to avoid the shortcomings of existing systems, to enlarge and to enrich the functionality by new desirable functions and at the same time it reflects the current contemporary trends in application development.

The second part of the thesis presents the results of this main goal (i.e. the creation of the universal testing environment) and of the particular goals (i.e. original solutions of tasks important for the main goal).

A proprietary language is introduced for the creation of testing questions, it enables to present them in rich graphic and interactive presentation, and it also offers the possibility to use random selections directly for individual questions. Thanks to that they can be used repeatedly without the risk of being interpreted mechanically, i.e. without the true understanding of the matter. The usage of this function does not limit in any way the automatic evaluation of the tests, on the contrary it offers a much wider possibility for the overall or partial evaluation of the testing results, including the support for the resemblance of two text strings calculation containing a typing error.

Problems created by the development of the testing environment are presented in the following subchapters. They were solved by original procedures based on current know-how and research. The guidelines for these subsystems were also designed and described so that they could be used also outside of this project as the basis or an overall methodology for the research and development of similar systems.

For example a new way for the selection of test questions was created which takes into account the previous results and the training curve. Characteristic strings again enable advance work with random elements inside the testing topics and their use for the generating of more varied than just merely random values. The specific protected communication between the client application and the server via the internet is described, from the user-end, the coding of the data, and also automatic efficient exchange and object-relational data mapping. For the administrator interface the access to the configuration framework was designed, created and described, it represents an alternative to the classical frameworks programmed structurally or in objects. It brings numerous advantages namely for the development of desktop or RIA information systems.

In many cases the optimal solution was reached through newly created mini-languages which can be an inspiration not only for their direct use in future projects but namely as a guidance for the solution of some new problems.

In the concluding chapters you will find examples of the presently used universal testing environments and their integration into third party application interface.

OBSAH

SEZNAM TABULEK

SEZNAM VZORCŮ

SEZNAM GRAFŮ

SEZNAM OBRÁZKŮ

SEZNAM UKÁZKOVÝCH KÓDŮ

SEZNAM PŘÍLOH

SEZNAM ZKRATEK

AES – Advanced Encryption Standard, pokročilý šifrovací standard

AICC – Aviation Industry Computer-Based Training Committee, standard v e-learningu

AJAX – Asynchronous JavaScript and XML, technologie vývoje interaktivních webových aplikací

API – Application Programming Interface, aplikační programové rozhraní

AOP – Aspect-Oriented Programming, aspektově orientované programování

ARGB – Alfa, Red, Green, Blue, aditivní barevný model s průhledností

ASCII – American Standard Code for Information Interchange, kódová tabukla znakové sady

AVG – AVeraGe, průměr

BLOB – Binary Large OBject, binární datový typ v databázi

CIDR – Classless Inter-Domain Routing, schéma pro přidělování IP adres

CMS – Course Management Systém, systém pro řízení kurzů

CPU – Central Processing Unit, procesor

CSS – Cascading Style Sheets, kaskádové styly

DB – DataBase, databáze

DES – Data Encryption Standard, standard pro šifrování dat

EDI – Electronic Data Interchange, elektronická výměna dat

GUI – Graphical User Interface, grafické uživatelské rozhraní

GUID – Globally Unique IDentifier,jedinečný globální identifikátor

HDD – Hard Disk Drive, jednotka pevného disku

HTML – HyperText Markup Language, hypertextový značkovací jazyk

HTTP – HyperText Transport Protocol, protokol pro přenos hypertextových dokumentů

HTTPS – Secure HyperText Transport Protocol, šifrované http

ID – jednoznačná IDentifikační hodnota

IL – též CIL nebo MSIL, Common Intermediate Language, nízkoúrovňový jazyk používaný v .NET

IP – Internet Protocol, číselná adresa počítače

JSON – JavaScript Object Notation, textový zápis objektových dat

LDAP – Lightweight Directory Access Protocol, protokol pro přístup k datům na adresářovém serveru

LINQ – Language-Integrated Query, integrovaný jazyk pro dotazování

LMS – Learning Management Systém, systém pro řízení výuky

M-J – Mini-Jazyk

OOP – Object-Oriented Programming, objektově orientované programování

ORM – Object-relational mapping, objektově relační mapování

OS – Operating Systém, operační systém

QML – Questions Markup Language, značkovací jazyk pro tvorbu testových otázek

RAM – Random Access Memory, operační paměť

RDBS – Relational Database Systém, relační databázový systém

REST – Representational State Transfer, rozhraní posílání dat v distribuovaném prostředí

RGB – Red, Green, Blue, aditivní barevný model

RIA – Ritch Internet Aplication, bohatá internetová aplikace

RO – Read Only, pouze pro čtení

RPM – Revolutions Per Minute, počet otáček za minutu

SHA – Secure Hash Algorithm, bezpečnostní hashovaní algoritmus

SCORM – Shareable Content Object Reference Model, referenční model pro e-learning

SOAP – Simple Object Access Protocol, protokol pro posílání zpráv přes internet

SRP – Secure Remote Password, protokol pro ověřování identity na základě hesla

SQL – Structured Query Language, strukturovaný dotazovací jazyk

SSL – Secure Sockets Layer, vrstva zabezpečující komunikaci

SVG – Scalable Vector Graphics, jazyk pro zápis vektorové grafiky

TCP – Transmission Control Protocol, protokol pro posílání dat na internetu

TSL – Transport Layer Security, protokol zabezpečené komunikace

URL – Uniform Resource Locator, webová adresa

UUID – Universally Unique IDentifier, jedinečný unikátní identifikátor

W3C – World Wide Web Consortium, mezinárodní konsorcium vyvýjející webové standardy

WCF – Windows Communication Foundation, komunikační platforma .NET Frameworku

WPF – Windows Presentation Foundation, technologie pro tvorbu bohatého uživatelského rozhraní

WSDL – Web Services Description Language, popis funkcí webové služby

WSE – Web Services Enhancements, doplněk pro .NET Framework

WYSIWYG – What You See Is What You Get, typ textového editoru

XAML – Extensible Application Markup Language, značkovací jazyk pro grafické rozhraní

XML – Extensible Markup Language, rozšiřitelný značkovací jazyk

XSLT – eXtensible Stylesheet Language Transformations, transformace XML dokumentů

◎ – označení dílčího cíle této práce

1 ÚVOD

V současné době nám počítače usnadňují práci téměř ve všech odvětvích, v některých jsou dokonce nepostradatelné. Patří mezi ně i oblast testování znalostí, která sahá do mnoha různých oborů. Mezi nejobvyklejší úlohy patří ověřování dosažených znalostí, hodnocení jejich úrovně, a to nejen z hlediska evaluačního, ale i sebepoznávacího. Samotné testování přitom nemusí zastávat pouze funkci hodnotící, může sloužit též jako výukový materiál, kdy se testovaný jedinec učí z vlastních chyb.

V oblasti testování lze využít některý z mnoha existujících počítačových testovacích systémů, které jej dokáží ve většině aspektů zefektivnit a automatizovat, a tím usnadnit tuto edukační činnost jejich tvůrcům, hodnotitelům i testovaným subjektům. Hotová řešení lze nalézt v podstatě pro libovolný druh testování, ať se již jedná o testy známkou hodnocené, výukové či prezentační, integrované nebo integrovatelné do LMS, samostatně fungující nebo i testy umožňující efektivní samostatnou výuku vystavěnou na základě novodobých psychologických poznatků.

Stávající testové systémy jsou však obvykle úzce cíleny na jeden konkrétní účel, případně pevně integrovány do jiných systémů, bez nichž nedokáží samostatně fungovat. Univerzální testovací prostředí, použitelné pro různé druhy testování a fungující zcela samostatně, ale zároveň i schopné kooperace s jinými systémy zatím k dispozici není.

Existující testovací systémy také jen velmi okrajově využívají možností, které elektronická forma testování skýtá. Automatické vyhodnocení je jen základ, kterým by výčet předností rozhodně končit neměl. Je třeba se oprostit od omezení, která jsou zažitá z papírového testování a více využít nabízeného potenciálu. Řadu činností, které byly ještě do nedávna nepředstavitelné a pro zadavatele testů výpočetně nereálné, mohou nyní moderní algoritmy zcela automatizovaně zvládnout na pozadí během několika málo milisekund. Software tak může, bez úbytku na své uživatelské přívětivosti, obstarávat veškerý management znalostí svých uživatelů tak, aby pozitivní efekt na jejich vědomosti byl co největší. Na straně druhé také dokáže co nejpřesněji informovat zadavatele testů o stavu znalostí zkoušených.

Jen velmi málo ze stávajících nástrojů zároveň využívá bohatých možností, které nabízí současné technologie. Mezi stále preferovanější aplikační architekturu patří *cloud computing*. V něm nejsou aplikace nainstalovány přímo na počítači, na kterém se používají, ale na centrálním serveru poskytovatele a uživatelé s nimi pracují přes odlehčené klientské aplikace, spustitelné např. v rámci internetového prohlížeče. [1] *V současné době lze již konstatovat, že koncept cloud computingu skutečně představuje další „generaci IT“.* [2]

Nové trendy v oblasti uživatelských aplikací také směřují k propracovanější grafice, plynulosti ovládání, interaktivitě a multimediálním prvkům. Aplikace kombinující tyto přístupy jsou pak označovány zkratkou RIA, tzn. *Ritch Internet Applications*, čili bohaté internetové aplikace.

Efektivní grafické uživatelské rozhraní ale samozřejmě nesmí být na úkor přehlednosti, ovladatelnosti a především funkčnosti aplikace. Klasické webové formuláře obsahující prvky typu *radio button* či *checkbox* sice plní svůj účel stále dobře, nicméně uživatel od současných aplikací očekává víc, přičemž soudobé technologie jsou schopny na toto plně reflektovat.

2 CÍLE DISERTAČNÍ PRÁCE

Hlavním cílem této disertační práce je navrhnout a vytvořit ucelený systém pro on-line testovací prostředí. Tento systém se má skládat z celé řady dalších, na soudobých poznatcích a výzkumech originálně navržených subsystémů.

Vedlejšími cíli práce je vyřešit řadu dílčích úkolů, jež splnění hlavního cíle doprovázejí a to vědeckým a novátorským způsobem. Při realizaci přitom v klíčových částech nemá docházet pouze k přebírání již stávajících, byť osvědčených postupů, ale na jejich základě přicházet pokud možno s vlastními, originálními a dosud nepublikovanými řešeními, která kromě splnění účelu v rámci tvorby hlavního cíle, mohou být i inspirací pro další rozvoj v dané oblasti.

Systém jako celek má umožňovat tvorbu a správu interaktivních testů s otázkami nejrůznějších typů, s jejichž pomocí může být prováděno nejen efektivní testování ve smyslu zkoušení, ale i provozovat výukové testy schopné jejich uživatelům oživit a zábavnou formou každému individuálně vštípit potřebnou látku. Systém tak má být použitelný kromě školství i ve veřejném a soukromém sektoru, pro ověřování, opakování a výuku znalostí libovolného tématu zvolenou skupinou osob, a to jak prezenčně, tak distančně.

Rozhraní jednotlivých částí systému má být navrženo intuitivně dle současných standardů a možností RIA. Systém má být provozuschopný jak na vlastním serveru, tak i jako modul umístěný v jiných (na jiném serveru provozovaných) stránkách webových aplikací (LMS nevyjímaje) třetích stran, a s nimi vhodným způsobem kooperovat.

Součástí práce má být i podrobný průzkum stávajících testovacích systémů, stavu jejich rozšířenosti a požadavků jejich uživatelů na tyto systémy.

Hierarchicky strukturovaný seznam hlavních částí a jejich podcílů se nachází na následující straně, přičemž jednotlivé dílčí cíle jsou podrobněji rozepsány v následujících podkapitolách[1].

[1] pro lepší orientaci budou v dalším textu označeny vazby na cíle symbolem terče ◎ následovaným číselnou kombinací, kterou jednotlivým cílům udává následující strukturovaný seznam

I. Podrobně analyzovat aktuální situaci
    1. vyzkoušet a zhodnotit stávající vybrané testovací systémy
    2. zjistit rozšířenost, názor a požadavky na tyto systémy od jejich uživatelů
II. Umožnit vytváření bohatých testových otázek
    1. s podporou vlastní libovolné grafiky
        A. standardní vektorové objekty
        B. rastrové obrázky
        C. animace
        D. podpora řešení metodou drag&drop
    2. zjednodušené vytváření
        A. opakujících se částí otázek
        B. typových otázek
        C. různých zdlouhavých standardních definic
    3. automatické vyhodnocování
        A. normalizované fuzzy hodnocení
        B. částečné porovnávání textových vstupů
        C. zpětná vazba a podrobný rozbor každé otázky
        D. archivovatelný a zpětně čitelný protokol o testování
III. Vytvořit testovací rozhraní
    1. zpracovávat vytvořené otázky seskupené do testů
    2. odolné proti podvodům
        A. manipulace s časem během testu
        B. možnost omezit přepínání oken
        C. přerušení spojení s internetem
        D. zabezpečené spojení se serverem
        E. bezpečné uchování dat
    3. míchání
        A. možnost generování náhodného obsahu otázek libovolného rozsahu
        B. náhodný inteligentní výběr otázek pro sestavení testu
    4. konektivita
        A. schopnost samostatného fungování
        B. propojení s webovými aplikacemi
        C. podpora bezpečného předávání informací mezi aplikacemi
        D. možnost individuálního nastavení testu pro každé jeho spuštění
IV. Vytvořit administrační rozhraní
    1. umožňující vytvářet, zadávat a spravovat výsledky testů
    2. spouštět jeden test za různých podmínek
        A. s vlastním nastavením
        B. s možností zaměření se pouze na určitý typ otázek
        C. omezit pro konkrétní uživatele či skupiny uživatelů
        D. omezit dostupnost jen z konkrétních počítačů
        E. přístupný jen v určitý čas s možností jeho periodického vymezení
        F. s přesně stanovenými podmínkami pro absolvování testu
    3. vytvořené pomocí vlastního frameworku, umožňujícího úpravy bez nutnosti rekompilace
    4. s automatizovaným a znovupoužitelným způsobem komunikace se serverem

2.1 Analýza současného stavu [◎ I]

V první fázi je důležité podrobně analyzovat stávající testovací systémy [◎ I.1], zhodnotit jejich vlastnosti a zvážit, kde jsou jejich nedostatky a v čem by nový systém byl oproti nim přínosný. Zároveň by měl být proveden dotazníkový průzkum mezi uživateli těchto systémů, který by měl zmapovat, jaký způsob testování v současnosti převažuje a jaké jsou požadavky uživatelů na tyto systémy [◎ I.2]. Z porovnání zjištěného stavu s požadovaným pak mohou být vyvozeny relevantní závěry.

2.2 Tvorba otázek [◎ II]

Pro tvorbu otázek je třeba navrhnout vhodný způsob jejich definice. Ten by měl být co nejkomplexnější, aby umožňoval vytvářet graficky bohaté, interaktivní a multimediálně propracované otázky všech možných typů (standardních didaktických, konkurenčními systémy podporovaných a dalších originálních). Kromě textového zadání by měla být k dispozici podpora přímého vkládání standardních vektorových objektů (čáry, obdélníky, elipsy apod.) [◎ II.1.A]. Zde by měly být inspirací již existující otevřené vektorové formáty, na jejichž základě by měl vzniknout nový, v alespoň některých částech překonávající ty stávající.

Kromě vektorových objektů je nezbytná i podpora vkládání vlastních rastrových obrázků [◎ II.1.B]. Ty by měly být i znovu použitelné ve více otázkách, bez nutnosti jejich duplikace na serveru a následného vícenásobného stahování, bude-li obrázek v různých otázkách téhož testu. Dále by měly být podporovány alespoň základní animace (např. změna měřítka, rotace, pohyb, mizení apod.) [◎ II.1.C].

Mimo klasických prvků pro řešení, jako je zaškrtávací políčko, textový vstup apod. by mělo být možné v otázkách používat prvky, které zkoušený bude moci přesouvat metodou drag&drop[2] [◎ II.1.D], jež testy v HTML málokdy podporují.

Vhledem k požadované komplexnosti, kterou otázky mají podporovat, by měly být k dispozici nástroje zkracující zbytečně rozsáhlé či opakující se definice [◎ II.2.C] a to jak v rámci částí jednotlivých otázek [◎ II.2.A], tak i celkově, pro typové otázky [◎ II.2.B]. Výčet standardních typů podporovaných otázek by přitom neměl být konečný, ale libovolně uživatelsky rozšiřitelný.

2.2.1 Automatické vyhodnocování [◎ II.3]

Každý uskutečněný test by se měl v co nejvyšší možné míře vyhodnocovat automaticky. Musí tedy být vytvořen systém definice správných odpovědí pro každý možný bodovatelný objekt otázek a to nejlépe fuzzy způsobem, aby bylo možné posuzovat i částečně správná řešení a jejich míru [◎ II.3.A]. Zároveň je třeba stanovit uživatelsky snadno spravovatelný způsob určování vah, jak jednotlivých otázek v testu, tak i jejich dílčího obsahu.

Fuzzy hodnocení by se mělo týkat i otevřených textových odpovědí [◎ II.3.B], kde by mělo být možné posoudit míru podobnosti zadaného textu se správnou odpovědí a nehodnotit tak každou sebemenší odlišnost nulovým počtem bodů.

[2] drag&drop (táhni a pusť) je metoda, při níž se objekty na ploše přesouvají pomocí myši tak, že se nad objekt umístí kurzor, stiskne se tlačítko myši, čímž se objekt „uchopí", přetáhne se do požadované pozice, kde se opět „uvolní" tak, že se pustí do té doby stále stisknuté tlačítko myši

Veškerá tato automatická vyhodnocení by měla být natolik výpočetně optimalizovaná, aby je bylo možné provádět ihned v okamžiku jejich vzniku na straně klienta, s možností nastavení jejich zobrazování v reálném čase. To by bylo užitečné při prezentačních a výukových testech, ale také při jejich autorském „ladění".

Na základě dosažených výsledků by se, po dokončení testu či konečném uzavření otázky (povolí-li to zadavatel testu), měla zkoušenému kromě automatického procentního či bodového vyhodnocení zobrazit i autorem otázky zadaná zpětná vazba [◎ II.3.C]. Tou by mělo být zkoušenému nejen sděleno, bylo-li jeho řešení správné či nikoli, ale především proč tomu tak je.

I přes všechny uvedené automatické způsoby evaluace by měl hodnotitel testu mít také možnost ovlivnit výsledek testu jako celku bodovou penalizací či bonusem. Tyto ruční zásahy by však byly evidovány zvlášť a ve výpočtu celkového skóre rozlišitelně vyznačeny, kvůli zachování objektivnosti hodnocení.

Výsledný protokol ze zkoušení by měl být vždy uchován v databázi systému s možností exportu pro jeho případnou archivaci [◎ II.3.D]. Tento protokol by pak měl systém dokázat načíst, ať již v budoucnu dojde k jeho libovolnému upgrade, tzn., že musí být vždy zpětně kompatibilní.

2.3 Testovací rozhraní [◎ III]

Tato část [◎ III.1] bude sloužit výhradně pro testování, tj. spouštění připravených testů, jejich zpracování zkoušeným, odeslání výsledků na server a zobrazování rozborů testů zkoušeného včetně zpětné vazby.

V průběhu testování by měla být dostupná ochrana proti podvodům, jako je manipulace se systémovým časem [◎ III.2.A] a opisování z internetu [◎ III.2.B], tzn. přepínání oken a záložek internetového prohlížeče. Systém by měl být také odolný proti dočasnému přerušení připojení na internet [◎ III.2.C], což by nemělo nijak narušit průběh již započatého testování, ani zcela znemožnit následné odesílání výsledků na server. Komunikace se serverem pak musí být také zabezpečena [◎ III.2.D] proti útokům a podvrženým zprávám (podrobněji viz kap. 2.3.2).

2.3.1 Míchání [◎ III.3]

Má-li být test při konečném množství otázek několikanásobně použitelný, měl by být pokaždé zadán trochu jinak. Tím je myšleno v první řadě míchání otázek [◎ III.3.B], popřípadě jejich vhodný výběr z vícera možných, je-li jich k dispozici více, než má být vybráno. V druhé řadě jde o interní míchání uvnitř každé otázky [◎ III.3.A], aby tato při opakovaném použití nebyla stále stejná.

Interní míchání otázky [◎ III.3.A]

Každou jednotlivou otázku by bylo dobré mít možnost sestavit tak, aby se při testování dokázala vygenerovat v co možná největším množství různých variant, tzn. různé pořadí prvků, různé znění vět, různá slova, náhodné hodnoty, odlišný vzhled apod.

Tím by se při opakovaném položení téže otázky předešlo mechanickému naučení se odpovědí způsobem např. „správná odpověď je b)", „3. od konce", „nejdelší", „nejkratší", „začíná slovem „Není" apod. Zároveň, při hromadném zkoušení v jedné třídě, by se omezila možnost vzájemného opisování. Ideální by byla tak rozmanitá definice otázky, pro jejíž stoprocentní jistotu zodpovězení

libovolné verze by zkoušený musel dokonale chápat danou část tématu a odpověď neodhadnout, ale odvodit ze zadání.

Při generování každé verze otázky by přitom bylo dobré mít možnost nějakým způsobem zaznamenat pro každý náhodnostní prvek v otázce, jaká hodnota mu byla přidělena a při příštím generování témuž zkoušenému, bude-li tato funkce tvůrcem testu vyžadována, náhodné hodnoty co nejvíce obměnit. Např. při výběru ze tří možností je při rovnoměrném rozdělení 33% šance výběru téže možnosti jako posledně, čili bylo by vhodné, při výběru zvýhodnit zbylé dvě možnosti, které dosud vybrány nebyly.

Výběr otázek [◎ III.3.B]

I když se otázky dokáží do určité míry vnitřně modifikovat, takže je lze bez obav používat opakovaně, dojde-li k naučení se tématu zkoušeným, což je ostatně cílem výukového testu, je zbytečné mu tyto otázky pokládat stále znovu ve stejné míře jako ty, které se dosud dostatečně nenaučil. Jejich úplné vyřazení z výběru ovšem také není ideálním řešením.

Výběr z většího souboru otázek do testu, by mělo být možné nastavit ve více módech. Zcela náhodný výběr by byl jedním z nich. Další by měl dokázat přednostně vybírat otázky dosud neřešené, nebo řešené chybně. Za tímto účelem bude třeba vytvořit a implementovat spojitý matematický model, který bude optimalizovat výběr otázek uvedeným způsobem.

Také bude nezbytné vyřešit seskupování otázek, aby např. z dané skupiny byla do testu vybrána vždy maximálně jedna, dvě nebo jiný nastavitelný počet. Možná bude též varianta výběru právě jedné otázky, tedy její povinné zahrnutí do výběru (např. pokud se mají zahrnout 3 otázky probírající novou látku a libovolné další 4 z kterékoli starší látky).

2.3.2 Bezpečný přenos dat v internetu

Testovací portál bude v podstatě aplikací typu klient-server, přičemž veškerá komunikace mezi těmito dvěma stranami bude probíhat přes veřejnou síť internet. Je tedy nezbytné zvolit nebo vytvořit vhodný komunikační postup, který bude dostatečně rychlý, propustný, automatizovaný a bezpečný, jak z hlediska aktivních útoků, tak pasivního odposlouchávání probíhající komunikace.

Autentizace požadavků a dat zasílaných na server [◎ III.2.D]

Komunikaci mezi klientskou částí aplikace a serverem, bude zapotřebí zabezpečit, aby nemohlo dojít k narušení systému neoprávněnými osobami či útočnými programy. Za tímto účelem bude třeba navrhnout univerzálně použitelný systém autentizace každé instance klientské aplikace a zároveň i autorizace uživatele, který ji používá. Pouze na základě jejich ověření pak veřejně dostupná webová služba bude vykonávat příslušnou činnost.

Tento proces zároveň musí ochránit uživatelské heslo v případě zachytávání komunikace třetí stranou i bez použití protokolu https[3], který má také své slabiny [3]. Taktéž musí být odolný proti man-in-the-middle[4] útoku formou opakovaného zasílání týchž zpráv. Navíc by měl být použitelný na klasickém ASP.NET hostingu bez potřeby instalace jeho dalších rozšíření, respektovat bezstavovost webového serveru a být schopný autentizace v rámci každé jednotlivé komunikace se serverem, nezávislé na těch předchozích či budoucích.

[3] https – Secure HyperText Transport Protocol – http protokol s podporou SSL/TLS šifrování [124]
[4] man-in-the-middle – „člověk uprostřed" komunikačního kanálu ovlivňuje posílaná data

Vzhledem k možné potřebě kooperace s jinými systémy třetích stran by tento postup měl být snadno implementovatelný i v systémech, které nejsou založeny na platformě Microsoft .NET Framework a požívat pouze triviální algoritmy a výpočetní funkce dostupné i v jiných programovacích jazycích.

Šifrování dat [◎ III.2.E]

Data přenášená přes internet, nebude-li standardně použit zabezpečený komunikační protokol, by měla mít možnost být zabezpečena proti odposlouchávání třetí stranou. To platí i pro výstupní data aplikace, např. výsledky z testu, které se nepodaří přímo odeslat na server z důvodu přerušení spojení.

Šifrovacích algoritmů existuje celá řada, i přesto bude navržen zcela nový, a následně porovnán s těmi stávajícími, pro určení vhodnosti jeho použití v jednotlivých případech.

2.3.3 Konektivita [◎ III.4]

Výslednou aplikaci by mělo být možné provozovat ve dvou formách a to jako plnohodnotnou aplikaci [◎ III.4.A] s přihlášením, přehledem testů a předchozích výsledků, s testovací částí a možností zobrazení rozboru testu již ukončeného. Druhá varianta by umožňovala přímé spouštění testu [◎ III.4.B], přičemž autentizační údaje uživatele a volba testu by byly parametry předané aplikaci jiným systémem z vnějšku. Tím může být například libovolný LMS, ale i jakákoli jiná webová aplikace, schopná výměny dat.

Obr. 1 – Ilustrace propojení s on-line LMS v rámci jedné webové stránky

Vstupem budou údaje identifikující a jednorázově autentizující zkoušeného uživatele (viz Obr. 1). Identifikovat a jednorázovým způsobem bezpečně ověřit bude třeba i partnerský systém, aby účet v testovací aplikaci nemohl být zneužit mimo jeho rámec. Výstupem bude výsledek z proběhnutého testu, taktéž ověřený jednorázově použitelným „podpisem".

Je tedy zapotřebí definovat rozhraní pro výměnu těchto údajů [◎ III.4.C] mezi jednotlivými systémy a zároveň stanovit způsob a pravidla bezpečné, jednoznačné a jednorázové autentizace obou stran. Jednorázovost autentizace by měla zajistit ochranu proti jejímu zachycení a opakovanému použití bez vědomí partnerského systému.

Kromě autentizace partnerské aplikace a uživatele by také měla být možnost předat zabezpečené údaje o dodatečném nastavení testu pro aktuální testování [◎ III.4.D], aby se pro každou možnou kombinaci, jež by aplikace mohla vyžadovat, nemuselo předem připravovat nastavení tohoto testu.

V případě jednodušších nebo uzavřených systémů by měla být podpora i nezabezpečeného spouštění testů pouze na základě konstantní URL adresy. Takto by se daly spouštět alespoň výukové či prezentační testy, u nichž není nezbytná relevance výsledků pro konkrétní uživatele.

2.4 Administrační rozhraní [◎ IV]

Administrační rozhraní by mělo umožňovat správu systému, jeho jednotlivých částí a dat. Půjde především o správu uživatelů, jejich potvrzování, zařazování do skupin atd. Dále o sestavování a nastavování testů z předpřipravených otázek [◎ IV.1]. V neposlední řadě by měly být dostupné pro rozbor a hodnocení jak jednotlivé výsledky zkoušení, tak i jejich celkové přehledy, včetně možnosti exportu.

Jednotlivé vytvořené testy by mělo být možné nastavit pro opakované spouštění, ale za různých podmínek [◎ IV.2.A]. Tím je myšleno nejen různé nastavení pro testovací prostředí, ale i vymezení např. pouze určitého druhu (druhů) otázek [◎ IV.2.B] a také vymezení přístupu pouze pro určité uživatele či uživatelské skupiny [◎ IV.2.C], popř. jen konkrétní počítače (např. pouze určitou učebnu) [◎ IV.2.D].

Zpřístupňování testu by také mělo být možné omezit na určitý čas, včetně možnosti jeho opakování [◎ IV.2.E]. Za tímto účelem bude třeba vytvořit univerzální způsob zápisu periodických časových intervalů, přizpůsobitelný vlastnostem kalendáře (např. „každý týden" nebo „každý druhý pátek od 10:00 do 11:00", apod.), včetně možnosti omezit počet pokusů na každé takové opakování. Pro rychlé vyhledávání by zároveň mělo být snadné v takovýchto definicích rychle identifikovat aktuálně aktivní periody již při výběru dat přes SQL.

Výukové autotesty by sice měly sloužit jako dobrovolný nástroj pro tříbení vědomostí zkoušeného, mohlo by ale být žádoucí studenty k této dobrovolné činnosti nějak motivovat. Souhrnná známka, bonusové body nebo i připuštění ke zkoušce by mohlo být podmíněno tzv. absolvováním testu [◎ IV.2.F]. Princip by měl spočívat v nastavení kritérií, které musí zkoušený splnit, aby byl test absolvován. Mohlo by například jít o následující podmínky:

- Splnit test nad určitou minimální hranici
- Opakovaně X-krát splnit test nad určitou minimální hranici
- Průměrný výsledek posledních X pokusů musí přesáhnout určitou minimální hranici
- Průměrný výsledek za posledních X dní musí přesáhnout určitou minimální hranici
- X nejlepších výsledků z posledního týdne musí mít průměr nad určitou minimální hranici
- Průměrný či maximální výsledek z testování v rámci jednoho dne musí X dnů po sobě být nad určitou hranicí apod.

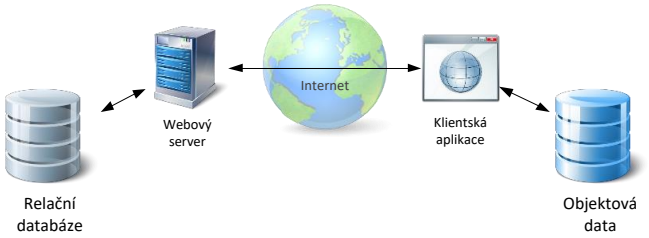
2.4.1 Framework [◎ IV.3]

Administrační rozhraní by nemělo být vytvořeno klasickým způsobem, kdy se jeho jednotlivá okna programují každé zvlášť, ale měl by pro něj být vytvořen a odzkoušen tzv. konfigurační framework. Při tomto přístupu se vytvoří pouze jádro systému, které bude jednotlivá okna automaticky generovat až v případě jejich potřeby, na základě jednoduchých externích konfiguračních dat. Ta by měla v co nejjednodušší, ale vše podstatné pokrývající formě, specifikovat veškerý obsah a vazby jednotlivých oken celého administračního systému.

Tento způsob vývoje by v případě úspěchu mohl být inspirací pro vývoj desktopových či RIA informačních systémů. Ty obvykle obsahují desítky oken téhož druhu a jejich individuální vývoj je zbytečně nákladný, stejně jako i následná údržba. Tento frameworky by přitom mohl umožnit rozsáhlé změny v systému nejen bez nutnosti jeho rekompilace, ale i restartu aplikace.

2.4.2 Přenos dat mezi serverem a klientskou aplikací [◎ IV.4]

Proces přenosu dat ze serveru na klientskou aplikaci by, i vzhledem k frameworkovému vývoji, bylo dobré navrhnout univerzálním jednotným způsobem. Ten by měl dokázat „zajistit" vždy a pouze požadovaná data, tzn. stáhnout je ze serveru nebo ověřit, že již byla stažena dříve a zbytečně nestahovat nic navíc. Stažená relační data by pak na klientské straně byla načtena do vzájemně propojených objektů a umožňovala tak využití mechanizmu LINQ to Object[5].



Obr. 2 – Schéma přenosu dat mezi serverem a klientem

Objekty v klientské aplikaci by měly být libovolně rozšiřitelné o další potřebné kalkulované vlastnosti, metody a události. Definice vztahů pro ORM[6] by přitom měla být co nejméně pracná a uvedena pouze jedenkrát na jednom místě, nejlépe v přímém propojení s definicí samotných objektů. Zvlášť by měly být řešeny přenosy rozsáhlejších hodnot (např. binární data).

Veškeré změny stažených zdrojových dat musí být evidovány s možností hromadného asynchronního odeslání těchto změn zpět na webový server, který je uloží do hlavní relační databáze. Klientská aplikace pak obdrží potvrzení úspěchu či neúspěchu tohoto uložení a v případě úspěchu také data nově vzniklá při jejich uložení (např. autoinkrementální ID), v případě neúspěchu chybové hlášení identifikující příčinu vzniklé chyby.

Výsledné řešení by mělo být natolik univerzální a nezávislé na tomto projektu, aby mohlo být bez nutnosti vlastního přizpůsobování použito i v projektech dalších.

[5] LINQ to Object – v .NETu integrovaný dotazovací jazyk nad objekty v paměti, podobný SQL [102 str. 63]
[6] ORM – objektově relační mapování, tzn. načtení dat z klasické relační databáze do objektů a zpět

3 ANALÝZA SOUČASNÉHO STAVU

3.1 Existující systémy testování [◎ I.1]

(z části publikováno v [ap-4])

3.1.1 Testovací systémy integrované do LMS

Systémy pro řízení studia (LMS – Learning Management System), zvané též někdy Systémy pro řízení kurzů (CMS – Course Management System) usnadňují tvorbu, používání a správu online kurzů především tím, že poskytují rozhraní, umožňující vytvářet prezentaci kurzu, soubor výukových nástrojů usnadňujících učení, komunikaci a spolupráci a soubor nástrojů administrativních, které pomáhají instruktorovi a manažerovi v procesu správy, vedení a zlepšování kurzu. [4]

Nástroje pro testování bývají v těchto systémech obvykle standardně integrovány, neboť jsou podstatnou součástí celého konceptu e-learningu. V této části tedy budou uvedeny některé z nich a podrobněji rozebrány jejich testovací subsystémy.

WebCT

WebCT je prostředí pro tvorbu a provoz on-line kurzů, které po jejich návrhářích nevyžaduje technickou odbornost. Poskytuje rozhraní, jež usnadňuje průběh tvorby (organizaci obsahu, provoz kurzů atd.), stejně jako soubor vzdělávacích a administračních nástrojů. Tyto nástroje zahrnují správu studentů a kontrolu přístupu, úložiště a distribuci známek, konferenční systém, chat, interní elektronickou poštu, oblast pro skupinovou prezentaci, automaticky vyhodnocované autotesty, on-line kvízy, slovník s vyhledáváním, přehled pokroku studentů, vyhledávání v obsahu kurzů atd. [5]

První verze WebCT (Web-based Course Tool) byla vytvořena již v roce 1996 na University of British Columbia. Následně byl projekt převeden pod vlastní společnost WebCT Educational Technologies Corporation, která byla roku 2006 prodána konkurenční firmě Blackboard Inc. [6]

WebCT na klientské straně používá HTML, JavaScripty a Java applety. Webová stránka je rozdělena na rámce, které byly používány především v době vzniku první verze WebCT a dnes se s nimi setkáme již spíše výjimečně.

WebCT, jako ostatní podobné virtuální prostředí, nabízí nástroje umožňující přípravu různých druhů testů s několika typy testových otázek. [7]

Modul pro testy je ve WebCT zpracovaný poměrně rozsáhle a obsahuje mnoho užitečných nastavení. Celkové nastavení testu například umožňuje test skrýt nebo zobrazit ve studentských seznamech přístupných testů, zobrazovat otázky v testu všechny najednou nebo postupně, časově test omezit (na minuty, hodiny ale i dny), omezit počet pokusů, nastavit minimální čas na pokus (dříve nepůjde test odevzdat), míchat otázky, určit, který z pokusů bude brán v potaz pro hodnocení (první, poslední, nejlepší nebo průměr ze všech), povolit nebo zakázat zobrazení zpětné vazby (vyhodnocení testu), nastavit rozsah tohoto zobrazení a upozornění e-mailem při splnění testu. Kromě toho lze testu nastavit čas (od – do), kdy pouze bude přístupný pro spuštění a určit IP adresu, včetně její masky, ze které pouze půjde test spustit, což zabrání, aby např. při použití testu pro ostré zkoušení s ním mohl manipulovat někdo mimo studentů ve vymezené učebně. WebCT také umožňuje všem svým položkám (včetně testů) nastavit kritéria zpřístupnění, v nichž lze specifikovat výčet studentů nebo skupin studentů, pro které pouze bude test viditelný a kdy (mohou se

tak naplánovat testy na jednotlivá cvičení, podle jejich času a účastníků) a také je možné zpřístupnění testu podmínit dalšími kritérii, jako například dosažením minimálního výsledku v testu jiném.

Testové otázky se vytvářejí nezávisle na jednotlivých testech a ukládají se do tzv. „databáze otázek", kde je lze hierarchicky třídit do skupin. Každá otázka obsahuje název (především pro účely jejich identifikace v přehledu databáze otázek) a zadání, neboli text s podporou HTML tagů a jeho vlastním jednoduchým WYSIWYG[7] editorem, podporujícím i vkládání složitých matematických vzorců. Pro jejich sestavování je k dispozici opět interní editor v JavaApplet, s možností importu vzorců zapsaných v MathML[8]. Text u některých typů otázek může obsahovat zvláštní znaky (obvykle proměnné v hranatých závorkách, např. `[x]`), které jsou pak nahrazovány hodnotami nebo jsou součástí odpovědí. Ke každému zadání lze také přiložit obrázek. Otázky, kromě dalších individuálních možností u jejich specifických typů, taktéž mohou obsahovat obecnou zpětnou vazbu (vysvětlení správného řešení) a interní poznámky designéra. Podporovány jsou následující typy otázek, přičemž podrobněji rozebereme i postup jejich zadávání.

- **Výpočty** – Volný text zadání, ve kterém jsou v hranatých závorkách vyznačeny proměnné. Následně se do zvláštní kolonky ve speciálním formátu, podobném zápisu vzorců v MS Excel, zapíše výraz, v němž se opět vyskytují tytéž proměnné. Tlačítkem „Analyzovat proměnné" se identifikují proměnné ve vzorci a je sestaven jejich seznam. Pro každou se pak nastaví rozsah, včetně podpory desetinných míst, a automaticky se vygeneruje 10 až 100 možných zadání příkladu, s vypočteným výsledkem. Náhodně vygenerované hodnoty lze upravit a výsledek přepočítat. Z této předpřipravené sady zadání se při testování vždy jedna náhodně vybere. Pro řešení je možné nastavit toleranci (± určitá hodnota nebo procento). Výsledek je vždy pouze jeden (ve vymezeném rozsahu) a hodnocení je buď 0, nebo 100%. Je také možné požadovat uvedení jednotek a této odpovědi přiřknout určitou procentní váhu z celkového výsledku, ovšem počítá se pouze s jedněmi konkrétními jednotkami, čili pokud např. zkoušený výsledek místo v kilogramech zapíše v gramech, bude řešení vyhodnoceno jako chybné.
- **Kombinace** – K otázce je zvlášť uvedena sada pojmů označených zástupnými znaky (A, B, C, …), z nichž se následně pro nabízené odpovědi vytváří kombinace, ze kterých se pouze jedna označí jako správná. Mimo kombinace zástupných znaků („A, B" nebo „B, C, D") lze ke každé kombinaci dopsat i krátký text (např. „Ani jedna z uvedených"). Hodnocení je tedy i zde buď 0, nebo 100%. Pořadí nabízených kombinací, jakož i pořadí v seznamu pojmů, je pevné a nelze nastavit jeho míchání.
- **Vyplňte prázdná pole** (cloze) – Stejně jako u výpočtů je otázka zapsána jako volný text, přičemž určitá slova mohou být uzavřena do hranatých závorek. Tato jsou označena pro doplňování a tlačítkem „Generovat odpovědi" se pro každé takové slovo vytvoří jeho podrobnější nastavení. Každému z nich se dá přiřadit procentní podíl na celkovém výsledku (součet musí být 100%). Pro porovnání zadaného a předepsaného textu je k dispozici operátor „obsahuje" (zadaný text se vyskytuje kdekoli v předepsaném), „je rovno" a „regulární výraz" (předepsaný text je regulárním výrazem[9] a zadaná odpověď mu musí odpovídat). Odpovědi mohu i nemusí

[7] WYSIWYG – What You See Is What You Get („co vidíš, to dostaneš"), typ editoru textu, v němž je přímo vidět jeho formátování
[8] MathML - Mathematical Markup Language – na XML založený formát zápisu matematických vzorců, viz [w81]
[9] regulární výraz – text popsaný pomocí vzorů, používají se pro ověřování údajů nebo vyhledávání [48]

být *case sensitive*[10]. Každé odpovědi lze vytvořit i alternativy se stejnými možnostmi nastavení, přičemž u nich nastavená procentní hodnota může v případě jejich volby být v součtu s ostatními i vyšší než 100%. Vzájemné vazby jednotlivých odpovědí možné nejsou (např. možnost záměny pořadí dvou slov).

- **Přeházená věta** – I zde se v textu otázky uzavřou určitá slova do hranatých závorek, tlačítkem „Generovat odpověď" jsou detekovány a lze jim pro toto pořadí nastavit procentní hodnotu. Tato slova totiž budou v textu otázky nahrazena rozevíracími seznamy, ve kterých zkoušený volí vždy jedno z nich. Každé slovo přitom smí být ve větě použito pouze jednou. Je možné vytvořit i alternativní pořadí s jinou procentní hodnotou za jeho sestavení. Přidat do seznamů distraktory[11] nelze.
- **Přiřazování** – V otázce je možné vytvořit alespoň pět párů výrazů nebo i delších textů. Při testování se pak ke každému výrazu zadanému vlevo vybírá z rozevíracího seznamu jedno ze slov, která byla zadána vpravo, přičemž je možné i jejich opakování. Hodnocení lze nastavit jako procentní podíl správně přiřazených dvojic z jejich celkového počtu, „všechno nebo nic" nebo „správné mínus chybné".
- **Výběr z odpovědí** (multiple-choice) – Pod zadáním otázky je možné vyplnit pět nebo více nabízených textových odpovědí. Ke každé lze uvést zpětnou vazbu, která se zobrazí pouze, je-li příslušná odpověď zvolena. U odpovědí se zaškrtává, je-li odpovědí správnou a v tom případě je její procentní hodnota automatická (100%), přičemž u nesprávných lze tuto hodnotu nastavit ručně, včetně záporných hodnot. Nastaven může být způsob výběru (dá-li se vybrat jen jedna z odpovědí nebo více), rozložení odpovědí (horizontální nebo vertikální), označení odpovědí (čísla nebo písmena), mají-li se odpovědi řadit náhodně, způsob hodnocení („všechno nebo nic" nebo kumulativní) a má-li být povoleno i celkové záporné skóre, nebo jej případně zastavit na nule.
- **Odstavec** – Jedná se o dlouho otevřenou odpověď, kterou musí ručně ohodnotit zadavatel testu. Kromě zadání otázky může být i nastaven text, který bude předvyplněn v položce pro vepsání odpovědi a také může být zvlášť uvedena správná odpověď, která se zkoušenému zobrazí po dokončení testu v rámci zpětné vazby.
- **Krátká odpověď** – Na zadanou otázku se odpovídá vepsáním textu. Řešení přitom může mít více různě hodnocených alternativ. Je také možné zobrazit pro vstup více než jednu položku. Možnosti porovnání textu jsou stejné jako u otázky typu „vyplňte prázdná pole".
- **Pravda nepravda** – Otázka musí být položena tak, aby se pro odpověď dalo jednoznačně zvolit mezi tím, je-li něco pravda nebo nepravda (dichotomická úloha). Popisek těchto odpovědí nelze změnit. Výsledkem je buď 0, nebo 100%.

Testy se vytvářejí tím způsobem, že se do nich již pouze přiřazují otázky z celkové databáze otázek. Každé z nich je možné pro aktuální test přiřadit jiné bodové hodnocení, které je maximem při dosažení 100% v rámci otázky, nebo procentním poměrem z těchto bodů při částečně správném řešení.

Otázky se mohou i v rámci testu seskupovat do skupin (pouze jednoúrovňových), u kterých lze definovat počet otázek (menší nebo roven celkovému počtu otázek ve skupině) pro jejich

[10] case sensitive – porovnávání dvou textů, při kterém závisí i na velikosti jednotlivých písmen
[11] distraktor – nesprávná odpověď, která je zkoušenému nabízena jako alternativa té správné [44 str. 35]

náhodný výběr z této skupiny. Celá skupina pak má definováno bodové skóre, na jehož dosažení se rovnoměrně podílí jednotlivé otázky této skupiny.

Zpětná vazba, je-li v nastavení testu povolena, se zkoušenému může zobrazit ihned po dokončení testu. V ní je vidět celkový bodový výsledek z testu a po jeho rozkliknutí i celý obsah testu včetně zvolených odpovědí. U každé otázky je přitom označeno, byla-li odpověď zkoušeného správná či nikoli, v tom případě je označena i správná odpověď a u určitých typů otázek je uveden i text zpětné vazby (u každé odpovědi nebo celkově za otázku). Totéž si může prohlédnout i zadavatel testu, včetně celkového statistického souhrnu za všechny uživatele.

Testy ve WebCT jsou poměrně dobře propracované a do umění jejich vytváření se dá rychle proniknout. Při správném nastavení mohou sloužit jako autotesty, dotazníky i plnohodnotné zkouškové testy. Díky možnosti časového omezení a návaznosti zpřístupňování jednotlivých testů lze sestavit plán testů na celý semestr dopředu tak, aby studenti byli motivováni k průběžnému studiu jednotlivých lekcí. Výhodou je i automatizovaný převod výsledků (známek) do celkového přehledu hodnocení, kde mohou být i výsledky jiných aktivit (např. úkolů).

Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment [8]) *je velmi populární LMS, poskytující vysoce konfigurovatelné on-line rozhraní, které zahrnuje širokou škálu činností obecně dostatečných pro standardní kurzy.* [9] *Moodle je open-source* (vydaný pod licencí GPL) *a byl přijat mnoha univerzitami jako jejich celouniverzitní LMS, což platí i o výzkumných skupinách nebo jednotlivých pedagozích. Popularita Moodle je podložena velkým počtem evidovaných instalací, jakož i počtem dostupných rozšíření.* [10] K datu 21.2.2012 bylo registrováno 66 127 instalací, s 5 851 638 kurzy a 56 862 180 uživateli, přičemž kurzy obsahovaly 112 489 517 testových úloh (otázek). [w47] Česká verze Moodle je zdarma dostupná na [w48].

Moodle standardně podporuje import obsahu ve formátech LAMS, SCORM, Social format, Topic format a Weekly format. [8] Lze do něho ale importovat i testy připravené např. v MS Excelu [11].

Testy se i zde skládají z otázek uložených v „bance úloh". Každé se dá pro daný test přidělit určitá bodová hodnota („známka"), která bude odstupňována dle procenta správnosti řešení této otázky. Úlohy se rovnou skládají na určité stránky testu (není-li aktivní jejich míchání), případně se naráz vkládají jejich celé skupiny.

V otázkách je pro každý text v testu (zadání otázky i odpovědí) k dispozici jednoduchý WYSIWYG editor HTML, který mimo jiné podporuje obrázky (jejich vkládání a zarovnávání do textu), tabulky, vlastní editor matematických vzorců v JavaApplet, vkládání formátovaného textu z Wordu a dokonce i kontroly pravopisu. Podporované typy testových úloh (otázek) jsou tyto (dle [w45]):

- **Výběr z možných odpovědí** (multiple-choice) – Umožňuje výběr jedné nebo více odpovědí ze seznamu.
- **Pravda/Nepravda** – Jednodušší varianta úlohy s více odpověďmi. Na dané tvrzení nabízí pouze dvě možné volby – „Pravda" a „Nepravda".
- **Krátká tvořená odpověď** – Odpověď je tvořena jedním nebo několika slovy, které jsou porovnány s různými modelovými odpověďmi. Ty mohou používat i zástupné znaky.

- **Numerická úloha** – Odpověď je tvořena číselným údajem doplněným případně o jednotky. Odpověď je hodnocena na základě číselného porovnání s danou tolerancí s různými modelovými odpověďmi.
- **Vypočítávaná úloha** – Vypočítávaná úloha se chová jako numerická úloha, ale konkrétní hodnoty jsou pro každého zkoušeného náhodně vybrány z jisté množiny.
- **Dlouhá tvořená odpověď'** – Umožňuje odpovědět několik vět nebo odstavců. Hodnocení musí poté proběhnout ručně.
- **Přiřazování** – Odpověď na každou podúlohu musí být vybrána ze seznamu možností.
- **Přiřazování pro náhodně vybrané úlohy s krátkou tvořenou odpovědí** – Chová se jako přiřazovací úloha, jejíž hodnoty jsou vybrány náhodně z úloh s krátkou odpovědí z dané kategorie.
- **Doplňovací úloha** (cloze) – Úloha tohoto typu je velmi flexibilní, ale může být vytvořena pouze zadáním zdrojového textu ve speciálním formátu. Tento text obsahuje kódy, které vytvářejí komplexní úlohu s vloženými dílčími úlohami s více odpověďmi, krátkou odpovědí či numerickou úlohou.
- **Jednoduchá vypočítávaná úloha** – Jednodušší varianta vypočítávané úlohy. Vypočítávaná úloha se chová jako numerická úloha, ale konkrétní hodnoty jsou pro každého studenta náhodně vybrány z jisté množiny.
- **Vypočítávaná úloha s více možnostmi** – Vypočítávaná úloha s více možnostmi se chová jako obyčejná úloha s více možnostmi, ale nabízené odpovědi se pro každého studenta vypočítávají jako výsledek daného vzorce s náhodně vybranými hodnotami z jisté množiny.

Nastavení testu umožňuje omezit přístupnost k testu na vymezené časové období (od – do), nebo přístup k němu podmínit známkou nebo výsledkem testu (či i více testů) jiného. Čas na řešení testu lze též omezit, stejně jako i počet pokusů na test, přičemž finálním výsledkem může být první pokus, poslední pokus nebo průměr ze všech pokusů. Lze též definovat minimální prodlevu mezi pokusy (odděleně mezi prvním a druhým a mezi těmi dalšími). Pořadí úloh (otázek) může být náhodné a je možné i nastavit, kolik otázek bude zobrazeno na jedné stránce. Přístup k testu může být zajištěn heslem i IP adresou (či více adresami zapsanými v CIDR[12] formátu). [12]

Během testování může být povinně zobrazena fotografie přihlášeného uživatele, aby se snadno dala zkontrolovat jeho totožnost při ostrém zkoušení. Po ukončení testu může být zkoušenému, dle dosažených bodů, zobrazena určitá celková zpětná vazba (celkové vyhodnocení testu). Zobrazení částečné zpětné vazby (u jednotlivých odpovědí, celkového řešení otázky, komentář, obecnou reakci, získané body a celkovou reakci testu) lze pro každý druh zvlášť povolit nebo zakázat po každém testu, omezit její viditelnost pouze po dobu zpřístupnění testu nebo ji naopak zobrazit až po uzavření testu.

Pro průběh testování lze nastavit zvláštní režim prohlížeče, v němž je pro test otevřeno samostatné okno roztažené přes celou plochu obrazovky, jsou skryty veškeré jeho možné ovládací prvky a JavaScript brání označování textu, jeho kopírování a vkládání. Okno však lze samozřejmě přepnout (Alt+Tab) na jiné nebo zmenšit pomocí myši.

Modul testy je v Moodle velmi dobře zpracovaný a lze jej využívat jak k autotestům pro opakování a oživení látky, tak i pro ostré známkované zkoušení. Je-li ovšem systém umístěn na exter-

[12] CIDR – Classless Inter-Domain Routing [w60]

ním vzdáleném serveru (mimo interní síť), je práce s nastavováním testu poněkud pomalá, neboť při každém potvrzení tlačítka dojde k „post backu", tedy přenačtení celé stránky, což působí poněkud rušivě a zpomaleně. Širší využití AJAXu[13] by mohlo práci s testy značně zefektivnit.

iŠkola

iŠkola.cz není LMS, ale *univerzální internetový server, umožňující každé škole vést elektronickou agendu a plně využívat moderní informační technologie ve výuce a při komunikaci školy mezi pedagogy, s žáky, rodiči a okolím.* [w38] Je tedy určena spíše pro základní a střední školy jako elektronická on-line třídní kniha, žákovské knížky apod.

Součástí této služby je i modul Testy, který umožňuje jejich tvorbu, zadávání, průběh i vyhodnocení. Otázky takového testu mohou být dvou typů a to výběr jedné z nabízených odpovědí (správná nemusí být pouze jedna odpověď) nebo krátká textová odpověď. Druhý typ otázky je ale potřeba vždy vyhodnotit ručně.

Každá otázka má počet správných bodů za své správné řešení nebo penalizačních bodů za řešení chybné. Výsledkem testu je suma těchto bodů, přičemž pro automatické známkování lze nastavit bodové rozsahy pro jednotlivé známky 1-5. Ty se pak hromadně dají přepsat do modulu pro klasifikaci studentů.

Text otázky ani odpovědí neumožňuje žádné formátování, lze však ke každé otázce přiložit max. jeden obrázek. Testy se pro aktuální okamžik zpřístupňují (znepřístupňují) jejich ruční aktivací (deaktivací). Počet pokusů na test není nijak jinak omezen. Časové omezení na splnění jednoho testu však možné je. Otázkám ani odpovědím není možné nastavit míchání.

Modul Testy v iSkola.cz je pouze doplňkem celého systému a v aktuální podobě není moc prakticky využitelný. Na jednorázové jednoduché krátké opakovací testy by však mohl stačit, za předpokladu, že škola již tento systém používá.

Odpovědníky

Agenda Odpovědníky je součástí informačního systému Masarykovy univerzity, která vyučujícím umožňuje vytvářet, zadávat a automaticky vyhodnocovat testy. Tento modul je plně integrovaný do informačního systému IS MU a umožňuje tak v jeho rámci správu uživatelů, kontrolu přístupu k testům a automatizovaný export výsledků do hodnotící agendy.

Vytváření Odpovědníků v IS MU je dvoustupňové. Vyučující si v systému v rámci předmětu vytváří a udržuje databázi otázek, které může třídit do skupin, např. dle obtížnosti. V případě potřeby pak z těchto otázek snadno připraví konkrétní test. [13]

Mezi podporované typy otázek patří *výběr jediné správné odpovědi, otázky typu multiple-choice, textové odpovědi, spojování souvisejících výrazů („matching"), úprava slovosledu, numerická odpověď (s možností nastavení intervalu) nebo odpověď matematickým výrazem v syntaxi LaTeX.* [14]

Otázky jsou v základu zapisovány v hybridním HTML kódu, který obsahuje speciální značky, jež jsou při zadávání testu nahrazeny příslušnými formulářovými prvky. Zároveň je z nich odvozena i správná odpověď, se kterou je během automatického vyhodnocování porovnávána odpověď zvolená či zadaná zkoušeným.

[13] AJAX – Asynchronous JavaScript and XML – technologie pro změnu obsahu webové stránky bez nutnosti jejího přenačítání [123]

Pro klasické typy otázek byly vytvořeny pomocné on-line aplikace, které umožňují zápis otázky provést v co nejjednodušším formátu a tyto aplikace jej následně převedou do plnohodnotného kódu zpracovatelného systémem (např. *cloze* [w35]). Nástroj Quest-o-mat [w36] umožňuje návrh a automatické generování kódu otázek složitějších na formátování, s využitím obrázků, libovolně rozmístěných formulářových prvků a spojnic (čar). Pro přidávání multimediálních objektů (video, audio a obrázek) slouží další pomocná aplikace GEM (Generátor kódu multimediálních prvků, viz [w33]), která opět vytváří kód, pro přímé vložení do kódu otázky.

Pro test lze nastavit míchání otázek i odpovědí, včetně možnosti výběru jen některých z databáze otázek nebo jejich podskupin. Rozsáhlejší interní míchání uvnitř obsahu otázky nastavit nelze, k dispozici je ale nástroj Multiquest [w34], který na základě jednoduchého nastavení dokáže vygenerovat velké množství zadání (stejných otázek) s různými parametry (číselnými hodnotami). To je vhodné např. pro vypočítávané úlohy, každá otázka však musí být zvlášť vložena do databáze.

Pro potřeby realizace ostrého zkoušení pomocí Odpovědníků v IS MU jsou pro učitele k dispozici možnosti nastavení co nejvíce eliminující podvádění. Typické nastavení Odpovědníků pro ostré zkoušení dovolí spustit Odpovědník pouze jednou, a to osobám přihlášeným na daný zkouškový termín, z IP adresy odpovídající rozsahu adres dané učebny, v časovém intervalu odpovídajícímu době konání zkoušky, navíc s časovým limitem. Pro lepší přehled může učitel v testu zapnout zobrazení barevného pruhu, který umožňuje učiteli zdálky monitorovat pracovní plochu počítačů studentů při vyplňování testu, a fotografie zobrazovaná v pruhu pomůže při identifikaci studenta. [14] Dlouhé otevřené otázky navíc mohou být zkontrolovány pomocí služby na odhalování plagiátů.

Systém umožňuje testy zadávat a vyplňovat i v případě, že není k dispozici počítačová učebna, a to pomocí vytištění zadání a skenování řešení. *Vytvoří-li učitel sadu otázek ke zkoušce, IS z ní umí „namixovat" originální zadání pro každého studenta. Ten pak vyplňuje odpovědi do Odpovědního listu a po ukončení zkoušky ho odevzdá učiteli. Po naskenování jsou odpovědní listy automaticky vyhodnoceny ISem a dle nastavené volby učitelem může být odpovědní list uložen každému studentovi (včetně volby data, kdy má student list k dispozici, například až na závěr zkouškového období).* [15]

Masarykova univerzita Odpovědníky používá jako autotesty, pro průběžné testování studentů, při ostrém hodnoceném zkoušení, u přijímacích řízení, ale i jako anketní dotazníky. Úspora času, snadnější správa a redukce chyb lidského faktoru oproti papírovým testům je totiž značná.

Dokeos

Dokeos je open-source LMS [w15], který k 21.2.2012 vykazoval 11 302 instalací s 347 159 kurzy a 3 922 985 uživateli po celém světě (zdroj [w16]). Systém má propracovanější grafiku, než jsou jen klasické formulářové prvky, ale není lokalizován do češtiny. Modul pro tvorbu testů podporuje otázky typu multiple-choice (výběr jedné či více odpovědí i s podporou „všechno nebo nic"), doplňování textu (výběrem či psaním), přiřazování, dlouhá otevřená otázka a kliknutí na správné místo (zóny v obrázku). Nastavení podporuje časové vymezení pro přístup k testu, míchání otázek a zpětnou vazbu.

3.1.2 Nezávislé testovací systémy

Testovaní je důležitou součástí LMS, avšak použití LMS nemusí být nezbytnou podmínkou pro provoz testovacího systému. Existují totiž i testovací systémy, které mohou fungovat zcela samostatně, nezávisle na systémech jiných. V této části budou představeny některé z nich.

Test park

Test park je nekomerční (nepočítáme-li reklamní banner) český portál [w76], určený pro volné vkládání libovolných testů s automatickým vyhodnocením. Otázky v testu mohou být pouze typu multiple-choice s výběrem jen jedné z odpovědí, přičemž otázky ani odpovědi se nemíchají. Vytvořené testy jsou veřejně přístupné všem zájemcům a odkaz na ně je zveřejňován přímo na úvodním přehledu stránek portálu v příslušné kategorii.

Automatické vyhodnocení u každé otázky vyznačí zvolenou a správnou odpověď, a zároveň jsou sečteny získané body a uvedeno procento úspěšnosti. Na základě celkového počtu bodů dojde k zobrazení příslušného textu (celkového vyhodnocení) zadaného autorem testu. Procentní skóre je také porovnáno s průměrným procentním skórem testu, kterého dosáhli ostatní uživatelé.

V době psaní tohoto textu byla možnost přidávat nové testy zablokována z důvodu množícího se vkládání nevhodných testů, s příslibem opětovného zprovoznění, po zavedení systému schvalování nových testů. Ke 21.2.2012 bylo na tomto serveru vytvořeno 2 139 testů.

Stránky rozhodně nelze využít pro ostré hodnocené testování, je však možné přes ně pro studenty připravit opakovací autotesty, kterými si mohou dobrovolně ověřit a připomenout své vědomosti.

ClassMarker

Classmarker je komerční zahraniční webová služba [w10], která umožňuje vytvářet a provozovat on-line testy, určené pro hodnocení znalostí. Lze ji provozovat ve dvou variantách:

- Class based – přístup k vytvořeným testům mají pouze registrovaní uživatelé, kterým to povolí autor testu za jím nastavených podmínek,
- External testing – přístup k testu mají všichni (vhodné např. pro dotazníky nebo autotesty).

Jediný typ otázek, které tento systém podporuje, je výběr z nabízených odpovědí, přičemž lze nastavit, aby bylo možné zvolit pouze jednu, nebo více odpovědí. Pořadí otázek i jejich odpovědí může být v testu předkládáno v náhodném pořadí. Text otázky a odpovědí může obsahovat pouze základní formátování (tučné, kurzíva, podtržené, …), a také odkazy, obrázky (z externích zdrojů) a YouTube video, vše formou BBCode[14].

Testu je možné nastavit časové omezení, stránkování (kolik otázek se má zobrazovat na jedné stránce), povolit nebo zakázat návrat k již zodpovězeným otázkám, zasílání výsledků na zadaný e-mail, možnost test přerušit a pokračovat v něm později, povolit přejít na další otázku, pouze je-li ta aktuální zodpovězena správně, minimální požadované skóre pro absolvování testu a omezit počet pokusů na test.

[14] BBCode – Bulletin Board Code – značkovací jazyk pro formátování textů, který se na straně serveru překládá do klasických HTML tagů

Výsledek testu včetně zpětné vazby si může prohlédnout jak zkoušený, tak zadavatel testu. Ten má k dispozici i statistický přehled za jednotlivé testy, studenty i otázky, přičemž vše může exportovat do Excelu.

Hlavní výhoda služby ClassMarker je, že pro jeho provoz není nutné cokoli instalovat, ani mít vlastní server či hosting, stačí pouze webový prohlížeč. Mezi nevýhody patří fakt, že webové rozhraní je pouze v angličtině, k dispozici je jen jeden typ otázek a omezené možnosti nastavení míchání (nelze náhodně vybírat z většího souboru otázek) a hodnocení (každá otázka má stejnou váhu).

Program Testy

Český komerční program Testy [w56] funguje pod Windows a je určen pro tvorbu a provoz testů ve vlastní režii. V roce 2005 získal tento software akreditaci Ministerstva školství, mládeže a tělovýchovy ČR. Skládá se ze tří podprogramů:

- Creator – vytváření testů,
- Tests – zkoušení z hotových testů,
- Viewer – prohlížení výsledků ze zkoušení.

Otázky vytvářené v podprogramu **Creator** mají vždy své zadání a až devět možných odpovědí. Otázka i jednotlivé odpovědi mohou obsahovat text jednotného formátu, jeden obrázek, zvuk a video. Odpovědi mohou být nastaveny tak, že lze zvolit pouze jednu, nebo některým z nich nastavit, aby je bylo možné zvolit (zaškrtnout) nezávisle na ostatních (např. tedy z odpovědí 1-4 pouze jednu plus kteroukoli nebo i všechny z těch dalších 5-9). U každé z odpovědi může být též kromě prostého zaškrtnutí vyžadována i textová odpověď.

Každé z odpovědí je stanoven počet bodů (mohou být i záporné), který je přičten k celkovému, je-li tato zvolena. U textových odpovědí jsou body přičteny, pouze pokud souhlasí i zadaný text s autorem testu předdefinovanou správnou odpovědí (musí si být rovny, na velikosti písmen nezáleží).

Test disponuje poměrně rozsáhlými možnostmi nastavení. Patří mezi ně míchání otázek, míchání odpovědí, výběr pouze několika otázek z celého testu, automatické vyhodnocení, zákaz vracení se k předchozím otázkám či jen nepovolit jejich opravování, časové omezení testu i otázek (všech stejné), zabezpečení testu heslem pro editaci a spuštění (různými hesly), zobrazení správnosti řešení hned po přepnutí otázky (zpětná vazba, pro výukové testy), ukládání výsledků (na lokální či síťový disk, na síťový server do sdílené složky nebo posílat e-mailem), penalizační body za neřešené otázky a vytvoření závěrečné zprávy o průběhu testování do textového souboru (pro zkoušeného). Kromě nastavení průběhu testu lze kompletně nastavit i vzhled celého programu. Možné také je, aby se před zahájením testu, zobrazil obrázek ze souboru podle jména (identifikátoru) testovaného, kdy je např. možné před testem dle fotografie ověřit jeho totožnost.

Testy se ukládají nikoli do souborů, ale do složek, do kterých jsou nakopírovány i veškeré multimediální soubory (obrázky, zvuky a videa). Obsah a nastavení testu je uloženo do textových, avšak silně šifrovaných souborů. Zabezpečit šifrou je možné i přiložené multimediální soubory, aby je nebylo možné prohlížet před zahájením testu. K tomu je určena externí aplikace „Cypher".

Vytvořené testy se spouští v podprogramu **Tests**. Není tedy třeba na každý počítač instalovat vše, ale stačí pouze tato malá podaplikace. Kromě této formy zkoušení se dají hotové testy

včetně klíče správného řešení také exportovat a to do textového souboru, Wordu, Excelu a HTML. Poslední možnost do výstupní stránky zahrne i JavaScript, který podporuje i nastavené míchání a automatické vyhodnocování testů.

Výsledky zkoušení se ukládají v zašifrovaných souborech na nastavené místo a obsahují kompletní rozbor testu. Ten se prohlíží v podprogramu **Viewer**, který umožňuje tisk, přičemž pro načtení rozboru je třeba zadat heslo pro editaci testu. Je též možné načíst všechny soubory s výsledky z určité složky do tabulkového přehledu a to buď za jednotlivá zkoušení, nebo jako rozbor podle otázek. V prvním případě je vidět, kdo získal jaký bodový zisk a je možné všechny tyto výsledky automaticky oznámkovat dle nastavených bodových rozsahů. Rozbor otázek pak ukazuje, kolikrát každou z možných odpovědí bylo odpovídáno na kterou otázku. Oba přehledy lze uložit do souboru, vyexportovat do Excelu nebo vytisknout.

Program Testy je užitečný pro testování v rámci interní sítě. Ve své první verzi vyšel již v roce 1999 a po svém dokončení se již příliš nevyvíjel. I přesto svou základní úlohu plní stále dobře, ovšem jeho GUI[15] již do dnešního trendu příliš nezapadá.

3.1.3 Výukové a prezentační testy

Ne vždy je cílem testu stanovit úroveň vědomostí jiné osoby. V některých případech spíše chceme posluchače nebo čtenáře více zapojit do probíraného tématu a přimět je tak právě nabyté vědomosti aktivně použít. Jednou z možností jak něco takového realizovat, je zeptat se jich na určitý fakt z právě probrané látky a následně jim poskytnout okamžitou zpětnou vazbu.

Tohoto se dá využít nejen při živé přednášce, kdy je do pléna vznesena otázka a dán určitý čas na její promyšlení, než je vyřčena její odpověď, ale je možné takto obohatit i samostatné studijní materiály, umístěné např. v LMS. Pro zapamatování si dané látky může být velmi přínosné, pokud je například po určité části učebního textu vznesena kontrolní otázka nebo i rychlý test. Student tak hned pozná, pochopil-li učivo dostatečně, nebo by se k němu měl ještě vrátit. Zároveň takovéto oživení strohého učení napomůže v udržení pozornosti a zapamatování příslušné látky.

V této části tedy uvedeme některé nástroje pro vytváření takovýchto výukových a prezentačních autotestů.

Hot Potatoes

Aplikace Hot Potatoes [w29] je vytvořena ve více verzích pro různé operační systémy, a to Windows, Linux i Mac OS. Pro výukové nekomerční účely je k dispozici zdarma, stačí pouze jednoduchá registrace.

Program využívá HTML jazyk a JavaScript, ale uživatelé je nemusí ovládat. Jediné co je potřeba, je vložit informace – otázky, odpovědi a zvolit formát, který chceme. Program vygeneruje webovou stránku sám. [16]

Vytvořené otázky lze ukládat a načítat z/do souborů vlastního formátu (jiný pro každý typ otázky), sestavovat z nich celé testy a exportovat je (testy i jednotlivé otázky) do formátu XHTML 1.1. JavaScript, který exportované stránky obsahují, je propracovaný a umožňuje takto vytvořeným stránkám s jednotlivými otázkami i určitý stupeň interaktivity, byť stránka neobsahuje žádné pluginy typu Flash, JavaApplet nebo Silverlight.

[15] GUI – Graphical User Interface (grafické uživatelské rozhraní)

Interaktivita spočívá především v tom, že po vyplnění každé otázky se kliknutím na tlačítko „Check" zobrazí zpětná vazba, tj. označení chybných a správných odpovědí, počet bodů a autorem testu předvyplněné vysvětlení. Také lze u některých typů otázek nastavit formu řešení pomocí metody drag&drop.

Podporované typy otázek jsou tyto:

- **JCloze** – Text s vynechanými slovy, která se doplňují buď písemně, nebo výběrem z nabízených možností.
- **JMatch** – Ke každému z jedné skupiny výrazů se z druhé přiřazují související výrazy (např. názvům států jejich hlavní města) a to buď výběrem, nebo přetahováním.
- **JQuiz** – Sada krátkých otázek těchto možných podtypů:
  - multiple-choice – výběr jedné z nabízených odpovědí,
  - short-answer – krátká psaná odpověď,
  - hybrid – krátká psaná odpověď, která, pokud je vyplněna chybně, je změněna na otázku typu multiple-choice,
  - multi-select – výběr více z nabízených odpovědí.
- **JCross** – Křížovka typu kris-kros.
- **JMix** – Skládání věty ze zamíchaných slov, nebo slov ze zamíchaných písmen.

*Pro správu lekcí s navazujícími cvičeními slouží poslední modul **Masher**. Základní SW je k dispozici jako freeware, použití plné funkčnosti Masher je podmíněno zakoupením licence.* [17]

Editory jednotlivých typů otázek tvoří samostatné desktopové programy, obsahující vždy jednoduchý formulář s editačními políčky, kam se vypisuje text otázky a odpovědí, každý s příslušnými možnostmi nastavení. Do textu je možné vkládat obrázky, odkazy a tabulky, vše ovšem pouze ve formě HTML kódu, který však doplní příslušný průvodce vložením objektu. Možné je též přidat multimediální objekty (audio a video), jako přiložené soubory otevírané v příslušném přehrávači přes odkaz.

Vyexportované soubory mohou být vloženy na libovolný web včetně kurzů v kterémkoli LMS (např. do WebCT, viz [18]), jako zvláštní stránky, nebo vloženy do rámců (iFrame[16]) uvnitř jiné stránky. Kromě exportu do HTML jsou k dispozici i tyto další možnosti:

- ZIP package – ke XHTML stránce jsou přiloženy veškeré související soubory a vše je zabaleno do archivu ZIP,
- SCORM[17] 1.2 – standardizovaný formát pro přenos materiálů mezi jednotlivými LMS,
- Export pro tisk – převede otázku včetně jejího správného řešení (odděleně) do prostého textového formátu, který uloží do schránky (clipboard), ze které je poté možno jej vložit do libovolného textového editoru a vytisknout,
- WebCT – speciální formát určený pro tento LMS.

Hot Potatoes je užitečný nástroj, jak rychle vytvořit efektivní autotesty použitelné on-line, bez nutnosti se zabývat jejich HTML kódem či JavaScriptem.

[16] iFrame (inline frame) - plovoucí rám vložený do jiné webové stránky
[17] SCORM – Shareable Content Object Reference Model [w61]

eXe

*Program eXe (**e**Learning **X**HTML **e**ditor) je tvůrčí prostředí pomáhající učitelům a akademickým pracovníkům při návrhu, vývoji a vydávání on-line vzdělávacích a výukových materiálů bez nutnosti odborné znalosti HTML nebo komplikovaných aplikací pro publikování na webu.* [w23]

V této open-source aplikaci [w24] pro Windows, Linux a Mac OS X, využívající prostředí prohlížeče Mozilla Firefox, se jednoduše pomocí přidávání předdefinovaných částí obsahujících WYSIWYG editor HTML (s podporou obrázků, vzorců v zápisu LaTeX, videí i zvukových souborů), vytvářejí graficky vzhledné webové stránky. Některé z těchto částí poskytují i určitý stupeň interaktivity prostřednictvím automaticky generovaného JavaScriptu či přidaných JavaAppletů. Patří mezi ně i některé druhy testových otázek, konkrétně doplňovačka (cloze), ano/ne, reflexe (prostá otázka, jejíž odpověď se zobrazí po kliknutí na tlačítko), kvíz (série otázek typu multiple-choice hodnocených procentním výsledkem), vícenásobná volba (výběr odpovědi s okamžitou individuální zpětnou vazbou) a vícenásobný výběr (volba jedné až všech nabízených odpovědí).

Kontrolními otázkami i dalšími položkami se tedy pouze prokládá samotný text stránky. Výstup může obsahovat i vícestránkovou strukturu včetně přehledné navigace (menu).

Webové stránky vytvořené v tomto programu lze vytvářet velmi snadno, rychle a přitom efektně. Kromě klasické HTML stránky je možný také export do Common Cartridge, SCORM 1.2, ISM, textového souboru a poznámek pro iPod. Výsledek by neměl být problém vložit do libovolného LMS (např. Moodle [19]) či jím obohatit kterýkoli web.

Alf

Česká komerční aplikace [w3] pro Windows vytvořená ve Flashi (spouští se v přibaleném Macromedia Flashplayeru 5.0) s propracovanou animovanou grafikou, umožňující vytvářet a spouštět testy. Ty se ukládají a načítají ze souborů ve vlastním textovém formátu. Po zodpovězené otázce se rovnou ukáže správné řešení, body se zobrazují u testu, ale nikde neukládají. Program je efektní především pro interaktivní tabuli na základních školách. Podporované typy otázek jsou:

- výběr jedné odpovědi,
- pexeso (hledané dvojice v něm tvoří dva různé, ale související výrazy),
- zařazení do skupin,
- přiřazení (k daným výrazům najít a přiřadit související),
- seřazení (bodují se pouze správné pozice).

Articulate Quizmaker

Quizmaker '09 je komerční aplikace [w8] ve které se dají vytvářet graficky propracované on-line testové kvízy a dotazníky ve Flashi. Aplikace je vytvořena v MS .NET WPF a používá uživatelsky přívětivé Ribbon[18] rozložení ovládacích prvků. *Autor testu může vytvářet audio scénář, který se vyvíjí v průběhu několika obrazovek (otázek), animovat objekty, používat předdefinovaných šablon a stylů pro testy či sestavovat lineární nebo větvený sled otázek.* [20]

Kvízy se vytvářejí ve dvou režimech a to jako hodnocený kvíz nebo dotazník. Kvíz podporuje otázky typu ano/ne, multiple-choice (výběr jedné či více z odpovědí), krátká psaná odpověď, výběr

[18] Ribbon – rozložení ovládacích prvků do „pásu karet" [w79]

jedné možnosti metodou drag&drop, přiřazování přetahováním, přiřazování výběrem, řazení přetahováním, řazení výběrem, číselná odpověď (správný výsledek může být definován sérií porovnávacích pravidel) a klinutí na správné místo (hotspot). Dotazník pak navíc může obsahovat otázky pro ohodnocení více faktů na společně definované stupnici a dlouhou psanou odpověď.

Otázky mohou být vytvářeny buď v režimu *form view*, kdy se pouze vyplňují hodnoty ve formuláři pro daný typ otázek, nebo *slide view*, kdy je kompletní obsah otázky plně pod kontrolou jejího tvůrce. Mezi těmito režimy se lze volně přepínat. Ovládací prvky a funkce pro sestavování designu otázky jsou velmi podobné slajdům v MS PowerPoint 2007. Součástí otázky mohou být též jednoduché animace počátku a konce jednotlivých grafických objektů, které se editují na vlastní časové ose animací. Program také obsahuje integrovaný audio editor, který umožňuje střih a základní úpravy zvukových souborů a nahrávek.

Otázky mohou být v kvízu tříděny do jednoúrovňových skupin, kterým lze nastavit počet otázek, takže za každou skupinu budou do testu náhodně vybrány pouze některé z nich. Je také možné nastavit větvení, při kterém je určena otázka, jež bude následovat po té aktuální, v případě že bude zvolena určitá odpověď.

Zpětnou vazbu lze nastavit za celý test, za každou otázku i za jednotlivé odpovědi (u některých typů otázek). Po dokončení testu si jej zkoušený může znovu projít a prohlédnout si, kde udělal chybu.

Vytvořené testy lze exportovat do formátů SCORM nebo AICC[19] a plně je integrovat do LMS podporujících tyto formáty (např. Moodle [w9]). LMS tak může kontrolovat nejen jejich zpřístupnění účastníkům kurzu, ale zároveň i získává zpětnou vazbu, byl-li test splněn či nikoli, na základě podmínek nastavených pro jeho absolvování (např. minimální dosažené skóre). Dále jsou podporovány exporty pro samostatný web, CD, Word a do dalších spolupracujících programů od této společnosti.

Quizmaker je kvalitně zpracovaný editor multimediálně bohatých kvízů, provozovaných v Adobe Flash, které se dají snadno integrovat i do LMS. Za jeho kvalitu je ovšem stanovena poměrně vysoká cena.

TestBuilder

TestBuilder je součástí komerčního software e-Learning Authoring Tool [w20], který umožňuje v uživatelsky přívětivé desktopové aplikaci vytvářet e-Learningové materiály pro kurzy v různých LMS, kam se integrují pomocí SCORM standardu. Podporované typy otázek jsou ano/ne, výběr jedné z odpovědí, výběr více z odpovědí a přiřazování. Možné je použít test jako autotesty nebo jako hodnocený test, jehož výsledek může být předán hostitelskému LMS systému.

3.1.4 Aplikace pro učení opakováním s vhodně zvolenou prodlevou

Programy tohoto typu nejsou určeny pro testování ve smyslu zkoušení ani prezentace, ale pro učení jednotlivce formou drilu. Průběh většinou spočívá v předložení jednoho výrazu (slovíčka, věty, otázky, obrázku, …) zkoušenému, který odpoví buď písemně, nebo ve většině případů jen sám sobě (v duchu). Následně se zobrazí správná odpověď a vyhodnotí se správnost odpovědi zkoušeného. U písemné odpovědi může být porovnání automatické, obvykle se však zkoušený hod-

[19] AICC – Adiabatic Isochoric Complete Combustion [w2]

notí sám, na vícebodové stupnici (např. 1-5), čímž vyjádří, znal-li doslovnou správnou odpověď a jak moc si jí byl jist. Hodnocení se zaznamená a pokračuje se dalším výrazem.

Tomuto způsobu se říká „**kartičky**" (anglicky flashcard), neboť bez použití počítače se používá tak, že pro každý pár výrazů (otázku a odpověď) je k dispozici kartička, na které je z jedné strany napsáno zadání a z druhé správné řešení.

Sofistikovanější systémy na základě výsledků pro jednotlivé otázky dokáží stanovit, za jak dlouho by si měl uživatel kterou z nich zopakovat, aby pravděpodobnost jejího zapomnění byla co nejnižší. Způsob určení tohoto intervalu se přitom v jednotlivých systémech různí (např. viz porovnání některých přístupů na [w41]).

Základy tomuto typu učení byly položeny již v roce 1885, kdy Hermann Ebbinghaus [21 stránky 94-96] zveřejnil výsledky svých pokusů s lidskou pamětí [22], ve kterých sestavil tzv. **křivku zapomínání** [23 str. 113] ukazující, jak s přibývajícím časem od naučení se určitých faktů dochází k jejich postupnému zapomínání. [24] Následné výzkumy (např. viz [25]) prokázaly, že včasným zopakováním určitého faktu lze sklon této křivky značně snížit a daný fakt si tak zapamatovat dlouhodobě, v ideálním případě i celoživotně. Tento interval je však individuální pro každého člověka a dokonce i pro každý učený fakt, ale dá se s určitou dosti vysokou pravděpodobností výpočetně určit. Je totiž důležité fakt neopakovat příliš často (mozek jej pak ponechává pouze v krátkodobé paměti a ustane-li opakování, fakt bude zapomenut) ani s příliš dlouhými rozestupy. Při vhodných intervalech opakování však mozek posoudí fakt jako pro další život důležitý a uloží jej do paměti dlouhodobé. [26]

P. Wozniak (narozen 1962 v Polsku) na základě těchto poznatků později sestavil algoritmus pro počítač [w69], který individuálně každému člověku dokáže pro každý učený fakt určit nejvhodnější dobu pro jeho zopakování, aby se s co největší pravděpodobností (okolo 90% [27]) uložil do dlouhodobé paměti. [28] Tento a podobné algoritmy pak byly použity v různých softwarech, které lze pro tuto formu výuky použít.

„**Spaced repetition**" (opakování s vhodně zvolenými prodlevami), jak se tento princip nazývá, je vhodný především pro učení se slovíček cizích jazyků a jednoduchých faktů různých druhů. Jedná se totiž o učení zpaměti (z gnoseologického hlediska tedy dogmatickou vyučovací metodu), nikoli o analýzu, syntézu, indukci, dedukci, srovnávání apod. [29 str. 187] Tento způsob učení také vyžaduje svůj čas. Nelze se s ním naučit hodně látky za krátkou dobu, ale učení musí být dlouhodobé a hlavně pravidelné, ovšem probíhá s menší námahou a je prokazatelně značně efektivní (např. viz [30]). [31]

SuperMemo

Vývoj aplikace SuperMemo [w72] začal již v roce 1985 (v Borland Turbo Pascal pro DOS [w68]) a pokračuje dodnes s využitím aktuálních moderních programovacích technologií, včetně použití on-line a verze pro mobilní telefony, s možností vzájemné synchronizace. Autorem prvních verzí je P. Wozniak, který se zároveň významně podílel na rozvoji počítačové podpory učení metodou *spaced repetition*. [w67] Algoritmus, který byl v prvních verzích aplikace použit, je veřejně dostupný [w69] a vychází z něj i řada dalších konkurenčních programů pro tento způsob učení.

SuperMemo slibuje, že při jeho každodenním používání po alespoň 10 minutách, bude učení 10-50x rychlejší než klasické metody, s průměrnou mírou zapamatování 95%. [w70] Program je

k dispozici zdarma, placené jsou pouze kurzy (soubory faktů) pro tento program vytvořené. Je však možné vytvořit si i kurzy vlastní.

Každý kurz je možné rozdělit na kapitoly obsahující jednotlivé stránky (kartičky). Ty mohou obsahovat buď pouze prosté informace (např. instrukce nebo zadání pro následující stránky) nebo funkční otázky. Rozložení stránky je přitom podobné slajdům v prezentaci PowerPoint. Stránky se skládají ze zadání otázky a odpovědi, která se při učení zobrazí až po kliknutí na tlačítko „Check answer". Otázka i odpověď mohou obsahovat text s možností základního formátování a prvky jako je tabulka, obrázek, zvuk a nápověda zobrazená po najetí myši nad určité slovo (hint). Součástí otázky navíc mohou být formulářové prvky a to editační políčko pro vepsání textu (spellpad), výběr jedné z odpovědí (radio buttons), výběr více odpovědí (checkbox), rozevírací seznam pro výběr jedné z položek (droplist) a volba typu ano/ne. U všech těchto prvků se dá vyznačit, která volba je správná, což bude po zobrazení řešení barevně vyznačeno.

Uživatel si tedy přečte otázku, sám pro sebe si odpoví, nebo zaškrtne příslušný formulářový prvek a zobrazí si správné řešení. Následně je dotázán, znal-li řešení či nikoli (stupnice 1-3), a pokud ne, je tato otázka zahrnuta do sekce „drill" určené pro zopakování otázek ještě tento den.

Aplikace na základě výsledků sama každému jejímu uživateli sestavuje dlouhodobý plán učení a opakování příslušných stránek, aby dosáhla jejich optimálního zapamatování. Tento plán si uživatel může prohlédnout v přehledném kalendáři.

SuperMemo je uživatelsky přívětivá aplikace s efektním ovládáním jak pro učení se, tak při vytváření nových studijních materiálů. Její největší síla je v dobře propracovaném a léty provozu ověřeném algoritmu pro učení se metodou *spaced repetition*. Uživatelské rozhraní je bohužel k dispozici pouze v anglické, německé a polské verzi. Profesionální editor pro tvorbu komerčních kurzů, umožňující navíc další funkce (např. analyzátor zvukového vstupu z mikrofonu), lze získat pouze po domluvě s autory programu [w71].

Anki

Anki je japonský open-source program [w6] pro Windows, Mac OS X, Linux, FreeBDS, iPhone, maemo, Android a některé kapesní počítače a mobilní telefony. Krom toho je k dispozici i online verze, jejíž obsah lze synchronizovat s verzemi desktopovými (je nutné si na webu [w7] vytvořit vlastní účet). Program disponuje lokalizacemi do mnoha světových jazyků (např. verze pro Windows má 43 lokalizací) včetně češtiny.

Soubory faktů lze vytvářet i stahovat (a nahrávat) již hotové z internetu přímo přes rozhraní programu. Kromě studijních materiálů je možné stahovat i zásuvné moduly, s jejichž pomocí lze různě rozšířit funkčnost programu.

Aplikace má v základu jednoduchý design, což je ku prospěchu její přehlednosti a intuitivnímu ovládání. Položky se vytvářejí ryze ve stylu kartiček, tedy její přední a zadní strana. Ty mohou obsahovat text se základním formátováním, ale také obrázky, zvuky a videa. Pro složitější zápisy lze text zapsat ve formátu HTML (přímo ve formě kódu, bez vlastního editoru) nebo přidat i matematické vzorce ve formátu LaTeX (opět zapsané jako kód, bez editoru). Pro kartičky s opakující se podstatnou částí obsahu může být vytvořena šablona, v níž se již pouze doplňují dvě slova (pro přední a pro zadní stranu kartičky). Kartičce je možné povolit i její oboustranné použití.

Učení a opakování tedy probíhá klasicky, zobrazením jedné strany kartičky, po promyšlení si odpovědi se odkryje i její druhá strana (pod sebou) a následně uživatel ohodnotí sebe sama, jak

moc byla jeho odpověď správná (stupnice 1-4, u každé z nich je i rovnou uvedeno, za jak dlouho bude při jejím zvolení kartička zobrazena znovu). Program vytváří přehledné statistiky pro každý balíček (soubor kartiček) i pro jednotlivé kartičky. K dispozici je též osm nastavitelných grafů zobrazujících aktivitu a plány daného uživatele.

Algoritmus, který plánuje včasné zopakování jednotlivých kartiček každému uživateli, je založen na SM2 (SuperMemo algoritmus verze 2 [w66]). [w3]

Anki je jednoduchý program vhodný především pro tvorbu a používání vlastních materiálů (materiály ke stažení se totiž převážně týkají japonštiny). Díky verzím pro různé platformy a možnosti synchronizace s webovou databází je jej možné používat ve volných chvílích téměř kdekoli a věnovat tak tento čas smysluplnému samostudiu.

Mnemosyne

Mnemosyne je japonská open-source aplikace [w78] pro Windows, Linux a Mac OS X vytvořená v programovacím jazyku Python [w58]. Její ovládání i design je ještě o něco jednodušší než tomu bylo v případě Anki. Hlavní okno aplikace je přímo tím, ve kterém probíhá učení a opakování. Program je přeložen do 36 jazyků včetně češtiny.

Nové kartičky lze importovat z XML nebo textových formátů, případně vytvářet vlastní přímo v programu. Hotové soubory kartiček se pak ukládají do souborů, obsahujících i informace o jejich procvičování.

Jednoduchý textový editor pro otázku a odpověď neumožňuje žádné WYSIWYG formátování, ale lze v něm použít HTML tagy. Podporován je také `<img>` pro obrázky, které je však nutné s uloženým souborem kartiček ručně kopírovat v příslušné relativní (nebo i absolutní) cestě. V textu je možné uvést také LaTeX zápis pro zobrazení matematických vzorců. Při ukládání nové kartičky lze zvolit, smí-li být použita i opačně (prohozena otázka a odpověď) a pokud ano, přidá se druhá varianta automaticky jako zvláštní kartička.

Při učení je vždy v horní části okna zobrazena otázka (přední strana kartičky) a ve spodní části je prostor pro správnou odpověď, která se zobrazí po kliknutí na tlačítko „Ukázat odpověď“. Následně uživatel sám sebe ohodnotí (stupnice 0-5, přičemž 0 je nejhorší výsledek a 5 nejlepší).

Kartičky mají v rámci souboru ještě možnost být tříděny do kategorií. Název kategorie, ze které je aktuální kartička, se pak zobrazuje i během učení. Pro správu kartiček je k dispozici třísloupcová (otázka, odpověď a kategorie) tabulka, s možností řazení a rychlého vyhledávání.

Statistiky jsou k dispozici pouze v textové formě a to za jednotlivé kartičky, za celé kategorie, dále procentní rozdělení jednotlivých sebehodnocení (0-5) a rozvrh ukazující počty kartiček, které je ve kterých následujících dnech potřeba zopakovat. Program podporuje též zásuvné moduly, které rozšiřují jeho základní funkčnost.

Algoritmus pro plánování včasného zopakování jednotlivých kartiček uživateli, je opět převzat z SM2 (SuperMemo algoritmus verze 2 [w66]), ovšem byl částečně modifikován. [w77]

Mnemosyne (pojmenováno po řecké bohyni paměti) je jednoduchý program, který se zaměřuje především na svou hlavní funkci, čili tvorbu kartiček a jejich správnou výuku, díky čemuž je velmi jednoduchý na ovládání a nezatěžuje uživatele dalšími, pro někoho i zbytečnými funkcemi.

Memostation

Memostation je česká aplikace [w42] vytvořená v Microsoft .NET pro Windows. Má propracované grafické uživatelské rozhraní i řadu pokročilých funkcí. Aplikace umožňuje vytvářet, ukládat, exportovat i importovat sady položek (faktů) pro jednotlivé učební předměty a lekce. Díky on-line podpoře je poměrně rozvinutá i výměna takto připravených materiálů a k dispozici jich je již značné množství (k 26.2.2012 je to 704 výukových souborů [w42]), různé kvality a s různým počtem položek.

Položky mohou obsahovat nejen prosté výrazy vyobrazené na virtuálních kartičkách, ale i rozsáhlejší formátovaný text včetně obrázků a tabulek (matematické vzorce nejsou podporovány), nebo otázky typu multiple-choice. Pro každou kartičku se dá nastavit, může-li být při testování použita oboustranně, a jestli na ni zkoušený smí odpovědět pouze písemně či použít sebehodnocení (stupnice 1-5). Pro automatické určování míry správnosti písemné odpovědi je použit Jaro-Winklerův algoritmus[20].

Aplikace dle výsledků z průběhu každého opakování sama určí sílu paměti uživatele, z ní stanoví tvar individuální křivky zapomínání a na jejím základě dle míry zapamatování jednotlivých položek dlouhodobě plánuje jejich opětovné zařazení do příštích opakování. Dlouhodobé plánování učení metodou *spaced repetition* program umožňuje pouze v placené verzi, existuje ale i varianta rychlého neplánového zkoušení, s následným drilem chybný odpovědí.

Dril

Dril je webová aplikace integrovaná do systému IS MU (Informační systém Masarykovy univerzity), vytvořená zaměstnanci Masarykovy univerzity (MU), kteří se inspirovali výše popsanými zahraničními systémy.

Dril simuluje tzv. kartičkovou metodu s využitím principu „spaced repetition", tzv. opakování s vhodně zvolenými prodlevami. Uživatel dá systému zpětnou vazbu, jak kterou kartičku ovládá a systém spočítá, kdy se má kartička nabídnout znovu k zopakování. [27]

Studenti MU si pomocí této aplikace mohou samostatně rozšiřovat svou slovní zásobu cizího jazyka, především angličtiny (fakulta informatiky) a latiny (lékařská fakulta). Součástí slovíček je i jejich správná výslovnost ve zvukovém formátu WAV. Studentovi je vždy předložen výraz v češtině, on si sám pro sebe řekne jeho překlad a následně si zobrazí správné řešení, které se mu i přehraje ze zvukového souboru. Svou úspěšnost a míru nejistoty student sám ohodnotí na stupnici od 1 do 6, přičemž stupeň 3-6 znamená, že by si dané slovíčko měl dnes ještě nejméně jednou zopakovat, aby jeho znalost byla pro tento den znovu stoprocentní. Slovíčka, která dokonale ovládá, může přitom z opakování vyřadit zcela.

Studenti mají též možnost si databázi faktů individuálně i kolektivně upravovat a rozšiřovat, díky čemuž v systému vznikají celé nové „učebnice" (soubory faktů k určitému tématu) z nejrůznějších oblastí. [w32], [w31]

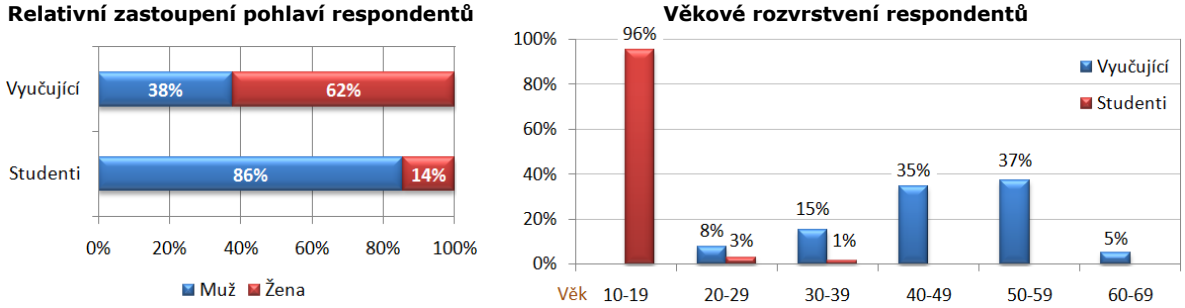
[20] Jaro-Winklerův algoritmus určuje míru shody dvou textových řetězců (viz 4.4, str. 76)

3.2 Dotazníkový průzkum [◎ I.2]

On-line dotazníkové šetření na téma *Testovací systémy* proběhlo v období 28.3. – 1.4.2011. Zapojilo se do něho celkem 152 respondentů, z čehož bylo 69 studentů a 83 vyučujících. Dotazník byl větvený a ne všechny otázky povinné, takže se u některých z nich celkové počty respondentů lišily.

Dotazník byl sestaven jako formulář ve službě *Dokumenty Google* [w17] a jeho samotné vytváření proběhlo ve dvou fázích. Nejprve byla sestavena první verze dotazníku [w73] a předložena zkušebnímu vzorku respondentů. Na základě následného interview s nimi došlo k úpravám dotazníku po stránce obsahové (změna formulace některých nejasných otázek) i rozsahové (zkrácení). Finální verze dotazníku [w75] pak byla s průvodním dopisem rozeslána pomocí e-mailu a Facebooku.

Dotazník byl vypracován jako anonymní, ovšem převážnou většinu respondentů z řad studentů byli studenti střední školy, ve které je zkušebně používáno testovací prostředí, jež je předmětem této práce. Vyučující respondenti pak byli z různých škol, převážně základních a středních v Královéhradeckém kraji, odkud bylo 94% vyučujících a 87% studentů. Pro úplnost uvádíme i relativní zastoupení pohlaví a věkové rozvrstvení respondentů v jednotlivých skupinách.



Graf 1 – Zastoupení pohlaví (vlevo) a věkové rozvrstvení respondentů (vpravo)

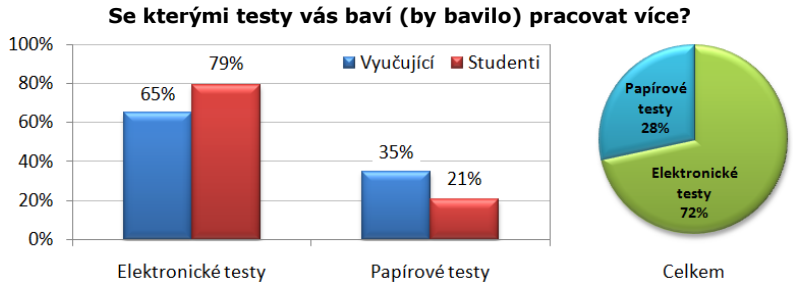
Rozbor výsledků, spadající do popisné statistiky, byl zpracován v MS Excel pomocí kontingenčních tabulek a grafů, pro testy hypotéz byl použit statistický software Statistica 9 [w65].

3.2.1 Elektronické a papírové testování

V první řadě byla zjištěna četnost, s jakou jsou používány elektronické testovací systémy oproti papírovým. Výhradně elektronické testy používá pouze 5% vyučujících respondentů, zatímco jen ty papírové jich preferuje 28%. Zbylých 67% používá oba způsoby testování.

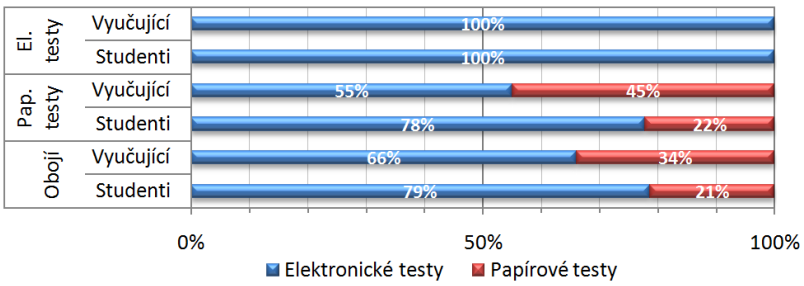
Graf 2 – Míra používání elektronických a papírových testů (resp.: 83 vyučujících a 69 studentů)

Nehledě na předchozí odpověď, byli všichni též dotázáni, s jakými testy je baví nebo by bavilo pracovat více (viz Graf 3). Pro elektronické testy se z dotázaných vyjádřilo 65% vyučujících a 79% studentů. Z T-testu i z Mann-Whitney testu vyplynulo, že se data obou skupin (vyučujících a studentů) na hladině významnosti 5% shodují (byť hraničně) a není mezi nimi statisticky významný rozdíl. V populaci tedy vyučující i studenti preferují elektronické testy přibližně stejnou měrou. Jelikož odpověď na tento dotaz je relevantní od každého respondenta, celkově lze říci, že 72% dotázaných preferuje elektronické testy před papírovými.



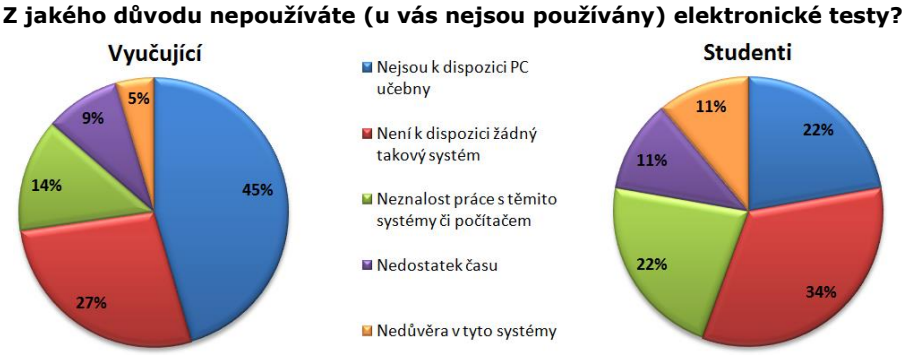
Graf 3 – Obliba elektronických a papírových testů (resp.: 80 vyučujících a 68 studentů, celkem 148)

Rozdělení těchto preferencí (Graf 3) na základě aktuálně používané formy testování (Graf 2) ukazuje Graf 4. Podle něho všichni respondenti (vyučující i studenti), kteří pracují s elektronickými testy, by se k papírovým již vracet nechtěli. Nejvíce se papírových testů „zastávají" vyučující respondenti, kteří aktuálně pracují pouze s nimi (45%). Respondenti z řad studentů, kteří používají testy papírové, samostatně nebo i v kombinaci s testy elektronickými, preferují v obou případech spíše testy elektronické (78%-79%). Vzhledem k nízkému zastoupení respondentů ve většině z těchto porovnávaných skupin ovšem nelze tyto závěry vztahovat na celou populaci, neboť Kruskal-Wallisův test na 5% hladině významnosti prokázal, že mezi jednotlivými skupinami není statisticky významný rozdíl.



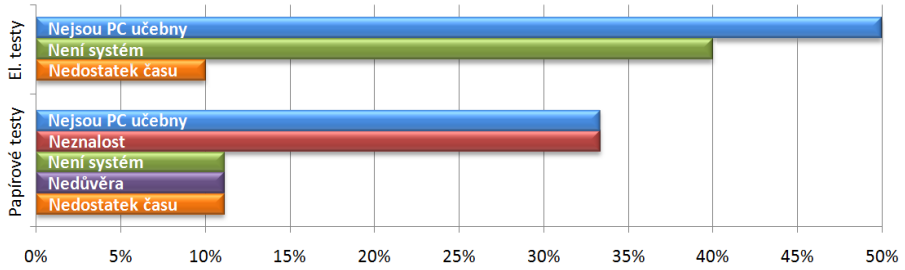
Graf 4 – Obliba elektronických a papírových testů dle aktuálně používaného způsobu testování

Respondenti, kteří odpověděli, že používají pouze papírové testy, byli dále dotázání na důvod, proč nepoužívají testy elektronické (viz Graf 5). V tomto případě jsou podstatnější odpovědi vyučujících, neboť studenti nemusí vždy znát konkrétní důvod, proč u nich není elektronický testovací systém používán. Z vyučujících respondentů pak pouze 45% uvedlo jako důvod nepoužívání elektronických testů nedostatek počítačů. Ostatní si myslí, že pro ně není k dispozici žádný testovací systém, nedokáží jej ovládat, nevěří těmto systémům nebo se obávají časové náročnosti při práci s ním.



Graf 5 – Důvody nevyužívání elektronických testů (resp.: 22 vyučujících a 9 studentů)

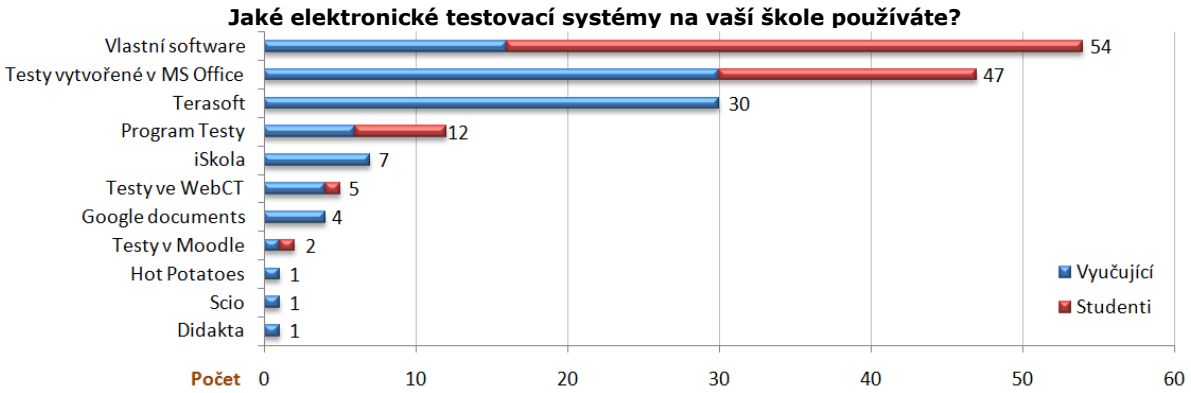
Rozdělení důvodů nepoužívání elektronického způsobu testování vyučujícími dle jejich preferovanější formy testování (viz Graf 3) ukazuje Graf 6. Polovina respondentů, kteří by raději používali elektronické testy, uvedlo jako důvod jejich nepoužívání nedostatek počítačových učeben. 40% pak nezná nebo nemá přístup k žádnému testovacímu systému. Vyučující, jimž stávající způsob papírového testování vyhovuje, z jedné třetiny uvádí jako důvod nepoužívání elektronických testů neznalost práce s těmito systémy nebo počítači a ve stejné míře nedostatek počítačů.



Graf 6 – Důvody nevyužívání elektronických testů vyučujícími dle preferovanějšího způsobu (19 resp.)

3.2.2 Používané testovací systémy

Následující dotaz byl položen pouze respondentům, kteří na předchozí otázku „Jaký způsob testování je na vaší škole používán?" (viz Graf 2) odpověděli, že používají elektronické testy nebo obojí (nikoli tedy pouze testy papírové). Respondenti zde mohli zvolit více možností i dopsat vlastní. Do výsledného grafu byly zahrnuty všechny jejich odpovědi, přičemž každému systému byl započten „bod" za každé jeho zvolení.



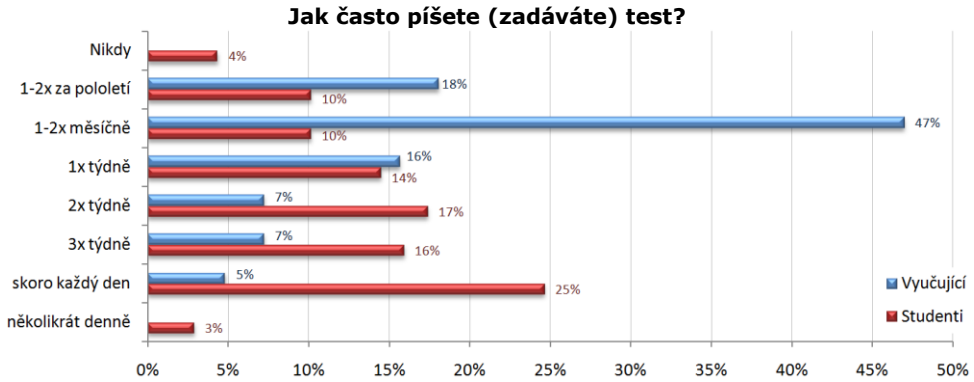
Graf 7 – Četnost používání konkrétních testovacích systémů (resp.: 55 vyučujících a 49 studentů)

Celkově nejčastěji byla vybrána volba *vlastní software*, tedy software vytvořený někým z dané školy, k čemuž nejvíce přispěli studenti a již zmíněný fakt, že většina z nich byli studenti střední školy, ve které je zkušebně používán vlastní testovací systém, jež je předmětem této práce. Z pohledu vyučujících byly nejčastěji zvoleny *testy vytvořené v MS Office* a *Terasoft*.

V prvním případě (office) nelze rozlišit, jestli vyučující neměli na mysli pouze přípravu testu např. v textovém editoru, který pak studentům zadávají v tištěné formě, nebo jestli kancelářský software využívají přímo pro testování na počítačích, jako např. v článku [32].

Terasoft je společnost vydávající výukový software především pro základní školy, na kterých je poměrně rozšířený (používá jej okolo 5 000 škol, viz [w74]). Testy, které jsou součástí těchto programů, jsou spíše výukové a prezentační autotesty, než testy, které lze použít pro hodnocení, nicméně použít se dají i tak.

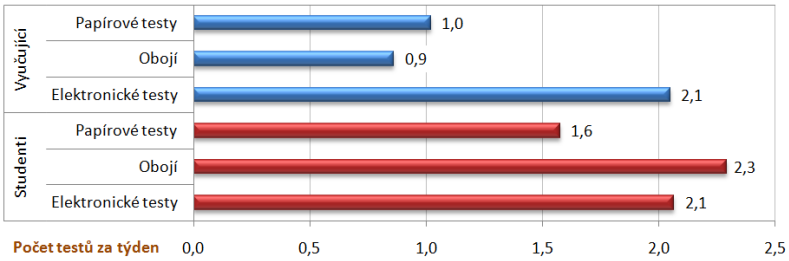
Pro posouzení, jak často respondenti používají zkoušení formou testů, jim byla položena následující otázka, jejíž výsledky ukazuje Graf 8.



Graf 8 – Frekvence používání zkoušení formou testů (resp.: 83 vyučujících a 69 studentů)

Téměř polovina (47%) vyučujících respondentů tedy zadává testy pouze 1-2x měsíčně. Respondenti z řad studentů však uvedli, že testy většinou píší častěji, z jedné čtvrtiny dokonce téměř každý den. Tento rozpor je logický, neboť studenti mají více předmětů, převážně vedených různými vyučujícími. Významný rozdíl mezi odpověďmi vyučujících a studentů na 5% hladině významnosti statisticky prokázal i Mann-Whitney test.

Aby bylo možné stanovit průměrnou frekvenci psaní či zadávání testů, byly jednotlivým možným odpovědím (viz Graf 8) přiděleny číselné hodnoty, které odpovídají jejich slovnímu vyjádření, přičemž jednotkou byl jeden týden (tj.: nikdy = 0, 1-2x za pololetí = 2/5/4 = 0,1, 1-2x měsíčně = 2/4 = 0,5, 1x týdně = 1, 2x týdně = 2, 3x týdně = 3, skoro každý den = 4, několikrát denně = 6). Za těchto podmínek vyšel celkový průměr za všechny respondenty 1,52 (tj. použití testů cca 1,5x týdně), přičemž průměrné zadávání testu vyučujícími bylo 0,96 a jeho psaní studenty 2,19. Graf 9 ukazuje tyto průměry vypočtené za jednotlivé skupiny respondentů a blíže rozdělené dle aktuálně používané formy testování (viz Graf 2).



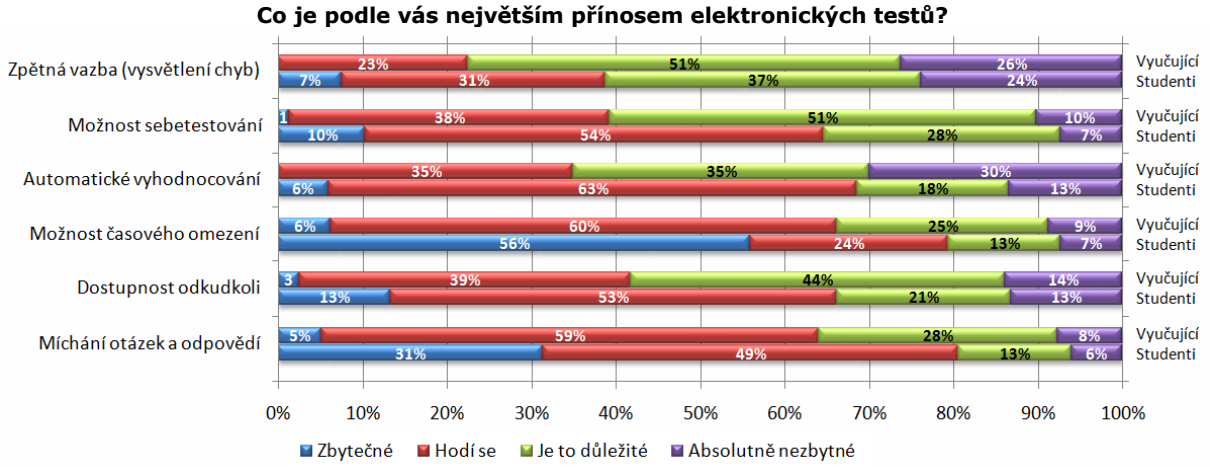
Graf 9 – Průměrná týdenní frekvence psaní (zadávání) testů za jednotlivé způsoby testování

Vyučující, na základě jejichž odpovědí byl sestaven Graf 9, a kteří používají výhradně elektronickou formu testování, tedy zadávají test více než 2x častěji než ti, jež používají výhradně papírové testy, nebo obojí. Studenti, testovaní elektronickou formou nebo kombinovaným způsobem, jsou pak také tetováni o něco častěji, nežli ti, kterým je zadáván pouze papírový test.

Vzhledem k nízkému počtu respondentů v některých skupinách však ze statistického hlediska není možné vztahovat veškeré tyto závěry na celou populaci. Neparametrický Kruskal-Wallisův test na 5% hladině významnosti potvrdil rozdíl pouze skupiny „Studenti – Obojí" oproti „Vyučující – Obojí" a také oproti „Vyučující – papírové testy". Ostatní skupiny jsou podle tohoto testu ve statistické shodě na dané hladině významnosti.

3.2.3 Funkce testovacích systémů

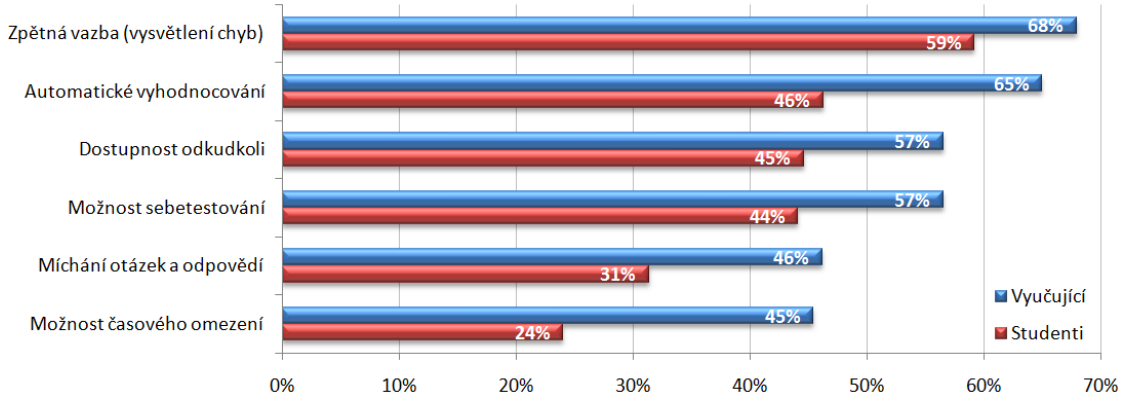
Další dotazy směřovaly na preference funkcí testovacích systémů. Nejprve respondenti měli na čtyřstupňové škále ohodnotit funkce, kterými disponuje většina testovacích systémů. Srovnání hodnocení vyučujících a studentů ukazuje Graf 10.



Graf 10 – Rozložení preferencí stávajících funkcí testovacího systému (resp.: 79 vyučujících a 68 studentů)

Graf 10 ukazuje, které z uvedených funkcí jsou jejich uživateli preferovány a jakou měrou. Abychom mohli tyto funkce seřadit dle oblíbenosti, je třeba jejich celkovou preferovanost za jednotlivé skupiny respondentů převést na ordinální hodnoty. Jednotlivým stupňům hodnocení tedy byly přiděleny číselné hodnoty (Zbytečné = 0, Hodí se = 1, Je to důležité = 2 a Absolutně nezbytné = 3), které byly za jednotlivé skupiny respondentů sečteny a vyděleny maximální možnou hodnotu, tj. počtem respondentů ve skupině vynásobeným třemi. Tyto odvozené relativní hodnoty oblíbenosti v sestupném pořadí ukazuje Graf 11, přičemž procentní hodnoty vyjadřují, za jak moc důležitou

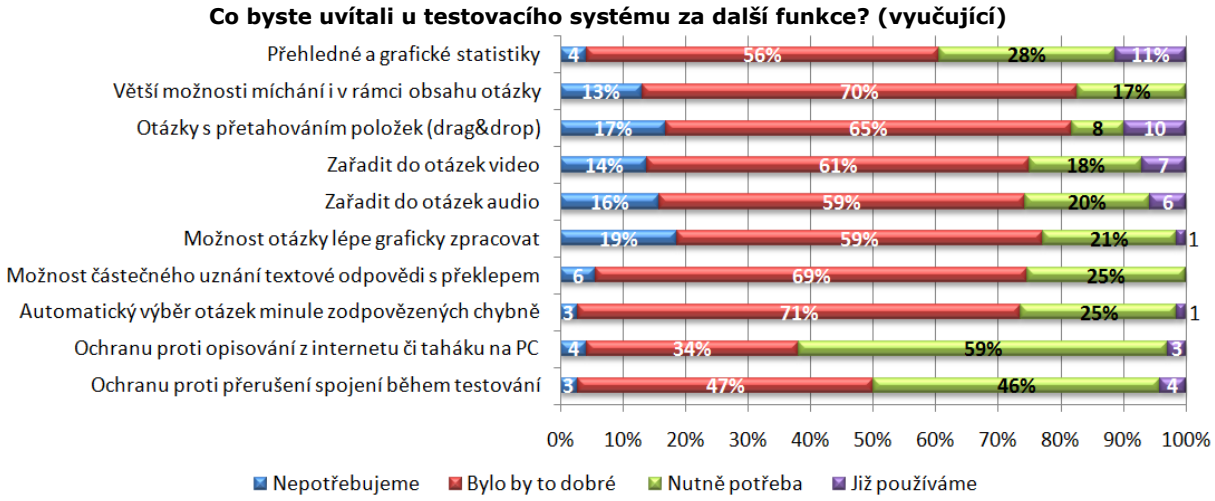
je příslušná funkce respondenty považována a 100% je maximum, kdy by funkci všichni označili jako absolutně nezbytnou.



Graf 11 – Seřazení stávajících funkcí dle jejich oblíbenosti

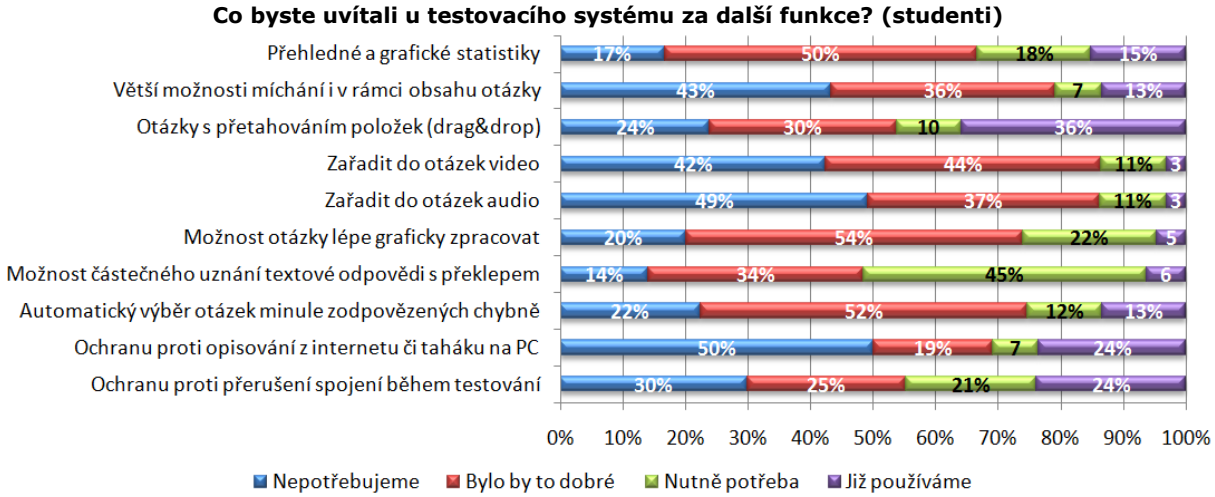
V tomto případě se dotazovaní vyučující a studenti na pořadí důležitosti funkcí shodli, ač studenti funkcím přikládali nižší hodnocení. Dle T-testu i Mann-Whitney testu ovšem statisticky prokazatelná shoda obou skupin nastala pouze v prvním případě, u funkce zpětné vazby a u ostatních funkcí shoda na 5% hladině významnosti prokázaná nebyla.

Další otázka směřovala na subjektivní hodnocení důležitosti funkcí, které u testovacích systémů nejsou až zas tak běžné. Preference vyučujících ukazuje Graf 12 a studentů Graf 13.



Graf 12 – Rozložení preferencí požadavků vyučujících na funkce testovacího systému (71 resp.)

Kromě těchto funkcí respondenti také mohli vepsat své další požadavky do rozsáhlého textového pole. Pouze jeden z nich této možnosti využil a uvedl, že by uvítal funkci pro zadání otázky formou obrázku, do kterého by se přímo vpisovaly požadované údaje. Druhým jeho požadavkem byla funkce, která by umožňovala učiteli sledovat, jak žáci v průběhu testu postupují (kolik již kdo zodpověděl otázek a jsou-li správně či chybně).



Graf 13 – Rozložení preferencí požadavků studentů na funkce testovacího systému (66 resp.)

I zde jeden student využil možnost vepsat neuvedenou funkci, o kterou by měl zájem, a jí byla „nápověda".

Také v těchto dvou případech byl proveden přepočet hodnocení na jednotnou ordinální hodnotu pro každou funkci a skupinu respondentů (Graf 12 a Graf 13). Číselné hodnoty byly tentokrát zvoleny tak, že stupeň Nepotřebujeme = 0, Bylo by to dobré = 1, Nutně potřeba = 2 a Již používáme = 2. Poslední stupeň totiž nevyjadřuje vyšší důležitost než ten předešlý, ovšem dá se z něho odvodit, že pokud je daná funkce již používána, pak je potřebná. Tentokrát bylo řazení vyučujících a studentů rozdílné, proto je v Tab. 1 uveden seřazený seznam pro každou skupinu zvlášť.

	Vyučující		Studenti	
1	Ochranu proti opisování z internetu či taháku na PC	79%	Možnost částečného uznání textové odpovědi s překlepem	69%
2	Ochranu proti přerušení spojení během testování	74%	Otázky s přetahováním položek (drag&drop)	61%
3	Přehledné a grafické statistiky	68%	Přehledné a grafické statistiky	58%
4	Automatický výběr otázek minule zodpovězených chybně	62%	Ochranu proti přerušení spojení během testování	57%
5	Možnost částečného uznání textové odpovědi s překlepem	60%	Možnost otázky lépe graficky zpracovat	53%
6	Zařadit do otázek video	56%	Automatický výběr otázek minule zodpovězených chybně	51%
7	Zařadit do otázek audio	55%	Ochranu proti opisování z internetu či taháku na PC	40%
8	Možnost otázky lépe graficky zpracovat	52%	Větší možnosti míchání i v rámci obsahu otázky	39%
9	Větší možnosti míchání i v rámci obsahu otázky	52%	Zařadit do otázek video	36%
10	Otázky s přetahováním položek (drag&drop)	51%	Zařadit do otázek audio	32%

Tab. 1 – Seřazení funkcí dle subjektivní důležitosti pro respondenty

T-test na 5% hladině významnosti prokázal statisticky významnou shodu v hodnocení vyučujících a studentů u funkcí 4, 5, 8, 10 a 3 (číslo označuje funkci uvedenou ve sloupci *Vyučující* na stejně označeném řádku Tab. 1). Mann-Whitney test dopadl obdobně, s tou výjimkou, že navíc prokázal shodu, ovšem velmi hraniční, u funkce 9.

3.3 Požadavky versus realita

V první části této kapitoly bylo představeno několik testovacích systémů z různých oblastí. Je zřejmé, že v případě potřeby lze nalézt testovací systém pro libovolný specifický účel, ať již jde o autotesty, hodnocené testování, prezentační testy nebo i programy pro efektivní samostatnou výuku.

Především testové moduly, jež jsou součástí LMS nabízejí bohaté možnosti nastavení a široké palety otázek. Nutnost provozu samotného LMS pro jejich užití však může být v některých případech omezením, stejně jako v jiných přínosem. Ostatní systémy či aplikace obvykle tak rozsáhlé možnosti neměly, případně byly uzpůsobeny právě pro vytváření a přidávání testu do LMS.

Ve druhé části byly uvedeny výsledky dotazníkového šetření právě na téma testovací systémy. Nastíněna tak byla stávající rozšířenost těchto systémů na českých školách a spokojenost jejich uživatelů s touto formou testování a funkcemi, které poskytují.

V následujících podkapitolách budou obě předešlé části porovnány. Na jedné straně budou diskutovány stávající elektronické testovací systémy, a na straně druhé aktuální stav jejich využívání a požadavky na ně kladené.

3.3.1 Důležité funkce testovacích systémů

Nejprve bude pozornost zaměřena na stávající obvyklé funkce, kterými disponuje většina testovacích systémů. Zdrojem pro uživatelské hodnocení těchto funkcí jsou data uvedená v kapitole 3.2.3, konkrétně Graf 10 a Graf 11.

Z obvyklých funkcí vyučující nejvíce oceňují **automatické vyhodnocování** a **zpětnou vazbu**, která je velmi ceněná i u studentů. Tyto funkce jsou totiž tím, co jim přináší největší výhodu oproti papírovým testům, neboť se samy opraví, vyhodnotí a studenti si mohou samostatně prohlédnout, jak dopadli a kde udělali chyby, bez nutnosti s tím zatěžovat vyučujícího. Tyto funkce měly všechny uvedené systémy pro uzavřené otázky. S těmi otevřenými si poradili pouze u krátkých jednoznačných odpovědí. Výjimkou byla pouze iŠkola (viz str. 15), kde bylo nutné ručně opravit všechny otevřené otázky.

Možnost sebetestování a **dostupnost testů odkudkoli** také vyučujícím připadají jako důležité vlastnosti dobrého systému. V ideálním případě by se díky této funkci student učil tak dlouho, dokud by autotest nesložil na plný počet bodů. Ovšem i v případě, že je test používán pro sebe výuku metodou pokus-omyl, přináší určitý efekt. Díky dostupnosti odkudkoli, tak navíc studenti mohou činit i z pohodlí svého domova. Tyto vlastnosti měly všechny uvedené systémy, které fungují on-line na webu. Desktopové aplikace jako Program Testy (viz str. 18), Alf (viz str. 21) nebo třeba Memostation (viz str. 26) sice dostupné odkudkoli nejsou, ale studenti si je mohou nainstalovat a používat i doma.

Jako nejvíce nežádoucí funkce u studentů vyšlo **časové omezení testu**. Tomu by se sice nevyhnuli ani u papírových testů, ovšem v elektronické verzi bývá časový limit přísnější a neúprosný, ale na druhou stranu i spravedlivější. Čas je měřen s přesností na vteřiny, obvykle bez možnosti prodloužení, což může na některé studenty působit stresově.

Druhou, studenty nejméně oblíbenou funkcí, je **míchání otázek a odpovědí**. Tato možnost totiž komplikuje vzájemné opisování a učení se stylem „v 1. otázce je správná odpověď b), ve 2. c), ve 3. a), …“. Tím ovšem studenty nutí, byť k nelibosti některých z nich, skutečně se učit.

3.3.2 Žádané funkce testovacích systémů

Tato část bude zaměřena na rozbor funkcí, které respondenti v dotazníku označili jako důležité a žádoucí. Hlavním faktorem bude dostupnost těchto funkcí a jejich variabilita v uvedených systémech. Vzhledem k odlišným preferencím respondentů z řad vyučujících a studentů budou v případech, kde nedošlo ke shodě na pořadí, funkce rozebrány odděleně dle preferenčního žebříčku obou skupin respondentů. Zdrojem pro uživatelské hodnocení těchto funkcí jsou data uvedená v kapitole 3.2.3, konkrétně Graf 12, Graf 13 a Tab. 1.

Vyučující

Funkce, kterou by uvítala drtivá většina vyučujících je **ochrana proti opisování z internetu či taháků na PC**. Během ostrého zkoušení plné učebny studentů pomocí nějakého on-line testovacího systému je mnohdy velmi těžké ohlídat, aby se někdo v momentě, kdy není sledován, nesnažil v jiném okně prohlížeče vyhledat odpovědi na otázky z testu, případně se přes chat neradil s někým jiným či naopak nenapovídal kolegovi. Neobvyklé také není hledání ve vlastních poznámkách v textovém editoru či prezentaci z přednášky, k čemuž ani není zapotřebí připojení k internetu.

Z druhé strany je toto pochopitelně funkce, kterou studenti hodnotí jako jednu z nejvíce nechtěných. Takovouto funkcí žádný z probíraných systémů nedisponoval, pouze Moodle (viz str. 13) umožňoval překrýt oknem prohlížeče s omezenými funkcemi celou plochu obrazovky, avšak to se dalo velmi jednoduše zase přepnout, či zmenšit. JavaScript, blokující kopírování obsahu stránky, ovšem znesnadňoval zaslání zadání prostřednictvím internetu někomu dalšímu.

Druhou, vyučujícími nejžádanější funkcí (u studentů čtvrtou), je **ochrana proti přerušení spojení během testování**. Pokud totiž v průběhu testování dojde k přerušení nebo dočasné přetíženosti spojení se serverem, na kterém je testovací systém umístěn, může tím být narušen celý průběh testování. V případě testů realizovaných přes HTML stránky obvykle při překročení timeoutu[21] dojde k zobrazení chybové stránky a ztrátě, nebo alespoň zmizení všech ve formuláři vyplněných dat. Některé systémy, především ty integrované do LMS, umožňují průběžné ukládání otázek a v případě výpadku tak dojde pouze ke ztrátě dat z poslední vyplňované otázky. Zároveň umožňují test přerušit a pokračovat v něm později, třeba z jiného počítače (např. WebCT, viz str. 10), ovšem čas testu se obvykle odpočítává i během tohoto neplánovaného přerušení. Jsou-li všechny otázky testu zobrazeny na jedné stránce, krátké výpadky v průběhu testu vadit nemusí, avšak dojde-li k němu během odesílání výsledků, může být ztráta o to větší.

Z on-line systémů pouze Quizmaker (viz str. 21), který je vytvořen jako flashová aplikace, není po svém načtení, do chvíle odeslání výsledků zpět do LMS, závislý na nepřetržitém spojení, byť se jednotlivé otázky nachází na oddělených slajdech. Při dlouhodobém výpadku ovšem ani zde nezbývá, než celý test zrušit a zopakovat jej jindy. I v případě výpadku chvilkového, kdy pouze dojde ke zdržení a minimální či žádné ztrátě dat, se na toto ovšem mohou odvolávat méně úspěšní studenti a zpochybnit tak hodnocení nejen své, ale i všech ostatních. Z uvedených systémů mohou tomuto čelit pouze Odpovědníky (viz str. 15), a to jen ve své tištěné formě určené pro skenování.

Další žádanou funkcí jsou **přehledné a grafické statistiky**. Většina uvedených systémů sice poskytovala podrobné seznamy výsledků i jejich rozbory např. dle jednotlivých otázek, ovšem

[21] timeout na webu – maximální čas, po který se prohlížeč pokouší spojit se serverem, než operaci zruší

grafické zhodnocení scházelo. Přitom díky němu se lze obvykle v přehledu rychleji zorientovat a odvodit z něho nějaké závěry. Pokud je však kvůli tomu nezbytné data vyexportovat a ručně v Excelu upravovat, uživatel se většinou raději spokojí s možností přehled seřadit a hlouběji nezkoumat. Grafické vyhodnocení by zároveň uvítali i respondenti z řad studentů.

Automatický výběr otázek minule zodpovězených chybně umožňovaly především systémy určené pro opakování s vhodně zvolenou prodlevou (viz kap. 3.1.4), ovšem za jiným účelem. U testů používaných výhradně ve škole pro občasné zkoušení nemá aplikace metody *spaced repetition* smysl.

Podstata této funkce má dva rozměry, a to při autotestech, kdy je žádoucí nepředkládat studentovi otázky, které již dříve úspěšně vyřešil a někdy i při zkoušení, kdy může být potřeba ověřit, doučil-li se student látku, kterou minule neznal. Do této funkce lze zahrnout i schopnost systému preferovat pro výběr z rozsáhlé databáze otázek ty, které dosud zkoušenému předloženy nebyly. Tuto volbu testovací systémy schopné provozu pro ostrý hodnocený test ovšem nenabízejí.

Možnost zadat otázku formou **obrázku s přímým vpisováním údajů**, kterou navrhl jeden respondent (vyučující), umožňovaly pouze Odpovědníky při vytváření přes Quest-o-mat [w36] (viz str. 15) nebo Quizmaker (viz str. 21). Co se týče možnosti **sledovat postup studentů v průběhu testu**, tak tou např. disponuje autorský program představený v článku [33], z uvedených systémů však žádný.

Studenti

Studenty nejžádanější funkcí byla **možnost částečného uznání textové odpovědi s překlepem**, kterou i vyučující zhodnotili jako přínosnou. Samozřejmě že ne vždy je takováto funkce žádoucí, ale existují případy, kdy by mohla více zobjektivnit hodnocení. Pro tuto možnost ovšem téměř všechny uvedené systémy neposkytují žádné zcela uspokojivé řešení. Obvykle lze zadat více variant, se kterými je zadaný text porovnáván, zrušit *case sensitive* komparaci, vzorový text doplnit různými maskami nebo jej vyhodnotit pomocí regulárního výrazu jako např. ve WebCT (viz str. 10). Pouze Memostation (viz str. 26) dokáže vyhodnotit zadaný text Jaro-Winklerovým algoritmem, který určuje procentní shodu dvou textů, a tak i jeden překlep neznamená zcela chybnou odpověď.

Jak již bylo zmíněno, většina respondentů z řad studentů již pracovala s aplikací, jež je předmětem této práce, což může být důvodem, že se na druhém místě jejich žebříčku umístila funkce podporující otázky řešené metodou **drag&drop**. Ty jsou totiž u studentů celkem oblíbené, spíše však pro svou efektnost a interaktivitu, než z důvodu, že by jejich řešení snad bylo snazší. Vyučující, u nichž byla funkce zařazena až jako nejméně žádoucí ze všech předložených, naopak tento systém neznají a mohou mít obavy jednak z náročnosti přípravy takových otázek, jednak z nemožnosti jejich použití v tištěné formě a také proto, že každý vyučující má obvykle za svou praxi nastřádánu určitou databázi otázek, kterou mohou někteří považovat za dostatečnou.

Možnost **nápovědy**, kterou zmínil jeden z respondentů, může být sice brána jako studentský žert, nicméně při hlubším zamyšlení je neexistence této funkce při testování, ať již elektronickém či na papíře, jedním ze zásadních rozdílů oproti ústnímu zkoušení. Při něm totiž, pokud si zkoušený student nemůže vybavit určitý fakt, ale je patrné, že látku ovládá, může zkoušející studenta vhodně nasměrovat tak, že si naučené vědomosti vybaví. I kdyby měla takováto nápověda vést k horšímu hodnocení, nemusí to díky ní ale být stupeň nejhorší.

Uvedené systémy sice umožňují formulovat otázku jakkoli, ale její vícestupňovou, na popud studenta postupně zobrazovanou konkretizaci, neumožňuje žádný z nich. Pouze *Hot Potatoes* (viz str. 19) v otázce *JQuiz* typu *hybrid* dokáže při chybně psané odpovědi zobrazit seznam jejích možných znění. Nikde však není možnost nastavit podmínky užití pro takovéto nápovědy, např. penalizační body za její zobrazení.

3.3.3 Podpora požadovaných funkcí ve stávajících systémech

Poznatky ze zkoumaných stávajících testovacích systémů ve vztahu k požadavkům na ně kladeným shrnuje následující tabulka (viz Tab. 2).

Testovací systémy	LMS					Nezá-vislé			Výukové					Drilovací				
Nedostatky	**WebCT**	**Moodle**	**iŠkola**	**Odpovědníky**	**Dokeos**	**Test park**	**ClassMarker**	**Program Testy**	**Hot Potatoes**	**eXe**	**Alf**	**Quizmaker**	**TestBuilder**	**SuperMemo**	**Anki**	**Menomosyne**	**Memostaion**	**Dril**
Nedostačující interaktivita	X	X	X	X	-	X	X	X		X				X	X	X	X	X
Řešení pomocí drag&drop	X	X	X	X	X	X	X	X	-	X			X	X	X	X	X	X
Pouze předdefinované typy otázek	X	X	X		X	X	X	X	X	X	X		X	X	X	X	X	X
Pro grafiku je třeba nakreslit a vložit obrázek	X	X	X	X	X	/	X	X	X	X	X		X	X	X	X	X	X
Animace = video	X	X	/	X	X	/	X	X	X	X	/		X	/	X	/	/	X
Úzká specializace na konkrétní typ testování			X			X				X	X			X	X	X	X	X
Provázanost s jinými systémy	X	X	X	X	X								X					X
Nemožnost propojení s jinými systémy	X	X	X	X	X	X	X	X			X		X	X	X	X	X	X
Není dostupné odkudkoli								X	-	-	X	-		X		X	X	
Potřeba instalace a údržby	X	X	X	/	X			X	X	X	X	X	X	X	X	X	X	/
Nezbytný vlastní či pronajatý server	X	X		/	X								X					/
Inteligentní výběr otázek do testu	X	X	X	?	X	X	X	X	X	X	X	X	X					
Interní míchání v otázkách	-	-	-	-	-	X	X	-	-	X	X	X	-	X	X	X	X	X
Přerušení spojení naruší průběh testování	-	-	X	-	X	X	-	/	/	/	/	-	X	/	X	/	/	X
Není ochrana proti opisování z internetu	X	-	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Fuzzy hodnocení jednotlivých prvků v otázce			X			X	X				X			X	X	X	X	X
Fuzzy hodnocení textových odpovědí s překlepem	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
Nechráněno proti „hackerským" zásahům									X	X				/	/	/	/	
Nedostatečné možnosti časování dostupnosti	-	-	-	-	-	/	-	X	/	/	/	/	-	/	/	/	/	/
Nedostatečné nastavení podmínek absolvování	-	-	X	-	-	X	-	X	X	X	X	-	-	/	/	/	/	/
Nemožnost zařazení penalizované nápovědy	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Vysoká cena	X		-	/			X	-			X	X	X	X				/

Tab. 2 – Podpora požadovaných funkcí ve stávajících testovacích systémech

V Tab. 2 křížek (X) značí, že testovací systém obsahuje příslušný nedostatek, pomlčka (-), že nedostatek je řešen ovšem jen částečně, lomítko (/), že tato funkce není podporována vůbec, popř. není pro daný systém relevantní a otazník (?), že se danou vlastnost nepodařilo ověřit.

3.3.4 Shrnutí

Elektronických testovacích systémů existuje pro každou formu testování celá řada. Rozsah jejich funkcí je různý, v základu však všechny umožňují minimálně jejich hlavní účel užití, tzn. sestavovat a provozovat testy.

Možnosti, které tyto systémy odlišují od klasického testování na papír (např. automatické vyhodnocování a míchání otázek), již dnes většina uživatelů považuje za samozřejmý standard a ceněny jsou spíše funkce, které více využívají potenciálu, jež sebou elektronická forma testování přináší. To samozřejmě nesmí být na úkor ovladatelnosti a uživatelské přívětivosti těchto systémů.

Jak je patrné ze závěrečného přehledu v Tab. 2, každý ze zkoumaných stávajících testovacích systémů má řadu nedostatků. Ty jsou ovšem v současné době již řešitelné, pomocí existujících nebo odvoditelných postupů. I přes nemalé množství těchto systémů tak zde stále existuje prostor pro vznik nového, uvedených nedostatků prostého, univerzálního testovacího prostředí.

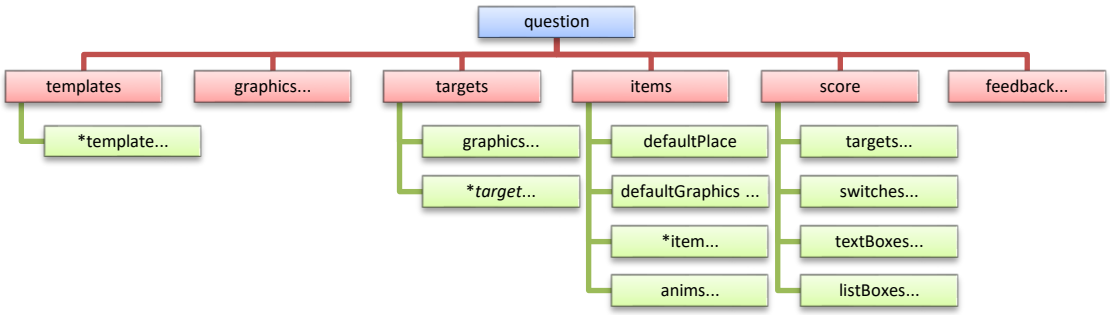
4 VÝSLEDKY DISERTAČNÍ PRÁCE

Systém pro testování, který je předmětem této práce, je vyvíjen jako cloud RIA aplikace na platformě Silverlight, což *je softwarový plugin pro vývoj bohatě vybavených internetových aplikací spouštěných v rámci webového prohlížeče, vyvinutý společností Microsoft, jež je menší verzí .NET Frameworku a podporuje kód napsaný v různých jazycích .NET, zejména C# a Visual Basic.* [34]

4.1 Jazyk pro tvorbu otázek [◎ II]

(z části publikováno v [ap-8])

Každý test se skládá z jednotlivých otázek, přičemž zde je ke každé z nich přistupováno jako k samostatnému dílu. Jelikož zamýšlenou variabilitu designu a funkcí otázek nebylo možné pokrýt žádným existujícím způsobem ani databázovým schématem, byl pro jejich zápis vytvořen vlastní jazyk, pracovně nazývaný **QML** (Questions Markup Language), který, jak již jeho název napovídá, je založen na klasickém XML. Podobně jako jiné jazyky odvozené z XML i tento definuje vlastní strukturu elementů, které se v dokumentu mohou vyskytovat, na jakém místě a jaký mají význam v definici otázky.



Obr. 3 – Schéma základních elementů QML

Základním kořenovým elementem v QML je `<question>`. Ten se dále větví na další základní elementy, definující jednotlivé viditelné a funkční části otázky. Obr. 3 ukazuje tuto strukturu, přičemž *kurzívou* označené elementy mají ve skutečnosti jiný název z několika různých možností, hvězdička (*) před názvem elementu znamená, že na tomto místě může být uveden opakovaně vícekrát a tři tečky (…) za názvem elementu naznačují jeho další větvení.

4.1.1 Grafika otázky [◎ II.1]

Pro tvorbu grafického vzhledu otázky byl v QML vytvořen mini-jazyk (QML-graphics), který umožňuje používat základní grafické objekty a to jak vektorové (čáry, elipsy, křivky apod.) tak i rastrové (obrázky). Inspirací pro něho přitom byly již existující standardy a to SVG a XAML.

SVG *(Scalable Vector Graphics) je W3C schválený XML jazyk pro vektorové kresby. Definuje prvky, které představují polygony, obdélníky, elipsy, čáry, křivky a další. Nové tvary mohou být definovány pomocí jednoduchého jazyka. Barevná schémata a vzory lze aplikovat na tvary pomocí oříznutí, maskování, kompozice, výplní a přechodů. Kromě toho se mohou obrazce na stránce pohybovat. JavaScript může umožnit i reakci tvarů na vstupu uživatele. SVG je kompletní formát pro detailní popis dynamické vektorové grafiky. Pro statickou grafiku je SVG téměř na stejné úrovni*

Adobe EPS formát a výrazně lepší než CGM (Počítačová grafika Metafile). Pro animované obrázky je asi na podobné úrovni jako proprietární SWF formát používaný v Macromedia Flash. [35 str. 849]

*Jazyk **XAML** (eXtensible Application Markup Language) byl vytvořen společností Microsoft výhradně pro účely propojení s jeho .NET Frameworkem. XAML dává vývojářům možnost definovat rozložení všech prvků uživatelského rozhraní, jako jsou tlačítka, text, grafika atd. pomocí XML. Každý XAML tag odpovídá konkrétní .NET třídě, jejíž vlastnosti jsou definovány pomocí atributů XML. XAML elementy představují Common Language Runtime (CLR) třídu, runtime engine pro Microsoft .NET Framework. CLR je podobný Java Virtual Machine (JVM), kromě toho, že JVM může spouštět pouze programy v jazyce Java, zatímco CLR může spustit aplikace napsané v řadě .NET jazyků, jako je C#, J# a VB.NET.* [36 str. 3]

Vzhledem k faktu, že systém je vyvíjen na platformě Silverlight, která stejně jako WPF[22] používá pro definici designu aplikace jazyk XAML, je i QML-graphics, především co se týče transformací a animací, odvozena spíše z jazyka XAML než SVG. Ve spoustě aspektů si jsou ovšem oba srovnávané jazyky podobné, přičemž QML-graphics má snahu použít z obou vždy tu pro jeho účely lepší formu zápisu a eliminovat jejich nedostatky. Kromě toho přináší řadu dalších novinek (např. aktivní prvky umožňující automatizovanou interakci s uživatelem), které jsou nezbytné či jen užitečné pro tvorbu funkčních testových otázek.

Element graphics [◎ II.1.A]

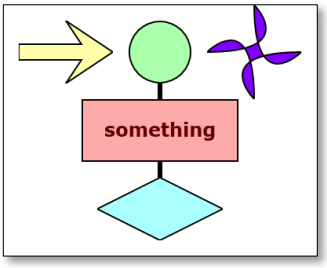
Element `<graphics>`, který se nachází jak v kořenovém elementu otázky, tak i pod jednotlivými cíli `<targets>` a položkami `<items>`, může obsahovat širokou paletu nejrůznějších grafických prvků, které tvoří vzhled i obsah každé otázky.

```
<question width="200" height="160">
  <graphics color="000000" thickness="1">
    <font family="Verdana" size="12" weight="bold" color="660000" />
    <line x="100" y="30" x1="0" y1="0" x2="0" y2="100" thickness="3" />
    <ellipse x="80" y="10" width="40" height="40" background="AAFFAA" />
    <rectangle x="50" y="60" width="100" height="40" background="FFAAAA" />
    <text x="C" y="72">something</text>
    <rhombus x="60" y="110" width="80" height="40" background="AAFFFF" />
    <polygon x="10" y="10" background="FFFFAA"
        points="60,20;30,40;40,25;0,25;0,15;40,15;30,0" />
    <path x="130" y="0" background="8000FF"
        data="M 30,0 c 0,0 30,30 -30,30 c 0,0 30,-30 30,30 c 0,0 -30,-30 30,-30
              c 0,0 -30,30 -30,-30 z" />
  </graphics>
</question>
```

Kód 1 – Ukázka zápisu základních grafických elementů

Kód 1 ukazuje zápis základních grafických elementů, konkrétně úsečky (`line`), elipsy (`ellipse`), obdélníku (`rectangle`), textu (`text`), kosočtverce (`rhombus`), polygonu (`polygon`) a cesty (`path`). Element `<font>` pouze definuje vlastnosti písma pro všechny texty na stejné nebo nižší úrovni hierarchie. Může být také použit přímo v elementu `<text>`. Vynechané atributy jsou pak nastaveny na hodnoty uvedené na vyšších pozicích hierarchie, nebo na hodnoty výchozí. Výsledek tohoto kódu je zobrazen na Obr. 4.

[22] WPF – Windows Presentation Foundation



Obr. 4 – Ukázka grafického výstupu základních elementů (viz Kód 1)

Grafické elementy jsou přes sebe vykreslovány v tom pořadí, v jakém jsou zapsány, takže později zapsaný prvek překryje prvek zapsaný dříve (překrývají-li se), např. čára je překryta elipsou, obdélníkem a kosočtvercem.

V kódu je také použit princip dědičnosti a to například u atributu `thickness`, který určuje sílu čar a obrysů. Ten je definován v nadřazeném elementu `<graphics>` a jeho podelementy jej berou jako výchozí, nemají-li sami, nebo na hierarchicky bližší vyšší úrovni, definovánu jinou hodnotu (jako má např. `<line>`). Stejné pravidlo by platilo i pro další vlastnosti, u kterých má dědičná definice smysl, zde tedy barvu čáry (`color`) a barvu pozadí (`background`). Kromě plných či částečně průhledných barev lze v QML použít i barevné přechody a to jak lineární tak radiální (viz kap. 4.12.4, str. 152).

Objekt `<polygon>` je uzavřený obrazec složený z úseček. Ty jsou definovány jednotlivými body. Souřadnice [0,0] je určena pozicí prvku (`x` a `y`), šířka a výška pak pozicí nejvzdálenějšího bodu v daném rozměru. Body jsou zapisovány v atributu `points` v párech rozdělených čárkou (`x,y`) a jednotlivé body jsou od sebe odděleny středníkem. Pokud souřadnice posledního bodu neodpovídá souřadnicím prvního bodu, je automaticky spojnice těchto dvou bodů doplněna a objekt polygon tak vždy uzavřen.

Obdobně funguje i objekt `<path>`, který má svou definici uvedenou v atributu `data`. Formát zápisu tvaru objektu se provádí a v tzv. *Path Mini-Language*, který kromě úseček umožňuje zadávat i různé druhy eliptických oblouků a kubických, kvadratických a „vyhlazených" Beziérových křivek. Stejný formát tohoto mini-jazyka používá XAML i SVG (např. viz [36] a [37] ale i [38] a [35]).

Rastrové obrázky [◎ II.1]

Rastrové obrázky se nahrávají na server přes administrační rozhraní (viz kap. 4.3.2, str. 72), přičemž je jednak vypočten jejich hash a také je jim přidělen jedinečný textový kód UUID. Hash je určen pro případ, že by se v budoucnu došlo k pokusu nahrát na server týž obrázek znovu. Pak je podle hashe identifikována shoda, zamezeno nahrání duplicitního obrázku a použit je obrázek původní. UUID slouží pro používání obrázku v otázkách, jelikož právě tento kód se uvádí do atributu `src` v elementu `<img>` (viz Kód 2).

```
<graphics>
  <image src="Sg65a10t5e" stretch="Fill" x="10" y="10" width="640" height="680" />
</graphics>
```

Kód 2 – Ukázka použití rastrového obrázku

Dalším nepovinným atributem, kromě pozice a určení rozměrů, je `stretch`, který určuje, jakým způsobem se bude obrázek přizpůsobovat vymezené ploše (viz [w30]).

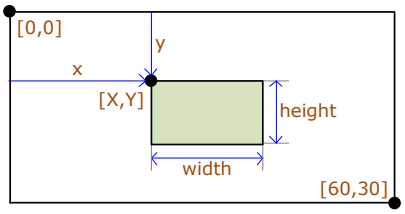
Okno „preview"

V otázce nemusí být vždy veškerý její obsah zobrazen hned od začátku. Element `<preview>` umožňuje vytvořit grafický objekt, na který když zkoušený klikne, zobrazí se nad otázkou na daných souřadnicích okno, které může obsahovat další grafiku a informace. Okno lze otevřít v režimu `modal`, kdy je zbytek otázky znepřístupněn a překryt poloprůhledným tmavým panelem, nebo klasicky tak, aby se během jeho zobrazení dalo s otázkou i nadále pracovat. Okno zkoušený zavře kliknutím na něj, v případě `modal` zobrazení pak kliknutím kamkoli do otázky.

Tohoto okna lze využít např. pro nápovědu nebo podrobnější vysvětlení zadání, především u autotestů. Je možné také omezit počet, kolikrát lze toto okno zobrazit a zároveň i čas, po který může být zobrazeno (po jeho vypršení se okno automaticky uzavře).

Souřadnicový systém [◎ II.1]

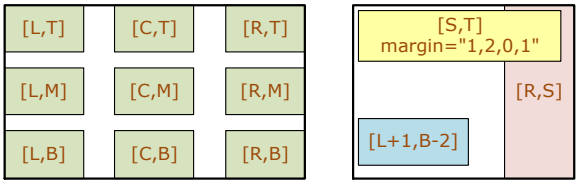
Souřadnicový systém pro rozmisťování grafických prvků je zde řešen klasickým způsobem, tj. jako IV. kvadrant osového souřadnicového systému, přičemž hodnoty na ose Y jsou absolutní (kladné). V levém horním rohu je tedy počátek [0,0], doleva se přičítá souřadnice X a dolů Y (viz Obr. 5).



Obr. 5 – Souřadnicový systém

Každý prvek má tedy polohu svého levého horního rohu určenou dvěma souřadnicemi. Od tohoto bodu se počítá jeho šířka a výška. Ta může být definována pevnými hodnotami, nedefinována vůbec, přičemž je určena automaticky dle obsahu prvku, nebo stanovena tak, že se roztáhne přes celou výšku či šířku kontejneru[23], ve kterém je prvek umístěn.

V rámci kontejneru mohou být grafické prvky rozmisťovány nejen absolutně, ale i relativně a to vzhledem k okrajům či středu kontejneru. V takovém případě je v příslušné souřadnici místo čísla uveden znak této pozice, tj. pro X je to `L` (left), `C` (center) nebo `R` (right) a pro Y je to `T` (top), `M` (middle) nebo `B` (bottom) (viz Obr. 6 vlevo). Za tímto znakem může též následovat znaménko („+" nebo „-") a hodnota, o kterou se má prvek posunout oproti znakem určenému zarovnání. Tedy např. `x="C-10"` znamená zarovnání na střed posunuté o 10 pixelů doleva.



Obr. 6 – Ukázky relativních souřadnic

[23] kontejner – grafický poziční element umožňující v sobě mít další podelementy, pro které je on hlavním pozadím (kreslícím plátnem)

Při roztažení prvku přes některý z rozměrů kontejneru nemají pozice souřadnic smysl, proto je možné je v tomto případě využít rovnou pro toto nastavení a to hodnotou `S` (stretch). Pak již není třeba nastavovat rozměry prvku. Při roztažení je také možné použít vlastnost `margin` (odsazení od okrajů), která zde oproti XAML funguje shodně jako v CSS pro HTML [39 str. 25], tj. má 4, 2 nebo 1 hodnotu. Prvek pak nemusí být roztažen až k úplnému okraji kontejneru, ale odsazen od něho právě o hodnotu `margin` pro příslušnou stranu (viz Obr. 6 vpravo).

Poziční prvky

Kromě vizuálních grafických prvků lze používat i standardně neviditelné prvky sloužící pouze pro snazší a uspořádanější rozložení layoutu otázky. Jsou jimi `grid` (pro uspořádání do tabulky), `stack` (pro uspořádání vedle sebe nebo pod sebe) a `wrap` (pro uspořádání vedle sebe s možností zalamování řádků). Grid i stack (StackPanel) jsou převzaty přímo z jazyka XAML, wrap pak využívá komponentu ze Silverlight Toolkit[24].

Prvek grid je užitečný pro relativní, automatické polohovací strategie, při kterých je nutné mít nad umístěním prvku určitou kontrolu. Grid je podobný tabulce (table v HTML) a poskytuje jednotlivé buňky, ve kterých mohou být umístěny prvky. Velikost tabulky i buněk může být explicitně nastavena číselnou hodnotou v pixelech, jako procentní podíl celkové dostupné šířky a výšky nebo odvozena automaticky na základě obsahu. [36]

Tyto prvky jsou tedy kontejnery, které obsahují další grafické podprvky. Pro ně je pak tento brán jako hlavní kreslící plátno. Absolutní pozici prvků lze přitom stanovovat pouze v kontejneru typu `grid` (nahrazuje tak zároveň i kontejner typu `Canvas` používaný v XAML). V ostatních dvou je možné používat pouze `margin` pro odsazení od ostatních prvků, případně lze objekt zarovnat k okraji nebo na střed souřadnice kolmé na směr řazení objektů v daném kontejneru.

4.1.2 Transformace a animace [◎ II.1.C]

Transformace umožňují objektům i jejich skupinám, umístěných na pozičních prvcích, změnu jejich měřítka nebo otočení. Pouze díky nim lze vytvořit např. pootočenou elipsu.

Animace jsou pak jednou z výhod, kterými papírový test nikdy disponovat nebude, i proto v QML nesmí chybět jejich podpora. Zápis animací v XAML a SVG se celkem liší, ač výsledku lze dosáhnout téhož. Příloha 1 porovnává některé rozdílně řešené zápisy definic animací všech tří jazyků.

Jak vypadá jednoduchá animace zeleného obdélníku s pozicí středu na [300, 200] zapsaná v jednotlivých jazycích ukazuje Příloha 3. Tento obdélník má donekonečna postupně měnit šířku a výšku s iluzí mačkané houby, tj. při natahování do šířky se snižuje jeho výška a při následném stlačování šířky výška narůstá (obě strany v rozmezí 95-105% originálních rozměrů 200x100px). V tomto konkrétním případě je délka zápisu v QML pouze 45% délky zápisu v SVG a dokonce jen 27% vzhledem k zápisu v XAML (bez *white-spaces*[25]).

QML nyní podporuje animace měřítka (`scale`), otočení (`rotate`), pohybu (`move`), průhlednosti (`opacity`) a barev (`color`).

[24] *Silverlight Toolkit obsahuje další ovládací prvky, které nejsou standardně zahrnuty v základních vývojářských kitech. Je šířen jako Open Source na adrese [w63].* [37]
[25] white-space, neboli prázdné místo, označuje mezery, tabulátory a znaky pro nový řádek [126]

4.1.3 Aktivní prvky

Aktivní prvky poskytují zkoušenému interaktivitu, tedy možnost do předložené otázky zasáhnout, něco vykonat a tím na ni odpovědět. [ap-8]

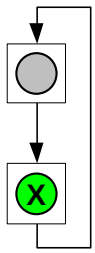
Přepínače

Jednou z nejobvyklejších testových otázek je otázka s výběrem jedné (*multiple-choice*) či více (*multi-select*) z nabízených odpovědí (a, b, c, …). Volbu předpokládané odpovědi zkoušený obvykle provádí jejím vyznačením (zaškrtnutím, zakroužkováním, podtržením apod.). Pro tyto účely slouží právě přepínače (`switch`).

Ty umožňují definovat takové grafické objekty, které se po kliknutí na ně změní v jiné (např. prázdný kroužek v zaškrtnutý kroužek, viz Obr. 7). Původní myšlenku však překračují, neboť mohou mít v podstatě neomezené množství stavů (nejen dva), přičemž grafický obsah každého z nich, kromě vymezené polohy a rozměrů, není nijak závislý na ostatních.

```
<graphics color="000000" thickness="1">
  <switch id="s1" x="20" y="20" width="20"
      height="20" group="1" stateOff="off">
    <state id="off">
      <ellipse x="S" y="S" background="C0C0C0" />
    </state>
    <state id="on">
      <ellipse x="S" y="S" background="00FF00" />
      <text x="C" y="M">X</text>
    </state>
  </switch>
</graphics>
```

Kód 3 – Ukázka kódu dvoustavového přepínače



Obr. 7 – Stavy přepínače (viz Kód 3)

Pro zobrazení zaškrtávacích políček je tedy možné použít libovolný vzhled. Lze také vytvářet propracovanější více stavové přepínače s bohatou grafikou (viz kap. 4.1.1) i animacemi (viz kap. 4.1.2). Pokud by se měl vzhled jednotlivých stavů lišit jen částečně (např. viz Příloha 2), je vhodné pro jejich vykreslení použít šablonu (viz kap. 4.1.4).

Každý přepínač má určen svůj výchozí „vypnutý" stav (`stateOff`, viz Kód 3), do kterého je nastaven na začátku. Přepínače je totiž možné seskupovat pomocí identifikátoru skupiny (`group`) a pokud je tato hodnota definována, pak může být mimo výchozí stav přepnut vždy pouze jeden z přepínačů této skupiny. Pokud se z něho přepne jiný, ostatní se automaticky vrátí do výchozího stavu. Díky tomu je možné simulovat tzv. „radio buttons" a vytvářet otázky, kde lze zvolit pouze jednu z nabízených odpovědí. Navíc zde odpadá problém s návratem k „nehodnocení", což u klasických „radio buttons" standardně nelze.

Textový vstup

Pro krátké otevřené otázky je možné použít textový vstup. Tím je klasické editační políčko, do kterého zkoušený vpisuje text, resp. předpokládané znění správné odpovědi na položenou otázku. Tento prvek je zastoupen elementem `<text>` a kromě jeho polohy, rozměrů a barev lze přednastavit i jeho výchozí obsah.

Řazení

Otázka, nebo její část může po zkoušeném vyžadovat, aby seřadil nějaké položky do správného pořadí („uspořádací úloha" dle [40 str. 49]). To je realizováno metodou drag&drop, kdy se jednotlivé položky mezi sebou přetahují tak, že se některá z nich umístí mezi jiné dvě a vše se okamžitě posune tak, aby nikde nezbyla mezera (viz Obr. 8).

Obr. 8 – Ukázka úlohy typu řazení, během přesunu položky

Položky pro řazení nemusí být jenom jednořádkový text, ale mohou obsahovat libovolné plnohodnotné grafické prvky a jejich skupiny, včetně obrázků. Řazení je kromě vertikálního podporováno i horizontální včetně funkce „wrap". Pokud se tedy při horizontálním řazení položky nevejdou v jednom řádku do definované maximální šířky, je tento řádek zalomen a pokračuje se pod ním na řádku dalším. Nachází-li se v jedné otázce více řadících jednotek (listboxů) je možné nastavit i to, že se prvky každého z nich dají přesunout nebo dokonce zkopírovat do jiného.

Položky a cíle [◎ II.1.D]

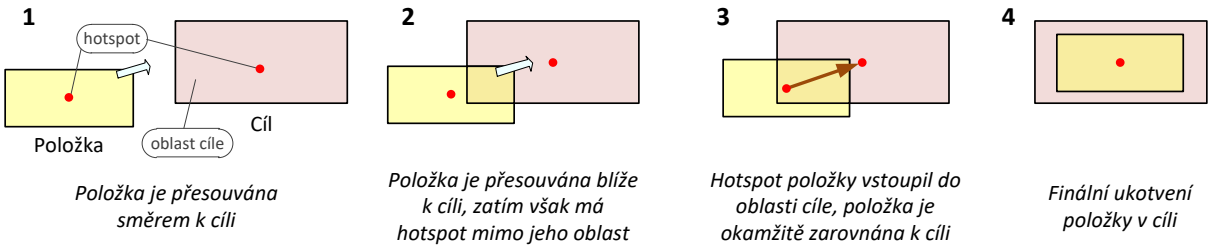
Součástí otázky mohou být i položky a cíle. Jedná se o velmi efektní typ úlohy, který se v klasickém HTML realizuje jen velice těžko, pomocí složitých JavaScriptů a obvykle má vedlejší nežádoucí efekty, jako např. označování textu na stránce během tažení položky (jako např. v Hot Potatoes, viz str. 19). *Obdobný typ úloh byl představen např. v článku* [41]*, kde byly testové položky zadávány jako JavaBeans*[26]*.*

V této podúloze jsou samostatně a nezávisle na sobě (jejich vztah určuje až bodování v elementu `<scores>`, viz kap. 4.1.7, str. 53) stanoveny cíle a položky. Položky jsou plovoucí objekty s vlastní grafikou, které lze myší přesouvat po celé ploše otázky metodou drag&drop.

Cíle jsou stanovené oblasti, do kterých se mají položky umísťovat. Grafika cílů by měla být definována v rámci grafiky otázky, ovšem je možné jim i v jejich elementu nastavit společnou grafickou značku. Cíle zároveň mají tu vlastnost, že pokud se do jejich vymezené blízkosti během tažení dostane položka, mohou ji „přitáhnout" k sobě a ta tak jakoby zapadne na své místo. Položky totiž mají v rámci své grafiky stanoven tzv. *hotspot*, pro který je při jejich přesunu kontrolováno, nedostal-li se do oblasti spadající pod některý z cílů, a pokud ano, je položka zarovnána na plochu tak, aby její *hotspot* byl přesně umístěn na *hotspotu* cíle (viz Obr. 9).

Každá položka přitom zapadá do každého cíle stejně, takže pouhým zkoušením přítomnosti této „magnetizace" různých položek v různých cílech nelze odhadnout, která položka kam patří. Pokud je cíl obsazen, další položky již nepřitahuje. Ukotvenou položku lze samozřejmě z cíle uvolnit, stačí za ni myší „táhnout" mimo oblast cíle.

[26] JavaBean – třídy programovacího jazyku Java



Obr. 9 – Ilustrace ukotvení položky do blízkého cíle

Cíle se definují třemi způsoby a to jako obdélník (`rectangle`), kruh (`circle`) nebo barva (`color`). Tyto typy mají vliv pouze na „magnetický rozsah" cíle, nikoli na jeho vzhled. Obdélníkový cíl (např. viz Příloha 4) je vymezen buď některým ze svých rohů či středem a šířkou a výškou, zatímco kruhový cíl pouze svým středem a poloměrem. V obou případech se souřadnice pozice cíle udávají pouze v rámci souřadnicového systému grafiky otázky (viz kap. 42), tedy bez možnosti použití pozičních prvků.

Barevný cíl je oproti tomu určen pouze kódem konkrétní barvy nebo více barev a svým *hotspotem*. Ten může mít také buď absolutní souřadnice, nebo určovat relativní posun od standardního umístění, kterým je střed barevné oblasti. To je velmi výhodné, mají-li se cíle vztahovat např. k určitým oblastem v grafice otázky (např. viz Příloha 7), které jsou rozmístěny pomocí pozičních prvků a jejichž poloha může být, vzhledem k možnostem interního míchání otázky, různá. Barevné cíle přitom mohou být rozlišeny i jediným stupněm na barevné paletě, který je pouhým okem nerozeznatelný (např. viz Příloha 5) a ani v případě strojového dekódování barvy nelze určit, který cíl je určen pro kterou položku. Navíc je k dispozici i možnost náhodného výběru z více barev.

Pro výchozí polohu položek je určen obdélníkový prostor, kde jsou na začátku náhodně rozmístěny a to včetně své `z` souřadnice (položky s vyšším `z` překrývají položky s nižším `z`, přičemž po kliknutí na položku je tato přesunuta nejvýše). Aby položky nesplynuly s ostatní grafikou, je jim možné nastavit i společnou animaci, která bude probíhat pouze po dobu jejich volného stání, tzn., dokud nebudou taženy či umístěny do některého z cílů. Počet cílů a položek může být různý.

4.1.4 Šablony [◎ II.2.A]

V některých otázkách se mohou určité části jejího QML zápisu opakovat. Například u otázek typu *multiple-choice* je třeba pro každou nabízenou odpověď znovu vytvořit přepínač a v něm vykreslit zaškrtávací kolečko pro oba stavy (zaškrtnuto a nezaškrtnuto). Šablony byly navrženy, aby takovéto opakující se zápisy nebylo nezbytné v otázce provádět více než jednou. Pro nejvhodnější způsob jejich použití v QML byly brány v potaz formáty používané v jazycích SVG, XAML a XSLT.

SVG používá pro znovupoužitelné šablony seskupování pomocí elementu `<g>` (rozlišené identifikátorem v atributu `id`), které je pro samotné vykreslení voláno pomocí `<use>` s týmž `id`. Při tomto volání může být do kresby promítnut jiný styl, definovaný v atributu `style` (na bázi CSS[27]). Definovat další úpravy pro vykreslení uvnitř skupiny (šablony) při jejím volání nelze. Je však možné do sebe skupiny vícenásobně vnořovat. Kromě skupin SVG podporuje také `<symbol>`, který funguje naprosto totožně jako skupina, pouze umožňuje nastavit způsob chování měřítka při vykreslování na plochu rozdílné velikosti. [38 str. 53]

[27] CSS – Cascading Style Sheets

XAML, podobně jako formát aspx v ASP.NET [42 str. 699], umožňuje vytvářet uživatelské komponenty (UserControls). Ty mohou obsahovat vlastní kód, včetně definice vlastností, jež se při jejich použití mohou nastavovat přímo přes XAML. Tyto komponenty přitom podporují dědičnost i polymorfizmus objektově orientovaného programování. Takovýto způsob je velmi univerzální, vyžaduje však delší zápis kódu. [36]

XSLT umožňuje definovat šablony `<xsl:template>` v kořenovém elementu, přičemž rozlišeny jsou atributem `name`. Jejich volání kdekoli v dokumentu se pak provádí elementem `<xsl:call-template>`, kde je atribut `name` nastaven na název šablony, která je volána. V tom místě je do dokumentu vložen veškerý obsah příslušné šablony. XSLT podporuje i parametry pro volání šablony, včetně povinné definice jejich výchozích hodnot, nejsou-li při volání uvedeny. K těm je pak v rámci šablony přistupováno jako k proměnným. [43 str. 56]

QML se v tomto případě inspirovalo nejvíce způsobem použitým v XSLT. V kořenovém elementu otázky může být element `<templates>` a v něm nadefinované jednotlivé šablony `<template>`, rozlišené identifikátorem v atributu `id`. Tyto šablony lze v dalším QML zápisu použít (volat) stejným elementem se stejným `id`, včetně volání šablony z jiné šablon.

Šabloně mohou být předány i parametry. Jejich výčet však není explicitně uveden v hlavním elementu šablony, ale její vnitřek může na libovolném místě obsahovat zvláštní textové řetězce, které budou při zpracování šablony nahrazeny hodnotami předanými pro příslušné parametry. Proměnné jsou standardně označeny z obou stran znakem `%` (např. `%jmeno%`), který však lze změnit v atributu šablony `paramBounds`. Hodnoty se šabloně předávají v hodnotových podelementech v `<template>`, který šablonu volá. Zde se již znak uvozující proměnné neuvádí a elementy mají přímo název těchto proměnných (např. `<jmeno>Adam</jmeno>`).

4.1.5 Transformace pomocí XSLT [◎ II.2.B]

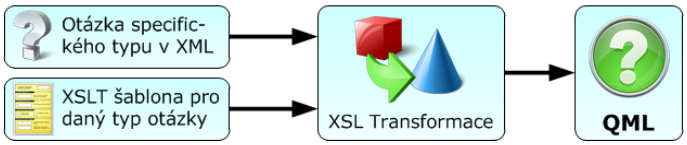
Způsob zápisu QML umožňuje vytvářet graficky efektní a přitom funkčně rozsáhlé otázky, kde jejich autor může plně realizovat svůj tvůrčí potenciál. V mnoha případech ovšem může být tato univerzálnost i na obtíž. Zapsat například klasickou otázku typu „výběr správné odpovědi" [44 str. 30], je v QML celkem zdlouhavé a pracné (nastavení fontů, rozmístění jednotlivých odpovědí, vytvoření několika stejných přepínačů apod.), byť s využitím šablon (viz kap. 4.1.4). Přitom lze předpokládat, že stejný typ otázky může být používán celkem často a bylo by tak nezbytné psát (kopírovat) stále tentýž kód do mnoha otázek. Tento problém se samozřejmě týká i jiných druhů klasických otázek nejen didaktického testu.

Problém by mohl částečně vyřešit WYSIWYG editor otázek, který by tento opakující se kód generoval sám, avšak i v něm by bylo celkem pracné jednotlivé grafické prvky znovu vytvářet a rozmísťovat. Mimo to by se v databázi otázek ukládal stále týž kód a zbytečně tak zvětšoval její velikost, přenosnost a čitelnost (z uživatelského hlediska).

Jinou možností by bylo vytvořit specifické QML pro jednotlivé typy otázek a ty zpracovávat odlišným postupem. Tím by však byla striktně vymezena množina použitelných typů otázek. Navíc by bylo velmi pracné vytvářet obslužný kód pro každý takový typ.

Jak tedy umožnit definování jednodušší struktury QML pro často se opakující typy otázek tak, aby kvůli každé z nich nemusel být upravován aplikační kód? Je-li třeba převést jedno XML (stručněji zapsanou otázku) na jiné XML (QML), je logickou volbou použití XSLT.

XSLT (eXtensible Stylesheet Language Transformations) je jazyk založený na XML pro transformování XML dokumentů do jiných textových formátů. Nejčastěji se XSLT používá pro transformaci jednoho typu XML dokumentu do jiného typu XML dokumentu, což pomáhá zmírnit nekompatibilitu. [45 str. 85] Nejběžnější transformací je obvykle převedení souboru dat strukturovaných v XML na formát HTML pro přímé zobrazení [43]. Aktuální verzí XSLT je 2.0 [w83].

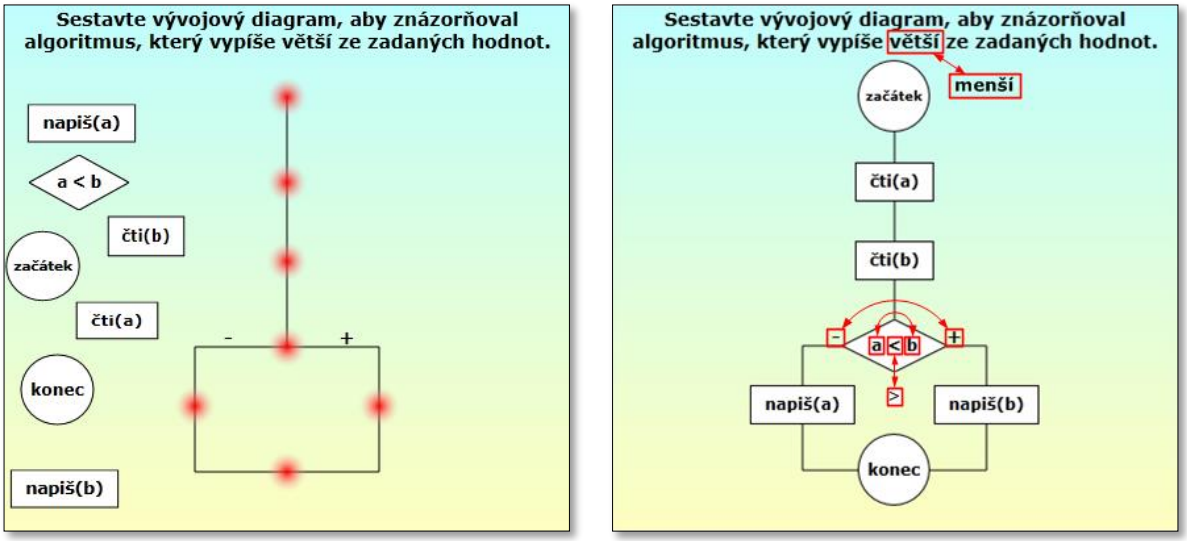


Obr. 10 – Schéma transformace zjednodušeného XML zápisu otázky do QML pomocí XSLT

Možnost zařadit XSL Transformaci do procesu generování otázky vyřešila řadu nesnází. Díky ní lze pro jakýkoli opakující se typ otázky vytvořit XSLT šablonu, pomocí které bude otázka před svým načtením převedena z individuálního jednoduchého XML do libovolně graficky bohatého QML (viz Obr. 10). Navíc je veškerý složitý kód pro definici jednotlivých prvků grafiky uveden pouze jedenkrát a to v této XSLT šabloně. Psaní otázky pak již může být velmi snadné a z několika málo triviálních XML elementů se sama vygeneruje propracovaná, plně funkční otázka. V případě potřeby úpravy některé části tohoto typu otázky přitom stačí upravit pouze tuto šablonu a není třeba úpravu provádět u každé takové otázky zvlášť.

4.1.6 Náhodné prvky [◎ III.3.A]

QML podporuje elementy pro generování náhodných čísel, znaků, slov i barev. Určité části kódu mohou také být podmíněny pro jeho použití při testu náhodným výběrem z několika variant. Kód může, podobně jako XSLT, používat podmínky, větvení, cykly a interní šablony. Dokáže též vyhodnocovat matematické i logické výrazy. Jedna otázka tak může mít obrovské množství variant. [ap-4]



Obr. 11 – Ukázka otázky s cíli a položkami ve stavu výchozím a vyřešeném, s vyznačením částí, které se v zadání náhodně mění a jakým způsobem

Obr. 11 např. ukazuje otázku, kde se mění celkem 4 části, každá dvěma možnými způsoby, což dává celkem $4^2 = 16$ variant téhož příkladu. Přitom není bodový rozdíl, pokud jsou zaměněny kroky `čti(a)` a `čti(b)`, naopak největší váhu na výsledné hodnocení otázky má správné umístění příkazů `napiš(a)` a `napiš(b)`, neboť ty jediné musí být v různých verzích jinak rozmístěny (viz str. 55).

Pro správné vyřešení této otázky již nestačí si zapamatovat, že `napiš(a)` je vždy vlevo a `napiš(b)` vpravo, nebo rafinovaněji, že `napiš(a)` je na téže straně, jako je zapsán znak `a` v podmínce, ale je třeba si vždy podrobně prohlédnou celé zadání a výsledek určit až na jeho základě, dle naučených principů fungování diagramu.

Verze

Systém verzí umožňuje náhodný výběr z několika verzí zadání otázky. Je určen pro případy, kdy se náhodná volba proplétá celou otázkou na různých místech jejího zápisu. Při této možnosti je v hlavním elementu otázky uveden element `<versions>`, v němž jsou vyjmenovány jednotlivé verze jednoznačně rozlišené identifikátorem v atributu `id`. Libovolný element v celém zápisu otázky pak může obsahovat atribut `version`, jehož hodnota je rovna `id` jedné (nebo i více, odděleno čárkami) z těchto verzí.

Při generování otázky je pak nejprve ze seznamu verzí jedna z nich náhodně vybrána. Při následném zpracování otázky je u každého elementu kontrolována existence atributu `version`, a pokud existuje, je element použit pouze v případě, že je jeho hodnota rovna nebo obsahuje zvolenou verzi. V opačném případě je celý tento element včetně jeho případných podelementů zcela ze zpracování otázky vyřazen.

```
<question width="200" height="80">
  <versions>
    <version id="1" />
    <version id="2" />
  </versions>
  <graphics thickness="1" color="000000" background="EEEEEE">
    <rectangle x="S" y="S" version="1" />
    <ellipse x="S" y="S" version="2" />
    <text x="C" y="M" version="1,2">Co je toto za obrazec?</text>
  </graphics>
</question>
```

Kód 4 – Ukázka zápisu míchání pomocí verzí

Kód 4 demonstruje způsob zápisu při použití verzí. V něm je náhodně zvoleno mezi 1 a 2, přičemž pro 1 se vykreslí obdélník, pro 2 elipsa. Text je vypsán v obou případech, a jelikož více možných variant není, nemusel by se u něho atribut `version` vůbec vyskytovat. Aby využití verzí bylo smysluplné, měl by na její volbě záviset i nějaký jiný element v jiné části. Například by bylo vhodné použít jej v definici hodnocení (`scores`) pro bodování, zvolil-li zkoušený na základě vypsaného textu správný obrazec. Podobně omezení hodnocení obsahuje např. Kód 13 (str. 56).

Proměnné

Výstupní hodnoty níže vysvětlených prvků `<rnd>` (náhodná hodnota), `<exp>` (výraz) a `<switcher>` (větvení) lze kromě přímého dosazení používat opakovaně jako proměnné. Podmínkou je přítomnost atributu `id`, který musí mít jedinečnou hodnotu v celém dokumentu (např. `id="x"`). Takovouto proměnnou je pak možné opětovně používat znovu pro dosazení, jako element, který již tentokrát bude obsahovat pouze atribut `id` (např. `<rnd id="x" />`).

Druhou možností je hodnotu takové proměnné dosadit v rámci atributu jiného elementu. Zde je použita stejná syntaxe, jako v jazyce PHP, při dosazování hodnot do textových řetězců, a sice mezi složené závorky s prefixem znaku dolar (např. `value="{$x}"` nebo `value="Číslo je {$x}."`, také viz Kód 5).

Hodnota proměnné se při svém prvním nastavování zároveň i dosadí do textu. Pokud se tak při této její deklaraci stát nemá, lze do elementu přidat atribut `mem="1"`. Tím se výstupní hodnota elementu pouze uloží do paměti pod příslušným `id`, ale k jejímu vypsání nedojde.

Náhodná hodnota

QML umožňuje na libovolném místě použít element `<rnd>` (random), který generuje náhodné hodnoty. Je možné jej použít jak pro náhodné číslo z daného rozsahu tak i pro výběr jednoho z možných textových řetězců či znaků.

```
<graphics>
  <text>
    <rnd id="otazka" type="string" values="Kolik je,Určete,Vypočtěte" />·
    <rnd id="promenna" type="char" min="a" max="z" except="abc" />, když
    <rnd id="promenna" /> =
    <rnd id="x" type="int" min="1" max="90" minLen="2" padChar="0" />·
    <rnd id="znamenko" type="char" values="+-*" />·
    <rnd id="y" type="int" min="{$x}" max="99" except="10,20,50" />
  </text>
</graphics>
```

Kód 5 – Ukázka použití náhodné hodnoty typu textu, číslo a znak

Kód 5[28] ukazuje použití všech těchto typů náhodných hodnot. Nejprve je zvoleno náhodné znění otázky (typ `string`). Následuje výběr náhodného znaku pro proměnnou v rozsahu `a` až `z` (dle ASCII tabulky), přičemž nesmí být použita písmena `a`, `b` ani `c`. Pokračuje text otázky, za nímž je znovu použit znak téže proměnné, tentokrát již pro rovnici. První člen výrazu, interně označený jako `x` je náhodně vybrán z rozsahu 1 až 90 (včetně) a v případě potřeby zleva doplněn nulou, aby vypsaná hodnota měla vždy dva znaky. Poté je vypsáno znaménko (`+`, `-` nebo `*`). Hodnota pro druhý člen výrazu `y` je vybrána z rozsahu `x` až 99, přičemž `x` je hodnota dříve určená pro prvního člena výrazu (platí tedy `x <= y`), a zároveň pro `y` nesmí být zvoleny hodnoty 10, 20 ani 50.

Výsledek pak může vypadat např. takto: „`Vypočtěte h, když h = 07 + 51.`“[29]

[28] znak střední tečky „·“ zapsaný v kódu, je ve výstupu zobrazen jako mezera a požívá se pro oddělení dvou po sobě jdoucích elementů, neboť jsou-li mezi nimi pouze mezery, budou při parsování XML vynechány
[29] pro text na jednom řádku, musí být v QML zapsán též do jednoho řádku, což formát tohoto dokumentu neumožňuje

Náhodný výběr

Nestačí-li použití náhodných hodnot, je možné náhodně vybírat i z celých částí QML. Za tímto účelem je zde element `<mix>`. Jím lze „obalit" v podstatě jakoukoli jinou sekci, přidat její alternativy a z nich některé náhodně vybrat, případně je jen promíchat.

```
<graphics>
  <stack>
    <text>Jaká je barva oblohy při jasném počasí?</text>
    <mix id="q1" mix="1" count="2">
      <item id="1" required="1"><text>modrá</text></item>
      <item id="2"><text>zelená</text></item>
      <item id="3"><text>červená</text></item>
    </mix>
  </stack>
</graphics>
```

Kód 6 – Ukázka použití náhodného výběru

Kód 6 ukazuje jednu z možností, jak může být využit element `<mix>`, a to pro zamíchání pořadí podlementů `<text>` (`mix="1"`) a výběr pouze 2 ze 3 alternativ (`count="2"`), přičemž správná odpověď bude ve výběru zařazena vždy (`required="1"`).

Element `<mix>` lze ovšem použít i opačně, tedy pro vytvoření více položek, než jich je uvedeno. Stačí *mixu* přidat atribut `build="1"` a nastavit atribut `count` na požadovaný počet. Pak bude obsah generován tak, že se v cyklu s počtem opakování rovným `count` vždy náhodně vybere jedna z položek `<item>` a její obsah bude přidán do kódu otázky. To má samozřejmě smysl pouze s dalšími prvky pro náhodné generování, které budou obsahem jednotlivých položek (např. viz Kód 5, který by mohl při jednom zápisu zadání vygenerovat celou sadu příkladů).

```
<graphics>
  <stack>
    <text>Vypočtěte následujících deset příkladů</text>
    <mix id="q1" mix="0" count="10" build="1" buildIndexCode="%i%">
      <item id="1">
        <text>%i%) ·
          <rnd type="int" min="0" max="10" /> *
          <rnd type="int" min="0" max="10" />
        </text>
      </item>
    </mix>
  </stack>
</graphics>
```

Kód 7 – Ukázka použití elementu `<mix build="1">`

Při zapnuté funkci `build` může být též v elementu `<mix>` nastaven atribut `buildIndexCode` na libovolný text, který pokud bude použit v položkách mixu, bude nahrazen číslem kroku, který je aktuálně pro cyklus budování používán (viz např. Kód 7 příklady čísluje „1)" – „10)").

Větvení

Aby bylo možné lépe využít možnosti náhodného generování v širších souvislostech, bez nutnosti pro každou možnou kombinaci vytvářet zvláštní verzi (`version`), je možné použít element `<switcher>` pro vícenásobné větvení. Jeho funkčnost byla inspirována příkazem `switch` z jazyka C# [46 str. 55], přičemž v jazycích založených na XML podobného větvení využívá např. XSLT (`<xsl:choose>`) [43 str. 167].

```
<graphics>
  <text>Číslo <rnd id="x" type="int" min="1" max="999" /> je
    <switcher value="{$x}">
      <case operator="ends" value="0">dělitelné deseti</case>
      <case operator="ends" values="2,4,6,8">sudé</case>
      <default>liché</default>
    </switcher>.
  </text>
</graphics>
```

Kód 8 – Ukázka použití větvení

Kód 8 ukazuje příklad použití větvení na příkladu vypsání náhodného čísla a uvedení, je-li dělitelné deseti, a pokud ne, pak je-li sudé či liché.

V elementu `<switcher>` je v atributu `value` uvedena hodnota (s možností vložení proměnné z náhodně generované hodnoty), která se postupně porovnává s hodnotami a za podmínek nastavených v jednotlivých podelementech `<case>`. Hodnoty se porovnávají jako textový řetězec způsobem definovaným v atributu `operator`. Aktuálně jsou podporovány operátory `=` (výchozí volba), `!=`, `begins`, `ends` a `contains`. Hodnota pro porovnávání je buď jedna v atributu `value`, nebo více v atributu `values`, kde jsou odděleny čárkami.

Porovnávání probíhá postupně, jakmile je nalezena shoda, je do QML vložena hodnota vyhovujícího `<case>` a další porovnávání neprobíhá. Hodnota elementu `<default>` je použita, pokud nevyhovuje žádný `<case>`. Není-li `<default>` definován a žádný `<case>` nevyhovuje, je výsledkem větvení prázdný řetězec.

Element `<switcher>` lze použít i ve zkrácené verzi nepárového elementu, ovšem pouze pro jednoduché textové hodnoty. Vstupní hodnota je opět předána atributu `value`, hodnoty, s nimiž se tato porovnává, jsou pak za sebou vypsány v atributu `cases`, odděleny čárkou, a jednotlivé výstupní hodnoty jsou uvedeny v atributu `values`, také odděleny čárkou (viz Kód 9). Oddělovací znak lze změnit přes nepovinný atribut `separator`. I zde může být ještě atribut `default`, jež by určoval výstupní hodnotu v případě, že by ta vstupní neodpovídala žádné z porovnávaných.

```
<graphics>
  <text>Slovíčko <rnd id="x" type="string" values="red,green,blue" /> označuje
    <switcher value="{$x}" cases="red,green,blue"
              values="červenou,zelenou,modrou" /> barvu.
  </text>
</graphics>
```

Kód 9 – Ukázka použití zjednodušené verze větvení

Výrazy

Výrazy umožňují výpočty s hodnotami jiných proměnných (např. náhodné hodnoty z `<rnd>`). Pro jejich použití je k dispozici element `<expr>` (expression – výraz), v jehož atributu `value` se uvede matematický výraz. Ten je vyhodnocen a výsledek dosazen do kódu jako text (viz Kód 10).

```
<graphics>
  <text>
    <rnd id="x" type="int" min="1" max="9" /> +
    <rnd id="y" type="int" min="1" max="9" /> =
    <expr value="{$x}+{$y}" />
  </text>
</graphics>
```

Kód 10 – Ukázka použití výrazu

Ve výrazech lze používat i matematické a logické funkce, jako např. Sin, Cos, Log, Abs, Round, Floor, Sign, Sqrt, If apod.

4.1.7 Automatické vyhodnocování [◎ II.3]

Automatické vyhodnocování správnosti odpovědí na testové otázky je nezbytnou součástí celého systému. V této části bude proto podrobněji vysvětlen princip, na kterém toto hodnocení funguje a možnosti jeho definice.

Váhy [◎ II.3.A]

Úspěšnost každé otázky je hodnocena procentuálně, resp. v intervalu ⟨0, 1⟩. V otázce však může být mnoho různých možností, jak toto skóre zvýšit či snížit a bylo by velmi složité, v některých případech dokonce nemožné, definovat jednotlivým případům takový podíl na skóre, aby v součtu všech nejlépe hodnocených voleb dával právě 1 (100%). Z tohoto důvodu je uvnitř každé otázky stanoven **bodový subsystém**, kdy se jednotlivým případům přidělují určité bodové váhy. Jejich rozsah a stupnice může být v každé otázce zvolena libovolně. Systém totiž při generování otázky sám vypočítá maximální možný bodový zisk a jím dělí počet bodů získaných, čímž je výsledné skóre převedeno zpět do jednotného rozsahu ⟨0, 1⟩.

Mnohem snáze se definuje, že např. správné řešení podotázky a) je 3x složitější nežli podotázky b). Pokud by v otázce byly pouze tyto dvě, maximální bodový zisk by byl 3+1=4, takže část a) by zahrnovala 3/4 = 0.75 (75%) a část b) 1/4 = 0.25 (25%) podíl na celkovém skóre otázky.

Váhy některých řešení mohou být definovány i **zápornou hodnotou**. Ty však nejsou do výpočtu celkového maximálního skóre zahrnuty a vyjde-li díky nim bodový výsledek záporný, je skóre vyhodnoceno jako 0 (0%). Záporné hodnoty mohou v některých případech posloužit jako penalizace za „zvlášť špatné řešení", nebo vykrýt statistické minimální skóre, není-li pro hodnocení žádoucí (tzv. korekce na hádání).

Přepínače

Hodnocení přepínačů lze ve výchozí formě realizovat tak, že je pro každý stav každého z přepínačů definován libovolný počet bodů. Kód 11 ukazuje způsob zápisu bodového hodnocení jednotlivých stavů přepínačů. Atribut `id` je identifikátorem přepínače, `value` je identifikátorem jeho

stavu a `score` určuje počet bodů, které tato kombinace přináší. Výchozí bodová hodnota neuvedených kombinací je 0.

```
<scores>
  <switches>
    <switch id="1" value="1" score="1" />
    <switch id="1" value="2" score="-0.5" />
    <switch id="1" value="3" score="-0.5" />
    <switch id="2" value="yes" score="-1" />
    <switch id="2" value="no" score="1" />
  </switches>
</scores>
```

Kód 11 – Zápis bodového ohodnocení jednotlivých stavů přepínačů

Kód 11 kromě definice pozitivních bodových hodnot za správné řešení nastavuje i záporné penalizační body za řešení nesprávná a to přesně v poměru pro „korekci na hádání". Ta se používá, především pokud má být v celkovém výsledku vyrušena náhodná složka, tj. body za odpovědi správně zvolené jen náhodou.

Pokud zkoušený v části otázky volí mezi dvěma až třemi nabízenými odpověďmi, doporučuje se pro správnou volbu korekčního hodnocení chybných odpovědí Vzorec 1., který vychází z předpokladu, že zkoušený, který odpověď hádá, se dopouští častěji chyb než ten, který úlohy skutečně řeší a odpovídá jedině tehdy, když odpověď zná. [44 str. 33]

$$s_0 = s_n - \frac{n}{y-1}$$

Vzorec 1 – Korekce dosažených bodových výsledků na hádání [44]

Proměnná s_0 *ve vzorci je tzv. opravené skóre,* s_n *neopravené skóre,* n *počet nesprávných odpovědí zkoušeného a* y *počet možných voleb podotázky, přičemž možnost neřešení se nezapočítává.* [44]

Vzorec 1 počítá s hodnocením 1 bod za správnou odpověď. Kolik bodů je pak optimální odečíst za nesprávnou odpověď lze určit přeformulováním a doplněním na Vzorec 2.

$$s_k = -\frac{b}{y-1}$$

Vzorec 2 – Výpočet korekčního skóre (s_k) pro chybnou odpověď

Výsledná proměnná s_k je korekčním skóre, b je počet bodů, které maximálně tato podotázka přináší a y, stejně jako v předchozím případě, počet možných variant řešení, mimo výchozího.

Například u dichotomických[30] podotázek jsou nabízeny dvě možnosti ($y = 2$), takže korekční skóre vychází -1 bod, při výběru ze tří možností by pak tato hodnota byla -0,5 bodu. Pokud by tedy otázka obsahovala 10 podotázek typu ano/ne po jednom bodu za správnou odpověď, bez korekce na hádání by v průměru mělo zcela náhodné řešení znamenat 5-ti bodovou úspěšnost. Bude-li však za chybnou volbu penalizace -1 bod, pak by teoreticky kladný výsledek měly přinést až správné odpovědi nad rámec náhody.

[30] dichotomická úloha – otázka, na kterou jsou zkoušenému předkládány právě dvě alternativy odpovědi s tím, že má zvolit (označit) tu správnou, přičemž má i možnost úlohu neřešit vůbec [44]

Optimální penalizační záporné hodnoty bodů za chybnou odpověď pro jednotlivé hodnocené přepínače v otázce určuje Vzorec 2. Práci s výpočtem a zápisem těchto hodnot pro nesprávné varianty lze usnadnit pomocí vestavěné funkce hodnotícího algoritmu tak, že se zapíše pouze varianta se správným řešením, body, jež její volba přináší a k tomu se přidá atribut `otherStates="penalize"`. Tím je určeno, že volba kteréhokoli z ostatních stavů mimo zde ohodnoceného a výchozího (`stateOff`) bude hodnocena zápornými body, které určuje Vzorec 2. Kód 11 se dá tedy zapsat i jako Kód 12.

```
<scores>
  <switches>
    <switch id="1" value="1" score="1" otherStates="penalize" />
    <switch id="2" value="no" score="1" otherStates="penalize"  />
  </switches>
</scores>
```

Kód 12 – Zkrácený zápis bodového hodnocení přepínačů se standardní penalizací (korekce na hádání)

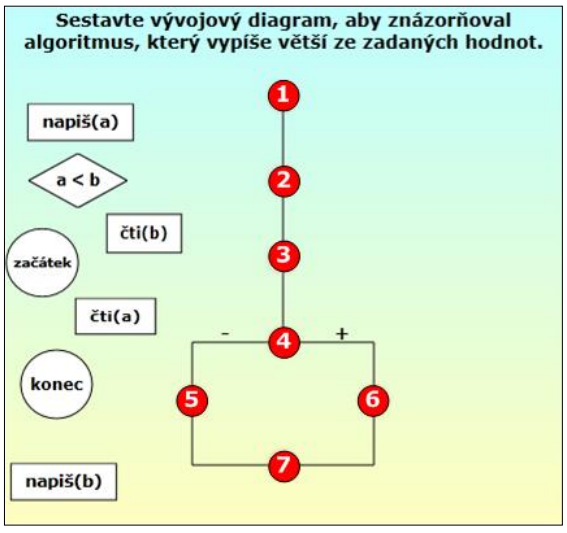
Celkový maximálně dosažitelný počet bodů je pro tento typ otázky definován jako součet maximálního skóre pro každý z přepínačů (viz Vzorec 3). Jím je pak při finálním přepočtu na procentní hodnocení otázky dělen bodový zisk.

$$\sum_{i}^{\{přepínače\}} \max\{\text{stavy přepínače}_i\}$$

Vzorec 3 – Výpočet maximálního možného počtu bodů pro část otázky typu přepínače

Položky a cíle

Efektním typem otázek je přetahování položek na vyznačené cíle metodou drag&drop. Zde se definují body za jednotlivé volby, resp. lze stanovit počet bodů pro každou kombinaci cíle a položky, pokud je tato do něho umístěna.



Obr. 12 – Zadání otázky s označením cílů

Položka	Cíl	1	2	3	4	5	6	7
1	začátek	1			-1			
2	čti(a)		2	2				
3	čti(b)		2	2				
4	a < b				1			
5	napiš(a)					6		
6	napiš(b)						6	
7	konec							1
	Maximum	1	2	2	1	6	6	1

Tab. 3 – Bodové hodnocení umístění jednotlivých položek (řádky) do jednotlivých cílů (sloupce)

Obr. 12 zobrazuje příklad tohoto druhu otázky s číselným označením jednotlivých cílů a Tab. 3 bodové hodnoty pro jednotlivé kombinace umístění položky (řádky tabulky) do určitého cíle (sloupce tabulky). Prázdné buňky tabulky jsou brány jako 0.

```
<scores>
  <targets>
    <target id="1" value="1" score="1" />
    <target id="2" value="2" score="2" />
    <target id="2" value="3" score="2" />
    <target id="3" value="2" score="2" />
    <target id="3" value="3" score="2" />
    <target id="4" value="4" score="1" />
    <target id="4" value="1" score="-1" />
    <target id="5" value="5" score="6" version="1,4,6,7,10,11,13,16" />
    <target id="5" value="6" score="6" version="2,3,5,8,9,12,14,15" />
    <target id="6" value="5" score="6" version="2,3,5,8,9,12,14,15" />
    <target id="6" value="6" score="6" version="1,4,6,7,10,11,13,16" />
    <target id="7" value="7" score="1" />
  </targets>
</scores>
```

Kód 13 - Kód zápisu bodového ohodnocení jednotlivých variant umístění položek do cílů (dle Tab. 3)

Kód 13 ukazuje způsob zápisu bodového hodnocení jednotlivých kombinací cíl-položka. Atribut `id` je identifikátorem cíle, `value` je identifikátorem položky a `score` určuje počet bodů, které tato kombinace přináší. Jelikož 0 je výchozí hodnota, není třeba zde vypisovat kombinace přinášející právě tento počet bodů. Stejný zápis lze provést i ve zkrácené verzi do atributu `data` pomocí min-jazyka (viz kap. 4.12.6, str. 156). Atribut `version` uvádí výčet identifikátorů verzí (viz str. 49), pouze při kterých je tato kombinace (element `target`) platná.

Celkový maximální počet bodů je pro tento typ otázky definován jako součet maximálního skóre pro každý z cílů (viz Vzorec 4).

$$\sum_{i}^{\{cíle\}} \max \{\text{položky v cíli}_i\}$$

Vzorec 4 – Výpočet maximálního možného počtu bodů pro část otázky typu cíle a položky

Pokud otázka obsahuje cíle a položky i část s přepínači, je celkový maximální bodový zisk otázky určen součtem maxim obou těchto částí. To platí i pro další druhy podotázek. Přírůstek na procentuálním výsledku otázky je přitom vždy určen počtem bodů za zvolenou část řešení dělený tímto celkovým maximem.

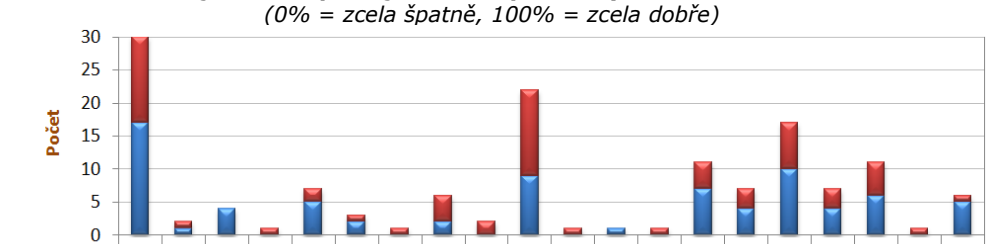
Řazení

Objektivní hodnocení uspořádací podúlohy, ve kterých mají být seřazeny položky podle určitého klíče, může být v některých případech velmi sporné, což ostatně dokládá i následující průzkum.

V rámci výše uvedeného dotazníkového šetření (viz kap. 3.2, str. 27) byla respondentům položena i následující otázka (viz Graf 14). Pro uvedený příklad konkrétně existuje několik interpretací, s nimiž může být výsledek ohodnocen přes celou hodnotící škálu, tj. 0%-100%. V dotazníku tedy byli respondenti dotázáni, jak by příklad ohodnotili oni.

Kolika procenty byste ohodnotili řešení otázky, v níž měly být seřazeny zpřeházené kartičky s čísly 1-6 od nejmenšího po největší a odpověď byla tato: "6-1-2-3-4-5"?

(0% = zcela špatně, 100% = zcela dobře)



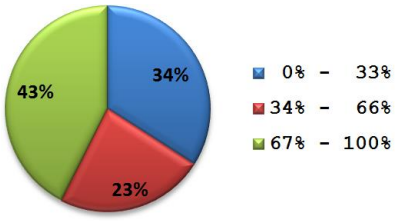
Hodnocení [%]	0	8	10	15	20	25	33	40	45	50	55	60	66	70	75	80	85	90	95	100
Studenti	13	1		1	2	1	1	4	2	13	1		1	4	3	7	3	5	1	1
Vyučující	17	1	4		5	2		2		9		1		7	4	10	4	6		5

Graf 14 – Četnosti hodnocení příkladu s řazením (resp.: 77 vyučujících a 64 studentů)

Graf 14 ukazuje kolik vyučujících a studentů uvedlo, jaké procentní hodnocení příkladu považuje za spravedlivé. Vzhledem k tomu, že hodnoty obou skupin (vyučující a studenti) zjevně nemají normální rozdělení, což ostatně potvrdil i Kolmogorov-Smirnovův test normality, místo studentova t-testu bylo na porovnání dvou skupin použito pouze Mann-Whitneyho testu, který na 5% hladině významnosti prokázal shodu v hodnocení obou skupin. Můžeme tedy s daty těchto skupin pracovat jako s jednoskupinovým datovým celkem.

Celkový průměr ze všech uvedených hodnocení je 48,69%, přičemž za vyučující je to 48,81% a za studenty 48,55%, což je velmi podobné. Směrodatná odchylka pak dosahuje poměrně vysoké hodnoty 34,1.

Kategorizaci četnosti hodnocení do tří skupin rovnoměrně pokrývajících celou stupnici ukazuje Graf 15. Je vidět, že největší část respondentů (43%) by se klonila spíše k hodnocení v horní třetině stupnice, avšak i ostatní dvě skupiny mají značné zastoupení.



Graf 15 – Kategorizovaná četnost hodnocení příkladu s řazením (141 resp.)

Jak je z tohoto průzkumu patrné, neexistuje jednotný názor, jak tuto úlohu hodnotit. Proto byla v testech implementována podpora hned několika různých typů automatického hodnocení těchto úloh. Tvůrce otázky si může zvolit libovolný z nich tak, aby nejlépe vyhovoval jeho záměrům. Typ hodnocení je určen atributem `type` příslušného elementu `listBox`. K dispozici jsou tyto alternativy:

- `none` – řazení nehodnotit (netřeba jej do `scores` vůbec uvádět),
- `all` – všechno nebo nic,
- `individual` – individuální hodnocení pozic (výchozí volba),
- `distance` – hodnocení vzdáleností,
- `pairs` – párování.

Všechno nebo nic

Nejjednodušším (avšak málo citlivým) způsobem hodnocení je všechny body přidělit pouze za zcela správné uspořádání a za všechna ostatní řešení není bodový zisk žádný. [40 str. 51]

```
<scores>
  <listBoxes>
    <listBox id="one" score="3" type="all" order="o1,o2,o3,o4,o5" />
    <listBox id="grp" score="2" type="all" order="g1,g2,(g3,g4),g5" />
  </listBoxes>
</scores>
```

Kód 14 – Zápis bodového hodnocení pro uspořádací úlohu typu „všechno nebo nic"

Zápis této varianty ukazuje Kód 14. Zde je v řadící jednotce s `id="one"`, hodnocené třemi body, uveden seznam identifikátorů jednotlivých položek oddělených čárkou ve správném pořadí v atributu `order`. Ve druhé jednotce s `id="grp"`, hodnocené dvěma body, je uvedená varianta, kdy lze položku `g3` a `g4` zaměnit, aniž by řešení bylo klasifikováno jako chybné. Takto zaměnitelné identifikátory sousedících položek stačí uzavřít do závorek a dát tak najevo, že na vzájemném pořadí mezi nimi nezávisí. Pro povolení záměny pořadí položek, které spolu nesousedí, je třeba možné kombinace podrobněji rozepsat do jednotlivých elementů a uvést všechny přípustné kombinace (viz Kód 15, jen u jednotlivých elementů `item` není již uvedena hodnota `score`).

Individuální hodnocení pozic

Individuální hodnocení pozic, podobně jako je tomu u položek a cílů, umožňuje přidělit pevné bodové hodnoty za přiřazení určité položky na specifickou pozici (viz Kód 15). Je tedy explicitně vyčísleno, jaký přínos, jaká položka, na jaké pozici přináší. Neuvedené kombinace opět znamenají 0.

```
<scores>
  <listBoxes>
    <listBox id="main" score="3" type="individual">
      <item id="i1" index="1" score="4" />
      <item id="i2" index="2" score="3" />
      <item id="i3" index="3" score="2" />
      <item id="i3" index="4" score="1" />
      <item id="i4" index="4" score="1" />
    </listBox>
  </listBoxes>
</scores>
```

Kód 15 – Zápis bodového hodnocení pro uspořádací úlohu typu „individuální hodnocení pozic"

Kód 15 ukazuje definici bodových hodnot pro řadící jednotku (`listBox`) s identifikátorem `main`, za který lze získat maximálně 3 body (`score`). Na tuto hodnotu se skládají 4 položky (`item`, identifikátor každé z nich je uveden v `id`), přičemž váhy jejich přispění k celkovému skóre při umístění na jednotlivé pozice (`index`) určuje hodnota v jejich atributu `score`. Při součtu maximálního možného skóre každé položky (4+3+2+1=10) je tedy určena hodnota, která v přepočtu odpovídá hodnotě skóre dané řadící jednotky (10 bodů z položek = 100% = 3 body celkem pro otázku).

Konkrétně tedy položka s `id="i1"` má `score="4"`, což je po přepočtu $\frac{4}{10} \cdot 3 = 1{,}6$ bodu a to při umístění na 1. pozici. Položka `i2` pak na 2. pozici zvyšuje bodový zisk otázky o 1,2 bodu, položka `i3` na třetí pozici o 0,8 (na 4. by to bylo pouze o 0,4) a položka `i4` na 4. pozici o 0,4. Pouze položka `i3` může být na 3. i 4. pozici a přinese kladný bodový zisk, avšak 3. pozice je 2x výhodnější než 4., nehledě na to, že tím zabere jediné bodované místo pro položku `i4`.

Pokud by hodnota `score` nebyla v elementu `listBox` definována, pak by každá položka přispívala k celkovým bodům otázky právě tou hodnotu, která je u ní pro příslušnou pozici definována.

Tento způsob tedy umožňuje definování všech možných stavů, které mohou nastat. To je ovšem při vyšším počtu položek dosti pracné a nepřehledné (`m!` kombinací, kde `m` je počet položek) a stanovení bodů stylem jedna položka pouze na jedné pozici přináší nějaké body, může poškodit zkoušeného. Například pokud by veškeré položky seřadil správně a pouze tu poslední umístil na první místo (viz Graf 14), pak by byly všechny posunuty o jedno místo dolů a tím pádem umístěny chybně. Ač je tedy otázka „téměř správně" bylo by za ni přiděleno 0 bodů a tedy i **0%**.

Hodnocení vzdáleností

Při hodnocení vzdáleností mají položky, stejně jako u typu „všechno nebo nic", pevně určeny jediné správné pozice, ovšem výsledek je počítán na základě jejich vzdálenosti od těchto pozic. Zápis hodnocení může být tedy podobný, jako ukazuje Kód 14 pro `listBox` s `id="one"` (záměna prvků není povolena).

Toto hodnocení vychází z Byčkovského metody [47], který navrhl následující algoritmus výpočtu skóre pro tento typ úlohy, vhodný především při řazení více než 5-ti položek. Výpočet skóre zde probíhá na základě relativní míry odchylek, tj. vzdálenosti položky od své správné pozice. Například pokud je správné umístění položky na 2. pozici a je umístěna na 5., je odchylka 5-2=3. Pro výpočet se nejprve určí maximální možná odchylka pro každou položku (d_{max}) při nejhorším možném celkovém řešení. Rozdíl jejich sumy a odchylek zkoušeným vytvořených (d), vydělený sumou d_{max} dává skóre relativní správnosti (x). Tento vztah vyjadřuje Vzorec 5.

$$x = \frac{\sum d_{max} - \sum d}{\sum d_{max}}$$

Vzorec 5 – Výpočet skóre pro úlohu řazení [47]

Tento druh výpočtu skóre počítá s tím, že každá položka má právě jednu správnou pozici, ovšem její nedodržení ještě neznamená nulový bodový zisk. Výsledná hodnota `x` je tedy z rozsahu $\langle 0, 1 \rangle$ a pokud je v elementu `listBox` uvedena hodnota skóre, je výsledný příspěvek k bodovému hodnocení otázky násobkem vypočteného `x` a této `score` hodnoty.

Pro příklad uveďme hodnoty z již zmíněného případu, kdy jsou všechny položky seřazeny správně, pouze poslední je zařazena na první pozici (viz Tab. 4).

Správné pořadí	Obrácené pořadí	d_{max}	Příklad řešení	d
1	6	5	6	5
2	5	3	1	1
3	4	1	2	1
4	3	1	3	1
5	2	3	4	1
6	1	5	5	1
∑		18		10

Tab. 4 – Ukázka postupu při výpočtu skóre relativní správnosti nesprávného řešení v uspořádací úloze [44]

V tomto případě by byl výpočet skóre následující: $x = \frac{18-10}{18} = 0{,}444$, tj. cca **44%**. Jako nejhorší možné řešení příkladu Tab. 4 používá seřazení v opačném pořadí než je správné. Ač by se pro jednotlivé položky daly najít pozice s vyšší odchylkou, než poskytuje toto řešení, je přesto právě toto řešení skutečně celkově nejhorší, co se sumy odchylek týče.

Jak ukazuje Tab. 4, sloupec d_{max} obsahuje jednotlivé odchylky nejhoršího řešení, jež začínají na hodnotě $m-1$ (m je počet položek) a pak dále klesají po dvou až do minimální nezáporné hodnoty (pro sudé m je to 1, pro liché 0) a následně se tyto hodnoty opakují v opačném pořadí. Suma maximálních odchylek ($\sum d_{max}$) je tedy rovna dvojnásobku sumy aritmetické posloupnosti, kde $a_1 = m \bmod 2 + 1$, $a_n = m-1$, $n = m \operatorname{div} 2$ a $d = 2$ (viz Vzorec 6).[31]

$$\sum d_{max} = 2s_n = 2\frac{n(a_1 + a_n)}{2} = n(a_1 + a_n)$$

Vzorec 6 – Výpočet sumy maximálních odchylek od správných pozic jako suma aritmetické posloupnosti

Pokud se bude rovnice řešit zvlášť pro sudé a pro liché m, lze jednotlivé rovnice zapsat následovně (viz Vzorec 7).

$$\text{S: } \frac{m}{2}(1 + m - 1) = \frac{m^2}{2}, \quad \text{L: } \frac{m-1}{2}(2 + m - 1) = \frac{m^2-1}{2}$$

Vzorec 7 – Výpočet sumy maximálních odchylek pro sudý (S) a pro lichý (L) počet položek (m)

Jediným rozdílem obou výsledků je u lichého -1 v čitateli. V případě univerzálního zápisu výsledku lze tedy výpočet sumy maximálních odchylek zapsat jako Vzorec 8.

$$\sum d_{max} = \frac{m^2 - (m \bmod 2)}{2}$$

Vzorec 8 – Výpočet sumy maximálních odchylek od správných pozic

[31] *operátory* `div` *a* `mod` *jsou definovány pro operace s celými čísly; v případě operace* `m div n` se jedná o výsledek *při celočíselném dělení čísel* `m`, `n` *(celá část jejich podílu) a v případě operace* `m mod n` o *zbytek po celočíselném dělení čísel* `m`, `n` [ap-16]

Párování

Další možností je hodnocení dvojic přiřazení. Například má-li správná odpověď znít „Jaro – Léto – Podzim – Zima", hodnotíme, zda žáci správně uspořádali léto za jaro, podzim za léto, zimu za podzim. V uvedeném příkladu by se tedy hodnotily 3 uspořádané dvojice. [40 str. 51]

Při tomto postupu se tedy výsledné skóre `x` vypočte tak, že se nejprve určí počet správně seřazených párů `p` a vydělí se celkovým počtem párů, který je o jednu menší než počet položek `m` (viz Vzorec 9).

$$x = \frac{p}{m-1}$$

Vzorec 9 – Výpočet skóre v uspořádací úloze při hodnocení párů

Tímto způsobem by skóre za řešení úlohy, jak ji popisuje Tab. 4, bylo $\frac{4}{6-1} = 0{,}8$ čili **80%**.

Textový vstup

Na krátké otevřené otázky zkoušený odpovídá zapsáním textu do editačních políček (viz str. 44). Hodnocení těchto odpovědí nemusí být pouze na základě přesné shody s určitou vzorovou odpovědí, ale lze definovat komplexní sadu podmínek, jež určí míru správnosti dané odpovědi.

```
<scores>
  <textBoxes>
    <textBox id="edit1" score="3.5">
      <answersRoot>
        <answer cs="0" trim="1" compare="equal">vzor odpovědi</answer>
      </answersRoot>
    </textBox>
  </textBoxes>
</scores>
```

Kód 16 – Ukázka zápisu hodnocení jednoduché textové odpovědi

Kód 16 ukazuje zápis hodnocení textové odpovědi v textovém vstupu (`textBox`) s identifikátorem `edit1`. Zadaná odpověď musí být shodná se vzorem uvedeným v hodnotě elementu (`vzor odpovědi`), přičemž nezáleží na velikosti písmen (to určuje atribut `cs` – case sensitive) a před porovnáváním budou z odpovědi umazány případné mezery z jejího začátku a konce (atribut `trim`). V tomto případě musí jít o přesnou shodu obou slov, což určuje atribut `compare` hodnotou `equal`. Následující seznam obsahuje kompletní výčet dostupných operátorů porovnání.

- `equal` (rovná se) – zadaná odpověď musí být shodná se vzorem
- `notEqual` (nerovná se) – zadaná odpověď nesmí být shodná se vzorem
- `begins` (začíná) – zadaná odpověď musí začínat vzorem udanými znaky
- `ends` (končí) – zadaná odpověď musí končit vzorem zadanými znaky
- `contains` (obsahuje) – zadaná odpověď musí obsahovat souvislou sadu znaků odpovídající vzoru
- `notContains` (neobsahuje) – zadaná odpověď nesmí obsahovat souvislou sadu znaků odpovídající vzoru
- `regEx` – zadaná odpověď musí odpovídat regulárnímu výrazu definovanému vzorem
- `similarity` – zadaná odpověď musí být podobná vzoru alespoň na danou hranici podobnosti

Při porovnání operátorem `regEx` je místo vzorové odpovědi uveden regulární výraz (viz [48]), jehož pravidlům musí odpovídat zadaná odpověď, aby byla uznána za správnou. Porovnání operátorem `similarity` umožňuje správnost odpovědi posuzovat na základě míry její podobnosti se vzorem, prostřednictvím některého z podporovaných algoritmů porovnání (viz kap. 4.4, str. 76).

```
<textBox id="edit2" score="2" recountScore="0">
  <answersRoot>
    <answer compare="similarity" simAlg="JW" simMin="0.75">logaritmus</answer>
  </answersRoot>
</textBox>
```

Kód 17 – Ukázka zápisu hodnocení textové odpovědi na základě podobnosti se vzorem

Kód 17 ukazuje zápis podmínky pro uznání textové odpovědi podle její podobnosti se slovem *logaritmus*. Shoda zkoušeným zadané odpovědi a vzoru je určena Jaro-Winklerovým algoritmem (zkratka `JW` v atributu `simAlg`, dle seznamu zkratek všech dostupných algoritmů, viz Tab. 5, str. 76, kap. 4.4) a tato musí být minimálně 75% (hranici určuje v intervalu $\langle 0, 1)$ atribut `simMin`), aby byla odpověď uznána za správnou. V tomto případě by tedy možné bodové hodnocení této odpovědi bylo 0 bodů při podobnosti v intervalu $\langle 0, 0.75)$ nebo 2 body při podobnosti v intervalu $\langle 0.75,\ 1\rangle$.

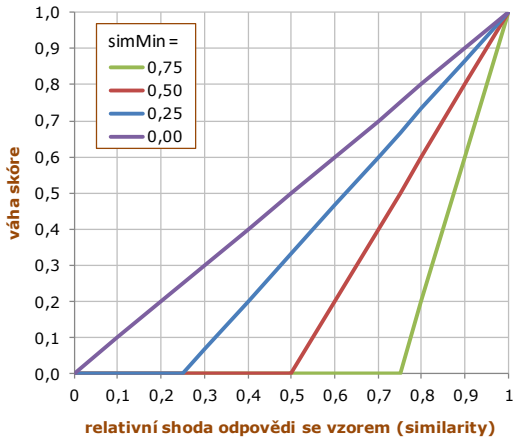
Je však také možné tento bodový zisk přímo odvodit z výsledné hodnoty porovnání, tzn. maximální možný bodový zisk odpovědi (v tomto případě 2 body) vynásobit mírou shody obou slov. Aby došlo k takovémuto přepočtu hodnot, je třeba v atributu `recountScore` v elementu `<textBox>` nastavit 1 (true) a přidat atribut `simScore` také s hodnotou 1 (true) do elementu `<answer>`. První úprava zajistí rozdělení celkového skóre mezi jednotlivé možné odpovědi (v tomto případě jednu) a druhá změna (`simScore="1"`) odvodí váhu této odpovědi z míry shody odpovědí určené zvoleným algoritmem. V tomto případě hodnota v atributu `simMin` udává hodnotu míry podobnosti, od které (níže) již bude váha odpovědi 0 (např. podobnost v intervalu $\langle 0,\ 0.75\rangle$ tak představuje nulovou váhu a tedy i nulový bodový zisk) a zároveň i hodnotu, od které (výše) je tato váha počítána až do 1 (např. $\langle 0.75, 1\rangle$) podle následujícího vztahu (viz Vzorec 10, odvozený z klasického směrnicového tvaru rovnice přímky v rovině[32] [49 str. 81]).

$$váha = \frac{similarity - 1}{1 - simMin} + 1$$

Vzorec 10 – Výpočet váhy skóre na základě podobnosti (**`similarity`**) odpovědi a vzoru a minimální hranice (**`simMin`**) pro částečného uznání odpovědi, platný v intervalu $\langle$**`simMin`**, 1)

Vliv míry podobnosti zadané a vzorové odpovědi na váhu pro výpočet skóre pro různé hodnoty `simMin` ukazuje následující graf (viz Graf 16). Z tohoto vztahu například vyplývá, že při hodnotě `simMin` 0,75 (v grafu zelená křivka) a podobnosti odpovědi 0,9 (90%) bude váha skóre 0,6, čili při 2b hodnotě odpovědi bude bodový zisk 0,6 · 2 = 1,2 bodů.

[32] y = kx + q



Graf 16 – Ukázka vztahu váhy skóre a míry podobnosti odpovědi se vzorem pro 4 různé hodnoty **`simMin`**

Vícenásobné hodnocení odpovědí

Element `<asnwersRoot>` může také obsahovat více než jen jednu podmínku pro hodnocení správnosti odpovědi. V takovém případě pro uznání odpovědi stačí, bude-li splněna alespoň jedna z nich. Je-li třeba, aby odpověď splňovala více podmínek současně, lze je seskupit do elementu `<and>` (a zároveň, tj. musí platit všechny přímo podřízené podelementy, viz Kód 18). Ten lze zároveň libovolně dále větvit a doplňovat i podelementy `<or>` (nebo, tj. stačí, když platí alespoň jeden z přímo podřízených podelementů).

```
<textBox id="gravitace" score="5">
  <answersRoot>
    <and>
      <answer compare="contains">Einstein</answer>
      <or>
        <answer compare="contains">Albert</answer>
        <answer compare="contains">A.</answer>
      </or>
    </and>
    <answer compare="contains">Newton</answer>
  </answersRoot>
</textBox>
```

Kód 18 – Ukázka zápisu hodnocení textové odpovědi s použitím seskupování podmínek

Jednotlivé podmínky první úrovně v elementu `<asnwersRoot>` také mohou každá zvlášť přímo ovlivňovat výši celkového skóre dané odpovědi. Tuto vlastnost určuje již zmíněný atribut `recountScore` v elementu `<textBox>`. V tomto případě je celkové skóre odpovědi (atribut `score` v elementu `<textBox>`) rozděleno na části určené počtem podmínek první úrovně a o započtení každé z nich rozhoduje splnění této podmínky. Díky tomu lze např. požadovat vypsání určitého výčtu slov oddělených čárkou v libovolném pořadí do jednoho editačního políčka, přičemž za každé správné se hodnocení zvýší. Je-li třeba některé z podmínek přidělit vyšší podíl na celkovém skóre, lze i jednotlivé podmínky doplnit atributem `score` a v něm tuto váhu nastavit, přičemž výchozí hodnota této váhy je 1. Třetí podmínka v následující ukázce (viz Kód 19) tak bude mít dvojnásobnou váhu než první dvě otázky, k maximálnímu celkovému skóre (2 body) tak při svém splnění připočte 1 bod (1. a 2. podmínka přitom každá svým splněním připočítává pouze 0.5 bodu).

```
<textBox id="samohlasky" score="2" recountScore="1">
  <answersRoot>
    <answer compare="contains">Ostrava</answer>
    <answer compare="contains">Brno</answer>
    <answer compare="contains" score="2">Praha</answer>
  </answersRoot>
</textBox>
```

Kód 19 – Ukázka zápisu hodnocení textové odpovědi rozpočítávající skóre na základě tří podmínek

Jednu sadu podmínek je též možné vztáhnout i na více editačních políček současně. Stačí do elementu `<textBox>` přidat podelement `<ids>` a do něho v samostatných elementech `<id>` vyjmenovat identifikátory všech editačních políček, na které se mají podmínky vztahovat (viz Kód 20). Při tomto typu hodnocení nelze použít rozpočítávání celkového skóre na jednotlivé podmínky (`recountScore` musí být 0, tj. false, nebo nesmí být uveden).

```
<textBox id="vyjSlovZ" score="2">
  <ids>
    <id>z1</id>
    <id>z2</id>
    <id>z3</id>
    <id>z4</id>
  </ids>
  <answersRoot>
    <answer cs="0" compare="equal">brzy</answer>
    <answer cs="0" compare="equal">jazyk</answer>
    <answer cs="0" compare="equal">nazývat</answer>
    <answer cs="0" compare="equal">Ruzyně</answer>
  </answersRoot>
</textBox>
```

Kód 20 – Ukázka zápisu hodnocení textových odpovědí zapsaných do 4 různých editačních políček, hodnocených jednou sadou podmínek

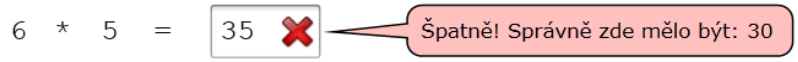
Při tomto postupu jsou pro každou odpověď v každém uvedeném editačním políčku testovány postupně všechny podmínky, přičemž pokud některá z nich platí, je tato podmínka vyřazena z testování pro ostatní odpovědi a zároveň daná odpověď již není dále testována na ostatní podmínky. Díky tomu lze např. požadovat vypsání určitého výčtu slov (každé slovo do vlastního editačního políčka), přičemž jejich pořadí může být libovolné a není možné opakovaně použít totéž slovo. Splnění každé z podmínek zároveň znamená připočtení bodů k celkovému skóre otázky v hodnotě uvedené v atributu `score` v elementu `<textBox>` (tzn., že např. 4 podmínky při skóre 2 mohou přinést až 8 bodový zisk, viz Kód 20).

4.1.8 Zpětná vazba [◎ II.3.C]

Díky automatickému vyhodnocování systém dokáže ihned poznat, je-li odpověď zkoušeného zcela správná, zcela chybná, případně správná jen z části a z jaké. Na základě tohoto rozdělení nejen přidělí procentuální hodnocení, ale může i v rámci zpětné vazby vyznačit jednotlivé části (aktivní prvky) v otázce dle tohoto stavu. K tomu jsou použity tři různé ikony (viz Obr. 13), jež sdělují správnost dílčího řešení otázky. Jejich polohu vzhledem k prvku, který hodnotí, zadává tvůrce otázky ve speciálním mini-jazyku (viz kap. 4.12.5, str. 155) do atributu `rip` (Result Icon Position), jehož hodnotu lze definovat jak globálně, tak pro každý typ prvků i jednotlivé prvky zvlášť.

Obr. 13 – Ikony oznamující dílčí výsledky otázky

Další možností při následném rozboru testu je (kromě vyznačení chybných a správných odpovědí v každé otázce) i funkce, která zkoušenému vypíše, jak to mělo být správně, což dokáže systém (resp. XSLT šablona) generovat automaticky (viz Obr. 14), a proč, což již samozřejmě musí zadat tvůrce otázky. Text této zpětné vazby se zobrazí ve speciální „bublině" po najetí myší nad hodnotící ikonu.



Obr. 14 – Ukázka automaticky generované zpětné vazby

Podmínky a znění zpětné vazby se v otázce zapisuje do elementu `<feedback>`. Ten se, obdobně jako element `<score>`, dělí na podelementy podle jednotlivých druhů aktivních prvků (`textBoxes`, `switches`, `targets`, `items` atd.). V každém z nich pak lze individuálně zapsat zpětnou vazbu pro každý z prvků tohoto druhu, jež se v otázce vyskytují, přičemž rozlišeny jsou dle jejich `id` (viz Kód 21).

```
<feedback>
  <text state="right">Správně!</text>
  <textBoxes rip="R+4M">
    <text state="wrong">Špatně! Správně zde mělo být: </text>
    <text state="neutral">Zde mělo být: </text>
    <textBox id="priklad_1">
      <text noState="right">30</text>
    </textBox>
    <textBox id="priklad_2">
      <text noState="right">5</text>
    </textBox>
  </textBoxes>
</feedback>
```

Kód 21 – Zápis zpětné vazby pro dva textové vstupy

Element `<text>` se v zápisu zpětné vazby používá pro výpis textové hodnoty (text je uveden v hodnotě elementu). Kde bude daný text ve finále uveden, určuje jednak jeho poloha v rámci elementu `<feedback>` a také podmínky, které definují jeho atributy. Například text uvedený přímo v hlavním elementu zpětné vazby, bude její součástí u všech prvků, text pod `<textBoxes>` bude pouze u zpětné vazby textových vstupů apod.

Podmínka uvedená v atributu `state`, udává hodnotu stavu (popř. více hodnot oddělených čárkou), která se zvlášť pro každý prvek ověřuje a pouze v případě shody je daný text zařazen do jeho zpětné vazby. Možné hodnoty stavu jsou `right` (zcela správné řešení), `wrong` (zcela špatné řešení), `neutral` (neřešeno, nebo řešeno tak, že přírůstek bodů nedosáhl možného maxima) a `none` (neřešeno, přičemž žádné řešení by zde body nezvýšilo). Jako podmínku lze též použít atribut `noState`, který naopak ověřuje, že stav prvku není z uvedeného seznamu (negace `state`).

Další možné podmínky mohou být zapsány v atributu `val` (popř. negace `noVal`). Ta testuje hodnotu prvku, což je u textových vstupů zapsaný text, u přepínačů `id` aktuálního stavu, u cílů `id` přiřazené položky apod. Atribut `rightVal` pak porovnává hodnotu správné odpovědi prvku, jež odvodí ze skóre pro jednotlivé stavy prvku (viz Kód 22). Atribut `isSet` může určovat logickou hodnotu, která omezuje platnost textu na prvky, které mají nebo naopak nemají nastavenou nějakou hodnotu. Atribut `score` umožňuje pomocí speciálního mini-jazyka (viz kap. 4.12.2, str. 151) omezit platnost dle rozsahů procentuální správnosti prvku. Podobně atribut `valNumeric` (popř. negace `noValNumeric`) stejným způsobem určuje rozsahy, ovšem pouze u odpovědí typu číselná hodnota.

Mimo elementu `<text>` lze použít též `<br>`, který v textu zpětné vazby vynechá řádek (viz Kód 22). Další možností je `<grp>`, jež umožňuje seskupit více podlementů do jednoho a tím pádem uvést podmínku pouze 1x. Jeho další výhodou je možnost změny orientace řazení výstupu, který je standardně pod sebe (vertikální), ale pokud se atribut `o` (orientace) nastaví na `h` (horizontální), bude se text ve skupině řadit vedle sebe.

Poslední možností je použití tzv. zdrojů (resources), které umožňují vícenásobné použití delších textů, aniž by se musely vypisovat více než právě jednou. Ty jsou pomocí identifikátoru, propojeny se svou definicí a vyjma toho, že sami nemají žádný obsah, se používají stejným způsobem jako `<text>`. Tyto zdroje se definují jako jednotlivé elementy v hlavním elementu `<resources>`, umístěném přímo ve `<feedback>` (viz Kód 22). Zdroje je vhodné použít např. pro vysvětlení u podotázek, jež mají více podobných verzí (např. různé formulace téže věty) a musely by se tak vypisovat vícekrát.

```
<feedback>
  <switches rip="1-8M">
    <text state="wrong">Špatně!</text>
    <text state="right">Správně!</text>
    <text noState="right" rightVal="1">Správná odpověď je ANO.</text>
    <text noState="right" rightVal="0">Správná odpověď je NE.</text>
    <br />
    <switch id="1"><resource id="molekula" /></switch>
    <switch id="2"><resource id="led" /></switch>
  </switches>
  <resources>
    <text id="molekula">Molekula je svazek několika atomů.</text>
    <text id="led">Led vzniká zamrznutím vody při teplotaách od 0°C níže.</text>
  </resources>
</feedback>
```

Kód 22 – Zápis zpětné vazby s použitím zdrojů pro opakované použití

4.2 Sestavování testu [◎ IV.2]

Test je v podstatě souborem otázek a nastavení. Otázky se do něho přiřazují z celkové databáze otázek, přičemž je lze i v rámci testu hierarchicky seskupovat do skupin, které mohou mít vlastní nastavení platné pouze pro danou **skupinu** (např. viz Příloha 8). Patří mezi ně, nastavení počtu otázek, jež budou z celé skupiny vybrány, mají-li se otázky v této skupině míchat, mají-li držet při sobě (při nadřazeném míchání by otázky ze skupiny byly vždy u sebe, neproložené otázkou z jiné skupiny), má-li být tato skupina v testu zastoupena vždy (v případě, že by byla součástí nadřazeného částečného výběru), její pevnou pozici mezi otázkami, celkovou bodovou váhu (na tu by se pak otázky skupiny skládaly v poměru svých individuálních vah a správnosti jejich řešení) a úroveň.

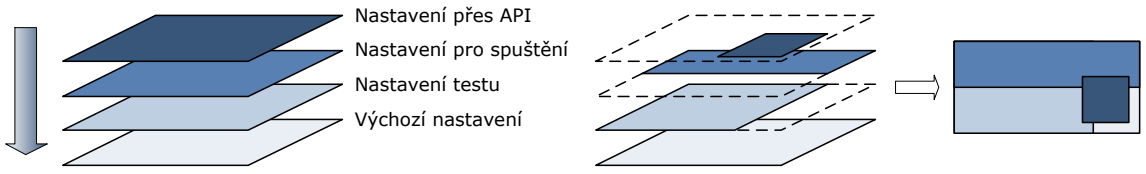
Skupinám otázek i jednotlivým otázkám v testu je možné nastavit identifikátor **úrovně** (*level*), jímž lze pro jednotlivé spuštění testu hromadně vymezovat, které otázky mají být v testu zahrnuty (popř. i v jakém počtu) a které ne. Ten samý test pak lze použít pro každou úroveň zvlášť, ale při závěrečném hodnocení i pro libovolnou kombinaci těchto úrovní. Volba úrovně se provádí ve speciálním mini-jazyku (viz kap. 4.12.7, str. 157).

Jednotlivým **otázkám**, ať již jsou v jakékoli skupině, či přímo v testu, lze taktéž nastavit bodovou váhu, povinné zahrnutí do testu, pevnou pozici, úroveň a navíc i časový limit.

4.2.1 Nastavení a otevření testu [◎ IV.2.A]

Nastavení testu umožňuje individuálně povolit či zakázat mnoho různých možností. Patří mezi ně i postranní panel se seznamem otázek, který může (nebo nemusí) obsahovat číslo otázky, její název, identifikátor stavu řešení, pomocné zaškrtávací políčko (zkoušený si jím může označovat otázky, ke kterým se již nechce vracet) a procentní ukazatel správnosti řešení. Ten může reagovat buď okamžitě v reálném čase na každou změnu v otázce, až po přepnutí otázky nebo až po jejím uzamčení. S tím souvisí i další nastavení jako je možnost zakázat vracení se v otázkách zpět, vracení se pouze k již řešeným otázkám, možnost upravovat otázky, které již byly řešeny a přepnuty apod. Dále je možné zobrazovat i celkové skóre testu (i s možností jeho okamžité aktualizace nebo přepočtem až po přepnutí či uzamčení otázky), zapnout ochranu proti přepínání oken, nastavit způsob protokolování průběhu testu, časový limit testu, omezit počet otázek apod. Nastavení testu se provádí ve speciálním editoru nastavení testu (viz Obr. 44 na str. 145).

Systém má své výchozí nastavení, které lze pro každý test v každém jeho bodu změnit. Totéž je možné provést i pro každé individuální otevření testu a i toto nastavení může být ještě upraveno při spuštění testu přes API (viz kap. 4.3.3, str. 72), přičemž je uplatňován princip dědičnosti (viz Obr. 15) jako např. u CSS kaskádových stylů (viz [39 str. 62]).

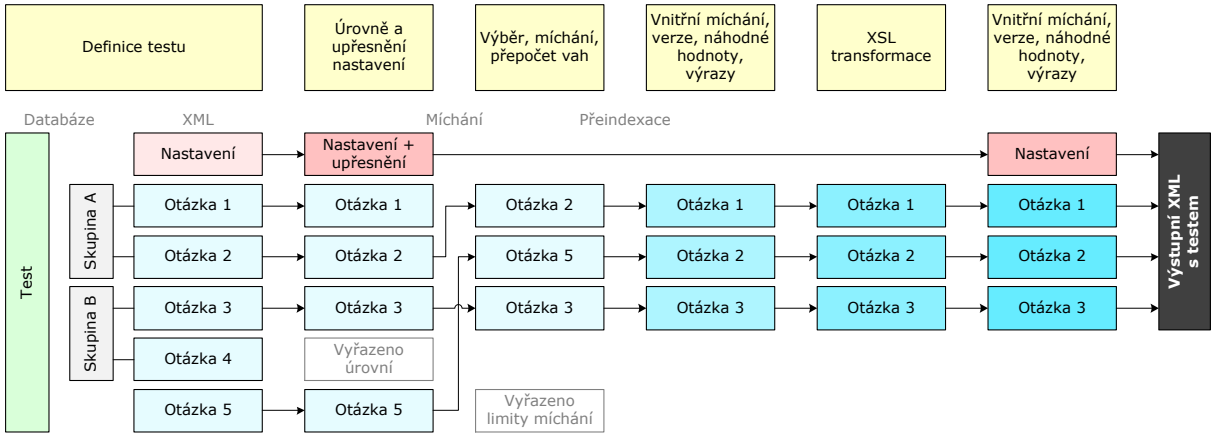


Obr. 15 – Ilustrační ukázka překrývání platnosti různých úrovní téhož nastavení

Otevření testu zde znamená jeho zpřístupnění dalším uživatelům systému (zkoušeným) pro jeho spuštění. To se provádí na určitý datum a čas, s možností opakování (např. „každé druhé pondělí od 8:00 do 9:00“, viz kap. 4.12.8, str. 158), pro vybrané uživatele nebo skupiny uživatelů (s hierarchickou podporou, dle [ap-1]), pro vymezené IP adresy (viz kap. 4.12.3, str. 151), s možností individuálního nastavení, vymezení použitých úrovní (viz kap. 4.12.7, str. 157), povolených počtů opakování, podmínek pro prohlížení vyhodnocení testu a jeho případné absolvování (viz kap. 4.12.9, str. 162).

4.2.2 Generování testu

V okamžiku, kdy uživatel spustí klientskou část aplikace a zapne si nový test, je na server vyslán požadavek na jeho sestavení. To se provádí pokaždé znovu na základě jeho aktuálního nastavení. Průběh generování testového zadání ukazuje Obr. 16.



Obr. 16 – Schéma postupu generování testu

Nejprve se načte nastavení a definice celého testu včetně seznamu jeho otázek a skupin otázek a určí se, pro které jeho *otevření* je test spouštěn. Na základě úrovně definované v otevření testu se z výběru vyřadí příslušné otázky a skupiny. Se zbytkem je provedeno nastavené míchání otázek, případně náhodné výběry otázek z větších skupin, dle metody popsané v kap. 4.6 (str. 84), je-li tato volba nastavena. Pro aktuální výběr dojde k přepočtu vah otázek z víceúrovňových skupin. V dalším kroku jsou na úrovni XML překontrolovány jednotlivé elementy každé otázky a dojde ke zpracování elementů pro jejich interní míchání (viz kap. 4.1.6, str. 48). Následně je s každou otázkou, která má danou volbu nastavenou, provedena příslušná XSL transformace. Výsledek je opětovně překontrolován na elementy pro interní míchání, neboť ty mohou generovat i některé šablony. Na závěr jsou otázky sestaveny do jednoho XML souboru, je k nim přiložena kompletní definice nastavení testu a celý tento výsledek je odeslán zpět klientovi.

4.2.3 Protokol o testování [◎ II.3.D]

O průběhu každého zkoušení musí být sestaven a uložen protokol, který umožní kdykoli v budoucnu zobrazit si přesně tentýž test, jaký zkoušený dostal zadán a v něm musí být zaznamenány veškeré jeho odpovědi, včetně těch nesprávných, či jen částečně řešených (jako kdyby odevzdal test řešený na papíře).

Za tímto účelem se ukládá vždy kompletní vygenerovaný test s pro testování použitým nastavením a všemi otázkami ve finálním QML (po XSL transformaci i veškerém interním míchání). Odpovědi na otázky jsou pak zaznamenány zvlášť do dalšího XML dokumentu, z něhož se po načtení testu dají snadno nastavit veškeré jeho hodnoty tak, jak provedl zkoušený uživatel, než byl test ukončen. Veškerá bodová hodnocení, výsledky a váhy jsou také součástí tohoto souboru, aby byly čitelné i při editaci nezašifrované verze tohoto dokumentu v obyčejném textovém editoru.

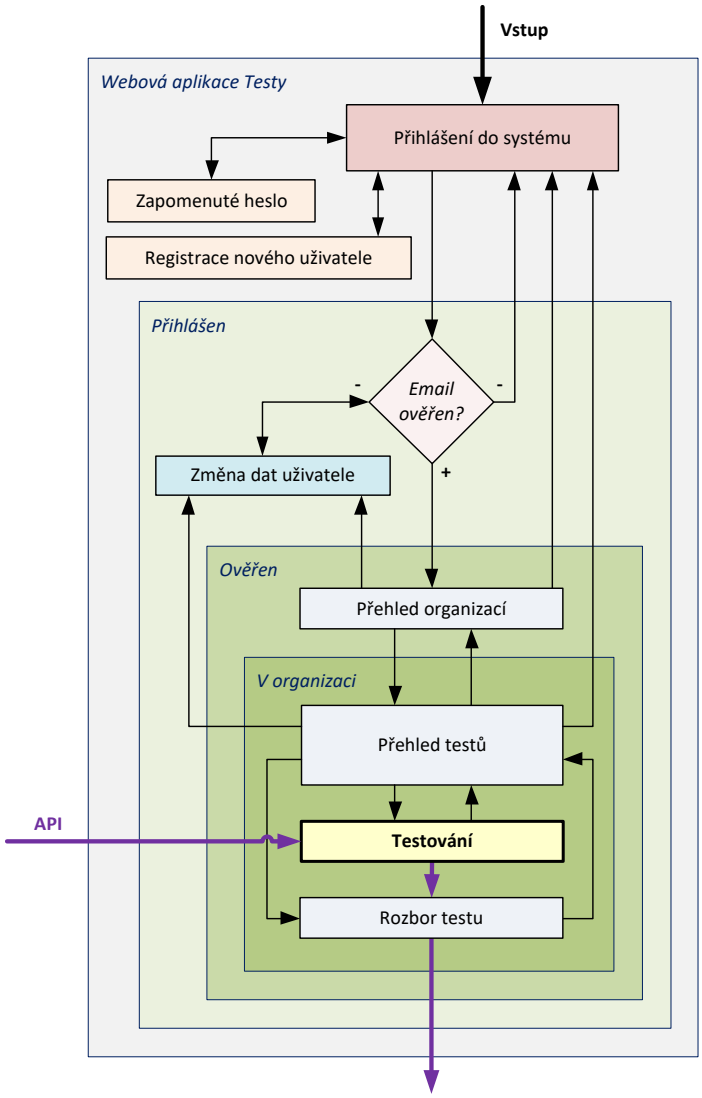
Díky tomu lze kdykoli zpětně zobrazit jakýkoli proběhnutý test i v případě, že se zadání jeho otázek, výběru otázek nebo příslušných transformačních šablon změní. Jelikož jsou protokoly uchovávány ve formátu XML, budou zpětně kompatibilní i v případě že by se v budoucnu měla měnit i přímo jejich struktura, resp. byly do ní přidány další dodatečné údaje.

4.3 Rozhraní

Testovací prostředí se skládá ze tří rozhraní a to testovacího (uživatelského), v němž se provádějí připravené testy, administračního, kde se testy sestavují a spravuje systém a aplikačního, jež umožňuje propojení se systémy třetích stran.

4.3.1 Testovací rozhraní [◎ III]

Testovací část sytému umožňuje vlastní realizaci průběhu testování z předpřipravených testů[33]. Kromě testování (viz Příloha 7) zároveň poskytuje i komplexní rozhraní umožňující plnohodnotné fungování této části jako samostatné aplikace. Disponuje tedy také registrací a přihlašováním uživatelů, včetně ověření platnosti e-mailové adresy a obnovy zapomenutého hesla, výběrem organizace pro vstup, přehledem dostupných testů (viz Příloha 6) a z nich dosažených výsledků včetně možnosti jejich detailního rozboru a zpětné vazby. Strukturu celé testovací části znázorňuje Obr. 17.



Obr. 17 – Struktura stránek rozhraní testovací části

[33] průběh testu si lze vyzkoušet na [w80]

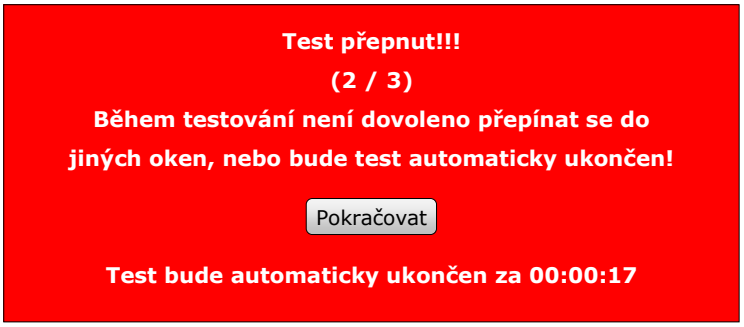
Přihlášený uživatel (zkoušený) si tedy v přehledu dostupných testů zvolí jeden z nich. Jeho podepsaný (viz kap. 4.8.4, str. 106) požadavek je odeslán na server, kde se znovu ověří dostupnost testu pro daného uživatele a podle zadaných kritérií se mu sestaví (viz kap. 4.2, str. 67) a odešlou data s připraveným testem. Klientská aplikace test načte, stáhne případné obrázky (viz str. 41), jsou-li součástí některých otázek a vyčká uživatelova pokynu pro zahájení testu. Pokud je test časově omezen, pak až v tuto chvíli je zahájen jeho odpočet.

V průběhu samotného testování již dalšího spojení se serverem není třeba, takže případné výpadky spojení proces testování nijak neovlivní [◎ III.2.C]. Po skončení testu, ať již uživatelem nebo z důvodů vypršení časového limitu, se test automaticky uzavře a znemožní tak zkoušenému další zásahy do řešení otázek. Přitom se sestaví protokol o zkoušení (viz kap. 4.2.3, str. 68), který se odešle na centrální server. Pokud je připojení na internet nedostupné právě v tuto chvíli, po překročení timeoutu je uživateli zobrazeno ohlášení této skutečnosti a nabídnuta možnost pokus zopakovat. To lze provádět tak dlouho, dokud se odeslání výsledků nezdaří. Je zde také možnost výsledky uložit do šifrovaného souboru (viz kap. 4.9, str. 113) a ten do systému nahrát později.

Až po úspěšném odeslání výsledků na server se zkoušenému zobrazí výsledné skóre, kterého dosáhl a je-li to tvůrcem testu povoleno, umožní se mu otevření rozboru testu a zpětné vazby (viz kap. 4.1.8, str. 64).

Během testování je aktivních několik ochranných mechanizmů. Jedním z nich je neustálé sledování systémového času, aby nedošlo k jeho manipulaci [◎ III.2.A], neboť s jeho pomocí je měřen časový limit na test. Každých 100ms je tak tento čas ověřován a pokud je jeho hodnota mimo toleranci od hodnoty předpokládané[34], je test automaticky ukončen s oznámením této skutečnosti.

K dispozici je i ochrana proti opisování z internetu [◎ III.2.B]. Je-li totiž okno prohlížeče nebo jeho záložka během testování přepnuta, test je překryt červeným panelem a je zahájen odpočet jeho automatického ukončení, které lze zastavit pouze kliknutím na tlačítko „Pokračovat" (viz Obr. 18). Čas pro tento odpočet i počet tolerovaných přepnutí lze nastavit pro každý test zvlášť.



Obr. 18 – Ukázka hlášení ochrany proti opisování z internetu (přepínání oken)

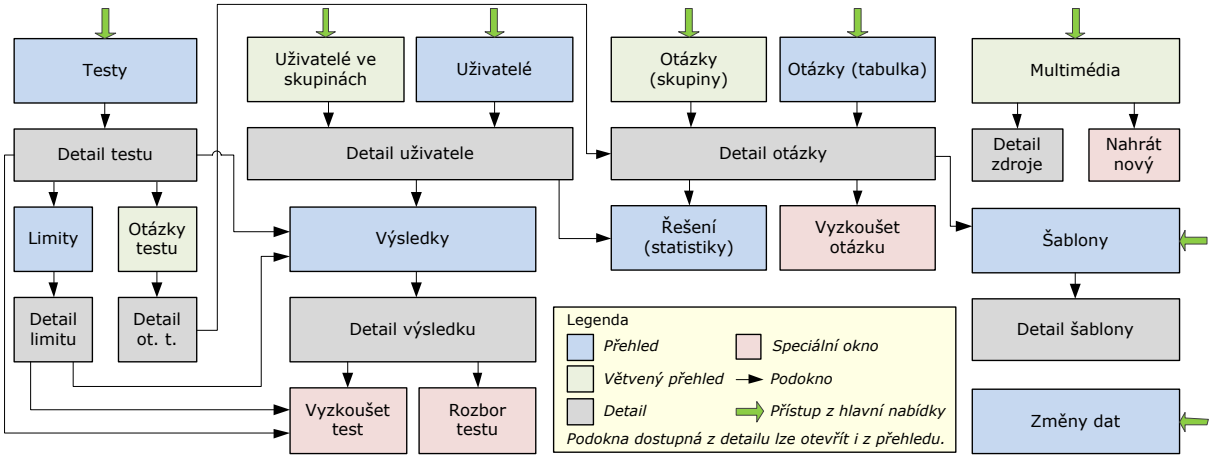
Další ochrana v průběhu testování brání nechtěnému ukončení testování z důvodů jako je např. vypnutí okna prohlížeče, vypnutí záložky prohlížeče, znovu načtení stránky, vrácení se o stránku zpět, přechod na jinou webovou adresu apod. Tyto události hlídá JavaScript, aktivovaný (či deaktivovaný) Silverlight aplikací pouze během testování, který v případě potřeby zobrazí dialog upozorňující zkoušeného, že tato akce nenávratně přeruší testování a nabídne mu možnost akci odvolat a vrátit se k testu.

[34] s přechodem na letní a zimní čas není zatím v tomto případě počítáno, časově omezené testy by se tak v tyto okamžiky neměly spouštět

4.3.2 Administrační rozhraní [◎ IV]

Administrační rozhraní umožňuje správu uživatelů, multimédií, otázek a sestavování, zadávání a vyhodnocování testů. Stejně jako testovací rozhraní, je i toto realizováno jako RIA aplikace, na platformě Silverlight, tedy plně dostupné online, pouze s použitím prohlížeče a tohoto pluginu.

Administrační GUI má standardní intuitivní rozložení (viz Příloha 8). V horní části se nachází ovládací prvky rozmístěné na nástrojových panelech v ribbon rozhraní, vlevo je v ukotveném plovoucím okně hierarchický přehled dostupných oken (1. úroveň seznamu – hlavní nabídka) a z nich otevřených podoken a v hlavní části se pak na záložkách otevírají jednotlivá okna (stránky) aplikace. Jejich vzájemnou provázanost ukazuje následující schéma (viz Obr. 19).



Obr. 19 – Schéma vazeb jednotlivých oken administračního rozhraní aplikace

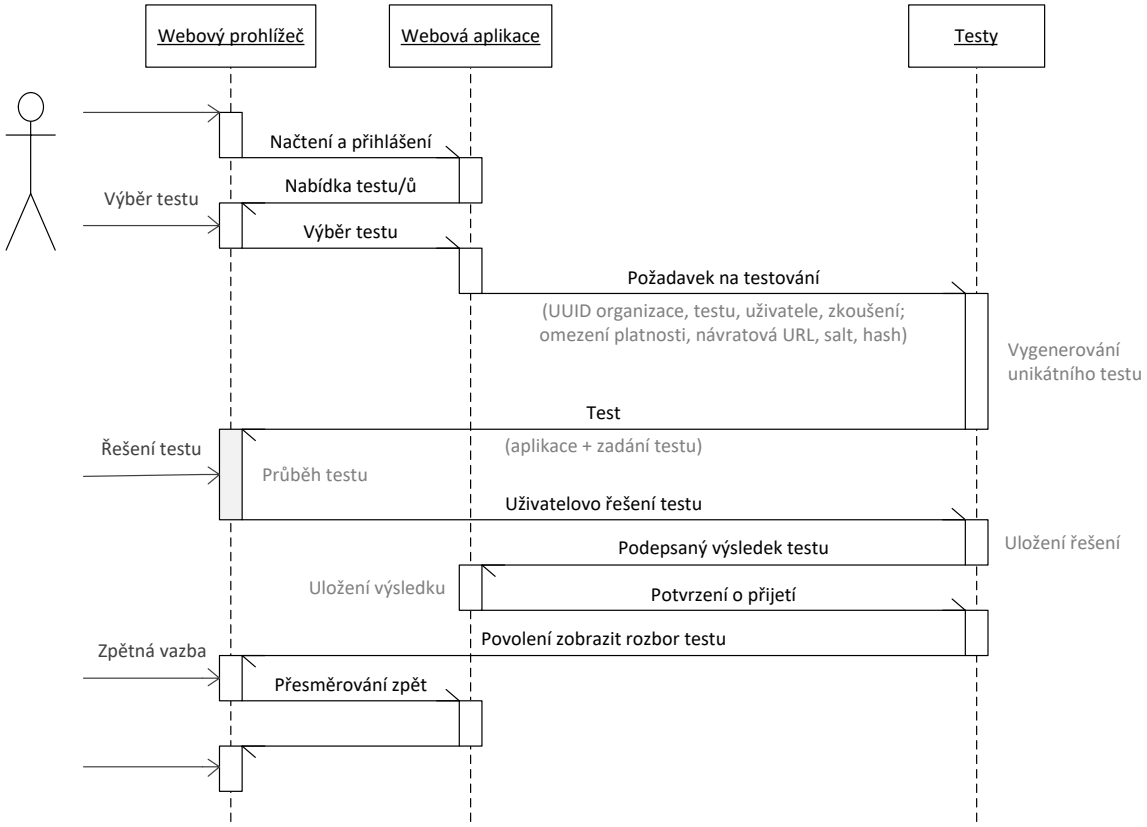
Struktura administračního rozhraní je plně konfigurovatelná pomocí jednoduchého XML souboru (viz kap. 4.11, str. 132) a pro spojení se serverem využívá vlastní komponenty pro bez stavový přístup k centrální databázi (viz kap. 4.10, str. 125).

4.3.3 Aplikační rozhraní [◎ III.4.B]

Testovací část aplikace, kromě přímého uživatelského použití, disponuje i rozhraním API[35], tedy možností jejího částečného ovládání prostřednictvím aplikací třetích stran. Tyto aplikace tak mohou například pomocí parametrizovaných URL spouštět přímo konkrétní testy pro předdefinované uživatele a po jejich ukončení přijímat výsledky z těchto testů přes návratovou či skrytou URL. Veškerá komunikace mezi aplikacemi samozřejmě podléhá přísnému zabezpečení proti neoprávněným zásahům, s využitím jednorázově platných podpisů obou stran (podobně jako v kap. 4.7, str. 91). [ap-8]

Testy tak mohou být spouštěny nejen přes integrované rozhraní, ale i přímo přes odkazy či tlačítka jiné webové aplikace a to jak na samostatné stránce prohlížeče, tak i v rámci jiné stránky. Testy tak lze přímo propojit s obsahem nejen libovolného LMS, ale i kterékoli jiné aplikace. Na server webové aplikace přistupující k testům přitom není třeba instalovat jakýkoli další software, k testovacímu prostředí se přistupuje výhradně jejím voláním na centrálním serveru, podobně jako fungují video objekty na YouTube [w84].

[35] API - Application Programming Interface – aplikační programové rozhraní



Obr. 20 – Sekvenční diagram komunikace přes aplikační rozhraní

Sekvenční diagram na Obr. 20 ukazuje postup komunikace webové aplikace s testovacím prostředím přes URL. Webová aplikace si zde sama obstará autentizaci uživatele a na jejím základě nabídne testy, které mu chce zprostředkovat. Uživatel si zvolí test a server webové aplikace sestaví a „podepíše" URL adresu s požadavkem, na kterou uživatele přesměruje (viz Kód 23). To lze realizovat různými způsoby, např. kompletním přesměrováním celé stránky, otevřením v novém okně či záložce prohlížeče, nebo i uvnitř designu stránky webové aplikace ve vlastním rámci (frame či iFrame, podobně jako v [50]).

Server s testovacím prostředím podle zaslaných parametrů vygeneruje test a zobrazí jej v prostoru vymezeném webovou aplikací. Uživatel (zkoušený) vyplní test a po ukončení se jeho výsledky odešlou na server s testy, který je archivuje a bylo-li to webovou aplikací požadováno, skrytě jí je odešle jako parametry v zadané autentizované URL (parametr `secretUrl` stejné struktury jako `redirectUrl`, viz Kód 23). Následně testovací aplikace uživateli zobrazí vyhodnocení testu se zpětnou vazbou (viz kap. 4.1.8, str. 64) a po ní přesměruje stránku (popř. rámec) na zadanou URL adresu, jejíž součástí opět mohou být parametry s identifikací a výsledky testu.

```
http://www.alltest.eu/TestApp.aspx?user=OUE50f14gsg&org=Ouhk&test=Tks67Mas9&id=demo
&expire=20120125224107&redirectUrl=www.lms.cz%3fresult%3d%23result%23%26score%3d%23
score%23%26id%3d%23id%23&settings=settings(limits(testTimeLimit[600])mixing(questio
nsCount[3]))&salt=SjRsUaFFCHnffH1Q&hash=25B9606CFD5B3FBA0C816CF

Dekódovaná redirectUrl: www.lms.cz?result=#result#&score=#score#&id=#id#
```

Kód 23 – Ukázka URL s parametry pro komunikaci přes API

URL pro spuštění testu může obsahovat následující parametry.

- `user` – identifikátor uživatele v organizaci přidělený testovací aplikací
- `org` – identifikátor organizace přidělený testovací aplikací
- `test` – identifikátor testu přidělený testovací aplikací
- `id` – libovolný identifikátor, který bude uložen u výsledku testů pro pozdější třídění
- `expire` – datum a čas (ve formátu „rrrrMMddHHmmss“) určující konec platnosti URL
- `redirectUrl` – návratová adresa, kam bude po skončení testu uživatel přesměrován (bude doplněna o parametr s podpisem pravosti, tj. `salt + hash`, podobně jako v kap. 4.8.4, str. 106); může obsahovat následující parametry:
  - `result` – identifikátor záznamu s výsledkem testu přidělený testovací aplikací
  - `score` – uživatelem dosažený výsledek (skóre) z testu
  - `id` – stejný identifikátor, jež byl v parametru `id` na vstupu
- `secretUrl` – adresa se stejnými možnostmi jako `redirectUrl`, kterou server skrytě zavolá, při ukládání výsledků
- `settings` – dodatečná nastavení testu (viz kap. 4.12.10, str. 164)
- `salt` – 1x použitelná hodnota unifikující kontrolní součet (viz kap. Salt, str. 103)
- `hash` – kontrolní součet (viz kap. Hash, str. 102)

Veškeré URL, přes jejichž parametry spolu oba servery (s webovou aplikací a testovacím prostředím) komunikují, jsou chráněny proti zásahům (úpravám) třetích stran. To zajišťuje parametr `hash`, což je kontrolní hash vypočtený z textového řetězce, jež obsahuje hodnoty všech ostatních parametrů a navíc je doplněn o tajný klíč přidělený organizaci testovací aplikací. Ta jej samozřejmě zná také, takže pro veškeré příchozí požadavky může stejným způsobem sama vypočítat kontrolní hash a porovnat jej s tím příchozím. Pokud byla hodnota některého z parametrů někým nepovolaným pozměněna, kontrolní součet nebude souhlasit a požadavek bude zamítnut. Jelikož by tajný klíč organizace nikdo znát neměl, neměl by být ani schopen vypočítat nový hash, platný pro upravenou URL. Zároveň tento klíč sám osobě ani v zašifrované formě posílán není, a z hashe jej zpětně odvodit nelze, tudíž je komunikace z tohoto hlediska dostatečně zabezpečena.

Opakovanému použití jedné URL pak zabraňuje parametr `salt`, který musí být vždy jedinečný a díky jehož zařazení je pokaždé jiný i kontrolní hash. Jedinečnost tohoto saltu je kontrolována, obdobně jako při komunikaci klientské části testovací aplikace se serverem (viz kap. 4.8.4, str. 106) a v případě zjištění jeho duplikace, je příchozí požadavek zamítnut.

Uživatel tedy nemůže parametry měnit, ovšem je-li také nežádoucí, aby je mohl vidět, lze je poslat v zašifrované formě. Veškerá parametrová část URL (za otazníkem), kromě parametru `org` se tak zašifruje pomocí algoritmu AES, přičemž jako heslo je použit tajný klíč organizace, a výsledek se doplní do URL jako parametr `cypher`. Možné je též parametry pouze skrýt před přímou čitelností bez nutnosti šifrování, popř. šifrování o toto ještě doplnit a skrýt tak i hodnotu `org`. Lze totiž celou parametrovou část URL zakódovat pomocí Base64, které je sice snadno dekódovatelné, ale pro člověka bez dalších pomůcek nečitelné, a výslednou hodnotu dát do jediného parametru `data`.

Volný přístup

Ne vždy je možné výše popsaný zabezpečený přístup z každé aplikace zajistit, například pokud je webová aplikace, do níž se test integruje, uzavřená, bez možnosti vkládání vlastního kódu na straně serveru. Také zabezpečený přístup nemusí být zcela nezbytný, pokud je jejím prostředkem zpřístupňován nehodnocený výukový test, který není nezbytné nijak omezovat.

Pro tyto účely je možné místo uvedených parametrů uvést pouze jeden a to `free`, jehož hodnota bude udávat identifikátor UUID otevření testu. Toto otevření testu musí mít zároveň povoleno, že má být dostupné pro volný přístup z aplikací a musí mu být přiřazen jeden konkrétní uživatel, pod kterým se budou ukládat výsledky z jednotlivých testování. URL adresa pro spuštění takovéhoto testu je pak platná vždy a odkudkoli, přičemž je stále možné její dostupnost v rámci nastavení podmínek otevření testu omezit podle IP adresy (viz kap. 4.12.3, str. 151) či počítače a pro určitý čas (viz kap. 4.12.8, str. 158). Další parametry této adresy nejsou povinné, ale uvést a použít lze `redirectUrl` a `id`. Identifikátor zkoušení v parametru `id` přitom může být využit libovolně, např. pro podpis či e-mail uživatele, který zadá do webové aplikace před samotným spuštěním testu.

Jelikož v některých aplikacích může být test žádoucí nespouštět přímo, ale např. až po klinutí na nějaké tlačítko (např. u kontrolních otázek v přehledu některých LMS by jejich hromadné načítání již při zobrazení stránky přehledu mohlo být zbytečně zpomalující), byla na serveru s testy vytvořená úvodní stránka (TestFree.aspx), jež tuto funkci umožňuje. Jí se předají tytéž parametry jako přímo aplikaci (TestApp.aspx), s tím, že na ni je stránka přesměrována až po kliknutí na toto tlačítko. Zároveň je mezi URL parametry přidán `redirectUrl`, který po skončení testu zajistí zpětné načtení této úvodní stránky, umožňující spustit stejný test znovu.

4.4 Podobnosti textových řetězců [◎ II.3.B]

Při automatickém vyhodnocování otevřených textových odpovědí mělo být kromě klasického porovnávání na základě vzoru možné hodnotit i částečně správné odpovědi, tj. např. hodnoty s překlepem. Vyjádřit však pro porovnání veškeré varianty možných překlepů, byť s pomocí např. regulárních výrazů, je vzhledem k možnému počtu kombinací v podstatě nemožné. Z těchto důvodů byla do systému přidána funkce pro porovnávání dvou textů fuzzy způsobem, čili vyjádřením míry podobnosti zadaného textu se správnou odpovědí.

Algoritmů, které dokáží určit tuto míru podobnosti dvou textových řetězců, již přitom existuje celá řada, proto byl v tomto případě před vývojem vlastního postupu upřednostněn rozbor a následná implementace těch stávajících.

Sada nejpoužívanějších algoritmů pro porovnávání textových řetězců již byla programově zpracována do open source knihovny SimMetrics [w64] v rámci grantu GR/N15764/01[36] [w28] na univerzitě UK Sheffield University. Zdrojové kódy této knihovny jsou volně dostupné v jazycích Java a .NET C# a implementují algoritmy uvedené v Tab. 5. Jejich zkratky budou pro přehlednost uváděny v dalším textu a zároveň se používají pro určení, který algoritmus má být použit při automatickém hodnocení textových odpovědí (viz str. 62). Typ určuje, s jakými útvary daný algoritmus při porovnání prioritně pracuje, tj. znaky (Z), slovy (S) nebo délkou řetězce (D) [51 str. 52].

Algoritmus	Zkratka	Název algoritmu	Typ
BlockDistance	BD	Algoritmus blokové vzdálenosti	S
ChapmanLengthDeviation	CLD	Chapmanova odchylka délky	D
ChapmanMeanLength	CML	Chapmanova střední délka	D
CosineSimilarity	CS	Kosinová podobnost [52]	S
DiceSimilarity	DS	Diceho míra podobnosti	S
EuclideanDistance	ED	Euklidova vzdálenost	S
JaccardSimilarity	JS	Jacardova míra podobnosti	S
Jaro	J	Jarův algoritmus [53]	Z
JaroWinkler	JW	Jaro-Winklerův algoritmus [54]	Z
Levenstein	L	Levenshteinova vzdálenost [55]	Z
MatchingCoefficient	MC	Koeficient shody	S
MongeElkan	ME	Monge-Elkanova vzdálenost [56]	Z
NeedlemanWunch	NW	Needleman-Wunschův algoritmus	Z
OverlapCoefficient	OC	Koeficient překrytí	S
QGramsDistance	QGD	Vzdálenost q-gramů[57]	Z
SmithWaterman	SW	Smith-Watermanův algoritmus	Z
SmithWatermanGotoh	SWG	Smith-Waterman-Gotohův algoritmus	Z
SmithWatermanGotohWindowedAffine	SWGWA	Smith-Waterman-Gotohův algoritmus s afinními okny	Z

Tab. 5 – Přehled podporovaných algoritmů pro určení míry podobnosti dvou textových řetězců

[36] název projektu je Interdisciplinary research collaboration in advanced knowledge technologies - AKT

4.4.1 Jednoslovná porovnávání

Pro určení vhodnosti použití jednotlivých algoritmů pro daný účel bylo provedeno několik měření[37]. Nejčastějším případem textových odpovědí jsou jednoslovné výrazy, pro srovnání jejichž hodnocení bylo porovnáváno slovo *logaritmus*[38] s jeho různými modifikacemi. Výsledky tohoto měření ukazuje Příloha 9 a časy potřebné pro tyto jednotlivé výpočty Příloha 10.

Jak je patrné z tohoto měření (Příloha 9), algoritmy BD, CS, DS, ED, JS, MC a OC téměř ve všech případech, vyjma mezery před a za porovnávaným slovem, nedetekovala žádnou schodu (0%) obou výrazů, ač tato byla téměř ve všech případech zjevná. Podobně shodu hodnotil i CML, který ovšem při těchto změnách vracel hodnotu 15%, avšak stejnou hodnotu určil i při porovnání dvou totožných slov. Algoritmus CLD hodnotí shodu pouze z ohledu počtu znaků, tudíž v podstatě ignoroval veškeré změny týkající se pouze záměny písmen, při které vždy stanovil 100% shodu. Algoritmy ME, SW, SWG a SWGWA pak byly při určování shody netolerantnější u změn spočívající ve vynechání či přidání jednoho písmene, kde opět ve většině případů určovaly 100% shodu.

Výsledky zbylých algoritmů (J, JW, L, NW a QGD) se pak více blížily k pro člověka očekávaným hodnotám, s různou úrovní tolerance různých druhů změn porovnávaného slova. Například algoritmy J a JW při porovnávání berou v potaz podobnost písmen jak z grafického hlediska (např. „t" a „f", nebo nula „0" a „o"), tak významového (např. „i" a „y"), ale i z hlediska výslovnosti (např. „s" a „z") [58]. Dokonce byly lépe hodnoceny i změny v diakritice (např. „s" a „š", ale i „u" a „ů") než při změně písmene za úplně jiný znak. Jako zásadní rozdíl je však vnímána změna velikosti písmene, načež by před porovnáváním měly být všechny znaky převedeny na stejnou velikost, není-li právě ta předmětem hodnocení. Neobvyklý je i výsledek porovnání slova s jeho pouhým prvním písmenem, kdy algoritmus JW určil dokonce 73% shodu. Pro tento algoritmus je totiž důležitější začátek slova než jeho konec či střed.

Při jednoslovných srovnáních jsou tedy jednotlivé algoritmy vhodné pro různé účely, přičemž u každého z nich je třeba počítat s jinou stupnicí hodnocení. Další obdobná měření lze nalézt také třeba v [59] a [60].

Z hlediska času potřebného pro výpočet shody (viz Příloha 10, kde časy jsou vždy mediánem z deseti měření téhož a výsledek je v mikrosekundách) si nejhůře vedly algoritmy ME, SWG a SWGWA, které byly zhruba 20x pomalejší (průměrně 126 μs), než nejrychlejší CLD a CML (průměrně 6 μs). I s nimi je však průměrná doba všech těchto výpočtů pouhých 31 μs, tudíž by použití žádného z těchto algoritmů pro jednoslovná porovnání nepředstavovalo na klientské stanici znatelné zpomalení.

4.4.2 Víceslovná porovnávání

Dále byly testovány výsledky porovnávání delších textových útvarů různých délek. Nejprve šlo pouze o dvouslovný výraz „disertační práce". Tab. 6 ukazuje výsledné procentuální hodnocení porovnání jednotlivými algoritmy (sloupce) s různými úpravami tohoto výrazu, jež jsou uvedeny v prvním sloupci.

[37] parametry počítače použitého pro měření uvádí Příloha 25
[38] slovo *logaritmus* bylo pro měření zvoleno (pomocí [w62]), protože každé písmeno v něm je pouze 1x, obsahuje znaky možné zaměnit za jiné podobné znaky (L, O), písmena doplnitelná o diakritiku (o, a, i, u, s) a z jeho délky (10 znaků) bude dobře patrný poměrný vliv jednoho znaku na výsledek porovnání

"disertační práce"	BD	CLD	CML	CS	DS	ED	JS	J	JW	L	MC	ME	NW	OC	QGD	SW	SWG	-WA
disertační práce	100	100	23	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
práce disertační	100	100	23	100	100	100	100	72	72	25	100	100	50	100	61	63	63	63
disertačnípráce	0	94	23	0	0	0	0	98	99	94	0	100	94	0	86	97	93	93
dyzertační práce	50	100	23	50	50	29	33	92	93	88	50	85	94	50	78	81	81	81
bakalářská práce	50	100	23	50	50	29	33	58	58	38	50	60	69	50	33	38	38	38
diplomová práce	50	94	23	50	50	29	33	64	71	50	50	61	72	50	46	40	40	40
disertační	67	63	19	71	67	42	50	88	93	63	50	60	63	100	67	100	100	100
práce	67	31	16	71	67	42	50	0	0	31	50	60	66	100	40	100	100	100
disertační obhajoba	50	84	25	50	50	29	33	76	85	58	50	60	71	50	56	69	69	69
d. p.	0	31	16	0	0	0	0	42	48	19	0	50	50	0	8	40	40	40
Průměr	**53**	**80**	**21**	**54**	**53**	**40**	**43**	**69**	**72**	**56**	**50**	**74**	**73**	**60**	**57**	**73**	**72**	**72**

Tab. 6 – Podobnost dvouslovného výrazu „disertační práce" (16 znaků) s jeho různými modifikacemi dle jednotlivých srovnávacích algoritmů

Tentokrát již nenulové výsledky určily i algoritmy CS, DS, ED, JS, MC i OC. Je tedy patrné, že ač se pro porovnání jednoslovných výrazů nehodí, což dokazuje i varianta zápisu slov bez mezery, pří víceslovných výrazech může být jejich nasazení na místě. Na rozdíl od algoritmů úspěšnějších v předchozích případech totiž porovnávání provádějí na úrovni slov, nikoli znaků.

Při dalším měření byla délka řetězce rozšířena na 100 znaků (14 slov), které tvořila jedna věta, konkrétně šlo o upravené znění Archimédova zákona (viz Tab. 7).

Typ změny věty	BD	CLD	CML	CS	DS	ED	JS	J	JW	L	MC	ME	NW	OC	QGD	SW	SWG	-WA
prohodit slovo	100	100	87	100	100	100	100	96	98	90	100	100	94	100	96	82	87	87
umazat slovo	96	87	85	96	96	81	92	86	92	87	93	96	87	100	92	93	96	96
vymazat interpunkci	75	97	87	80	80	50	67	98	98	96	71	100	97	83	90	98	98	98
bez diakritiky	43	100	87	38	38	24	24	77	80	89	43	87	95	38	69	78	82	82
jiná věta	7	100	87	8	8	4	4	64	64	16	7	69	56	8	13	8	10	10
překlad do AJ	0	91	89	0	0	0	0	56	56	18	0	83	55	0	2	3	4	4
lorem ipsum	0	100	87	0	0	0	0	57	57	11	0	91	55	0	3	4	5	5
Průměr	**46**	**96**	**87**	**46**	**46**	**37**	**41**	**76**	**78**	**58**	**45**	**89**	**77**	**47**	**52**	**52**	**54**	**54**

Tab. 7 – Podobnost věty (100 znaků) s jejími různými modifikacemi dle jednotlivých srovnávacích algoritmů

Z výsledků v Tab. 7 je patrné, že kromě slov jsou v předchozím měření zmíněné algoritmy citlivé i na interpunkční znaménka (dvě čárky a tečka) a slova bez diakritiky považují za jiná. Správně však v posledních třech případech odhalily, že věta byla nahrazena jinou (Pythagorova věta), přeložena do jiného jazyka (pouze pomocí překladače [w57]) a nahrazena nesmyslnými texty *lorem ipsum* (pomocí české verze generátoru [w54]).

V další fázi byl pro porovnávání použit již celý odstavec textu, který tvořily 2 věty o 45 slovech a 337 znacích.

Typ změny odstavce	BD	CLD	CML	CS	DS	ED	JS	J	JW	L	MC	ME	NW	OC	QGD	SW	SWG	-WA
prohodit slovo	96	100	100	95	95	79	91	86	91	95	96	100	96	95	98	94	95	62
umazat slovo	99	95	100	99	99	89	98	93	96	95	98	99	95	100	98	98	99	64
přidat slovo	99	97	100	99	99	90	98	93	96	97	96	100	97	100	98	98	99	66
prohodit větu	100	100	100	100	100	100	100	82	82	50	100	100	63	100	98	75	75	75
umazat větu	86	75	100	86	85	62	74	92	95	75	76	91	75	100	86	100	100	100
přidat větu	88	79	100	88	87	65	77	89	94	79	75	100	79	100	88	87	95	75
bez diakritiky	49	100	100	47	47	29	30	79	87	91	49	89	96	47	75	82	86	86
jiný odstavec	9	91	100	8	8	5	4	73	76	18	6	91	55	9	16	2	4	2
překlad do AJ	4	98	100	4	4	2	2	67	70	24	4	83	58	5	14	4	7	4
lorem ipsum	4	100	100	2	2	2	1	68	68	14	2	88	54	2	6	1	2	2
Průměr	**63**	**93**	**100**	**63**	**63**	**52**	**58**	**82**	**86**	**64**	**60**	**94**	**77**	**66**	**68**	**64**	**66**	**54**

Tab. 8 – Podobnost odstavce textu (337 znaků) s jeho různými modifikacemi dle jednotlivých algoritmů

Pro doplnění bylo ještě vyzkoušeno, jak si algoritmy poradí při porovnávání značně delšího slohového útvaru vypravování (povídky) o 14 248 znacích, přičemž jedinou změnou byla změna jmen všech postav (viz Tab. 9). V tomto případě již porovnání pomocí posledních dvou algoritmů SWG a SWGWA provedeno nebylo z důvodu v praxi nepoužitelné extrémní časové náročnosti. Porovnávají totiž z různých hledisek každý znak vzoru s každým znakem druhého textu a jejich výpočetní náročnost tak roste exponenciálně.

BD	CLD	CML	CS	DS	ED	JS	J	JW	L	MC	ME	NW	OC	QGD	SW
95	99	100	98	98	78	96	82	89	95	95	100	97	98	95	93

Tab. 9 – Podobnost slohového útvaru textu (14 248 znaků) s jeho upravenou verzí dle jednotlivých algoritmů

Z algoritmů hodnotících podobnost podle obsahu, rozpoznal vysokou podobnost obou textů např. CS, který se také používá při odhalování plagiátů (např. viz [61]).

Časovou náročnost použitých algoritmů shrnuje Tab. 10 a to zpětně za všechna provedená měření. Časy v jednotlivých řádcích jsou průměrnou hodnotou z časů naměřených v jednotlivých pokusech pro dané textové útvary, přičemž ty byly vždy mediánem z většího počtu opakování téhož měření (pro eliminaci náhodných výkonnostních výkyvů během měření). V posledním řádku je v některých případech použito znaku „k" (kilo), jež značí, že daná hodnota je ještě 1 000x větší (např. 190k = 190 000).

Typ a délka textu	BD	CLD	CML	CS	DS	ED	JS	J	JW	L	MC	ME	NW	OC	QGD	SW	SWG	-WA
slovo (10)	10	6	6	10	10	9	10	10	10	14	9	125	13	10	36	15	126	126
dvě slova (16)	12	7	6	12	12	11	12	13	13	23	11	293	22	12	66	27	373	294
věta (100)	96	25	22	97	98	95	97	345	322	1 520	62	9 724	782	43	1 116	958	34 395	34 625
odstavec (337)	535	33	32	517	511	472	472	2 203	1 713	6 676	131	100 080	5 165	197	10 838	7 265	961 045	513 997
povídka (14 248)	190k	210	216	132k	140k	188k	132k	2 056k	2 078k	11 414k	93k	189 625k	10 088k	133k	4 112k	12 466k	-	-

Tab. 10 – Průměry ze středních hodnot časů (v μs) potřebných na výpočty podobnosti různých typů textových útvarů dle různých srovnávacích algoritmů

Jak je patrné z Tab. 10, některé algoritmy není vhodné používat pro porovnávání delších (více větných) textů nejen z důvodů nevhodného výsledku porovnání, ale také kvůli výpočetnímu času, jež na tento úkon potřebují. S tím je třeba počítat a pro různé případy zvolit algoritmus vhodný z obou těchto hledisek.

4.5 Náhodné barvy [◎ III.3.A]

Testové otázky mohou obsahovat prvky, jež svůj obsah náhodně generují, či vybírají z více variant. Jedním z hlavních účelů otázek je totiž ověřit studentovu znalost určité problematiky, čili na základě zadání provést správné řešení. Při opakovaném použití otázky může v některých případech ovšem převážit mechanická interpretace odpovědi na základě některého z vizuálních prvků otázky. Ta může být vztažena jak k formulaci jejího znění tak i vzhledu (rozložení prvků, jejich tvarů, barev apod.). Veškeré tyto vlastnosti otázky mohou být generovány náhodně v takovém rozsahu, aby došlo k eliminaci tohoto typu propojení a rozhodujícím faktorem pro správné řešení bylo vždy pouze aktuální zadání. Z tohoto důvodu byla mezi náhodné prvky zařazena i podpora náhodných barev.

Výběr náhodné barvy zde však není až zas tak náhodný, resp. negenerují se zcela náhodné barvy, ale náhodně se vybírá z jejich seznamu. Ten je zapsán jako jeden textový řetězec s čárkami oddělenými hodnotami barev, zapsaných v klasickém formátu, jež se provádí v hexadecimálním vyjádření jejich tří (RGB) popřípadě čtyř složek (ARGB, kde A je alfa neboli průhlednost, přičemž 0 je zcela průhledná a 255 neboli „FF" je zcela neprůhledná, což je i výchozí hodnota, není-li tato složka uvedena), podobně jako v HTML a CSS.

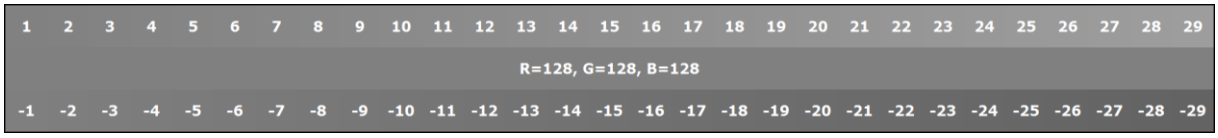
Těchto seznamů barev v otázce může být i více a lze určit, ze kterého z nich se má vybírat. To se hodí v případech, kdy je třeba zajistit dvě kontrastní barvy, např. barvu textu a barvu pozadí, na kterém bude text napsán. Dále lze každý náhodný výběr z každého seznamu barev identifikovat číslem. To je užitečné jednak v případě, kdy je třeba tutéž barvu použít i někde jinde v otázce a také pro zajištění toho, aby v příštím výběru z téhož seznamu nebyla zvolena stejná barva. Posledním nepovinným parametrem je číslo určující vzdálenost od této barvy na barevné paletě. To je důležité, především pokud jsou cíle pro položky určeny nikoli polohou ale svou barvou (viz str. 45). Ta je pak vždy vybírána jako co nejvíce podobná té základní, přičemž stupeň její vzdálenosti od barvy výchozí na barevné paletě určuje právě tento parametr. Díky tomu jsou takto odstupňované barvy pouhým okem nerozlišitelné, avšak ani jejich strojová identifikace by zkoušenému neumožnila odhalit, na kterou z nich patří která položka.

Parametry pro výběr barvy jsou tedy tři, přičemž ten poslední je nepovinný. Barva se pak nastavuje jejím zapsáním v následujícím formátu „P-1-2-3", kde jsou její jednotlivé části odděleny pomlčkou. První znak „P" sděluje, že barva není zapsaná v klasickém formátu (to by prvním znakem byla číslice nebo znaky A, B, C, D, E či F), ale bude se vybírat z palety barev. První číslo, v tomto případě 1, tedy určuje identifikátor palety, ze které se barva vybírá. Druhé číslo (2) identifikuje výběr jako druhý, tzn., je-li barva s tímto číslem nastavována poprvé, náhodně se vybere z palety barva, která dosud nebyla zahrnuta do výběru s jiným číslem (např. 1). Pokud již byl výběr s tímto číslem (2) proveden, použije se pro toto číslo dříve zvolená barva. Třetí číslo (3) určuje, že se k této barvě vyhledá barva o 3 stupně rozdílná (viz násl. podkap. 4.5.1). Kdyby tento třetí parametr uveden nebyl (zápis by vypadal takto: „P-1-2"), byla by použita přímo tato barva (tj. druhého výběru z první palety).

4.5.1 Odstupňování barev

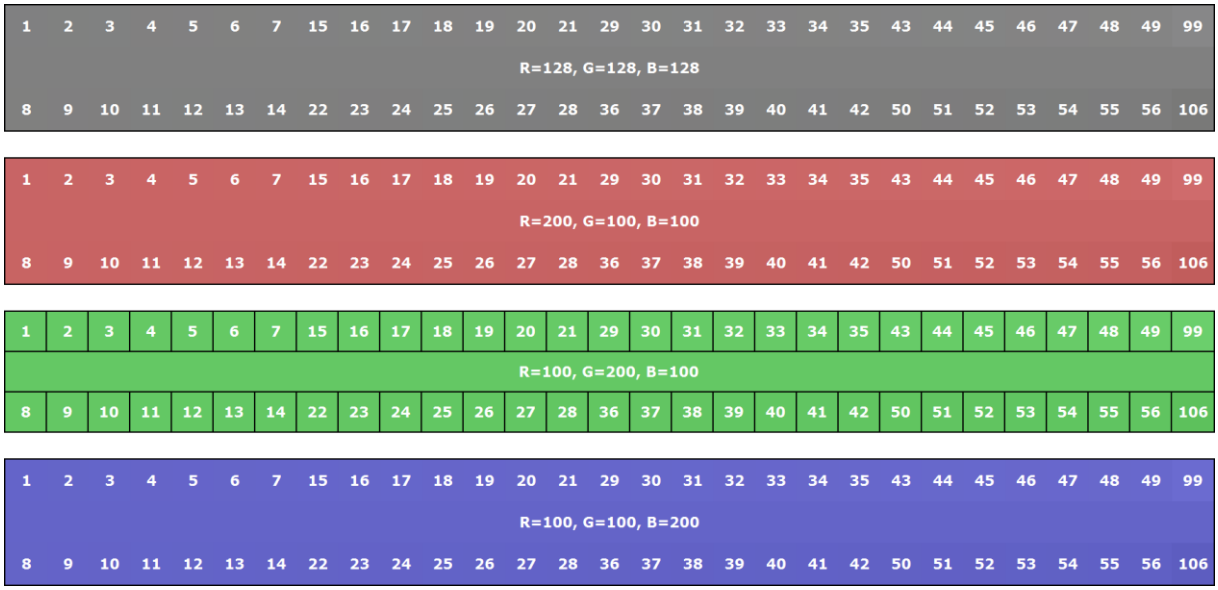
Pro určení odstupňované barvy co nejpodobnější té základní byl vytvořen vlastní postup jejího výpočtu. Na barevné paletě je přitom postupováno tak, aby se měnil pouze jas a sytost této barvy nikoli odstín. Jednotlivé složky RGB tak zůstávají téměř ve stejném poměru jako u výchozí barvy, tj. jsou zvyšovány či snižovány téměř o stejnou hodnotu.

Přechod o jeden barevný stupeň ve všech třech složkách (např. z „100,100,100" na „101,101,101") je pro lidské oko nerozeznatelný, i když se obě barvy budou nacházet bezprostředně vedle sebe. Při této stupnici by ovšem bylo k dispozici pouze 255 stupňů barev a to pouze u odstínů šedé[39], přičemž velmi brzo by se dosáhlo odstínů zřetelně rozdílných od barvy výchozí (viz Obr. 21).



Obr. 21 – Ukázka přímého odstupňování výchozí barvy (uprostřed) o +29 (nahoře) a -29 (dole)

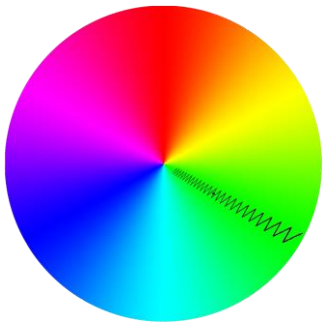
Aby se barvy nevzdalovaly od té základní příliš rychle a byla tak vyšší rezerva pro případné využití většího počtu barev, byl přechod mezi jednotlivými stupni rozmělněn tak, aby se využily veškeré jejich kombinace, které mezi těmito stupni jsou (např. mezi „20,20,20" a „21,21,21" jsou barvy „20,20,21", „20,21,20", „21,20,20", „20,21,21", „21,20,21", „21,21,20"). Došlo tak sice k vychýlení barvy z jejího odstínu, avšak vždy maximálně o jeden stupeň, čili pro člověka nerozeznatelně. Zároveň tak z původního jednoho stupně vzniklo 7 (tj. 2^3 - 1). Při tomto způsobu odstupňování již nedochází k rozdílům od základové barvy tak rychle a lze proto použít více barev (viz Obr. 22).



Obr. 22 – Ukázka popisovaného odstupňování barev pro čtyři výchozí barvy (vždy uprostřed)

[39] u jiných barev, než jsou stupně šedi, mají jednotlivé složky RGB různé hodnoty, díky čemuž by při jejich odstupňování některá z nich mohla dosáhnout dolního (0) či horního (255) limitu dříve než ostatní a odstín barvy by se tak mohl změnit

Toto odstupňování používá všechny kombinace tří barevných složek mezi jednotlivými stupni téhož odstínu výchozí barvy. Jednotlivé kombinace jsou přitom určeny pomocí binárního třístavového počítadla. Obr. 22 ukazuje čtyři různé palety tímto způsobem odstupňovaných barev a je patrné, že na rozdíl od předchozího případu (Obr. 21) jsou rozdíly mezi jednotlivými barvami výrazně hůře rozeznatelné, ač poslední dvě byly posunuty o 50 stupňů, oproti předchozí. U zelené barvy bylo zároveň přidáno orámování jednotlivých stupňů, které odhalení různosti barev ještě více komplikuje. V tomto obrázku, stejně jako v předchozím, je jednotlivé odstupňování rozděleno do tří řádků, a sice kladné přírůstky nahoře, základní barva uprostřed a záporné přírůstky dole. Čísla v horní a dolní řadě pak každému stupni udávají jeho posun od výchozí barvy. Z hodnot je patrné, že se přírůstky kladné a záporné pravidelně střídají, a to z důvodu, aby byla odstupňovaná barva co nejdéle co nejpodobnější té výchozí.



Obr. 23 – Ilustrační ukázka popisovaného odstupňování podobných barev od výchozího bodu na paletě

Za účelem efektivní práce s těmito barevnými stupni bylo zapotřebí stanovit jednotný vztah mezi výchozí barvou a odstupňovanou barvou pouze na základě čísla stupně posunu. Pro jednotlivé barevné složky byly sestaveny následující tři rovnice, kde **r**, **g**, **b** jsou složky odstupňované barvy, **R**, **G**, **B** jsou složky původní barvy a **p** je stupeň posunu.

$$r = R + \ sgn\left(\left\lfloor\frac{6+p}{7}\right\rfloor \ mod\ 2 - 0{,}5\right)\cdot\left\{\left\lfloor\frac{13+p}{14}\right\rfloor - \left(\left\lfloor\frac{(p-1)\ mod\ 7+1}{4}\right\rfloor + \ 1\right)\ mod\ 2\right\}$$

$$g = G + \ sgn\left(\left\lfloor\frac{6+p}{7}\right\rfloor \ mod\ 2 - 0{,}5\right)\cdot\left\{\left\lfloor\frac{13+p}{14}\right\rfloor - \left(\left\lfloor\frac{(p-1)\ mod\ 7+1}{2}\right\rfloor \ mod\ 2 + 1\right)\ mod\ 2\right\}$$

$$b = B + \ sgn\left(\left\lfloor\frac{6+p}{7}\right\rfloor \ mod\ 2 - 0{,}5\right)\cdot\left\{\left\lfloor\frac{13+p}{14}\right\rfloor - \ (\,[\,(p-1)\ mod\ 7+1]\ mod\ 2+1)\ mod\ 2\right\}$$

Vzorec 11 – Výpočet jednotlivých barevných složek (r,g,b) na základě výchozí barvy (R,G,B) a čísla posunu (p)

Při výpočtu je použito operátoru pro celočíselné dělení (`div`), jež v uvedených rovnicích zastupuje funkce $\lfloor x \rfloor$, který vrací celou část čísla **x** (např. $\lfloor 1{,}85 \rfloor = 1$ nebo $\lfloor -1{,}53 \rfloor = -1$). Operátor `mod` (modulo) pak vrací zbytek po celočíselném dělení (např. `7 mod 3 = 1`) a funkce `sgn(x)` vrací znaménko hodnoty x (tj. -1 pro x < 0, 0 pro x = 0 a 1 pro x > 0). Tyto rovnice byly následně zjednodušeny do jednotné funkce $f(x,y)$ (viz Vzorec 12), jejímž parametrem **x** je vždy **p** (posun) a parametr **y** se různí pro jednotlivé barevné složky.

$$f(x,y) = sgn\left(\left\lfloor\frac{6+x}{7}\right\rfloor \ mod\ 2 - 0{,}5\right)\cdot\left\{\left\lfloor\frac{13+x}{14}\right\rfloor - \ (y+1)\ mod\ 2\right\}$$

Vzorec 12 – Obecná funkce pro výpočet jednotlivých barevných složek odstupňované barvy

Následující rovnice (viz Vzorec 13) ukazují výpočet pro jednotlivé barevné složky s využitím předchozí funkce (viz Vzorec 12) a pomocného parametru **z**.

$$r = R + f\left(p, \left\lfloor \frac{z}{4} \right\rfloor\right), \quad g = G + f\left(p, \left\lfloor \frac{z}{2} \right\rfloor \bmod 2\right), \quad b = B + f(p, z \bmod 2), \quad kde\ z = (p-1) \bmod 7 + 1$$

Vzorec 13 – Výpočet jednotlivých barevných složek pomocí funkce **f** (Vzorec 12) a parametru **z**

I při tomto postupu ovšem stále zůstává problém s hraničními hodnotami. Odstupňování totiž nemůže pokračovat do nekonečna, ale hodnoty barevných složek RGB jsou striktně omezeny na hodnotu typu byte, tj. 2^8, čili 0-255. Nelze tedy kteroukoli z nich nastavit na vyšší hodnotu než 255, ani jít do hodnot záporných. S tímto omezením uvedené rovnice nepočítají, a proto bylo řešení tohoto mezního problému zpracováno programově.

V případě, že je výsledkem výpočtu některé z barevných složek hodnota mimo limit, je tato hodnota vrácena na hraniční hodnotu (např. 256 na 255, či -1 na 0). V tomto případě však může být výsledkem barva, která již byla použita dříve s jinou hodnotou posunu. Tuto skutečnost však lze snadno detekovat, neboť každá již použitá barva se během vytváření otázky zapisuje do dočasného seznamu, v němž se případné duplicity ihned odhalí. Dojde-li k tomuto případu, opakuje se výpočet pro další barvu v pořadí (např. je-li kolizní stupeň 10 se stupněm 9, zkusí se pro něj vypočítat stupeň 11, nevyhovuje-li ani ten, tak 12 atd.).

Tím ovšem může vzniknout problém, kdy výsledná barva nemusí vždy odpovídat výsledku rovnic (viz Vzorec 13) podle zadaného vstupního parametru **p**. Vrací-li např. stupeň 10 barvu pro stupeň 13, pak až dojde na řadu 13, vrátí tutéž barvu. Kontrola duplicit by sice vše odhalila a pro 13 by se zkusila vypočítat hodnota 14, ale dochází tak ke zbytečným vícenásobným výpočtům téhož, což má za následek zpomalení celého procesu. Řešení vychází z předpokladu, že barvy v otázce budou načítány postupně, resp. že je-li požadován např. stupeň 20, byly nebo budou požadovány i všechny stupně předchozí, tj. 1-19. Vytvořené barvy se přitom, jak již bylo uvedeno, zapisují do seznamu. Ten kromě vypočtených barev obsahuje i vstupní hodnotu posunu a zároveň i tu výstupní, resp. tu, pro kterou barvě odpovídají rovnice výpočtu, v případě, že je rozdílná od té vstupní. Eviduje se tedy jak stupeň poslední žádané barvy, tak i poslední „vydaný". Díky tomu lze ve výpočtech, včetně opravných posunů, vždy navázat přesně za do té doby posledním, aniž by se cokoli počítalo vícekrát než jednou. Jsou-li požadovány stupně na přeskáčku (např. nejdříve 5 a až potom 4), jsou doposud nepožadované mezistupně dopočítány do seznamu, neboť se předpokládá, že budou stejně následně vyžádány.

4.6 Míchání otázek [◎ III.3.B]

(publikováno v [ap-9])

Jednou ze stěžejních součástí testování je schopnost sestavovat vždy originální test s náhodně vybranými otázkami. Počet otázek, jež mají tvořit jednotlivé testy, obvykle bývá nižší, než je počet otázek, které jsou pro daný test k dispozici. Z takovéto databáze se pak náhodně vybírají otázky pro každý spuštěný test. Může jít o výběr např. 5 z 6, ale také z 500. Tento výběr ovšem nemusí být jen zcela náhodný, naopak v některých případech je žádoucí umožnit náhodnost výběru otázek ovlivnit tak, aby lépe plnila účel, pro který je test používán. Mezi účely vyžadující různé typy míchání při tom patří např. testování výukové, drilovací, opakovací či hodnocené závěrečné.

Zcela náhodné míchání je zapotřebí při závěrečném hodnoceném testování. U výukového testování by zase měly být upřednostňovány nové, zkoušeným dosud neřešené otázky. U testování drilovacího je naopak třeba, aby se již probrané otázky opakovaly ve specifických intervalech odpovídajících individuálním křivkám zapomínání jednotlivých zkoušených. Opakovací testování by pak při výběru mělo vhodným způsobem brát v potaz míru správnosti předchozích řešení každé otázky a zároveň dobu od jejího zodpovězení. A právě tímto posledním případem se budeme v této kapitole zabývat.

Opakovací testy kombinují didaktické testy ověřující a průběžné [44 str. 16]. Měly by rychle a objektivně prověřit znalost zkoušených (studentů) [40 str. 23]. Dovolují-li to podmínky, měly by být zadávány v pravidelných periodách, optimálně při každé hodině či cvičení z daného předmětu (tzv. „pětiminutovky"). Jejich obsahem by měla být předně látka probraná v předchozím cvičení, ovšem alespoň částečné zastoupení by mělo patřit i látce předešlé. To platí samozřejmě jen v některých předmětech, kde slouží tyto testy nejen k prověření aktuálního stavu znalostí studentů, ale i jako motivace k průběžnému studiu a přípravě na výuku.

4.6.1 Inteligentní výběr testových otázek

Pro optimální výběr otázek do periodicky zadávaného opakovacího testu je zapotřebí zohlednit několik faktorů. Ty by sice měly ovlivňovat pravděpodobnost výběru, avšak nikdy z něj žádnou otázku zcela nevyřazovat. Mezi tyto faktory patří počet řešení otázky v minulosti, doba uplynulá od těchto řešení a dosažený výsledek. Jejich vhodné zkombinování pro optimální výběr otázek bude předmětem této části.

Složka zapomínání

Složka zapomínání by měla zohledňovat dobu, jež uplynula od předchozího řešení dané otázky. Předpokladem je, aby nedávno zadaná otázka měla vyšší pravděpodobnost, že do nového výběru nebude zařazena, než ta, která byla řešena dříve.

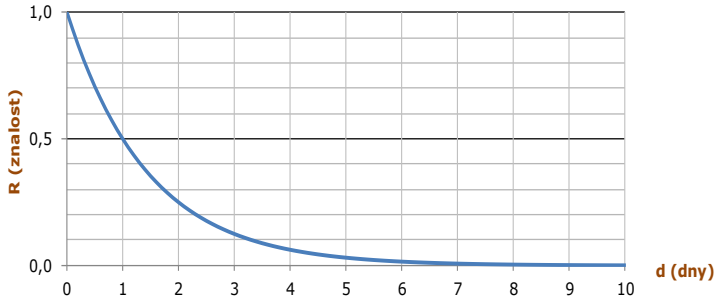
Pro stanovení tohoto časového vlivu se přímo nabízela tzv. křivka zapomínání, jejíž tvar zhruba odpovídá předpokladům využití a zároveň má i své vědecké opodstatnění z oblasti psychologie [62 str. 280]. Autorem této teorie je H. Ebbinghaus [21 stránky 94-96], který ve své práci [22] uvádí, že nejrychleji zapomínání probíhá v prvních hodinách od naučení se určitého faktu, přičemž do druhého dne jde až o 80% znalostí [63 str. 201].

Matematického modelu křivky zapomínání je zároveň využíváno ve výukových softwarech, kterými se uživatelé učí drilovací metodou, tj. opakováním faktů s vhodně zvolenou prodlevou mezi jednotlivými opakováními (*spaced repetition*, viz kap. 3.1.4, str. 22).

Z této teorie vychází i metoda Re-wise používaná např. firmou LANGMaster. Tato křivka předpokládá, že nově naučená fakta jsou zapomínána geometrickou řadou, oproti původnímu přístupu však pouze polovina z aktuálně pamatovaných faktů denně. Pomineme-li tedy efekt opakování a individuálnost jednotlivých osob a jejich vztahu k různým faktům, pak tuto křivku zapomínání vyjadřuje Vzorec 14 a znázorňuje Graf 17. [64 stránky 20-21]

$$R = \frac{1}{2^d}$$

Vzorec 14 – Rovnice křivky zapomínání dle metody Re-wise



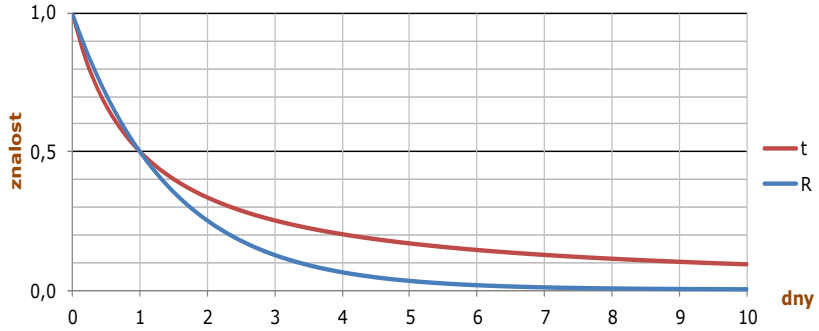
Graf 17 – Křivka zapomínání dle metody Re-wise

Tato verze křivky zapomínání ovšem velmi rychle klesá a již po čtyřech dnech odhaduje zapamatované znalosti pouze na 6,25%, po 7 dnech pak na pouhých 0,78%. Při takto rychlém vytěsnění časového efektu je tato křivka pro řešený model výběru otázek v podstatě nepoužitelná. Lze sice např. exponent jmenovatele **d**, vyjadřující počet dnů od naučení se sledovaného faktu, vydělit určitou hodnotou a zmírnit tak počáteční strmý pokles, avšak i po této úpravě je křivka stále příliš strmá pro dosažení kýženého efektu.

Vzhledem k tomu, že hodnocený opakovací test nemá za cíl zkoušené učit, ale naopak periodicky ověřovat jejich znalosti, není možné složku zapomínání použít v některém z výše uvedených tvarů. Proto navržený matematický model ovlivňující výběr otázek nevychází z křivek zapomínání [65], ale snaží se více reflektovat potřebu obměny otázek a pokrytí celého spektra testovaného učiva. Místo geometrické řady byla tedy časová proměnná přesunuta pouze do jmenovatele (viz Vzorec 15), což pro složku zapomínání **t** představuje zřetelně plošší křivku (pomaleji klesající) než v předchozím případě (viz Graf 18).

$$\mathrm{t} = \frac{1}{\mathrm{d}+1}$$

Vzorec 15 – Základní verze složky zapomínání



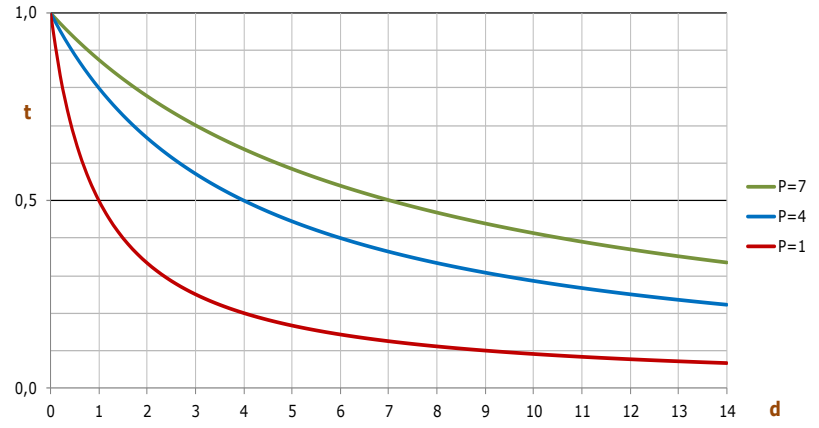
Graf 18 – Porovnání křivky zapomínání **R** (Vzorec 14) a navrhované křivky složky zapomínání **t** (Vzorec 15)

Složka zapomínání označená **t** tedy vyjadřuje míru zapomínání v čase ve zprava uzavřeném intervalu od 0 (žádná znalost) do 1 (plná znalost). Vzorec 15 byl následně ještě doplněn o parametr **p**, který umožňuje rychlost klesání křivky přizpůsobit periodě opakování testu (viz Vzorec 16).

$$t = \frac{p}{d + p}$$

Vzorec 16 – Složka zapomínání se započtením periody opakování

Parametr **p** je průměrná perioda mezi jednotlivými testy vyjádřena ve dnech (např. 7 pro „jednou týdně"). Doslovný význam parametru **p** je: *„Po kolika dnech má mít minule zcela správně zodpovězená otázka 50% pravděpodobnost zařazení do výběru oproti otázkám novým."* Graf 19 ukazuje průběhy jednotlivých křivek **t** pro tři různé hodnoty parametru **p**.



Graf 19 – Průběh křivek **t** pro různé hodnoty parametru **p** jak ukazuje Vzorec 16

Znalostní složka

Dalším faktorem, který je nezbytné při výběru otázek zohlednit je jejich předchozí výsledek. Na rozdíl od učení se prostým faktům, kde je výsledek obvykle dvoustavový (věděl – nevěděl), je zde výsledkem řešení jednotlivých otázek reálná hodnota z fuzzy stupnice v intervalu $\langle 0, 1 \rangle$.

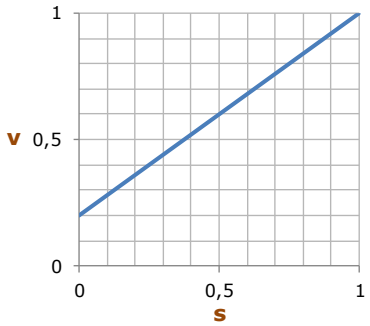
K výslednému skóre je však zapotřebí ještě připočítat „znalost", kterou dotyčný získal díky řešení této otázky. Po testu totiž může následovat podrobný rozbor řešení jednotlivých otázek, v němž se student vždy dozví, co udělal chybně a jak a proč to mělo být správně. Díky tomu znalost zkoušeného dané otázky, ač ji třeba zodpověděl zcela špatně (na 0%) vzroste nad nulovou úroveň. Pro vyjádření tohoto „poučení se z vlastních chyb" je do výpočtu zařazen parametr **n**, jež

vyjadřuje tuto hodnotu (viz Vzorec 17, odvozený z klasického směrnicového tvaru rovnice přímky v rovině [49 str. 81]).

$$v = (1 - n) \cdot s + n$$

Vzorec 17 – Předpokládaná znalost otázky se započtením poučení se z předchozí chyby

Vzorec 17 určuje míru předpokládané znalosti tématu otázky **v** po jejím zodpovězení a zobrazení rozboru (zpětné vazby). Proměnná **s** je zde skóre otázky získané za danou otázku v čase před **d** dny. Hodnota **v** je tedy navýšena o část **n** tak, že výsledek je opět v rozsahu $\langle 0, 1 \rangle$. Graf 20 ukazuje vztah mezi těmito dvěma hodnotami (**s** a **v**) pro hodnotu parametru **n** = 0,2.



Graf 20 – Vztah hodnot **v** a **s** pro **n** = 0,2

Váhy pro výběr

Předchozí dvě vypočtené hodnoty, tj. složka zapomínání **t** (Vzorec 16) a znalostní složka **v** (Vzorec 17) jsou následně zkombinovány jejich součinem do hodnoty **z** (viz Vzorec 18).

$$z = t \cdot v$$

Vzorec 18 – Aktuální předpokládaná znalost tématu otázky dříve 1x řešené (obecně)

Výsledkem **z** je aktuální předpokládaná znalost tématu otázky, vzhledem k jejímu dřívějšímu řešení. V případě, že otázka byla řešena vícekrát (**c**-krát), lze obecně tento vzorec pro **i**-té řešení (**i** ∈ {1, 2, …, **c**}) rozepsat následovně (viz Vzorec 19).

$$z_i = \frac{p}{d_i + p} \cdot [s_i \cdot (1 - n) + n]$$

Vzorec 19 – Aktuální předpokládaná znalost tématu otázky dříve 1x řešené (podrobně)

Výpočet celkové hodnoty aktuální znalosti otázky **Z** při dvou předchozích řešení (**c** = 2), pak, pro zachování výsledku v intervalu $\langle 0, 1 \rangle$, lze realizovat stejným způsobem, jako probíhá slučování pravděpodobností dvou nezávislých jevů (viz Vzorec 20).

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Vzorec 20 – Sloučení pravděpodobností dvou nezávislých jevů **A** a **B** [66 str. 6]

Výpočet **Z** by tedy pro dvě předchozí řešení (**c** = 2) konkrétně vypadal následovně (viz Vzorec 21).

$$Z = z_1 + z_2 - z_1 \cdot z_2$$

Vzorec 21 – Výpočet míry aktuální předpokládané znalosti **Z** otázky pro její dvě dřívější řešení (**c** = 2) $\mathbf{z_1}$ a $\mathbf{z_2}$

Při vícenásobném řešení (**c** > 2) vzorec pro slučování pravděpodobností **n** jevů vypadá následovně (viz Vzorec 22).

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(A_i \cap A_j) + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} P(A_i \cap A_j \cap A_k) + \cdots + (-1)^{n+1} P\left(\bigcap_{i=1}^{n} A_i\right)$$

Vzorec 22 – Sloučení pravděpodobností **n** (**n** > 2) nezávislých jevů $\mathbf{A_i}$ [66 str. 6]

Kromě přímého výpočtu však lze použít také výpočetně méně náročné postupné slučování pravděpodobností jednotlivých jevů, jak pro tři jevy ukazuje Vzorec 23.

$$P(A \cup B \cup C) = P(P(A \cup B) \cup C)$$

Vzorec 23 – Postup sloučení pravděpodobností tří nezávislých jevů **A**, **B** a **C** [66 str. 6]

Nejprve se tedy sloučí pravděpodobnosti prvních dvou jevů a až k tomuto výsledku se připočítá stejným způsobem ten třetí. Pro výpočet hodnoty **Z** lze tento postup zapsat do funkce pomocí rekurze (viz Vzorec 24), kde je její vstupní parametr **x** roven **c**, tj. celkovému počtu řešení zkoumané otázky daným uživatelem. Pro tento rekurzivní výpočet definujeme $f(0) = 0$, tzn. v případě, že uživatel tuto otázku ještě nikdy neřešil (**c** = 0), bude výsledkem funkce a tedy i **Z** rovno 0.

$$f(x) = \mathrm{z_x} + f(x-1) - \mathrm{z_x} \cdot f(x-1) = \mathrm{z_x} + (1 - \mathrm{z_x}) \cdot f(x-1), \quad f(0) = 0$$

Vzorec 24 – Funkce pro rekurzivní výpočet hodnoty **Z**

Celková aktuální znalost otázky **Z**, přičemž tuto otázku uživatel řešil **c**-krát, je tedy určena výše popsanou funkcí, kde jejím parametrem bude **c** (viz Vzorec 25).

$$Z = f(c)$$

Vzorec 25 – Výpočet **Z** pomoci funkce **f** při počtu řešení otázky **c**

Váha pro zařazení otázky do výběru, kterou označíme **P**, je pak doplňkovou hodnotou předpokládané znalosti otázky **Z** (viz Vzorec 26).

$$\mathrm{P} = 1 - \mathrm{Z}$$

Vzorec 26 – Váha pro zařazení otázky do výběru

Hodnota **P** tedy vyjadřuje váhu pro zařazení otázky do výběru, přičemž váhu 1 mají otázky dosud neřešené, ty ostatní ji mají nižší. Vzhledem k tomu, že jeden zkoušený nebude psát nikdy více testů v témže čase, bude složka zapomínání **t** vždy menší než 1, takže v praxi **P** nebude nikdy nulové (**t**<1 ⇒ **Z**<1 ⇒ **P**>0). Díky tomu nebude žádná otázka z výběru nikdy zcela vyřazena (viz Příloha 11).

Tab. 11 ukazuje postup výpočtu hodnoty **P** pro otázku, jež byla zkoušeným v minulosti již 4x řešena v uvedených datech a s uvedenými výsledky **s**. Hodnota **Z** je zde počítána kumulativně, tzn. postupným slučováním aktuálního $\mathbf{z_i}$ s do té doby již vypočteným **Z**. Konečné **Z** je pak v posledním řádku tabulky a je z něho přímo odvozena hodnota **P**.

Datum	d	t	s	v	z	Z (kumulativně)
25.01.2012 10:30	8,063	0,4819	0,8	0,84	0,40482	0,40482
19.01.2012 12:00	14,000	0,3488	0,5	0,60	0,20930	0,52939
12.01.2012 11:05	21,038	0,2628	0,3	0,44	0,11563	0,58381
25.09.2011 09:00	130,125	0,0545	0,4	0,52	0,02834	**0,59560**
Pro aktuální datum a čas 2.2.2012 12:00; p = 7,5; n = 0,2						**P = 0,40440**

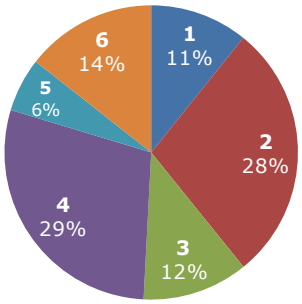
Tab. 11 – Ukázka výpočtu hodnoty P (viz Vzorec 26) pro jednu otázku

Výběr otázek podle vah

Sestavení testu probíhá náhodným výběrem ze všech dostupných otázek, přičemž pravděpodobnost výběru kterékoli z nich je přímo odvozena z ohodnocení **P**. Týž algoritmus výběru se používá v genetických algoritmech pro tzv. *selekci individuí* a je znám jako ruletový mechanizmus („roulette wheel selection" [67 stránky 28-29]). Při tomto postupu je každé otázce přiřazena kruhová výseč, jejíž plocha je přímo úměrná ohodnocení otázky **P** [68 stránky 22-23]. Poté se pomyslné kolo rulety roztočí a je vybrána ta otázka, na jejímž úseku se „kulička rulety" zastaví [69 str. 6]. Zvolená metoda selekce je pro tuto úlohu vhodná, jelikož zde nehrozí limitace jejími nedostatky, jako je např. požadavek na nezápornost řešení [70 str. 23].

Otázka	P	Podíl $\frac{P_i}{\sum P}$	Kumulativně	
			P	Podíl
1	0,375	0,108	0,375	0,108
2	0,984	0,284	1,359	0,392
3	0,404	0,117	1,763	0,508
4	1,000	0,288	2,763	0,797
5	0,206	0,059	2,969	0,856
6	0,500	0,144	3,469	1,000
∑	3,469	1,000		

Tab. 12 – Ukázka výpočtu velikosti výseče rulety



Graf 21 – Ukázka rulety jako koláčového grafu

Výše popsané „točení ruletou" lze programově realizovat mnoha způsoby. Jedním z nich je skutečně převést ohodnocení **P** každé otázky na jednotnou míru, a to vydělením každé z nich jejich celkovým součtem (viz 3. sloupec „Podíl" v Tab. 12). Výsledek pak v součtu dává hodnotu 1, přičemž generátory pseudonáhodných hodnot nejčastěji generují desetinnou hodnotu právě v rozsahu $\langle 0, 1)$ z rovnoměrného rozdělení. Kumulace těchto podílů (viz poslední sloupec v Tab. 12) pak vždy udává horní hranici intervalu, do něhož spadá-li vygenerované náhodné číslo, označuje jím otázku zařazenou do výběru (např. náhodná hodnota 0,637 by dle Tab. 12 označovala otázku č. 4). Na rozdíl od výběru individuí pro křížení v genetických algoritmech se ovšem jedna otázka v tomtéž testu nesmí vyskytovat více než jednou. Proto je pro výběr další otázky třeba celý postup znovu opakovat s vynecháním již zvolené otázky.

Méně náročnou variantou je vynechat přepočet ohodnocení **P** na jejich podíl a použít rovnou kumulace těchto hodnot (viz předposlední sloupec v Tab. 12). Pseudonáhodně vygenerovanou hodnotu v rozsahu $\langle 0, 1)$ z rovnoměrného rozdělení pak stačí vynásobit **∑P** a výsledek již přímo označuje interval vymezený těmito kumulativními hodnotami. Tím se, mimo další režie při výpočtu, ušetří počet $\frac{q(2Q-q+1)}{2}$ (kde **Q** je celkový počet otázek v databázi a **q** je počet otázek vybíraných pro test)

výpočetně nejnáročnějších operací dělení [71 str. 23], které jsou nahrazeny pouze **q** počtem operací násobení. (Např. $0{,}637 \cdot 3{,}469 \doteq 2{,}21$, čili i tímto postupem by byla vybrána otázka č. 4.)

4.6.2 Skupiny otázek

Výše popsaný postup se týká otázek, jež mají při prvním testu zkoušeného všechny stejnou pravděpodobnost na výběr (**P** = 1). V testech však bývá žádoucí, aby v nich určité skupiny otázek byly zastoupeny předem stanoveným počtem (např. 4 otázky z tématu minulé hodiny a 1 z čehokoli předešlého). Za tímto účelem některé testovací aplikace podporují dělení otázek do skupin, u kterých se zároveň definuje počet otázek, určující maximální zastoupení skupiny v testu.

V univerzálním testovacím prostředí je členění těchto skupin hierarchické, přičemž hloubka úrovní této hierarchie není nijak omezena. Výběr otázek do testu se pak provádí pomocí algoritmu prohledávání do hloubky, kdy je z každé již dále nedělené skupiny vybrán nastavený počet otázek pomocí zde popsaného postupu. Tato skupina je pak zrušena a z ní vybrané otázky přesunuty do vyšší složky. Tímto způsobem se postupuje až do nejvyšší kořenové složky, z níž jsou zvoleny otázky pro samotný test.

4.6.3 Shrnutí

Výběr otázek pro sestavování periodicky zadávaných opakovacích testů může probíhat zcela náhodně, ale také může mít důmyslně propracovaný postup, který zvýší jejich použitelnost. Jeden z takovýchto postupů byl představen i v této kapitole. I když se jedná o model použitelný pouze pro specifický druh testu a pouze pro některé učební předměty, může být inspirací nejen pro implementace budoucích testovacích aplikací, ale i pro další rozvoj teorií výběrových funkcí, ať již v tomto či zcela jiném odvětví.

4.7 Charakteristické textové řetězce [◎ III.3.A]

Interní míchání otázek podporuje řadu náhodných prvků (viz kap. 4.1.6, str. 48). Podobně jako při výběru otázek do testu (viz kap. 4.3, str. 70) je i v tomto případě žádoucí, aby interní mix nebyl vždy zcela náhodný, ale podporoval schopnost maximální možné obměny otázky při jejím opakovaném použití pro téhož zkoušeného. Díky tomu by měly být pro výběr každé z náhodných prvků otázky upřednostněny ty varianty, popř. jejich kombinace, které by byly co nejvíce odlišné od těch dříve použitých.

Určení míry odlišnosti jednotlivých kombinací náhodných prvků by přitom nemělo být výpočetně náročné (jako je např. vícerozměrná shluková analýza, viz [72]), mělo by být snadno databázově uchovatelné a struktura takovéhoto zápisu rozšiřitelná pro případný budoucí vývoj otázky.

4.7.1 Princip řetězcového porovnávání

Zvolený způsob řešení bude nejprve vysvětlen na názorněji prezentovatelné problematice porovnávání osob. Následující tabulka (viz Tab. 13) obsahuje pět vlastností, které budou u osob porovnávány, a popis jejich typů a rozsahů (tzv. metadata).

Vlastnost	Index	Typ	Minimum	Maximum	Rozsah	Jednotky	Váha
Výška	1	kardinální	40	220	181	cm	1,5
Váha	2	kardinální	1	256	256	kg	0,9
Věk	3	kardinální	0	127	128	roky	1,0
Pohlaví	4	nominální	1	2	2	-	10,0
Barva očí	5	nominální	1	6	6	-	2,0

Tab. 13 – Metadata vlastností osob potřebná pro jejich porovnávání

Parametr rozsah určuje, kolika možných hodnot může daná vlastnost nabývat a je vypočten následujícím způsobem (viz Vzorec 27).

$$rozsah_i = max_i - min_i + 1$$

Vzorec 27 – Výpočet rozsahu hodnot

Vlastnosti jsou rozlišovány na dva hlavní typy: kardinální a nominální. U kardinálních lze přímo přesně vypočítat rozdíly mezi jejich hodnotami u jednotlivých subjektů, zatímco u hodnot nominálních lze pouze stanovit, jsou-li shodné či nikoli [73 stránky 7-8]. Výpočty rozdílů pro oba tyto typy se proto budou lišit. Nejprve se zaměřme na porovnávání vlastností kardinálních.

Kardinální typy hodnot

Každý porovnávaný subjekt lze v podstatě vyjádřit jako vektor (např. $\vec{a}$ nebo $\vec{b}$) tvořený hodnotami jednotlivých vlastností. V případě, že by každá z těchto kardinálních vlastností měla stejnou váhu, tzn., že např. rozdíl jednoho roku u věku by při porovnání byl stejně významný, jako rozdíl 1cm výšky či 1kg váhy, tvořil by v tomto případě rozdíl skalární součin jednotkového vektoru a vektoru tvořeného absolutními rozdíly jednotlivých složek vektorů $\vec{a}$ a $\vec{b}$ (viz Vzorec 28).

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} je\ rozdílné\ od \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} |a_1 - b_1| \\ |a_2 - b_2| \\ \vdots \\ |a_n - b_n| \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

Vzorec 28 – Vektorové vyjádření výpočtu rozdílu kardinálních vlastností mezi osobou A (vektor $\vec{a}$) a B ($\vec{b}$)

Vzorec 28 lze též vyjádřit pomocí sumy absolutních hodnot rozdílů jednotlivých vlastností (viz Vzorec 29).

$$\sum_{i=1}^{n} |a_i - b_i|$$

Vzorec 29 – Výpočet rozdílů kardinálních vlastností subjektů A a B

Pokud by rozdíly u jednotlivých vlastností měly mít různý vliv na hodnocení celkové podobnosti dvou subjektů, pak lze ve výpočtu zohlednit i jejich váhy ($\vec{v}$, viz poslední sloupec v Tab. 13), jež jsou součástí metadat (viz Vzorec 30)[40].

$$R_{kar.} = \sum_{i=1}^{n} (|a_i - b_i| \cdot v_i)$$

Vzorec 30 – Výpočet rozdílů kardinálních vlastností subjektů A a B se započtením vah

Nominální typy hodnot

Pro nominální hodnoty, byť jsou tyto kategorie označeny čísly (kvůli kódování), nelze určit, jak moc jsou rozdílné, ale pouze jsou-li nebo nejsou shodné. V případě shody je výsledek 0, v případě rozdílu 1. V tomto případě lze tedy použít buď operátor porovnání (`if`) obou nominálních hodnot, nebo pro absolutní hodnotu jejich rozdílu provést funkci *sgn*[41] (viz Vzorec 31). Míru vlivu obou typů hodnot na celkový rozdíl pak lze stanovit vhodným nastavením váhy.

$$R_{nom.} = \sum_{i=1}^{m} (sgn|a_i - b_i| \cdot v_i)$$

Vzorec 31 – Výpočet rozdílů nominálních vlastností subjektů A a B se započtením váhy

Celkový rozdíl

Celkový rozdíl je tvořen součtem rozdílů kardinálních a nominálních hodnot (viz Vzorec 32).

$$R = R_{kar.} + R_{nom.}$$

Vzorec 32 – Výpočet celkového rozdílu subjektů A a B

Pro konkrétní případ porovnání dvou osob by byl postup následující (viz Tab. 14). Rozdíly pro jednotlivé typy hodnot jsou přitom vypočteny prostřednictvím výše uvedených vzorců, zvolených dle typu hodnoty (viz sl. Typ v Tab. 13).

Vlastnost	Osoba A	Osoba B	Rozdíl	Váha	Vážený rozdíl
Výška	175	165	10	1,5	15,0
Váha	85	60	25	0,9	22,5
Věk	35	25	10	1,0	10,0
Pohlaví	2	1	1	10,0	10,0
Barva očí	5	1	1	2,0	2,0
Celkem			**47**		**59,5**

Tab. 14 – Ukázka postupu výpočtu rozdílu mezi osobami A a B

[40] ve vektorovém vyjádření (viz Vzorec 28) by pak vektor vah $\vec{v}$ nahradil vektor jednotkový
[41] funkce *sgn* ponechá pouze vstupní hodnotu 0, všechny ostatní (kladné) hodnoty konvertuje na 1

Výsledný rozdíl osob A a B má tedy hodnotu 59,5. Budeme-li uvažovat např. ještě osobu C, se stejnými hodnotami všech vlastností jako osoba B, pouze s tím rozdílem, že její výška by byla 166cm, pak by rozdíl mezi osobou A a C měl hodnotu 58. Lze tedy říci, že osoba C by byla podobnější osobě A, nežli osoba B. Tímto postupem pak lze v libovolně rozsáhlém souboru osob najít osobu nejvíce či nejméně podobnou vzorové[42] osobě A (z hlediska porovnávaných vlastností), přičemž počet porovnání by byl roven pouze počtu osob v databázi.

Kódování

Dalším požadavkem na systém porovnávání byla schopnost snadného databázového uchování hodnot porovnávaných vlastností. V případě databáze osob obvykle bývá každá z hodnot atomicky uložena ve vlastní poli, resp. sloupci databázové tabulky. V takovém případě se sice musí pracovat individuálně s každým sloupcem, avšak potřebné výpočty lze provést již v rámci SQL dotazu. U složitějších struktur, jako je právě záznam interního míchání otázek, ovšem nelze předem předpřipravit datová pole, aby efektivně dokázala atomicky pokrýt všechny možné kombinace. Z tohoto důvodu bylo pro uchování kombinací náhodně vybraných hodnot vymezeno jen jedno pole.

Pro uložení více hodnot do jedné položky je nezbytné jejich kódování. V úvahu by přicházel např. datový typ *blob*, umožňující záznam textových či binárních dat nelimitované délky nebo *varchar*[43]. V případě datového typu *varchar* či textové verze *blobu* mohou být do položky ukládány textové řetězce ve zvoleném kódování. Z důvodů omezení znaků použitelných pro tištěnou formu bude pro kódování hodnot použito pouze 6 bitů z každého znaku řetězce, tj. o rozsahu 64 (2^6) možných znaků. V případě binárního *blobu* pak lze samozřejmě pro jednotlivé znaky využít celé škály bytu (2^8 = 256 znaků).

Znaky pro kódování 6-ti bytových hodnot mohou být zvoleny libovolně, obecně by však měly být vybírány mezi 32-126 znakem z ASCII tabulky, neboť právě tyto jsou téměř všechny zobrazitelné v tištěné formě a nepodléhají národnostnímu kódování. Odpadají tak veškeré potíže s kódováním znakové sady, není problém při jeho posílání přes URL parametr či v XML souboru, řetězec je snadno čitelný i pro člověka a dokonce zaznamenatelný v nedigitalizované formě.

Výběr znaků pro následující příklady byl inspirován formátem Base64, avšak vzhledem k tomu, že standardní implementované převodní metody pro tento účel nebylo možné použít, bylo jejich pořadí a dodatečné znaky vhodněji upraveny (viz Tab. 15).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	0	1	2	3	4	5

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	6	7	8	9	-	_

Tab. 15 – Převodní tabulka mezi hodnotami (0-63) a pro kódování zvolenými znaky

Znaky + a / použité v Base64 (viz [74 str. 3]) byly nahrazeny jinými, jelikož ty původní obvykle podléhaly ještě kódování při přenosu v rámci parametru URL. Navržené seřazení znaků je alespoň orientačně čitelné i pro člověka, neboť malá písmena a čísla od 0 do 5 značí první polovinu a velká písmena, zbylá čísla a ostatní znaky jsou v druhé polovině. Při dobré znalosti abecedy pak lze intuitivně odhadnout i přesnější hodnotu z každého znaku.

[42] vzorem je nazýván subjekt, s nímž jsou při vícenásobném porovnávání porovnávány subjekty ostatní
[43] varchar – databázový typ pro textový řetězec proměnlivé délky omezené shora [86 str. 149]

Dalším rozdílem od Base64 je i celkový přístup kódování. Base64 je totiž bezeztrátové a kóduje libovolně objemnou hodnotu přes jakýkoli potřebný počet znaků (viz Tab. 16).

Data (text)	O								k								o							
ASCII kód	79								107								111							
Bitové vyjádření	0	1	0	0	1	1	1	1	0	1	1	0	1	0	1	1	0	1	1	0	1	1	1	1
6-ti bitové číslo	19								54								45							
Base64	T								2								t							

Tab. 16 – Postup kódování znaků „Oko" do Base64 („T2tv")

Navržený způsob naopak vychází z toho, že hodnota každé vlastnosti bude uložena právě v jednom znaku, tzn., že kódovaná hodnota bude převedena na rozsah 0-63. Větší hodnoty tak sice budou uloženy ve ztrátovém tvaru a menší naopak nebudou zcela využívat celou škálu znaku, ovšem díky tomuto přístupu bude možné individuálně přímo porovnávat jednotlivé položky.

Pro převod hodnoty (a_i) vlastnosti na rozsah 0-63 je tato nejprve normalizována na rozsah $\langle 0, 1 \rangle$ (viz Vzorec 33) a následně vynásobena maximální hodnotou tohoto rozsahu, tj. 63. Výsledek zaokrouhlený na celé číslo (zde dochází ke ztrátě přesnosti) je pak hodnotou, jež bude převedena na znak dle kódové Tab. 15.

$$n_i = \frac{a_i - min_i}{rozsah_i - 1} = \frac{a_i - min_i}{max_i - min_i}$$

Vzorec 33 – Normalizace hodnoty i-té vlastnosti na rozsah $\langle 0, 1 \rangle$

Tento postup kódování lze použít na všechny číselné hodnoty, včetně nominálních, jsou-li jednotlivé kategorie nejprve převedeny na čísla. U hodnot s vyšším rozsahem než 64, je tento krok nezbytný, u diskrétních hodnot s nižším rozsahem nikoli. Od nich totiž stačí odečíst minimum a mohou být použity pro kódování přímo s tím, že nevyužijí celého rozsahu. Pro následující příklad byl právě zvolen tento jednotný postup pro všechny typy hodnot. Tab. 17 ukazuje postup zakódování hodnot vlastností dvou osob z předchozího příkladu.

Vlastnost	Osoba A				Osoba B			
	Hodnota	n_i	0-63	Znak	Hodnota	n_i	0-63	Znak
Výška	175	0,7500	47	P	165	0,6944	44	M
Váha	85	0,3294	21	v	60	0,2314	15	p
Věk	35	0,2756	17	r	25	0,1969	12	m
Pohlaví	2	1,0000	63	_	1	0,0000	0	a
Barva očí	5	0,8000	50	S	1	0,0000	0	a

Tab. 17 – Postup kódování hodnot vlastností na znaky

Dle příkladu v Tab. X je jsou tedy charakteristiky osoby A zakódovány do textového „charakteristického" řetězce „Pvr_S" a osoby B do „Mpmaa". Z řetězců je na první pohled patrné, že se jedná o dvě rozdílné osoby, které se neshodují ani v jednom parametru, včetně dvou posledních nominálních hodnot.

Porovnání charakteristických řetězců

Porovnání dvou charakteristických textových řetězců může probíhat několika způsoby, resp. v různých fázích zpětného převodu na původní hodnoty, přičemž v každém z nich je pro dosažení téhož výsledku třeba použít jiných vah.

Vlastnost	Znaky		Kódové hod. (0-63)			Normalizované h. ⟨0, 1⟩			Původní hod. (a_i)		
	A	B	A	B	rozdíl	A	B	rozdíl	A	B	rozdíl
Výška	P	M	47	44	3	0,746	0,698	0,048	174,3	165,7	8,6
Váha	v	p	21	15	6	0,333	0,238	0,095	86,0	61,7	24,3
Věk	r	m	17	12	5	0,270	0,190	0,079	34,3	24,2	10,1
Pohlaví	_	a	63	0	1	1,000	0,000	1,000	2,0	1,0	1,0
Barva očí	S	a	50	0	1	0,794	0,000	1,000	5,0	1,0	1,0
Celkem					16			2,222			44,9

Tab. 18 – Zpětný výpočet původních hodnot (a_i) z kódových textových řetězců (znaky)

Nejjednodušším způsobem jsou absolutní rozdíly hodnot převedených přímo z jednotlivých znaků (viz sl. „Kódové hod. – rozdíl" v Tab. 18). Druhou možností jsou absolutní rozdíly z hodnot normalizovaných na rozsah ⟨0, 1⟩ n_i, tj. předchozí hodnoty vydělené 63 (viz sl. „Normalizované h. – rozdíl" v Tab. 18). Třetí možností je porovnání hodnot převedených na ty původní inverzní operací z výpočtu n_i (viz Vzorec 33) jak ukazuje Vzorec 34.

$$a_i = n_i \cdot (rozsah_i - 1) + min_i = n_i \cdot (max_i - min_i) + min_i$$

Vzorec 34 – Výpočet původní hodnoty i-té vlastnosti

Původní váhy (viz Tab. 13) lze použít až ve třetí fázi, tj. při porovnávání původních hodnot. Lze však také jednorázovým přepočtem vah ušetřit nutnost opakovaného převádění kódovaných hodnot na ty původní, pro každou vlastnost, každého porovnávaného subjektu.

4.7.2 Mix otázky

Prvotní nastavení podmínek kódování (metadata) by pro konkrétní případy prováděl člověk, jež by posoudil, jaké rozsahy a váhy jsou pro daný účel nejvhodnější, popř. provedl pokusné měření a na jeho základě je dodatečně upravil. V případě testových otázek obsahujících různé náhodné prvky na nejrůznějších místech otázky, by takovéto nastavení pro tvůrce otázek bylo velmi komplikované, v některých případech (např. při míchání pozice obsahu) dokonce nemožné. Celý systém by tak měl pracovat na pozadí zcela automatizovaně.

Jelikož pořadí týchž náhodných prvků v otázce může být pokaždé jiné a některé části mohou být dokonce zcela vynechané, byly pro rozpoznání prvků použity jejich identifikátory (id). Pod nimi se při prvním zpracování otázky sepíší metadata čili údaje potřebné pro kódování hodnot, tzn. typ hodnoty a jejich minimum a maximum[44]. V případě potřeby jsou těmto prvkům (či jejich položkám) také přiděleny číselné identifikátory[45] pozic (`index`) a hodnot (`val`), pod nimiž budou následně kódovány. Pořadí položek (jednotlivých znaků) v kódovaném řetězci pak určuje pořadí jejich zápisu v tomto seznamu. Seznam (např. viz Kód 24) je ukládán v databázi přímo v rámci záznamu dané otázky (sl. `MIX_METADATA` v tab. `A_QUESTIONS` viz Příloha 13) a v případě potřeby (při přidání nových prvků, či odkrytí prvků kvůli náhodnému výběru dříve nedostupných) může být i rozšiřován.

[44] tyto informace jsou u náhodných hodnot z daného rozmezí (`<rnd>`) přímo uvedeny a v ostatních případech nejsou zapotřebí
[45] identifikátory zadané tvůrcem otázky nemusí být numerické

```
<metadata>
  <verisons index="1">
    <ver id="a" val="1" />
    <ver id="b" val="2" />
  </versions>
  <rnd index="2" id="x" type="car" min="10" max="99" />
  <rnd index="3" id="y" type="car" min="1" max="9" />
  <rnd index="6" id="var" type="nom" />
  <mix id="m1">
    <itm index="4" id="i1" />
    <itm index="5" id="i3" />
    <itm index="7" id="i2" />
  </mix>
</metadata>
```

Kód 24 – Ukázka možného zápisu metadat otázky do XML

V případě, že by některá z hodnot v seznamu z aktuálního mixu otázky byla z důvodu náhodného výběru vyřazena, musí být i přes to v kódovém řetězci zastoupena, aby jeho jednotlivé pozice odpovídaly vždy stejným prvkům. Pro tyto případy ekvivalentu hodnoty null[46] bude pro následující příklad použit speciální znak # (křížek), jež značí, že daná položka není v řetězci zastoupena[47]. V takovém případě jsou možné tři základní přístupy při porovnávání této hodnoty s jinou.

- Označit oba řetězce za neporovnatelné.
- Při porovnávání těchto konkrétních znaků stanovit nulový rozdíl pro danou vlastnost.
- Fakt, že vlastnost v jednom případě existovala a ve druhém nikoli, ohodnotit jako rozdíl maximální.

To, který přístup zvolit záleží na konkrétním případu použití, při porovnávání mixu otázek byla zvolena třetí možnost (maximální rozdíl).

Vícenásobné porovnání

Postup při porovnávání dvou subjektů, resp. jejich charakteristických řetězců byl vysvětlen v první části této kapitoly. Tento způsob lze přitom aplikovat i na libovolně rozsáhlý seznam a nalézt tak nejvíce či nejméně odlišný subjekt od vzorového. Porovnávání mixu otázky však má za úkol určit rozdíl nikoli pouze mezi dvěma subjekty, ale mezi jedním subjektem (generovaná otázka) a skupinou (dříve zadané otázky).

Pokud otázka byla v minulosti zadána zkoušenému vícekrát, je z důvodu úspory výpočetní náročnosti bráno v potaz maximálně pět posledních případů[48]. S každým z nich je pak vzorový řetězec porovnán a celkový rozdíl je součtem těchto hodnot. Jednotlivá porovnání však nemají stejnou váhu, je zde totiž použit podobný princip, jako byla složka zapomínání při výběru testových otázek (viz kap. 4.6 a str. 84). Ten je však v rámci urychlení výpočtů zjednodušen[49] na pouhé odstupňování váhy podle pořadí po jedné pětině[50].

[46] null je speciální hodnota použitelná v databázích a některých objektových programovacích jazycích napříč datovými typy, jež signalizuje, že poli databázového záznamu či proměnné nebyla nastavena žádná hodnota
[47] při kódování do rozsahu 256, lze tento rozsah o jednu snížit (na 255) a poslední hodnotu nechat vyhrazenu pro null
[48] použít pro porovnání lze samozřejmě libovolný počet předchozích případů, popř. i všechny
[49] lze samozřejmě použít i přímo časovou složku popsanou v uvedené kapitole
[50] poslední varianta otázky tak má váhu 1, předposlední 0,8, jí předcházející 0,6 atd. až pátá (nejstarší ve výběru) má váhu 0,2

4.7.3 Nejodlišnější náhodné hodnoty

Způsobů, jak vytvořit co nejodlišnější otázku vzhledem k jejím dřívějším verzím je opět více. Lze například vygenerovat celou otázku několikrát a použít její od minulých zadání nejodlišnější verzi, což vyhodnotí právě porovnání jejich charakteristických řetězců. Možné je také zohledňovat jednotlivé hodnoty již v průběhu jejího vytváření, což umožňuje jejich rozdělení do individuálních znaků. Tento způsob byl také vzhledem k náročnosti opakovaného parsování QML upřednostněn. Jelikož tedy nedochází k porovnávání otázky jako celku, ale pouze na úrovni jednotlivých prvků, není za potřebí ani nijak do výpočtu zohledňovat váhy a vzájemné rozsahy těchto parametrů.

Verze

Náhodná varianta otázky založená na verzích (`<versions>`, viz str. 49) definuje konečný počet možností, z nichž je jedna náhodně zvolena a zobrazení či skrytí jednotlivých prvků otázky jí je pak podmíněno. Z hlediska typu hodnot jsou verze nominální.

Pro nominální hodnoty je v tomto systému použito kódování, jež je (na rozdíl od předchozího příkladu porovnávání osob) nepřevádí na rozsah 0-63, ale hodnoty indexuje celými čísly od 0 výše. Nevýhodou je samozřejmě nemožnost zařazení do výpočtů posledních položek, bude-li jich výčet skýtat více jak 64. Výhodou ovšem je možnost kdykoli výčet rozšířit o další položky, aniž by došlo k zneplatnění dat z již proběhnutých testování.

Výběr verze tedy probíhá pětkrát opakovaným náhodným výběrem ze všech možných verzí. Pokud některá z nich nebyla v předchozích pěti případech zadání dané otázky použita, je tato rovnou prohlášena za nejvhodnější a proces dál nepokračuje. Jestliže však všechny z pěti náhodně vybraných „kandidátů" byly v předchozích pěti případech použity, je nejvhodnější z nich určen nejvyšším součtem vážených nominálních rozdílů od předchozích výběrů (viz příklad v Tab. 19).

Otázka		Předchozí verze		Náhodné verze					
Dat. použití	Váha	Znak	Verze	1	4	5	3	1	
8.2.2012	1.0	b	1	0.0	1.0	1.0	1.0	0.0	Vážený nominální rozdíl
1.2.2012	0.8	f	5	0.8	0.8	0.0	0.8	0.8	
26.1.2012	0.6	e	4	0.6	0.0	0.6	0.6	0.6	
5.1.2012	0.4	d	3	0.4	0.4	0.4	0.0	0.4	
15.12.2011	0.2	f	5	0.2	0.2	0.0	0.2	0.2	
Celkem				**2.0**	**2.4**	**2.0**	**2.6**	**2.0**	

Tab. 19 – Ukázka výběru nejodlišnější náhodné verze (z pěti možných) vzhledem k pěti předchozím volbám

Příklad v Tab. 19 ukazuje postup konkrétního případu náhodného výběru verze z pěti možných, přičemž všechny z navrhovaných možností (2x1, 3, 4 a 5) byly v předchozích pěti případech použity. Nejprve jsou předchozí případy použití otázky seřazeny sestupně podle data a na základě tohoto pořadí jim jsou přiděleny váhy. Dále je pro každý z těchto výběrů dekódován příslušný znak z charakteristického řetězce otázky na číselný identifikátor konkrétních použitých verzí. Následně jsou určeny nominální rozdíly předešlých a navrhovaných verzí (u shodných 0, u různých 1) a ty jsou vynásobeny vahami zohledňujícími aktuálnost předešlé volby. Jejich součet za jednotlivé „kandidáty" je pak rozhodující pro určení „vítězné" verze. Tou je v tomto případě čtvrtá z nich s identifikačním číslem 3 (`val="3"`), jejíž celková odlišnost od pěti předchozích dosáhla nejvyšší hodnoty 2,6.

Náhodné hodnoty

Při generování náhodných hodnot v otázce (<rnd>, viz str. 50) je i v tomto případě každá z nich vygenerována pětkrát. Všechny z této pětice jsou pak porovnány s příslušnými hodnotami pěti předchozích verzí a váženým součtem těchto porovnání jsou stanoveny rozdíly jednotlivých „kandidátů" pro použití v připravované otázce. Do otázky je zařazena hodnota s největším celkovým rozdílem (viz příklad v Tab. 20).

Otázka		Předchozí hodn.		Náhodné hodnoty					
Dat. použití	Váha	Znak	Hodnota	5	32	17	98	61	
8.2.2012	1,0	s	**29,29**	24,3	2,7	12,3	68,7	31,7	Vážený rozdíl
1.2.2012	0,8	Y	**89,00**	67,2	45,6	57,6	7,2	22,4	
26.1.2012	0,6	1	**43,43**	23,1	6,9	15,9	32,7	10,5	
5.1.2012	0,4	d	**5,71**	0,3	10,5	4,5	36,9	22,1	
15.12.2011	0,2	p	**24,57**	3,9	1,5	1,5	14,7	7,3	
Celkem				**118,7**	**67,2**	**91,8**	**160,3**	**94,1**	

Tab. 20 – Ukázka výběru nejodlišnější náhodné celé hodnoty (od 1 do 100) vzhledem k pěti předchozím

Příklad v Tab. 20 ukazuje postup konkrétního případu výběru náhodné hodnoty v rozsahu od 1 do 100 od dekódování předchozích hodnot ze zastupujícího znaku až po určení nejvhodnější hodnoty z pěti možných. Tou je čtvrtá z nich, 98, jejíž celková odlišnost od pěti předchozích výběrů dosáhla nejvyšší hodnoty 160,3.

Jako nominální jsou brány náhodné výběry z výčtu, např. u hodnot typu *string* (text) či *char* (znak). Jejich výběr probíhá podobně jako volba optimální verze. I v tomto případě není nezbytné výběr opakovat vždy pětkrát, neboť pokud je zvolena hodnota, jež v předchozích pěti výběrech nebyla, může být tato rovnou použita a v dalším výběru se u dané položky již nepokračuje.

Náhodný výběr

Postup při míchání obsahu otázky (<mix>, viz str. 51) je o něco komplikovanější. Každá z položek výběru mixu (<item>) má také vlastní identifikátor a díky němu je (s prefixem identifikátoru mixu či pod ním, viz Kód 24) evidována i v metadatech otázky. Hodnotou, jež je pod ní uchována, je pozice, na kterou byla v daném mixu umístěna[51]. V případě jejího vynechání kvůli početnímu omezení pro výběr je na její pozici dosazena hodnota null (#). Při porovnávání přidělené pozice s hodnotou null je rozdíl ohodnocen hodnotou rovnající se počtu vybíraných položek, jakožto maximální možnou neshodou. Jsou-li porovnávány dvě platné hodnoty, je jejich vztah ohodnocen absolutní hodnotou jejich rozdílu.

I v tomto případě se vygeneruje pět možných variant výběru a pozic položek a ty jsou porovnávány s pěti předchozími. Rozdíly jednotlivých položek jsou v rámci mixů sečteny a vynásobeny vahou zohledňující aktuálnost předešlé varianty (stejná jako v předchozích případech). Jejich součet za jednotlivé „kandidáty" náhodných výběrů je pak rozhodující pro určení „vítězné" varianty, kterou je opět ta s nejvyšší hodnotou tohoto součtu (viz Tab. 21).

[51] také v případě kódování pozic položek náhodného výběru nedochází k přepočtu na rozsah 0-63 z důvodů případné budoucí rozšiřitelnosti výběru o další položky

<table>
<tr><th colspan="8">Předchozí testování</th><th colspan="35">Náhodné výběry</th></tr>
<tr><th rowspan="2">Váha</th><th rowspan="2">Řetězec</th><th colspan="6">Pozice položek</th><th colspan="7">V1</th><th colspan="7">V2</th><th colspan="7">V3</th><th colspan="7">V4</th><th colspan="7">V5</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>1</th><th>2</th><th>-</th><th>3</th><th>-</th><th>0</th><th>vΣ</th><th>3</th><th>-</th><th>-</th><th>2</th><th>0</th><th>1</th><th>vΣ</th><th>0</th><th>2</th><th>-</th><th>-</th><th>1</th><th>3</th><th>vΣ</th><th>1</th><th>2</th><th>3</th><th>-</th><th>0</th><th>-</th><th>vΣ</th><th>3</th><th>2</th><th>1</th><th>-</th><th>0</th><th>-</th><th>vΣ</th></tr>
<tr><td>1,0</td><td>#c#bda</td><td>-</td><td>2</td><td>-</td><td>1</td><td>3</td><td>0</td><td>4</td><td>0</td><td>0</td><td>2</td><td>4</td><td>0</td><td>10,0</td><td>4</td><td>4</td><td>0</td><td>1</td><td>3</td><td>1</td><td>13,0</td><td>4</td><td>0</td><td>0</td><td>4</td><td>2</td><td>3</td><td>13,0</td><td>4</td><td>0</td><td>4</td><td>4</td><td>3</td><td>4</td><td>19,0</td><td>4</td><td>0</td><td>4</td><td>4</td><td>3</td><td>4</td><td>19,0</td></tr>
<tr><td>0,8</td><td>bcda##</td><td>1</td><td>2</td><td>3</td><td>0</td><td>-</td><td>-</td><td>0</td><td>0</td><td>4</td><td>3</td><td>0</td><td>4</td><td>8,8</td><td>2</td><td>4</td><td>4</td><td>2</td><td>4</td><td>4</td><td>16,0</td><td>1</td><td>0</td><td>4</td><td>4</td><td>4</td><td>4</td><td>13,6</td><td>0</td><td>0</td><td>0</td><td>4</td><td>4</td><td>0</td><td>6,4</td><td>2</td><td>0</td><td>2</td><td>4</td><td>4</td><td>0</td><td>9,6</td></tr>
<tr><td>0,6</td><td>d#acb#</td><td>3</td><td>-</td><td>0</td><td>2</td><td>1</td><td>-</td><td>2</td><td>4</td><td>4</td><td>1</td><td>4</td><td>4</td><td>11,4</td><td>0</td><td>0</td><td>4</td><td>0</td><td>1</td><td>4</td><td>5,4</td><td>3</td><td>4</td><td>4</td><td>4</td><td>0</td><td>4</td><td>11,4</td><td>2</td><td>4</td><td>3</td><td>4</td><td>1</td><td>0</td><td>8,4</td><td>0</td><td>4</td><td>1</td><td>4</td><td>1</td><td>0</td><td>6,0</td></tr>
<tr><td>0,4</td><td>#cb#ad</td><td>-</td><td>2</td><td>1</td><td>-</td><td>0</td><td>3</td><td>4</td><td>0</td><td>4</td><td>4</td><td>4</td><td>3</td><td>7,6</td><td>4</td><td>4</td><td>4</td><td>4</td><td>0</td><td>2</td><td>7,2</td><td>4</td><td>0</td><td>4</td><td>0</td><td>1</td><td>0</td><td>3,6</td><td>4</td><td>0</td><td>2</td><td>0</td><td>0</td><td>4</td><td>4,0</td><td>4</td><td>0</td><td>0</td><td>0</td><td>0</td><td>4</td><td>3,2</td></tr>
<tr><td>0,2</td><td>#bcd#a</td><td>-</td><td>1</td><td>2</td><td>3</td><td>-</td><td>0</td><td>4</td><td>1</td><td>4</td><td>0</td><td>0</td><td>0</td><td>1,8</td><td>4</td><td>4</td><td>4</td><td>1</td><td>4</td><td>1</td><td>3,6</td><td>4</td><td>1</td><td>4</td><td>4</td><td>4</td><td>3</td><td>4,0</td><td>4</td><td>1</td><td>1</td><td>4</td><td>4</td><td>4</td><td>3,6</td><td>4</td><td>1</td><td>1</td><td>4</td><td>4</td><td>4</td><td>3,6</td></tr>
<tr><td colspan="8">Celkem</td><td colspan="6"></td><td>39,6</td><td colspan="6"></td><td>45,2</td><td colspan="6"></td><td>45,6</td><td colspan="6"></td><td>41,4</td><td colspan="6"></td><td>41,4</td></tr>
</table>

Tab. 21 – Ukázka postupu vyhodnocení nejodlišnější varianty náhodného výběru (4 položek ze 6) vzhledem k pěti předchozím

Příklad v Tab. 21 ukazuje postup konkrétního případu náhodného výběru od dekódování předešlých výběrů z částí charakteristických řetězců až po určení nejvhodnější varianty výběru z pěti možných. Tou je V3, jejíž celková odlišnost od pěti předchozích výběrů dosáhla nejvyšší hodnoty 45,6. Podle ní má být ze 6-ti položek do výběru zařazena 1., 5., 2. a 6. a to v tomto pořadí. *Pro lepší představu jsou v Tab. 21 orámovány tři související hodnoty a to 4. položka z výběru V2 (2), tatáž položka u předminulé verze otázky (0) a taktéž jejich rozdíl v průsečíku obou souřadnic (|2-0|=2).*

4.7.4 Shrnutí

Charakteristické textové řetězce umožňují zakódovat do čitelných znaků hodnoty vlastností různých subjektů a vzájemně porovnávat jejich podobnost. V této kapitole byl předveden postup a způsob použití takovýchto řetězců pro generování více různých, než jen čistě náhodných, otázek pro téhož zkoušeného. Stejný postup je též možné použít i pro různé zkoušené při hromadné přípravě otázek v podobný čas, pro jednu IP adresu či počítačovou učebnu, jako prevenci proti možnému opisování. Nastíněny byly ale i další možnosti využití, včetně různých variant implementací celého postupu či jeho dílčích částí.

Díky kódovým řetězcům, mohou být například zjednodušeny některé databázové struktury, přičemž by v některých databázových systémech nemělo být náročné implementovat podporu pro vyhledávání a porovnávání těchto řetězců přímo do SQL funkce.

Porovnávání ovšem nemusí probíhat vždy oproti rozsáhlé databázi jiných subjektů, ale díky snadné přenositelnosti charakteristického řetězce může být realizováno i individuálně „v terénu" např. pomocí mobilního telefonu. Kromě textových znaků je pro uložení hodnot také možné použít i jiné datové struktury, např. čárový kód.

Díky separaci jednotlivých vlastností do individuálních znaků také lze porovnávat pouze jejich určité části, nezávisle na zbytku a bez nutnosti dekódování celého řetězce.

Oblasti využití jsou široké, a tento princip lze použít všude, kde je třeba uchovávat a porovnávat kardinální či nominální charakteristiky. Může jít např. o parametry zboží, vyhledávání osob, vozidel, seznamky atd.

4.8 Rychlá multiplatformní autentizace v internetu [◎ III.2.D]

(publikováno v [ap-3] a [ap-10])

Při komunikaci aplikací typu klient-server prostřednictvím internetu existuje řada rizik, se kterými je nutné počítat a předcházet jim. Pomineme-li technická úskalí, je zde stále možnost nežádoucího záměrného zásahu třetí osoby (útočníka). Ta může veškerou komunikaci mezi dvěma stranami odposlouchávat, zaznamenávat, analyzovat, případně do ní i aktivně vstupovat úpravou zasílaných dat, jejich opakovaným použitím, popřípadě i podvržením vlastních zpráv.

Představíme si tedy podrobněji některé druhy takovýchto útoků a zároveň i preventivní opatření proti nim. Zejména se zaměříme na pokročilou ochranu uživatelských hesel a také na univerzální, snadno implementovatelný způsob autentizace každé komunikační zprávy, zasílané klientskou aplikací na server.

4.8.1 Webové služby

Zvláštní pozornost v oblasti autentizace si zaslouží tzv. webové služby. *Webová služba je softwarová aplikace, ke které se vzdáleně přistupuje přes klasické protokoly založené na XML. Webová služba standardně definuje formát zprávy, specifikuje rozhraní, ve kterém má být zpráva posílána, popisuje konvence pro mapování obsahu zprávy pro vstup a výstup z programů provozujících službu a definuje mechanizmy pro zpřístupňování a zveřejňování rozhraní služby.* [75]

Například webové služby používané v Microsoft .NET Frameworku jsou implementovány jako metody tříd v některém z jazyků podporovaných .NET Frameworkem, které mohou být vzdáleně volány libovolnou aplikací, jež dodrží komunikační protokol SOAP[52]. Ten ovšem standardně neobsahuje žádné zabezpečovací mechanizmy, které by bránily zneužití webové služby nepovolanými osobami či aplikacemi.

Ze souboru platforem .NET pouze Silverlight z bezpečnostních důvodů nepovoluje volání mezi různými doménami. Toto opatření implicitně brání aplikacím Silverlightu, aby přistupovaly k jakékoli webové službě, která se nachází v doméně nebo kombinaci domény a portu, jež se liší od domény hostící aplikaci Silverlightu. Pouze cílový server může určit, které domény mohou přistupovat k jeho službám, pokud implementují soubor se zásadami Silverlightu (clientaccesspolicy.xml) v kořeni webového serveru. [76]

Toto omezení je ovšem pouze jakýmsi dobrovolným „sebeomezováním" Silverlightu (byť vyplývajícím z politiky hostujícího prohlížeče, která se podobně vztahuje i např. na aplikace vytvořené v Adobe Flash), který odmítá volat cizí webové služby, pokud mu to není výslovně povoleno ze strany serveru s webovou službou. Ostatní aplikace, včetně přímo webového prohlížeče však toto omezení respektovat nemusí a znají-li URL a strukturu webové služby, případně má-li tato vystavena svůj WSDL[53] dokument, pak jim nic nebrání tuto službu volat s libovolnými parametry a číst data, která vrací. Je tedy třeba zabezpečit přístup ke každé webové službě přímo v ní samotné.

V praxi lze celou situaci přirovnat k ulici (internet), kde má většina domů (serverů) dveře dokořán a jediným jejich zajištěním je na nich pověšený seznam osob, které mají povolen vstup.

[52] *SOAP – Simple Object Access Protocol je založený na XML, který je základem pro multiplatformní a na programovacím jazyku nezávislou komunikaci přes webové služby distribuovaných výpočetních aplikací* [75]
[53] WSDL – Web Service Definition Language – v XML dané struktury standardizovaný popis funkcí nabízených webovou službou

Microsoft v knize Web Service Security [77] nabízí několik standardizovaných řešení přímé i zprostředkované autentizace přes WSE[54]. Zprostředkovaná řešení zahrnují Kerberos, X.509 PKI a Security Token Service, které pochopitelně vyžadují pro svůj provoz další externí systémy a oddělenou správu identit uživatelů. V případě přímého řešení je pak správa a ověřování identit součástí nebo podsystémem aplikace přímo za webovou službou. Navrženy jsou implementace tří variant autentizace: pomocí Active Directory (přes LDAP[55]), dle databáze dodržující předdefinované schéma, nebo vytvořením vlastní autentizační třídy. Veškeré identifikátory jsou přitom součástí hlavičky SOAP obálky, takže přímo webová služba není zatěžována dalšími nadbytečnými parametry. Druhé dvě varianty by tedy mohly být univerzálním řešením, avšak automatizace jejich zpracování na klientské straně .NET technologiemi komplikuje využití v klientských aplikacích vytvořených na jiných platformách než .NET.

Přístup článku [75] taktéž využívá hlavičku SOAP obálky pro zasílání identifikačních údajů o uživateli, ale tato data jsou tam přidávána plně kontrolovaným kódem aplikace. Pro autentizaci je využita ruční implementace SRP[56] protokolu, který si při zahájení spojení se serverem vymění nezbytné údaje, z nichž je vygenerován autentizační „session key", platný pouze pro aktuální spojení.

Toto řešení není náročné na implementaci, ovšem vyžaduje pouze pro své účely vícero výměn dat mezi serverem a klientem, což zatěžuje server. Článek [75] v závěru uvádí měření, z něhož vyplývá, že použití tohoto postupu zpomaluje zpracování požadavku v průměru 4,74x oproti použití webové služby bez autentizace. Vzhledem k faktu, že tato autentizace není závislá na délce zprávy, je relevantnějším údajem spíše fakt, že zpomalení způsobené touto metodou bylo naměřeno 1 - 1,5 s, přičemž vliv na tuto dobu má i rychlost spojení se serverem, který byl však při tomto pokusu na témže počítači jako klientská aplikace.

Článek [78] navrhuje robustní bezpečnou autentizaci na základě hashe hesla zkombinovaného s náhodnou hodnotou (salt). Heslo ani jeho hash přitom nemusí nikdy putovat přes veřejnou síť, dokonce ani při prvotní registraci uživatele, podmínkou však je uchování nejen hesla, ale i složitých hash-kódů, buď v certifikačním souboru, nebo na čipové kartě. Tato podmínka velmi komplikuje využití pro aplikace založené na cloud computingu (jeho další specifická rizika např. viz [79]), jež by měly být použitelné kdykoli a odkudkoli, neboť je zapotřebí mít u sebe neustále soubor s certifikátem a nestačí pouze znát zvolené heslo.

U všech uvedených řešení navíc nelze zcela vyloučit nežádoucí opakované posílání týchž zpráv na server útočníkem typu man-in-the-middle.

4.8.2 Ochrana hesel v databázi

Nejprve si ukažme základní principy ochrany hesel uložených v databázi a jejich úskalí, neboť na nich prezentovaný postup bezpečné autentizace staví.

[54] WSE – Web Services Enhancements – přídavný modul pro Microsoft .NET Framework rozšiřující možnosti webových služeb, mimo jiné o možnost jejich zabezpečení na úrovni SOAP
[55] LDAP - Lightweight Directory Access Protocol
[56] SRP – Secure Remote Password – protokol pro ověřování identity na základě hesla, které ovšem není v žádné dešifrovatelné formě zasíláno mezi komunikujícími stranami [75]

Hash

Hash je jednosměrná (irreverzibilní) výpočetně efektivní funkce, která mapuje různě dlouhé binární řetězce na řetězce pevné délky, tzv. hash-hodnoty. Ty slouží jako kompaktní zástupce vstupního řetězce. V kryptografickém použití, je hash funkce H zvolena tak, aby bylo výpočetně nemožné nalézt dva různé vstupy, jejichž hash-hodnoty by byly shodné (tj. nalézt X a Y takové, aby platilo $H(X) = H(Y) \wedge X \neq Y$). Zároveň by také mělo být výpočetně nemožné určit vstup X pro danou hash-hodnotu Y (tj. $H(X) = Y$). Pravděpodobnost, aby n-bitová hash-hodnota (např. n = 256) náhodně vybraného řetězce měla konkrétní n-bitovou hash-hodnotu, je tedy 2^{-n}. [80]

Hashe mají ve výpočetní technice různá využití, jako například kontrolní součty souborů, přenášených dat, v digitálních podpisech nebo pro „ukládání" hesel.

Hesla

Ukládat hesla uživatelů přímo do databáze na serveru, byť tato databáze není žádným konvenčním způsobem zvenčí dostupná, není rozhodně bezpečné. Nikdy nelze vyloučit krádež této databáze schopným hackerem, či zaměstnancem s přístupem na server s databází. Ten by pak získal údaje o registrovaných uživatelích včetně čistého znění jejich hesel. Přitom nebývá výjimkou, že někteří uživatelé používají stejná hesla pro různé webové služby, z důvodu snazší zapamatovatelnosti. Útočník by tak rovnou získal přístup k jejich emailovým schránkám, profilům na sociálních sítích, případně i k bankovním kontům.

Tomuto je třeba předcházet a hesla v originálním znění do databáze, ať již je jakkoli zabezpečená, vůbec neukládat. Jednou z možností je hesla zašifrovat, ovšem pokud je hesla schopna rozšifrovat aplikace, pak útočník, který se již dokázal zmocnit chráněné databáze, zřejmě zvládne i aplikovat na serveru používaný dešifrovací algoritmus a získat tak uživatelská hesla.

Server však pouze potřebuje ověřit, je-li uživatelem zadané heslo shodné s tím v databázi a právě díky hashování pro tento úkon ani nepotřebuje znát jeho originální znění. Pokud se do databáze uloží pouze hash hesla a nikoli heslo samé, nebude možné z těchto hashů zpětně vypočíst originální hesla. Pro ověření přitom stačí z uživatelem zadaného hesla znovu vypočíst hash a ten porovnat s tím uloženým v databázi. Kvalitní hashovací algoritmy, jako například SHA-256 [81] spolu s faktem, že heslo smí obsahovat pouze určité znaky (písmena, čísla, podtržítko apod., nikoli ale binární znaky, jako např. ASCII znaky s kódem nižším než 32) garantují, že je statisticky téměř nemožné, aby existovalo jiné heslo se stejným hashem. A i kdyby, přijít na něj bude daleko složitější, než odhalit heslo původní.

Slovníkové útoky

Hash hesla dostatečně ochraňuje proti přímým výpočetním útokům na jejich prolomení. Je zde však ještě jedna možnost, jak se k heslu, jehož hash známe, dostat a tou je heslo „uhodnout". Pro tyto účely hackeři používají zejména slovníkové útoky nebo útoky hrubou silou.

Útok hrubou silou znamená, že útočník k odhalení hesla použije algoritmus, který postupně zkouší kombinace všech možných znaků, z nichž pokaždé vypočte hash, který porovná s hashem hesla uživatele. Pokud se rovnají, heslo je zjištěno. Tato metoda má vždy řešení, které je ovšem závislé na délce a složitosti hesla, s níž roste i doba potřebná k jeho zjištění. Například budeme-li

brát v potaz pouze znakovou sadu Base64[57], je pro tří znakové heslo 262 144 (64^3) možných kombinací ovšem pro minimální doporučovanou délku 8 znaků už je to 281,5 biliónů (64^8) kombinací. Při teoretické rychlosti milionu pokusů za sekundu dojde k prolomení tříznakového hesla nejdéle za 0,2s, kdežto na osmiznakové je již zapotřebí až téměř 9 let.

Útočník samozřejmě může použít distribuovanou výpočetní jednotku, díky čemuž tuto dobu podstatně zkrátí, ovšem i tak již nejde o krátkodobou záležitost. Luštit takto hesla několika set uživatelů je pak již značně dlouhodobá investice. Krom toho výpočet hashe není triviální matematickou operací, nýbrž komplikovaným algoritmem, jež zabere určitý výpočetní čas (např. viz Tab. 25).

Hackeři však mají k dispozici v záloze ještě metodu, která spoléhá především na lehkomyslnost uživatelů při volbě svého hesla, a tím je „slovníkový útok". Ten vychází z toho, že uživatelé pro snazší zapamatovatelnost volí taková hesla, která jsou určitými samostatnými slovy, jmény případně snadno zapamatovatelnými kombinacemi znaků (např. nejpoužívanější heslo „123456" [82]). Takovéto seznamy jsou dokonce volně dostupné na nejrůznějších webech, někdy dokonce rovnou s již vypočtenými nejpoužívanějšími hashi.

Díky tomu není třeba tyto hashe ani znovu vypočítávat a stačí pouze jednotlivé hashe ze zcizené databáze nalézt v tomto „slovníku", což je v moderních databázových systémech, při použití indexace, operace ještě mnohem rychlejší, než jakkoli jednoduchý výpočet. I v případě, že by hacker měl porovnat celý slovník některého jazyka řekněme o půl miliónu slovech, pokud obsahuje hledané heslo, bude mít výsledek za pár milisekund.

Nebývá přitom výjimkou, že právě uživatelé volící takto snadno rozluštitelná hesla je obvykle i používají na více místech, nehledě na fakt, že pro volbu snadno zapamatovatelných hesel odolávajících těmto druhům útoku existuje řada postupů (např. viz [83]). Navíc je zde i problém s tím, že pokud si zvolí dva či více uživatelů stejné heslo, budou mít i stejný hash a rozluštěním jednoho z nich útočník získá hesla hned několika uživatelů současně.

Salt

Problém s hashovými slovníky lze hackerům jednoduše zkomplikovat použitím tzv. saltu. V něm jde o to, že i když uživatel zvolí nevhodně jednoduché heslo obsažené v „hackerském hashovém slovníku" aplikace sama zajistí jeho dodatečné zkomplikování na mimo slovníkový rozsah. Pro každého uživatele totiž vygeneruje náhodnou sadu znaků (salt), kterou před výpočtem hashe přičte k zadanému heslu (viz Obr. 24). Salt poté uloží jako zvláštní, nikde nezobrazovanou hodnotu, mezi ostatní údaje o uživateli do databáze. [84]

Obr. 24 – Výpočet hashe hesla chráněného saltem

Při následném přihlašování uživatele pak postupuje stejně, pouze místo vygenerování nového saltu použije salt uložený v databázi. Ten tedy přidá k zadanému heslu uživatele, vypočte hash této kombinace a porovná jej s hashem uloženým v databázi.

[57] Base64 – sada 64 základních znaků, tj. 26 velkých písmen, 26 malých písmen, 10 čísel a „+" a „/"

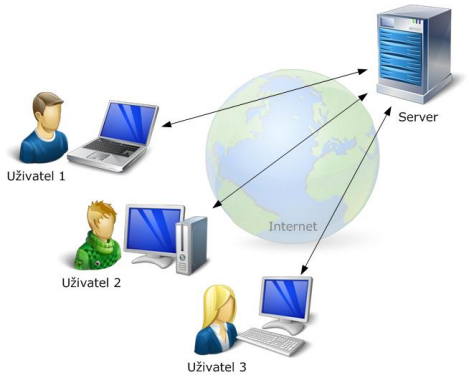
Salt přitom obvykle nijak zvlášť chráněn není a je uložen v originálním znění hned vedle hesla. Jedinou ochranou saltu může být způsob jeho kombinace s heslem (součtem textových řetězců, binárním součtem, přičtením hashe saltu, proložením znaků, …), nicméně na fakt, že útočník tento algoritmus nezjistí, nelze vzhledem k existenci mnoha dekompilačních programů[58] příliš spoléhat. Salt nemusí být vždy speciálně generován, ale lze použít již existující hodnoty jedinečné pro každého uživatele, jako jsou například login či ID, popřípadě hashe těchto hodnot.

I když totiž útočník zná hash hesla, salt i způsob jejich zkombinování, je pro něj rozluštění hesel uživatelů daleko komplikovanější, než pokud je použit pouze hash. Nemůže totiž vůbec použít hashový slovník a pro každého uživatele musí u všech zkoušených slov (hesel) znovu vypočíst jeho hash se započtením saltu. Jak již bylo zmíněno výše, výpočet hashe zabere určitý výpočetní čas a pokud heslo není ryze slovníkové, jeho odhalení zabere až nereálně dlouhý čas. Taktéž není rozeznatelné, používají-li někteří uživatelé stejná hesla.

Salt (sůl), stejně jako při přípravě pokrmů, v nadměrném použití snižuje „chuť na jídlo", v tomto případě na heslo, resp. jeho rozluštění. [80]

4.8.3 Aplikace klient-server v internetu

Nyní se zaměřme na problematiku autentizace aplikací typu klient-server, komunikujících přes veřejnou síť internet (viz Obr. 25). Mezi tyto aplikace lze zařadit klasické Windows aplikace, které pracují s daty na nějakém centrálním serveru, ať již přímo, nebo distribuovaně s občasnými synchronizacemi centrální a klientské databáze. Patřit mezi ně mohou také pokročilejší webové aplikace vytvořené pomocí JavaAppletů, Flashe nebo Silverlightu, ale i propracovanější Java skripty.



Obr. 25 – Aplikace klient-server v internetu

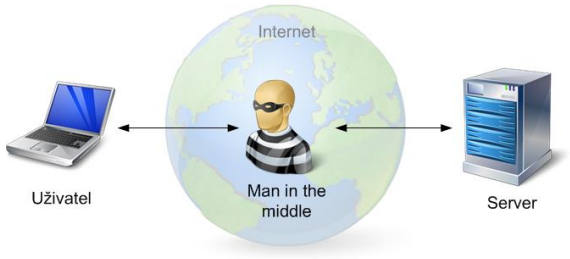
V těchto případech lze samozřejmě využít mnoha předpřipravených řešení (např. webové služby) a komunikačních protokolů (https[59]) s již zabudovaným určitým stupněm ochrany. Ne vždy je však těchto technologií možné či žádoucí použít nebo jsou považovány za dostatečnou ochranu.

Autentizací uživatele je myšlena jeho jednoznačná identifikace na serveru a zároveň bezpečné ověření, že je tím, za koho se vydává. Toto se obvykle ověřuje pomocí přihlašovacího jména (login) a hesla. Ač heslo v databázi může být chráněno výše popsanými metodami, při autentizaci je nezbytné jej nějak doručit z klientského počítače na server, kde bude teprve prověřeno. Na této cestě však může číhat řada úskalí.

[58] dekompilace programu – zpětné vygenerování zdrojového kódu ze strojového kódu (např. exe či dll souboru, viz např. [129])
[59] https – šifrovaný protokol pro posílání dat po internetu

Man in the middle

Základním problém při komunikaci přes veřejnou síť je tzv. „man in the middle" („člověk uprostřed" nebo „člověk mezi", viz Obr. 26). Zasílaná data totiž v internetu procházejí přes různé servery, které mohou být monitorovány oním „člověkem uprostřed". Pokud je takovýto útočník pasivní, tak komunikaci pouze odposlouchává, zaznamenává a analyzuje. Je-li však aktivní, může i zasílaná data různě modifikovat, nebo dokonce do komunikace vstupovat zasíláním vlastních zfalšovaných zpráv jedné či druhé straně.



Obr. 26 – Schéma „man in the middle" (člověk uprostřed)

S „člověkem uprostřed" je nezbytné počítat v každém případě, ať je již jeho očekávání jakkoli nepravděpodobné. Dokonce ani v případě komunikace pouze v interní neveřejné síti nelze obvykle takovýto útok zcela vyloučit.

Jsou-li data dostatečně zašifrována, nemělo by hrozit, že je útočník bude schopen v reálném čase číst. Jsou-li také bezpečně „podepsána" neměl by je dokázat ani modifikovat. Uvažujeme-li například synchronní šifrování (např. pomocí uživatelského hesla), může jako jednoduchý podpis sloužit třeba hash těchto dat, ke kterým byl nejprve přičten hash hesla. Jelikož jej útočník nezná, není ani schopen vytvořit takovýto kontrolní hash modifikované zprávy a bez něj jej server odmítne zpracovat.

Heslo samotné by nikdy nemělo být posíláno přes veřejnou síť v rozluštitelné formě. Již při registraci by hash hesla měla vytvořit přímo aplikace na klientské straně a na server poslat pouze ten. Ovšem ani prostý hash hesla by se neměl posílat jen tak. Jeho minimální ochranou je komunikace přes šifrovaný protokol https. Jelikož totiž heslo nikde na serveru uloženo není, používá se pro veškerou autentizaci právě jeho hash, případně jeho další hashové kombinace, jež je server schopen dopočítat s informacemi, které má.

Zaměřme se ale právě na problematiku útočníkem podstrčených falešných zpráv zasílaných na server. Nezná-li útočník heslo ani jeho hash, nemůže sice vlastní zprávy podepisovat, ani se serveru autentizovat, ovšem stále má ještě jednu možnost, jak uživateli minimálně zkomplikovat spojení. Může totiž v minulosti zachycené zprávy zasílané klientem na server ve vhodný okamžik znovu použít a sám je zaslat serveru. Tyto zprávy jsou totiž autorizované a správně „podepsané", takže by je server měl přijmout a znovu zpracovat. Útočník při tom ani nemusí znát přesný obsah takovéto zprávy.

Ne vždy je samozřejmě něco takového až zas tak závažné, avšak každý si jistě dovede představit určité případy, kdy by něco takového mohlo zásadně vadit. Například pokud by šlo o prostou komunikaci a útočník by zaznamenal klientovy zprávy „ano" a „ne" a následně je během on-line rozhovoru občas zaměňoval… Jindy by zase mohl zachytávat komunikaci s bankou a zaslat jí platební příkaz provedený uživatelem opakovaně apod.

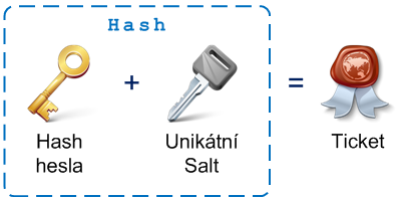
Jak tedy proti takovýmto útokům zabezpečit každou komunikaci, nikoli pouze spojení[60], klientské aplikace se serverem? Předveďme si zde jednoduchý postup, kterým lze efektivně zajistit bezpečnou autentizaci uživatelů přes tyto klientské aplikace, ať již samostatně nebo v kombinaci s existujícími bezpečnostními řešeními.

4.8.4 Bezpečná autentizace pomocí povinně unikátních saltů

Navržený princip uvažuje tzv. ticket, tj. bez znalosti hesla nezfalšovatelný kontrolní součet („podpis") unikátní pro každou komunikaci a to i v případě, že zasílaná data budou stále stejná (např. dotaz aplikace typu „Jsou pro mě nějaké novinky?" či odpověď serveru „Nejsou."). Přesněji řečeno, vycházet se bude pouze z hashe hesla, neboť heslo samotné server nezná.

Je-li hash hesla v databázi uživatelů zabezpečen ještě pomocí stálého saltu přiděleného konkrétnímu uživateli, je třeba tento salt předat klientské aplikaci, aby jej mohla zakomponovat do výpočtu hashe hesla po přihlášení uživatele. Jelikož tento salt nemusí být striktně utajován, jeho zaslání internetem při registraci klientské aplikace přes šifrovaný protokol by neměl být problém. V následujícím příkladu však pro jeho snazší pochopení od případné existence ochrany hashe hesla pomocí stálého saltu abstrahujeme, stejně jakož i od nutnosti ochrany posílaných dat. Jejich zašifrování a podepsání týmž nebo jiným principem je již pak triviální záležitostí.

Aby bylo dosaženo pokaždé jedinečného ticketu při autentizaci je pro každou komunikaci vždy znovu vygenerován nový unikátní salt. Výpočet kontrolního ticketu pro aktuální komunikaci se serverem proběhne jako hash výsledku součtu hashe hesla uživatele a unikátního saltu (viz Obr. 27).

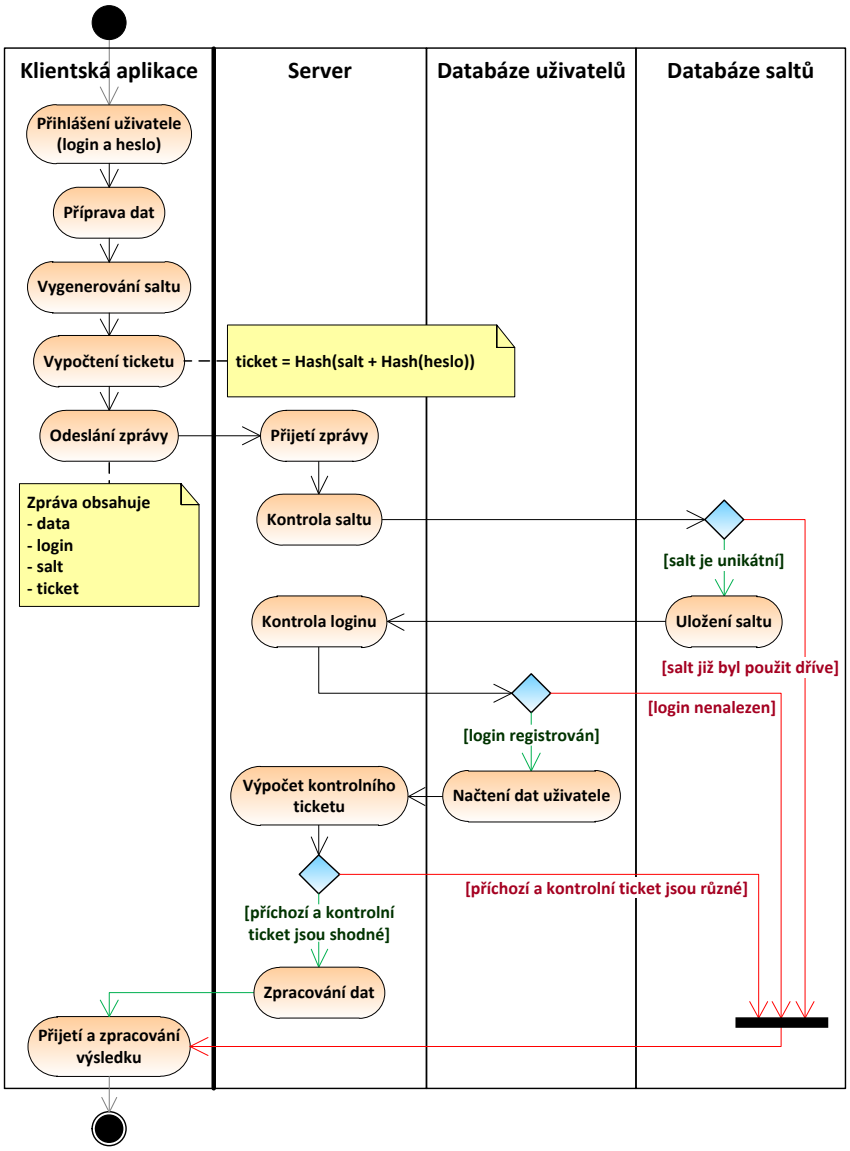


Obr. 27 – Výpočet ticketu

Unikátní salt a vypočtený kontrolní ticket jsou poté přiloženy spolu s daty ke zprávě a ta je odeslána na server. Jelikož útočník hash hesla, který není součástí komunikace, nezná ani jej nemůže zachytit nebo zpětně vypočítat, nedokáže ani vytvořit kontrolní ticket. Přitom díky faktu, že tento salt je pro každý požadavek zasílaný na server vygenerován znovu a vždy jinak, je i kontrolní ticket jiný a případná útočníkem opakovaně zaslaná zpráva bude serverem okamžitě odhalena, jelikož bude obsahovat již použitý salt.

Server by totiž měl veškeré příchozí salty od všech klientů ukládat do speciální databáze, aby jakákoli jejich duplicita byla okamžitě identifikována a taková příchozí zpráva zamítnuta jako podvržená. Celý proces takto chráněné komunikace zachycuje schéma na Obr. 28 (také viz Příloha 12).

[60] jedno spojení klienta se serverem obsahuje více samostatných komunikací mezi oběma stranami



Obr. 28 – Diagram činností zobrazující autentizační postup

Průběh zpracování je tedy následující (též stejně číslované schéma viz Příloha 12):

1. Uživatel v klientské aplikaci zadá své přihlašovací jméno (login) a heslo.
    a. Login si aplikace uloží tak jak je (uložením je myšleno uložení do proměnné aplikace, která s jejím vypnutím bude zapomenuta).
    b. Z hesla se vypočte jeho hash a až ten si aplikace uloží.
2. Když aplikace potřebuje komunikovat se serverem, postupuje následovně:
    a. Připraví data či zformuluje požadavek, který potřebuje zaslat serveru.
    b. Vygeneruje náhodný unikátní salt a to pokaždé znovu.
    c. Dle saltu a hashe hesla vypočte ticket platný pouze pro tuto komunikaci.
    d. Login, ticket, salt a data zašle na server. Tyto čtyři položky spolu tvoří tzv. zprávu.
3. Server ověří platnost příchozích dat (autentizuje uživatele):
    a. Uloží příchozí salt do databáze saltů. Kdyby byl tento Salt použit již v minulosti, databáze saltů by uložení mezi unikátní hodnoty odmítla a autentizace by byla rovnou zamítnuta.

   b. V databázi uživatelů dle loginu unikátního pro každého uživatele nalezne a vrátí jeho uložený hash hesla. Není-li tento login zaregistrován, je autentizace zamítnuta.
   c. Dle příchozího saltu a hashe hesla z databáze uživatelů vypočte kontrolní ticket.
   d. Porovná příchozí ticket a vypočtený kontrolní ticket.
   e. Jsou-li oba tickety shodné, data jsou zpracována, pokud ne, je autentizace zamítnuta a data jsou „zahozena".
4. Server informuje klientskou aplikaci o výsledku autentizace a byla-li tato úspěšná, pak také
   a. o výsledku zpracování dat,
   b. případně vrátí požadované údaje, je-li k tomu autentizovaný uživatel autorizován (oprávněn).

Ověření unikátnosti

Pro ověření unikátnosti saltu stačí na serveru jedna databázová tabulka o jednom sloupci, který je zároveň jejím primárním klíčem. Do něho se ukládají veškeré příchozí salty. Pokud by došlo k pokusu o uložení saltu již použitého, databáze tuto operaci zamítne a vygeneruje výjimku, kterou jakmile program zachytí, vyhodnotí, že se jedná o podvrh a dále již v autentizačním procesu nemusí pokračovat.

Toto řešení je nejjednodušší, obsah saltu se nemusí nikterak „zkoumat", je brán, tak jak přijde. Výhodou je i možnost, rozšířit tuto tabulku o další dodatečné informace, které by v případě pokusu o útok mohly napomoci v dalších protiopatřeních, případně sloužit pro statistické účely. U každého saltu lze například navíc evidovat, který uživatel a ze které IP adresy jej zaslal, v jaký přesný čas, atd. Nevýhodou takového řešení je nezbytnost absolutní záruky vždy originálního Saltu a také neustále do nekonečna narůstající databázový soubor s jejich historií.

Nejjednodušším způsobem, jak mohou klientské aplikace neustále „vymýšlet" nové originální salty, aniž by si musely cokoli pamatovat, je použití tzv. GUID[61] (někdy též UUID[62]). GUID je číslo (převedené na hexadecimální znaky) o rozsahu 2^{128}, pseudonáhodně vygenerované na základě „náhodných" aktuálních okolností (jako je např. aktuální datum a čas s max. přesností, MAC adresa počítače apod.). Uvádí se, že pravděpodobnost vygenerování dvou stejných GUIDů (kdekoli a kdykoli) je tak malá, že není třeba takovouto situaci vůbec řešit. [85] Nicméně určitě nebude od věci, pokud by v případě zamítnutí autentizace serverem aplikace alespoň jednou automaticky pokus zopakovala s jiným saltem.

Moderní programovací jazyky obvykle poskytují funkci pro automatické vygenerování GUID hodnoty (např. v C# je to `Guid.NewGuid()`), takže implementace takovéhoto postupu je velmi snadná.

Salt s informační hodnotou

GUID je sice vždy unikátní hodnotou, ale jinak nemá žádnou vypovídající schopnost a když už se v databázi ukládá, proč by nemohl obsahovat nějaké užitečné „zpracovatelné" informace přímo ve svém základu (bez nutnosti jejich ukládání do dalších sloupců databázové tabulky). Například by mohl obsahovat přesné datum a čas (např. pro 7.3.2011 12:34:56 sadu znaků „110307123456"), podle něhož je již možné například data řadit.

[61] GUID – Globally Unique IDentifier
[62] UUID – Universally Unique IDentifier

Takováto sada znaků by sama o sobě samozřejmě nebyla pro dosažení jedinečnosti dostačující, jelikož může nastat případ, že by různí uživatelé chtěli komunikovat se serverem ve stejnou sekundu. Pomoci by mohlo rozšíření třeba o ID uživatele, přidané před nebo za časové znaky (záleží na preferenci pro řazení). Nicméně i zde může nastat případ duplicitních hodnot, například u uživatele, který má aplikaci spuštěnou na více počítačích současně.

Do saltu se dá ovšem zařadit spousta dalších hodnot, jako třeba IP adresa, MAC adresa počítače, session ID apod. Jistou pojistkou unikátnosti pak na závěr může být vždy přiložen GUID nebo jeho část. Při kombinaci ID uživatele a data a času s přesností alespoň na sekundy a zároveň schopností aplikace zopakovat komunikaci v případě jejího zamítnutí serverem by měl stačit i jeden náhodný znak. Samozřejmě záleží zde též na frekvenci komunikace se serverem.

Ověření unikátnosti saltu bez nutnosti jeho archivace

Není-li třeba na serveru uchovávat historii saltů a komunikace klientských aplikací se serverem je tak častá, případně uživatelů příliš mnoho, a archivní databáze saltů by rychle narůstala, je možné zvolit i jiný přístup ověření unikátnosti příchozího saltu bez nutnosti ukládání každého z nich. V takovýchto případech je možné prvek náhodnosti ze saltu vyloučit zcela a stavět jej kompletně na konkrétně stanovených hodnotách. Ty ale musí obsahovat výhradně zpětně určitelné údaje, kvůli ověření prvotnosti jejich použití. Přitom nevadí, že ty následující budou kýmkoli předvídatelné, hlavně musí být vždy ověřitelně unikátní.

Použití ID uživatele v kombinaci se znaky sestávajících se z data a času, pomineme-li případ vícenásobného spuštění aplikace jedním uživatelem, by se mohlo jevit jako částečná záruka unikátnosti (čas bude pokaždé jiný). Server by pak automaticky zamítal všechny Salty obsahující kód data a času staršího než aktuální. Ovšem je třeba si uvědomit, že salt vzniká na klientském počítači a na něm nastavený datum a čas nemusí korespondovat s datem a časem serveru. Také zpoždění komunikace způsobené aktuální rychlostí internetového připojení zde hraje svou roli. Nelze tudíž s jistotou ověřit, je-li datum a čas v příchozím saltu „dostatečně aktuální".

Jinou možností je, že server může zveřejňovat tzv. „prefix dne" (popř. hodiny, týdne, měsíce, roku, …). To by byla veřejně dostupná hodnota, kterou by si ze serveru mohl přečíst každý a kterou by server striktně vyžadoval jako úvodní sadu znaků v každém příchozím saltu. Platnost tohoto prefixu by však byla omezena, stejně jakož i povinnost serveru si pamatovat všechny salty s tímto prefixem. Po určité době (dni, hodině, týdnu, …) by totiž server vyhlásil nový prefix, a od té chvíle už přijímal a ukládal pouze salty začínající jím. Salty se starým prefixem by tak mohl vymazat z databáze („zapomenout") a uložit si pouze tento starý prefix, aby jej v budoucnu nepoužil znovu. Pokud by jím byl například kód aktuálního data, nebylo by třeba ani archivace prefixů. Klientské aplikace by však musely obsahovat mechanizmus, že v případě zamítnutí zprávy kvůli saltu s neplatným prefixem, automaticky ze serveru načtou prefix nový a zopakují zaslání zprávy s ním.

Lepší variantou je použití tzv. auto inkrementální hodnoty, tj. např. čísla, které se při každé komunikaci se serverem automaticky zvýší o jedničku. Serveru si pak stačí pro každého uživatele pamatovat číslo z posledního saltu a povolit pouze ty s vyšší hodnotou. Tím se zároveň předejde i nebezpečí záměny pořadí v zasílaných zprávách.

Pro případy, kdy aplikace umožňuje vícenásobné spuštění jedním uživatelem současně, může kolizi auto inkrementálních čísel rozlišených dále pouze dle ID uživatele předejít např. zavedením tzv. sessions. Ty spočívají v tom, že klientská aplikace po spuštění naváže úvodní spojení se serve-

rem, který jí přidělí „Session ID", tj. jedinečný identifikátor pro toto spojení, které si obě strany budou až do odhlášení klientské aplikace (jejího vypnutí nebo určitou dobu nečinnosti klienta) pamatovat. Jeho unikátnost si přitom hlídá přímo server (např. použitím auto inkrementální hodnoty z jeho strany). Tato hodnota pak bude také součástí saltu a díky ní bude i rozlišena každá instance klientské aplikace, byť téhož uživatele.

4.8.5 Rychlost

Časová náročnost tohoto způsobu autentizace byla ověřena měřením na testovací aplikaci vytvořené na platformě Silverlight. Ta se připojovala k serveru, vytvořenému v ASP.NET, pomocí klasické webové služby (asmx). Databázový systém na serveru byl v tomto případě použit Firebird 2.5 [w26] (více např. viz [86]). Pro výpočet hashe a generování GUID byly na obou stranách použity algoritmy integrované v programovacím prostředí Microsoft .NET Framework 4.0, jazyk C#. Klientská aplikace i webový server byly při testu spuštěny na témže počítači, což mělo vliv pouze na kratší dobu komunikace obou stran, která ale nebyla předmětem měření. Parametry testovacího stoje uvádí Příloha 25.

Testovací aplikace pro autentizaci používala popsaný autentizační mechanizmus. Unikátnost saltu byla postavena na prosté GUID hodnotě, ukládané v samostatné databázi o jedné tabulce s jedním sloupcem. Samotná funkce webové služby byla ryze testovací. Kromě autentizačních údajů přijímala jen textový řetězec, který v případě pozitivního ověření uživatele převrátila a vrátila jej v opačném pořadí („ABC" → „CBA").

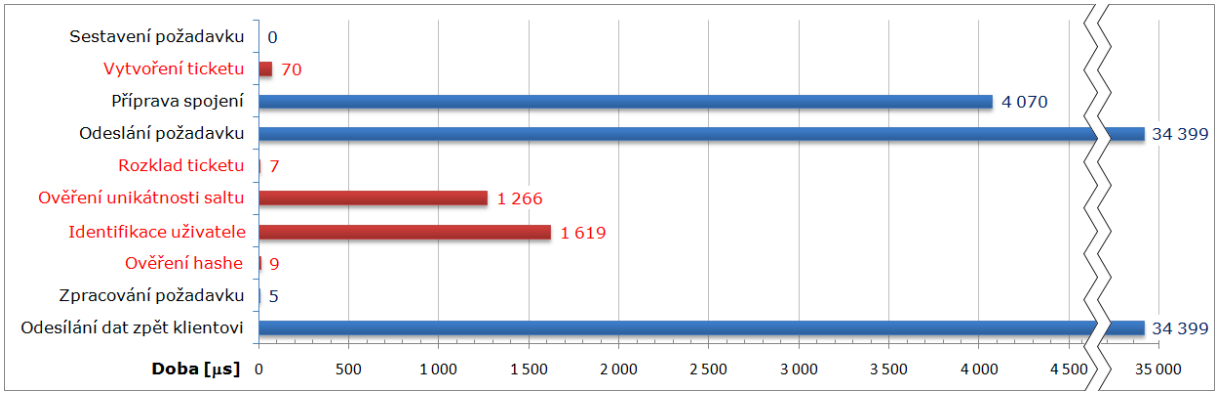
Měřeny byly začátky jednotlivých operací pomocí třídy „Stopwatch" v tzv. „Ticks" (1 tick = 100 ns, resp. 0,1 μs) a na základě rozdílů těchto začátků byla vypočtena časová rozmezí, potřebná pro vykonání dané operace. Čas na posílání dat mezi klientem a serverem byl kvůli nemožnosti existence společných stopek pro klienta a server vypočten jako rozdíl celkového času od okamžiku zahájení odesílání požadavku na server do okamžiku přijetí odpovědi a času, po který se prováděly veškeré operace na serveru. Operacím pro odeslání požadavku a jeho přijetí pak bylo přiznáno po polovině této výsledné hodnoty. Seznam jednotlivých měřených operací je následující:

1. **Sestavení požadavku** – Vygeneruje se náhodný text o délce 32 znaků.
2. **Vytvoření ticketu** – Vygeneruje se nový salt (hexadecimální vyjádření nové GUID hodnoty), ten se spojí s hashem hesla přihlášeného uživatele a vypočte se hash této hodnoty. Výsledný hash, salt použitý v tomto hashi a ID uživatele se spojí do jednoho textového řetězce (kompletní autentizační ticket).
3. **Příprava spojení** – Je vytvořena instance třídy SoapClient pro volání webové služby.
4. **Odeslání požadavku** – Data jsou asynchronně odeslána webové službě, která je kompletně přijme a zahájí svou činnost.
5. **Rozklad ticketu** – Na serveru je textový řetězec (autentizační ticket) rozložen zpět na jednotlivé části (hash, salt a ID uživatele).
6. **Ověření unikátnosti saltu** – Salt je uložen do databáze saltů, přičemž kód dokáže reagovat na výjimku způsobenou pokusem o uložení duplicitní hodnoty.
7. **Identifikace uživatele** – Dle ID je v databázi nalezen uživatel a načten hash jeho hesla.
8. **Ověření hashe** – Ze spojení hashe hesla uživatele načteného z databáze a přijatého saltu je vypočten hash, který je porovnán s příchozím hashem. Výsledkem je potvrzení nebo vyvrácení identity uživatele.

9. **Zpracování požadavku** – Znaky vstupního textu jsou převráceny na opačné pořadí.
10. **Odesílání dat zpět klientovi** – Výsledný text je odeslán zpět klientovi, který jej přijme.

Test byl opakovaně (50x) proveden za podmínky, kdy bylo v databázi saltů uloženo méně jak 500 hodnot a pak znovu 50x, když bylo v databázi uloženo již více než 500 000 saltů, za jinak stejných podmínek. Při každém z těchto celkem 100 pokusů byl vždy použit jiný vstupní řetězec (požadavek). Z naměřených hodnot byly vypočteny průměry, přičemž hodnoty jednotlivých měření se významně nelišily. Výjimku tvořilo pouze první provedení pokusu, které však bylo způsobené faktem, že ASP.NET při prvním volání provádí kompilaci webové aplikace (viz [42 str. 203]). Tyto hodnoty do výsledku samozřejmě započítány nebyly.

Naměřené hodnoty obou případů (s 500 a 500 000 salty v databázi) se však ani nyní statisticky významně nelišily, a to jak v celkových časech, tak i v rámci jednotlivých operací, včetně „Ověření unikátnosti saltu", kde jedině bylo relevantní skutečný rozdíl očekávat. Statistická shoda hodnot naměřených v obou případech byla ověřena Mann-Whitney testem na 5% hladině významnosti. Průměr z hodnot obou pokusů ukazuje Graf 22.



Graf 22 – Průměrná časová rozpětí (v µs) jednotlivých operací v testovaném pokusu

Pouze některé operace se týkaly procesu autentizace. Z nich časově nejnáročnější je identifikace uživatele a ověření unikátnosti saltu, což je logické, protože obě operace vyžadují připojit se k databázi a pracovat s ní (při použití jiného DBS by mohly být tyto hodnoty samozřejmě odlišné). I tak ovšem kompletní zpracování požadavku v průměru trvalo pouze necelých 76 ms (75 844 µs) a z toho pouze necelé **3 ms** (2 970 µs) zabraly veškeré operace pro práci s autentizačním ticketem. Při takovémto postupu není naměřený čas práce s ticketem závislý na velikosti posílaných dat, takže se jedná o konstantní časový náklad pro libovolný, jakkoli velký datový požadavek.

Tento způsob autentizace je tedy velmi rychlý a implementace celého postupu je programátorsky snadná. Vše je přitom multiplatformní, neboť postup lze implementovat ve většině programovacích jazyků. Při použití kvalitního hesla poskytuje samotné komunikaci klientské aplikace se serverem dostatečně bezpečnou autentizaci, i bez použití dalšího zabezpečení. Nutno však připomenout, že v testu nebylo nijak řešeno zabezpečení posílaných dat, tj. jejich šifrování a zahrnutí kontrolního součtu do autentizačního hashe.

4.8.6 Shrnutí

Prezentovaný algoritmus umožňuje bezpečnou autentizaci pro aplikace komunikující se serverem prostřednictvím internetu. Díky vždy unikátním saltům je vyloučena útočníkem podvržená autentizace bez znalosti správného hesla, včetně možnosti opakovaného zasílání předešlých zpráv. Princip staví na schopnosti serveru identifikovat již použité salty a striktně tak v příchozích zprávách vyžadovat jejich unikátnost.

Programátorská implementace uvedeného postupu je jednoduchá a při použití kvalitního uživatelského hesla (např. viz [83]) poskytuje komunikaci klientské aplikace se serverem dostatečně bezpečnou autentizaci, i pokud nejsou použity další zabezpečovací technologie. Přitom zpomalení komunikace, které si použitím tohoto principu každé individuální spojení vyžádá, je přijatelně nízké.

4.9 Šifrovací algoritmus [◎ III.2.E]

(publikováno v [ap-4])

Princip dokonalé šifry je znám již téměř celé století. Podmínky, které musí být při jejím používání dodrženy, však praktické nasazení stále komplikují. Šifra je tedy vhodná tam, kde se vyžaduje extrémně vysoké utajení, a kde nevadí mimořádně vysoké náklady spojené s výrobou a distribucí klíčů. [87]

V této kapitole shrneme princip dokonalé šifry a na jeho základě se s využitím moderních hashovacích algoritmů pokusíme navrhnout jednu z možných modifikací práce s klíči tak, aby byly zásadní faktory znemožňující praktické používání šifry eliminovány.

4.9.1 Základní pojmy

V následujícím textu budou používány známé výrazy, které ovšem mohou mít v různých kontextech různé významy. Upřesněme si tedy, co je pod jejich označením míněno zde. Ostatní pojmy budou vysvětleny přímo této kapitole.

- **Data** – zdrojová data, která jsou třeba ochránit šifrováním před jejich přečtením a zneužitím třetími osobami.
- **Zpráva** – je soubor informací zasílaných mezi dvěma komunikujícími stranami. Skládá se z **dat** (zašifrovaných) a dalších údajů, jako jsou identifikátor odesílatele, určení příjemce, datum a čas odeslání, apod.
- **Klíč** – tajná sada dat, s jejichž pomocí budou výpočetní operací zdrojová data šifrována a následně dešifrována.
- **Heslo** – slovo, fráze nebo kombinace znaků, kterými lze zašifrovaná data dešifrovat a naopak. Tzn. heslo se, buď použije přímo jako **klíč**, nebo se klíč vygeneruje na jeho základě, což bude i tento případ.

4.9.2 Dokonalá šifra

Roku 1917 Američan Gilbert Sandford Vernam (1890 – 1960) zkonstruoval zajímavý šifrovací systém. Ten vycházel z Vigenèrovy šifry z roku 1586 [88], ovšem obsahovala několik zásadních změn, údajně inspirovaných německým kryptologem Hermannem z roku 1892 [89]. Systém byl založený na použití jednorázového náhodně vygenerovaného hesla, které má stejnou délku jako zpráva sama.

Fakt, že je Vernamova šifra, zvaná též „one-time pad" (jednorázová tabulka), absolutně bezpečným (neprolomitelným) šifrovacím systémem, v roce 1949 matematicky prokázal Claude Elwood Shannon (1916 – 2001) [90]. Této dokonalé nerozluštitelnosti šifry lze dosáhnout ovšem pouze při dodržení tří přísných podmínek spolehlivosti:

1. Klíč musí být stejně dlouhý jako přenášená zpráva.
2. Klíč musí být dokonale náhodný.
3. Klíč nesmí být použit opakovaně. [91]

Jsou-li podmínky spolehlivosti dodrženy, je tato šifra zcela bezpečná proti jakémukoli pokusu o prolomení, včetně útoku hrubou silou. Není-li totiž znám správný klíč, neexistuje způsob proveditelný ani v libovolně dlouhém časovém horizontu, jak zprávu rozluštit. Je sice možné najít takový klíč, který by dokázal zašifrovaná data převést na srozumitelný text téže délky, ovšem takovýchto klíčů lze nalézt tolik, že tato data mohou v podstatě dávat libovolný smysl, přičemž nelze odhadnout, která z takovýchto zpráv byla tou odesílanou.

Vernam ve svém následném patentu [92] navrhl i jednoduchý přístroj, který pracoval s 31 znaky (26 písmen, mezera, znaky návrat vozíku (CR) a posun o řádek (LF) a signály "následují číslice" a "následují písmena"), což pokrývalo 5 bitů, jež přístroj šifroval náhodným klíčem pomocí operace XOR.

Operace XOR

Logická operace exkluzivní disjunkce, v originále „exkluzive or", zkráceně XOR se značí takto: „⊕". Jeho výsledkem je 0, pokud jsou obě vstupní hodnoty shodné a 1 jsou-li rozdílné.

$$A \oplus B = C$$

A	B	C
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Tab. 22 – Stavová tabulka hodnot operace XOR

XOR je komutativní operace, tudíž nezáleží na pořadí jednotlivých hodnot, mezi nimiž se operace XOR provádí.

$$(A \oplus B = C) \quad = \quad (B \oplus A = C)$$

Místo operace XOR lze také použít funkci modulo 2 ze součtu obou hodnot.

$$(A \oplus B = C) \quad \Leftrightarrow \quad (A + B) \bmod 2 = C$$

Z výsledku operace XOR lze následně další operací XOR s jednou z výchozích hodnot dopočítat tu druhou.

$$(A \oplus B = C) \quad \Leftrightarrow \quad (C \oplus A = B) \quad \Leftrightarrow \quad (C \oplus B = A)$$

Pokud tedy i-tý datový bit D_i zašifrujeme operací XOR s i-tým bitem klíče K_i do zašifrovaného znění C_i, pak z C_i zpětně dešifrujeme výchozí datový bit D_i opětovným provedením operace XOR s i-tým bitem klíče K_i.

Zašifrování: $D_i \oplus K_i = C_i$

Dešifrování: $C_i \oplus K_i = D_i$

Je-li zároveň klíč náhodný, pak pravděpodobnost, že $K_i = 0$, je stejná jako pravděpodobnost, že $K_i = 1$, je rovna ½ (tj. 2^{-1}), čili 50%. Stejně tak je tomu i u zašifrované hodnoty C_i. Není-li tedy znám klíč K_i, je pravděpodobnost „uhodnutí" správné hodnoty přesně poloviční. Pro celý znak sklá-

dající se z 8 bitů (1 byte) je pak za těchto okolností pravděpodobnost určení správného znaku již jen 2^{-8} (tj. 1/256), čili cca 0,4%.

Princip Vernamovy šifry

Původní verze této šifry pracovala pouze se základní anglickou abecedou o 26 písmenech. Těmto písmenům A až Z se přiřadila čísla 0 až 25 a pro i-tý znak utajovaných dat D_i s klíčem K_i (klíč se také skládal pouze ze znaků této abecedy) se znak šifrovaných dat C_i určil následujícím způsobem [88]:

$$C_i = (D_i + K_i) \bmod 26$$

Dešifrování pak proběhlo opačnou operací:

$$D_i = (26 + C_i - K_i) \bmod 26$$

Například slovo „AHOJ" by se s použitím náhodného klíče „SMHZ" šifrovalo jako je tomu v Tab. 23.

	Znaky				Čísla			
Data	A	H	O	J	0	7	14	9
Klíč	S	M	H	Z	18	12	7	25
Zašifrovaná data	S	T	V	I	18	19	21	8
Dešifrovaná data	A	H	O	J	0	7	14	9

Tab. 23 – Ukázka postupu šifrování znaků původní Vernamovou širou

Vylepšená verze této šifry pracuje s binární reprezentací dat. Jednotlivé bity dat převedených do binární podoby jsou šifrovány operací XOR s jednotlivými bity klíče. Výhodou byla možnost strojového zpracování této šifry. Vernam ve své době používal vlastní převodní tabulku pro znaky základní abecedy do binární formy, kterou místo hodnot 0 a 1 označoval znaky + a –. [92] V současnosti, kdy jsou data uchovávána a přenášena v elektronické formě bitů standardně, je situace pro podobný styl šifrování ještě daleko jednodušší.

Stejná data jako v předchozím případě by při binárním kódování (pro převod znaků na bity je použita standardní ASCII tabulka znaků) vypadala tak, jak ukazuje Tab. 24.

	Znaky				Bity			
Data	A	H	O	J	01000001	01001000	01001111	01001010
Klíč	S	M	H	Z	01010011	01001101	01001000	01011100
Zašifrovaná data	18	5	7	22	00010010	00000101	00000111	00010110
Dešifrovaná data	A	H	O	J	01000001	01001000	01001111	01001010

Tab. 24 – Ukázka šifrování dat Vernamovou šifrou na binární úrovni

Zašifrovaná data v příkladu jsou uvedena ve formě indexu znaku v ASCII tabulce, jelikož tyto jsou v textovém formátu nezobrazitelné. V tomto případě by zároveň náhodný klíč neměl využívat pouze byty znaků písmen ale vybírat z celé tabulky všech 256 znaků. Resp. generátor klíče by měl pracovat na úrovni bitů (náhodně volit sekvence 0 a 1), a na výsledné znaky vůbec nehledět.

Důsledky porušení podmínek spolehlivosti

Porušení podmínek spolehlivosti vede k nedostatečné bezpečnosti šifry a umožňuje její prolomení. Konkrétně nedodržení každé jednotlivé podmínky má následující důsledky.

Opakované použití klíče

V případě opakovaného použití klíče je možné tento klíč snadno určit pouze ze znalosti dvou zachycených zpráv šifrovaných týmž klíčem. Platí totiž následující vztah.

$$D1_i \oplus K_i = C1_i$$

$$D2_i \oplus K_i = C2_i$$

$$C1_i \oplus C2_i = D1_i \oplus D2_i$$

Kde $D1_i$ – i-tý znak 1. dat, $D2_i$ – i-tý znak 2. dat, K_i – i-tý znak klíče, $C1_i$ – i-tý znak zašifrovaného znění 1. zprávy a $C2_i$ – i-tý znak zašifrovaného znění 2. zprávy.

Výsledkem operace XOR dvou zašifrovaných dat je tedy XOR dvou původních dat. Tím dojde k odstranění veškeré náhodnosti klíče a z výsledku lze jednoduchou statistickou kryptoanalýzou získat oboje původní data a tím pádem i klíč. [80]

$$K_i = C1_i \oplus D1_i = C2_i \oplus D2_i$$

Díky tomu lze následně každou další zprávu zašifrovanou týmž klíčem dešifrovat již v reálném čase, bez nutnosti dalších kryptoanalýz.

Po prvním použití každého klíče je tedy třeba jej celý bezpečně „zničit", jak na straně příjemce, tak na straně odesílatele.

Krátký klíč

Pokud by klíč nebyl stejně dlouhý jako přenášená zpráva, muselo by dojít k jeho opakování pro šifrování částí dat, které nepokryl. To by mělo za následek týž efekt jako opakované použití klíče. V případě že by útočník znal některou z částí dat, získal by tak zpětným provedením operace XOR část nebo dokonce celý klíč a mohl jej použít na zbylé části dat, jež nezná.

Znalost části dat útočníkem je celkem běžný fakt. V dopisech bývá na začátku obvykle uvedeno „Dobrý den", „Ahoj" apod., na konci zase podpis odesílatele. Při posílání binárních dat je situace mnohdy ještě jednodušší, protože většina formátů souborů má vlastní hlavičku, která je vždy shodná (JPEG, ZIP, WAV, DOC, …) nebo je alespoň z konečné množiny možností. Struktura dokumentů textových editorů (RTF, XML, HTML, …) pak navíc opakovaně obsahuje známé formátovací sekvence znaků, které se dají frekvenční analýzou snadno detekovat.

Nedokonalá náhodnost

Předpoklad dokonalé náhodnosti celého klíče stejně jako jeho dostatečná délka zaručuje, že každý jednotlivý znak (bit) dat je zašifrován zcela nezávisle na ostatních znacích. Znalost jakékoli části dat tedy útočníkovi z výpočetního hlediska neprozradí nic o kterémkoli jiném jemu neznámém znaku dat ani klíče.

Pro dokonalou bezpečnost šifry by se neměly používat ani pseudonáhodné hodnoty. Ty jsou totiž generovány dle určitého algoritmu a jejich vygenerování je tak při dodržení stejných podmí-

nek zopakovatelné. Data jsou pak rozluštitelná v konečném čase, resp. lze nalézt takový klíč, který převede zašifrovanou zprávu na srozumitelná data a zároveň u něho bude možné prokázat vztah mezi jeho jednotlivými částmi nebo k nějaké výchozí hodnotě (seedu[63]) a tak identifikovat, která z možných rozluštění zpráv je ta pravá. Pro generování klíče je nejvhodnější užití fyzikálních metod, např. radioaktivity, o níž je prokázáno, že její charakter je skutečně náhodný. [87]

Kvantová kryptografie

Kvantová kryptografie využívá bezpečného komunikačního kanálu (optického vlákna) mezi dvěma komunikujícími stranami. Pro přenos jednotlivých bitů jsou použity ve smluveném směru polarizované fotony. Ty totiž nelze odposlouchávat jako klasický elektrický signál, jehož intenzitu je možné změřit, aniž by byl tok dat narušen. Foton je dále nedělitelná a neklonovatelná částice a jakákoli interakce s ním jej zásadně ovlivní. Případný odposlech lze tedy snadno odhalit. [93]

Aby se předešlo zachycení dat případným útočníkem odposlouchávajícím komunikaci (man in the middle), je nejprve poslán tímto kanálem klíč, který splňuje požadavky spolehlivosti na jeho délku a náhodnost. Pokud přenos klíče proběhne v pořádku (nedošlo k jeho odposlechu), jsou teprve pak odeslána data zašifrovaná tímto klíčem. V opačném případě je klíč „zapomenut" a zkusí se poslat jiný. Přenos dat již poté ani nemusí probíhat přes zabezpečený kanál, jelikož ta jsou bez klíče, při dodržení požadavků spolehlivosti, absolutně nedešifrovatelná. [91]

Tento princip přináší možnost zcela bezpečné komunikace, ovšem vyžaduje přímé spojení nepřerušovaným optickým kabelem mezi oběma stranami, což je podmínka splnitelná jen v některých výjimečných případech. Při komunikaci prostřednictvím veřejné sítě internet tedy globálně použít nelze.

Jinou možností bezpečného přenosu klíče je osobní předání datového média (např. CD) obsahujícího data klíče pro budoucí použití. Podobným způsobem je například zabezpečena horká linka spojující prezidenty Ruska a USA. [87]

Dlouhý a náhodný klíč

Podmínky spolehlivosti zaručují šifře nerozluštitelnost, zároveň však také komplikují její užívání. Konkrétně požadavek dokonalé náhodnosti klíče znesnadňuje jeho automatické softwarové generování. Také nutnost, aby jeho délka byla shodná s délkou šifrovaných dat, přináší (pomineme-li kvantovou kryptografii) týž problém, jako přenos dat samotných, na čemž se podílí i jednorázovost každého klíče. Může tedy být žádoucí, byť za cenu ztráty absolutní neprolomitelnosti šifry, umožnit, aby klíč resp. heslo mohlo být kratší, ne zcela náhodné (zapamatovatelné) a opakovaně použitelné.

Jak již bylo uvedeno, klíč složený z pseudonáhodných hodnot nezabrání v rozluštění zašifrovaných dat v konečném čase. Bude-li však tento klíč „kvalitně náhodný" a zároveň splňovat ostatní podmínky spolehlivosti, zůstane vyloučena možnost použití jakýchkoli výpočetních a statistických kryptoanalýz, s výjimkou útoku hrubou silou. Ten může útočník například použít na určení klíčů, které zašifrovaným datům (nebo jejich částem) dávají smysl a následně hledání souvztažnosti mezi jednotlivými částmi klíče. Účinnější formou útoku hrubou silou by pak bylo, v případě znalosti algo-

[63] seed – výchozí hodnota generátoru pseudonáhodných hodnot, v němž je každá následující hodnota odvozena od hodnoty předchozího kroku; při zadání téhož seedu lze tudíž zopakovat vygenerování stejné sady pseudonáhodných hodnot

ritmu generátoru pseudonáhodných hodnot pro klíče, určení výchozí hodnoty generátoru – seedu, resp. hesla. Bude-li toto dobře zvoleno, lze data rozluštit pouze „uhodnutím hesla" s použitím „hrubé síly" a onen konečný čas rozluštění dat tak může být nereálně dlouhý.

V případě, že budou pro účely šifrování přínosné výše zmíněné výhody ohledně hesla (krátké, nenáhodné a opakovatelné) a zároveň nebudou nepřekonatelnou překážkou uvedená omezení dokonalosti šifry (při „uhodnutí" hesla budou data dešifrovatelná), pak již zbývá jen vytvořit algoritmus, který z krátkého hesla dokáže opakovaně vytvořit libovolně dlouhý klíč, jenž bude statisticky prokazatelně náhodný v rovnoměrném rozdělení. To znamená, aby všechny hodnoty v rozsahu bytu[64] byly generovány se stejnou pravděpodobností, resp. generovaná sekvence bitů byla sama o sobě náhodná.

Hash

Popsané vlastnosti přímo zapadají do definice hash funkce (viz str. 102) a při jejím vhodném užití lze s její pomocí docílit veškerých požadovaných vlastností klíče.

Statistické testy náhodnosti hash-kódu generovaného algoritmem SHA-1[65] dle [94], [95] prokázali, že jím generovaná sekvence bitů vyhovuje ze statistického hlediska podmínkám rovnoměrného náhodného rozdělení. Jiné hashovací algoritmy (např. MD5, SHA-256, SHA-512, atd. nebo připravovaný SHA-3 [96]) by samozřejmě dle své definice měly mít tytéž vlastnosti, což je možné ověřit například dle postupů uvedených v [80] pomocí software popsaného v [94].

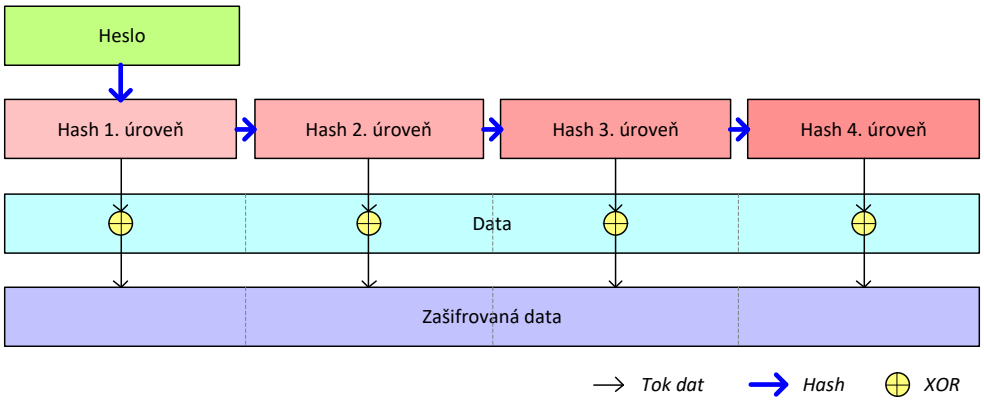
4.9.3 Heslo a klíč

Délka klíče

Pomocí hash algoritmu lze tedy z libovolného hesla vytvořit klíč vyhovující podmínce náhodnosti a neumožňující zpětný výpočet hesla. Jeho délka je ovšem předem určena na konstantní počet bitů dle konkrétního užitého algoritmu.

Zapotřebí je však klíč mnohem delší, než je hash-kód. Jednou z možností je použít jako klíč víceúrovňový hash. V tomto případě by byl hash původního hesla použit na zašifrování prvního bloku dat a následně posloužil jako vstupní hodnota pro vygenerování nového hash-kódu (hash 2. úrovně). Jím by se opět zašifroval další blok dat a znovu by z něho byl vygenerován hash 3. úrovně jako klíč pro další blok dat a tak dále až by byla pokryta celá datová zpráva (viz Obr. 29).

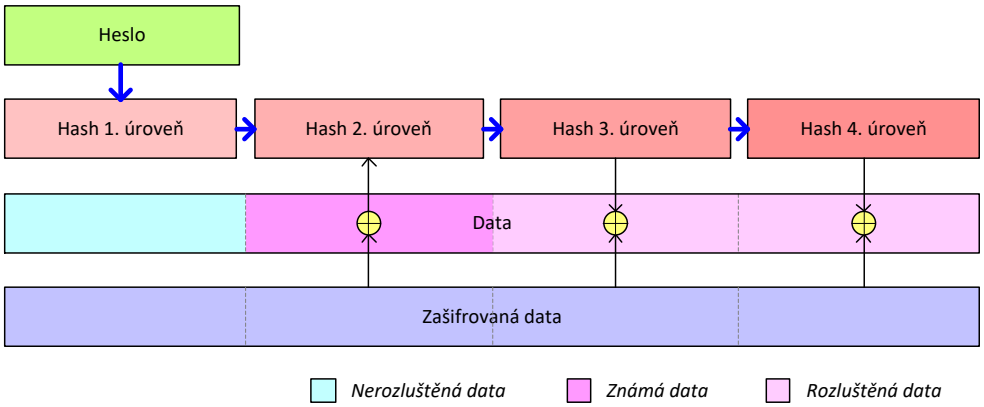
[64] rozsah bytu je 0-255, tedy 256 (2^8) možných hodnot
[65] SHA-1 - Secure Hash Algorithm, vracející hash-kód o délce 160 bitů, který byl navržen institutem NIST pro americké vládní aplikace [80]



Obr. 29 – Schéma šifrování dat operací XOR, kde klíč tvoří prostý víceúrovňový hash hesla

Zamezení útoku při znalosti části dat

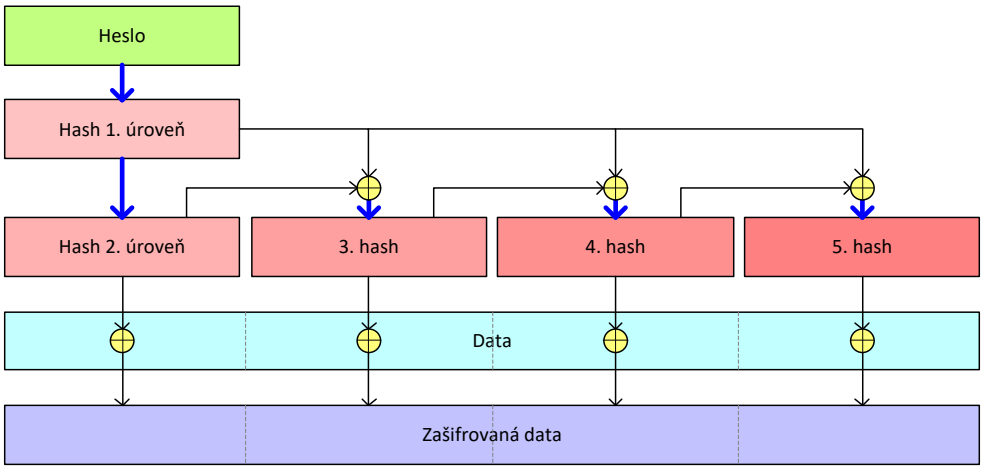
V uvedeném případě je ovšem útočník, který zná část dat, schopen rozluštit jejich i další neznámé části. Pokud by například znal data zašifrovaná hashem 2. úrovně, stačilo by mu provést operaci XOR mezi těmito a zašifrovanými daty a získal by část klíče (hash 2. úrovně). Z něho by sice nedokázal dopočítat hash 1. úrovně ani heslo, ovšem mohl by vygenerovat hash 3. úrovně, z něho pak 4. úrovně atd. Díky tomu by byl schopen dešifrovat data od bloku, jehož obsah znal, či např. slovníkovým útokem odhalil (viz Obr. 30).



Obr. 30 – Schéma rozluštění části dat zašifrovaných pomocí klíče z prostého víceúrovňového hashe hesla

Jednou z možností jak takovémuto útoku zabránit je před generováním hash-kódu každé další úrovně, vstup hash funkce (předchozí úroveň hashe) modifikovat takovým způsobem, který útočník nedokáže zopakovat, ovšem dešifrovací proces znalý správného prvotního hesla ano. Tato modifikace tedy musí přímo vycházet a záviset na tomto heslu. Lze například k hash-kódu hesla každé úrovně před generováním hashe další úrovně přičíst heslo samotné, avšak mnohem účinnější, bezpečnější a výpočetně efektnější je hash zkombinovat s jiným hashem. Oba totiž obsahují pseudonáhodné znaky z celé škály bytového rozsahu a také mají stejnou délku. Díky tomu lze opět využít operaci XOR.

Vzniklý blok bytů poslouží pouze jako vstup pro vytvoření hashe následující úrovně a sám o sobě nebude nikde použit. Pro tento účel ideálně poslouží hash hesla 1. úrovně, který by v tomto případě neměl být sám o sobě použit pro šifrování žádného z bloků dat a sloužil pouze pro kombinování s hashi vyšších úrovní (viz Obr. 31).



Obr. 31 – Schéma šifrování dat pomocí klíče z kombinovaného víceúrovňového hashe hesla

Pokud tedy útočník bude znát určitou část dat, dokáže sice rozluštit klíč, kterým byla tato část zašifrována, ale již nedokáže určit klíč pro následující (a samozřejmě ani předchozí) blok dat. K tomu by potřeboval znát buď hash hesla 1. úrovně nebo zdroj hashe pro klíč následujícího bloku dat. Oba požadavky by znamenaly určení reverze hashe, což je již dle základní definice této funkce výpočetně nemožné.

Jediný způsob, jak neznámé části dat určit je „uhodnout" heslo, popřípadě jeho hash 1. úrovně. Zde již závisí hlavně na „síle" zvoleného hesla, tj. jak dokáže odolat před slovníkovým útokem a útokem hrubou silou. Pro volbu snadno zapamatovatelných hesel odolávajících těmto druhům útoku existuje řada postupů (např. viz [83]).

Heslo delší než hash-kód nemá u jednorázového použití smysl, jelikož v takovém případě se útočníkovi vyplatí spíše určit 1. hash-kód tohoto hesla, neboť heslo samotné k rozluštění dat nepotřebuje. Pokud by však heslo mělo být používáno opakovaně, útočníkovi se vyplatí hledat spíše to, než jeho hash-kód, i když jeho určení bude o něco náročnější.

Jedinečný klíč pro každou zprávu

Poslední úskalí zůstává v požadavku na jedinečnost klíče pro každou komunikaci (každá data). Pro dosažení tohoto požadavku existuje několik možností. Je-li například souběžně s rychlým datovým potencionálně odposlouchávaným kanálem soustavně otevřen i další zabezpečený, byť třeba pomalý kanál, může být před zasláním každé datové zprávy tímto kanálem zasláno i nové heslo.

Druhou možností je, v případě souvislé komunikace mezi dvěma účastníky, používat stále další a další úrovně původního hesla a nezačínat je generovat vždy znovu od začátku. Klíčem pro šifru pak bude neustále jiný klíč. Nevýhodou ovšem je nezbytnost pamatovat si úroveň hashe, na které komunikace skončila a nemožnost zapojení více účastníků (při architektuře klient-server) do komunikace, aniž by neustále museli zbytečně dopočítávat příslušnou úroveň hashe dosaženou ostatními.

Jinou možností je využití faktu, že na kompletní změnu celého klíče sestávajícího se z mnoha úrovní hashe stačí změna jediného bitu v heslu či hashe 1. úrovně. Při tomto druhu změny může být její popis součástí zprávy obsahující i zašifrovaná data. Může se jednat například o dodatečný textový řetězec, který byl k heslu přičten před výpočtem hashe první úrovně. Takovýto přídavek

hesla se nazývá „salt" a je zároveň dobrou pomůckou proti slovníkovým útokům postaveným na předgenerovaných hashových slovnících (viz str. 103). Jeho zveřejnění přitom nijak nesnižuje obtížnost dešifrování dat, neboť pro výpočet klíče je stále nezbytné znát i původní část hesla.

Díky saltu je klíč pro data pokaždé kompletně jiný a ani případné určení jeho části, nebo i klíče celého, v některém z minulých datových přenosů nesnižuje zabezpečení přenosu dat budoucích, aniž by muselo dojít ke změně hesla. Je pouze třeba zabezpečit, aby byl salt pokaždé jiný, čehož lze například dosáhnout pomocí generátoru GUID hodnot.

Způsobů jak heslo a salt zkombinovat je nespočet. Například, pokud heslo bude „`heslo`" a salt „`SALT`" lze použít tyto způsoby:

- `hesloSALT`
- `SALTheslo`
- `hSeAsLlTo`
- `HASH(heslo) ⊕ HASH(SALT)`
- `...`

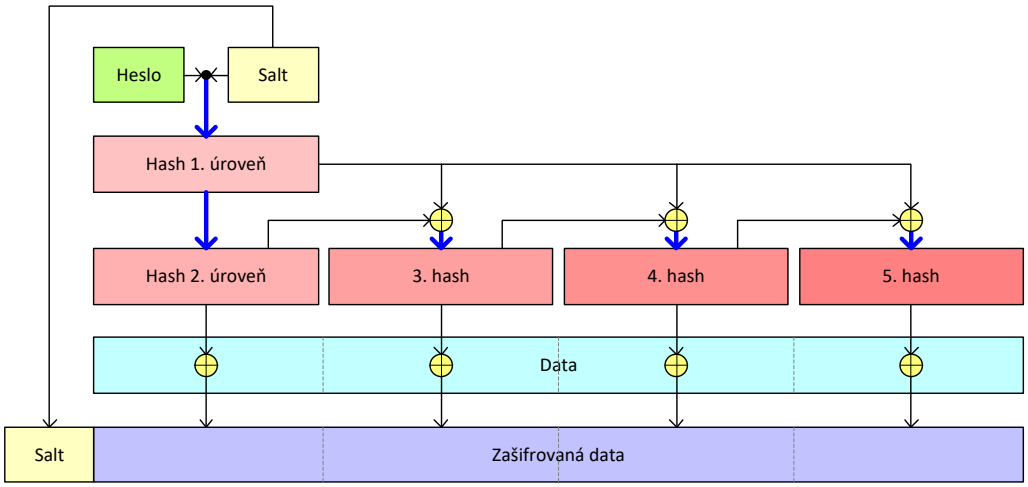
Dlouhodobé uchovávání saltu

Pokud není zaručeno, že pro každá šifrovaná data bude použito jiné heslo, je pro originalitu každého klíče nezbytné použít salt. Při komunikaci dvou stran může být salt jedním z předávaných parametrů zprávy. V případě použití šifry na dlouhodobě samostatně uchovávané soubory je třeba salt uložit tak, aby byl při potřebě soubor dešifrovat kdykoli dohádatelný a konkrétnímu souboru přiřaditelný, tedy nejlépe přímo do tohoto souboru.

Ani pozice saltu uloženého v souboru šifrovaných dat nemusí být vždy stejná a tím také dále znesnadňovat prolomení šifry. Salt totiž nemusí být přidán pouze na začátek nebo konec zašifrovaných dat, ale i na libovolnou pozici. Je také možné salt rozdělit na jednotlivé byty a ty různě mezi data rozmístit. Jejich pozice, případně saltu jako celku, i způsobu rozmístění by pak měla být jednoznačně určitelná na základě hesla, aby jej bylo možné při dešifrování zpětně dohledat a oddělit od dat. Jelikož by salt měl být zcela náhodný, stejně jako zašifrovaná data, nemělo by dojít k jeho identifikaci a pokusu o dopočítávání hesla. Každopádně nic nebrání použití zpětně nevypočitatelného postupu (rozmísťování saltu na základě hodnot hash-kódu hesla). Také je možné salt před uložením k datům zašifrovat pouze pomocí hashe hesla (náhodné bity zašifrované náhodnými bity bez klíče dešifrovat nelze).

V případě uchování saltu u dat ovšem již nejde o šifru `1:1`[66], ale `1:(1 + délka saltu)`. Tento postup také komplikuje blokové zpracování dat. Dešifrovací algoritmus musí nejprve přečíst salt a až po té s jeho pomocí může postupně dešifrovat data. Aby tedy nebylo nezbytné dvojí zpracování dat, je nejvýhodnější salt uložit hned na jejich začátek (viz Obr. 32).

[66] šifra 1:1 - jeden bit zdrojových dat je zašifrován právě do jednoho bitu zašifrovaných dat



Obr. 32 – Schéma šifrování dat se zapojením saltu

Při takto uloženém saltu může šifrování i dešifrování dat probíhat obvyklým blokovým i proudovým způsobem. Začátek dat, kde je uložen salt, ovšem musí být přečten vždy a nelze tak dešifrovat pouze určité úseky dat nezávisle na pořadí.

4.9.4 Rychlost

Navržený šifrovací algoritmus přímo staví na specifickém klíči, jenž tvoří víceúrovňový hash, který je navíc v každé úrovni znovu kombinován s hashem 1. úrovně. Výpočet hashe není samozřejmě triviální výpočetní operací, ale komplikovaným algoritmem, který zabírá určitý výpočetní čas. Je tedy zřejmé, že na rychlosti, či spíše „pomalosti" šifry, bude mít nevětší podíl právě výpočet tohoto klíče. Jelikož šifra dokáže pracovat s libovolným hashovacím algoritmem, byl pro ni zvolen ten nejrychlejší.

Za tímto účelem, bylo provedeno následující srovnání (viz Tab. 25). V něm byly pomocí jednotlivých hashovacích algoritmů (řádky tabulky) výše uvedenou metodou vypočteny víceúrovňové klíče daných délek (sloupce tabulky). Pokus byl opakován vždy 3x a výsledný průměr (buňky tabulky) je tedy v milisekundách vyjádřený čas potřebný na výpočet každého klíče. Základní vlastnosti hash-kódu (irreverzibilnost a náhodnost) byla již brána jako dále netestovaná samozřejmost.

Měření bylo prováděno za identických podmínek, tj. na stejném počítači při týchž běžících procesech. Parametry testovacího stoje uvádí Příloha 25. Pro výpočet klíče byly použity algoritmy integrované v programovacím prostředí Microsoft .NET Framework 4.0, jazyk C#, ve kterém byl implementován i následně testovaný šifrovací algoritmus.

Velikost [MB]	1	2	3	5	10	20	30	50	100
MD5	519	1 034	1 539	2 532	5 060	10 084	15 105	25 206	50 313
SHA-1	430	842	1 272	2 098	4 216	8 410	12 613	21 016	42 006
SHA-256	291	560	890	1 406	2 820	5 682	8 648	14 168	28 523
SHA-384	295	560	862	1 368	2 757	5 549	8 225	13 734	27 379
SHA-512	211	420	655	1 046	2 083	4 221	6 309	10 496	20 874

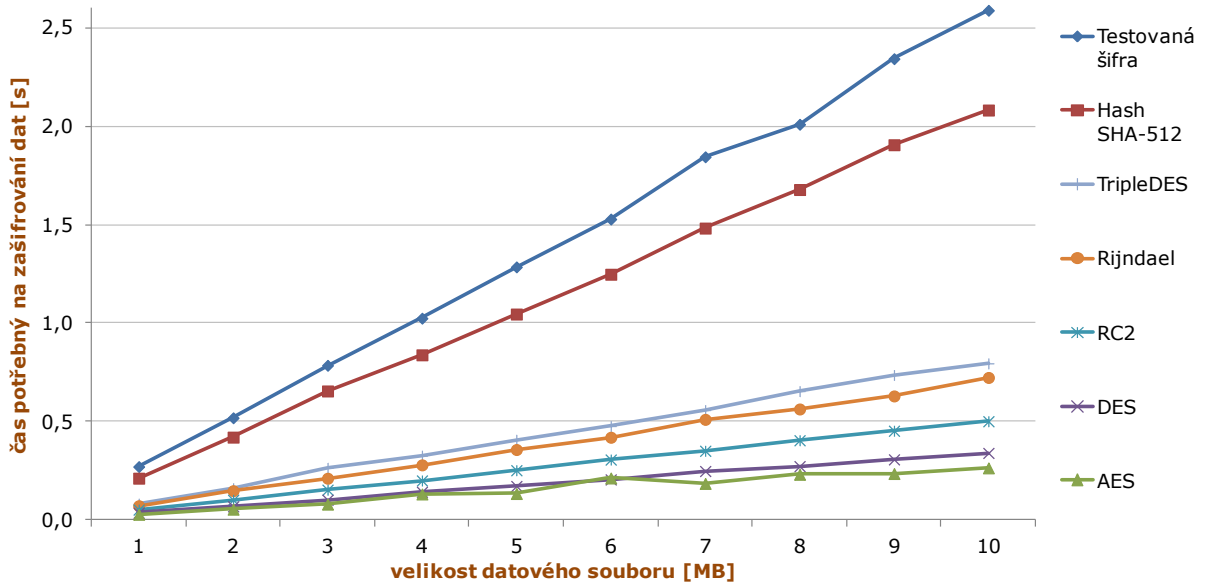
Tab. 25 – Porovnání rychlostí výpočtů [ms] víceúrovňového hashe dané délky jednotlivými algoritmy

Z porovnání v Tab. 25 vyplývá, že nejrychlejším z testovaných hashovacích algoritmů je SHA-512. Ten byl následně použit i pro test srovnání rychlostí tohoto a již existujících šifrovacích algoritmů (viz Tab. 26). Porovnání bylo provedeno podobným testem a za stejných podmínek, jako srovnání rychlostí hashovacích algoritmů. Tentokrát ovšem již nešlo pouze o výpočet hodnot v rámci paměti počítače, ale zdrojová data byla čtena ze souboru na pevném disku a výsledná zašifrovaná data na disk znovu ukládána. Data byla načítána, zpracována a ukládána proudově, po blocích o velikosti 5 kB.

Velikost [MB]	1	2	3	5	10	20	30	50	100
AES	28	54	79	133	264	531	818	1 263	2 797
DES	35	68	101	172	340	703	1 030	1 709	3 476
RC2	52	101	153	252	501	1 012	1 523	2 547	5 114
Rijndael	71	148	211	357	723	1 452	2 147	3 581	7 557
TripleDES	82	160	264	406	795	1 620	2 403	4 013	8 300
Testovaná šifra	**270**	**518**	**785**	**1 285**	**2 590**	**5 113**	**7 703**	**12 764**	**25 825**

Tab. 26 – Porovnání rychlostí šifrování [ms] datových souborů dané velikosti jednotlivými algoritmy

Porovnání v Tab. 26 a Graf 23 ukazuje, že navržený šifrovací algoritmus je oproti ostatním výrazně pomalejší. Zhruba 81% tohoto času ovšem zabírá výpočet klíče, byť nejrychlejším z testovaných hashovacích algoritmů SHA-512. Použitím rychlejšího hashovacího algoritmu by tedy mohlo dojít i k výraznému zrychlení této šifry. Tuto vlastnost by mohl přinést např. připravovaný hashovaní algoritmus SHA-3[67].



Graf 23 – Graf porovnání rychlostí šifrování datových souborů dané velikosti jednotlivými algoritmy (zahrnuta je i rychlost generování klíče pomocí hashovacího algoritmu SHA-512)

Ostatní šifrovací algoritmy mají navíc výhodu, neboť jsou implementovány v .NET řízeném (managed) kódu, ale jsou zabudovány přímo do operačního systému, který je zpracovává pod aplikační vrstvou bez zbytečné režie přes Windows CryptoAPI [97 str. 3]. Tento rozdíl je patrný při porovnání rychlostí šifrování AES a Rijndael, což je v obou případech stejný algoritmus, avšak

[67] soutěž o SHA-3 viz [w11]

Rijndael je jako jediný reimplementován .NET, stejně jako testovaná šifra. Ostatní algoritmy by pak za stejných podmínek byly také výrazně pomalejší, zhruba v témže poměru. Nutno však dodat, že rychlejší CryptoAPI zde i v Tab. 25 využívají také všechny hashovací algoritmy.

Rychlost šifrování tedy limituje použití při klasickém šifrování v reálném čase, například při on-line komunikaci dvou stran, kde je rychlost spojení jedním z hlavních parametrů. Její využití by tak mohlo být spíše v případech, kdy má úroveň zabezpečení vyšší prioritu, nežli čas, potřebný na zašifrování.

4.9.5 Shrnutí

Na základě původního principu Vernamovy dokonalé šifry byla navržena modifikace práce s klíči, jejichž správa zatím velmi komplikuje její praktické využití. Při kombinaci s moderními hashovými algoritmy lze šifrovat data pomocí matematicky prokazatelně zpětně nevypočitatelných postupů a přitom i opakovaně používat „jednoduchá" hesla. Síla šifry je pak vždy přímo úměrná síle zvoleného hesla.

Implementace uvedeného postupu je přitom programově velmi snadná. Rychlost šifrování a dešifrování dat nejvíce závisí na rychlosti výpočtu hash-kódu, tedy na zvolené hash funkci. Připravovaná funkce SHA-3 přitom slibuje mnohem rychlejší výpočet a zároveň i vyšší bezpečnost než ty stávající, byť dosud neprolomené. [96]

Navržená šifra lze, díky své stávající nižší rychlosti, používat pro ochranu přenosu dat přes veřejnou síť internet zatím pouze v případech, kdy nevadí zpomalení potřebné pro šifrování dat. V případě šifrování archivů a souborů pro dlouhodobou úschovu, kde je obvykle přednější jejich bezpečnost před časem potřebným na zašifrování, může tato šifra nalézt své uplatnění již nyní.

4.10 Efektivní předávání dat v internetu [◎ IV.4]

(publikováno v [ap-7])

Během návrhu administračního rozhraní aplikace, vznikl požadavek na jednoduchý způsob předávání dat mezi webovým serverem a klientskou aplikací. Přitom bylo žádoucí tento postup unifikovat, aby pokrýval veškeré potřeby nejen dané aplikace, ale byl univerzálně použitelný i v jiných projektech.

Ideálním kandidátem pro takovouto oboustrannou výměnu relačních dat by byl **DataSet**. *Objekt třídy DataSet v .NET představuje úložiště určitých dat v paměti počítače a je jako objekt předáván mezi střední vrstvou a klientskou aplikací případně webovou službou. Objekty třídy DataSet se dají snadno serializovat a deserializovat do a z dokumentů XML. To znamená, že data, společně se souvisejícími údaji o schématu, lze přesouvat mezi vrstvami aplikace velmi volným způsobem.* [98 str. 220]

Objekt DataSet ovšem není v Silverlightu podporován, z důvodů minimalizace instalačního balíčku Silverlight verze .NET Frameworku a ani do budoucna se jeho přidání do Silverlight neplánuje [w49]. Z tohoto důvodu vznikl nezávislý projekt, který se snažil vytvořit DataSet pro Silverlight, jež měl obsáhnout veškerou jeho funkčnost a umožnit výměnu dat s webovým serverem pomocí serializovaných[68] dat kompatibilních s originálním DataSetem (viz [w1]). Základem byl překlad serializovaných dat na dynamicky vytvářené objekty přímo technikami jazyka MSIL[69]. Byť tyto objekty plně spolupracovaly se standardními datovými komponentami, bylo velmi obtížné jejich doplnění a práce s nimi, neboť jejich třídy v době návrhu aplikace neexistovaly. Tento projekt ovšem jeho autoři nedokončili (např. nefunkční relační vztahy mezi tabulkami) a předčasně jej uzavřeli.

Architektura **WCF**[70] umožňuje přenášet objekty, jejichž třídy jsou definovány na straně serveru. Při vývoji klientské části aplikace dokáže návrhové prostředí z WSDL dokumentu zjistit strukturu těchto tříd, automaticky je vytvořit a později i aktualizovat jejich Silverlight verzi. Má ale například základní nevýhodu, že při serializaci nedodržuje reference mezi objekty, resp. každý podobjekt, na který se odkazuje ten serializovaný, je také serializován a to pokaždé znovu. Pokud se tedy například přes WCF bude posílat seznam výsledků ze zkoušení, a každý z výsledků bude mít i referenci na objekt uživatele, který byl zkoušen, pak se pro 100 různých výsledků pošlou data o 100 uživatelích, i kdyby třeba patřily všechny výsledky témuž uživateli. Při deserializaci na klientské straně pak samozřejmě vznikne 100 nezávislých objektů třídy *Uživatel* s týmiž daty. Tento fakt jednak zvyšuje objemnost přenášených dat a také značně komplikuje jejich objektovou reprezentaci a zpracování.

Služby **RESTful**[71] předávají data ve formátu XML nebo JSON[72], přičemž veškeré požadavky na ně jsou součástí URL adresy. Ta obsahuje například i podmínky filtru, autentizační klíč apod. Součástí požadavku zasílaného přes http protokol mohou být samozřejmě i v URL přímo neuvedené

[68] *serializace ukládá stav objektu do zvoleného úložiště* (např. jako textový řetězec) *a deserializace z něho zpětně rekonstruuje originální objekt* [131]
[69] MSIL, zkráceně také IL – Intermediate Language – nízkoúrovňový jazyk s jednoduchou syntaxí založenou na číselných kódech, do kterého jsou překládány (kompilovány) programy .NET před jeho převodem do přímo spouštěného strojového kódu [46 str. 12]
[70] WCF – Windows Communication Foundation
[71] REST – REpresentational State Transfer
[72] JSON – JavaScript Object Notation, viz [w37]

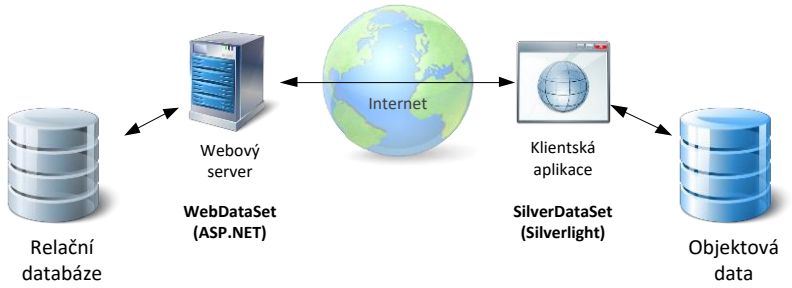
údaje, přiložené metodou POST, PUT a DELETE. Ty jsou využívány pro úpravu dat (POST = vložení nového záznamu nebo kolekce, PUT = změna dat existujícího záznamu nebo kolekce a DELETE = výmaz záznamu nebo kolekce). [99] Služby RESTful obvykle poskytují servery, pro které je žádoucí podpora vývoje nezávislých aplikací pracujících s jejich daty (např. Amazon, viz [w4]).

Výměna dat mezi aplikacemi různých tvůrců v současnosti nejčastěji probíhá pomocí XML (popřípadě JSON) dokumentu. Jeho strukturu obvykle definuje autor jedné z komunikujících aplikací a ostatní se jí přizpůsobí. **OData**[73] je jedním z protokolů, které standardizují jak tuto strukturu dat, tak i formu dotazování se a manipulaci s nimi. Aplikace, které podporují OData výměnu dat jsou pak spolu schopny komunikovat bez nutnosti zásadního přizpůsobování vstupů či výstupů. Online aplikace podporující OData jsou podobně jako služby RESTful standardně dotazovány přes URL, včetně výše zmíněných obtíží se serializací dat.

Vzhledem ke zmíněným komplikacím při přenosu objektových dat bylo navrženo a vytvořeno vlastní řešení – Silverlight DataSet.

4.10.1 Silverlight DataSet

Pro účely výměny dat byly vytvořeny dvě knihovny tříd. První je určena pro platformu Silverlight na klientské straně a zprostředkovává sestavování požadavků na data, deserializaci příchozích dat, jejich správu a evidenci seznamu změn. Druhá knihovna na straně serveru přijímá požadavky, serializuje požadovaná data a zpracovává seznamy změn (viz Obr. 33).



Obr. 33 – Schéma přenosu dat mezi serverovou a klientskou částí DataSetu

Část pro Silverlight definuje třídu `DataObjectBase`, která poskytuje základní vlastnosti pro jednotlivé datové záznamy. Veškeré třídy, zastupující tabulky v relační databázi na serveru, musí být odvozeny z této třídy.

Provázání tříd se záznamy v tabulkách a jejich vlastností se sloupci těchto tabulek je prováděno přímo v kódu dané třídy. Každá taková třída musí mít atribut[74], ve kterém lze definovat, základní parametry pro provázání s tabulkou, tj. především její databázový název. Podobně tomu tak je i u vlastností této třídy. Ty také mohou mít atributy, které určují se kterým sloupcem dané tabulky je tento svázán a zároveň i další příznaky, např. je-li hodnota pouze pro čtení, je-li povinná (`not null`), odložená a v případě textových řetězců jejich maximální možnou délku (viz Kód 25). Vlastnosti, které tento atribut nemají, jsou DataSetem ignorovány a mohou být tvůrcem aplikace libovolně použity.

[73] OData – Open Data Protocol, viz [w51]
[74] vlastní atributy umožňují deklarativně anotovat konstrukce kódu metadaty, která pak mohou být za běhu dotazována a dynamicky interpretována jiným kódem [130 str. 435]

```
[DataObjectAttribute("A_USERS")]
public class User : DataObjectBase
{
    private string prijmeni;
    [DataObjectPropertyAttribute("SURNAME", true, 30)]
    public string Prijmeni
    {
      get { return prijmeni; }
      set { prijmeni = value; ValueChanged("Prijmeni"); }
    }
    ...
}
```

Kód 25 – Ukázka kódu definice třídy provázané s databázovou tabulkou pomocí atributů

Kód 25 ukazuje způsob mapování pomocí atributů [100 str. 449]. Třída `User` je zde provázána s tabulkou `A_USERS` a vlastnost `Prijmeni` na sloupec `SURNAME`. Zároveň je v konstruktu atributu určeno, že příjmení je povinná položka (2. parametr) a jeho maximální délka je 30 znaků (3. parametr).[75] Volání metody `ValueChanged` v `set` kódu vlastnosti je zde jednak pro možnou datovou návaznost vlastnosti s editační vizuální komponentou, aby na případnou změnu hodnoty mohla patřičně reagovat (viz [76 stránky 67-90]), a zároveň jsou s její pomocí evidovány změny dat, pro jejich pozdější uložení zpět na server do databáze.

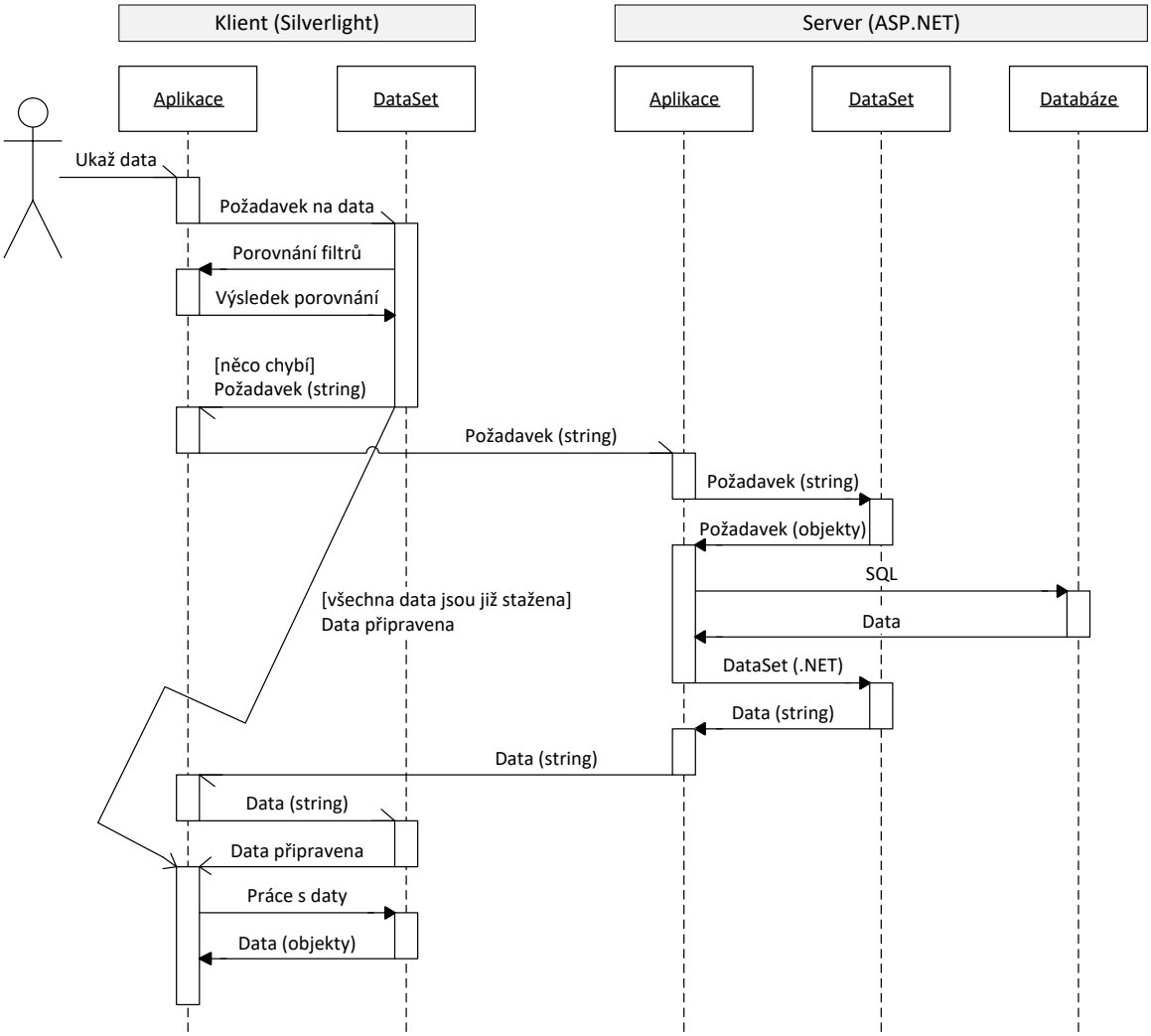
Aby nebylo nutné seznamy těchto datových tříd jinde znovu sepisovat, provádí sestavení jejich seznamu DataSet sám pomocí reflexe [101 str. 489]. Stačí při vytváření hlavní zprostředkovatelské třídy předat `Assembly`[76] ve které jsou datové aplikační třídy definovány a ty jsou na základě příslušných atributů automaticky nalezeny, zmapovány a uloženy do strukturovaných seznamů, aby s nimi mohl následně DataSet bez dalších prodlev pracovat.

Veškerá data jsou stahována ve svém původním relačním formátu (viz Příloha 13) a třídy, které je na klientské straně zpracovávají, s tím musí počítat. Reference na jiné tabulky (třídy) jsou tak v základu realizovány pouze přes hodnotové vazby (cizí klíče), nikoli objektově, avšak tuto funkcionalitu není díky funkci LINQ[77] to Object (viz [102 str. 63]) těžké odprogramovat (viz Příloha 14).

4.10.2 Načítání dat

Jakmile aplikace potřebuje nějaká data (např. chce-li uživatel zobrazit určitý přehled), je požadavek na ně předán DataSetu. Ten může být buď objektový, nebo ve formě textového řetězce či XML elementu dané struktury (viz kap. 4.11.3, str. 133). Sekvenční diagram na Obr. 34 ukazuje, jak je následně požadavek na data asynchronně zpracován.

[75] podobný způsob definice vazeb používá také např. XPO od DevExpress [w53]
[76] *assembly (sestavení) je logická jednotka obsahující kompilovaný kód určený pro platformu .NET* [46 str. 25]
[77] LINQ - Language INtegrated Query (integrovaný jazyk pro dotazování)



Obr. 34 – Sekvenční diagram čtení dat přes DataSet

Požadavek může obsahovat odkazy na více tabulek současně. Pro každou z nich dojde k porovnání filtru pro požadovaná data s filtry dat již stažených. Vyhodnotí-li DataSet, že byla všechna data stažena již dříve, oznámí rovnou aplikaci, že může pokračovat v práci. V opačném případě sestaví požadavek do textového řetězce, který předá aplikaci. Ta jej odešle na server (např. přes webovou službu), kde je předán webové části DataSetu, jež ho deserializuje na strukturovaný datový objekt a ten vrátí serverové aplikaci. Aplikace zpracuje požadavek tak, že z databáze načte požadovaná data do klasického objektu *DataSet*, který je součástí .NET Frameworku [42 str. 350]. V tuto chvíli také aplikace může přihlížet na autorizaci uživatele, který data požaduje, má-li pro jejich čtení příslušné oprávnění. *DataSet* s načtenými daty je předán zpět třídě `WebDataSet`, která tato data serializuje do textového řetězce, jež vrátí webové části aplikace, aby jej zaslala zpět klientovi.

Klientská aplikace přijme řetězec s daty a předá je své části Silverlight DataSetu, který je zpracuje (deserializuje do příslušných tříd a přidá záznamy do objektových seznamů) a následně oznámí aplikaci, že je vše připraveno. Aplikace pak může začít s daty pracovat.

Z celého procesu tedy vyplývá, že veškerá data zasílaná prostřednictvím internetu jsou posílána jako textový řetězec (string). Přitom však aplikace pracuje pouze s objektovými daty. O veškerou práci se serializací, deserializací a správou těchto dat se stará příslušná knihovna Da-

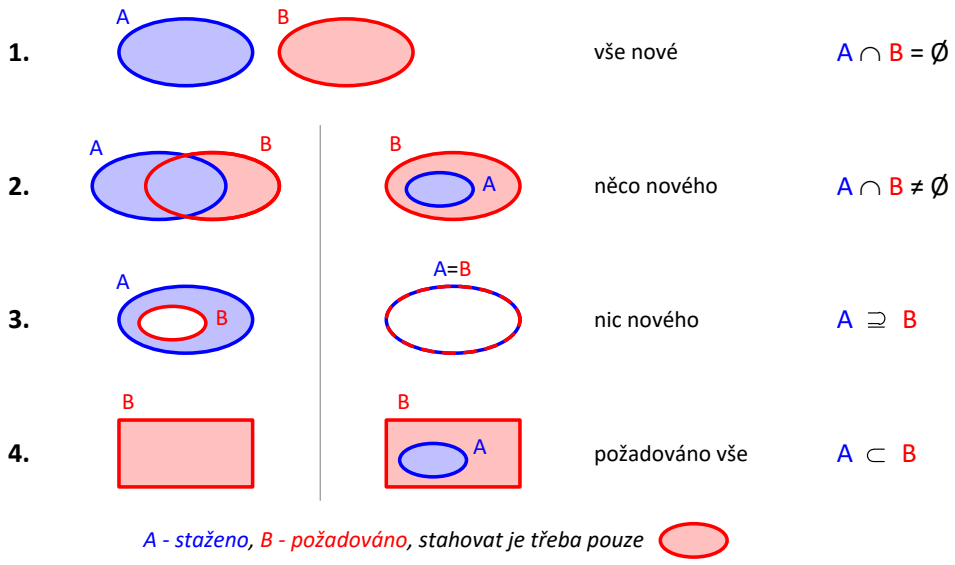
taSetu. Spojení s databází a zasílání a přijímání dat zajišťuje sama aplikace. DataSet je tak nezávislý na prostředí ve kterém je používán, a jen maximálně usnadňuje veškerou rutinní práci s daty.

Filtry

Filtry jsou důležitou součástí DataSetu. Hlavním předpokladem totiž je, že se vždy ze serveru stahují pouze ta data, která jsou právě potřeba a aby se pokaždé nemusela stahovat kompletní data celé tabulky, jsou tu právě filtry, aby určily, jaká její část (horizontální) je aktuálně žádána.

Filtry jsou řešeny na bázi textového řetězce, který definuje tvůrce aplikace používající DataSet a vzhledem k tomu, že přímé spojení s databází realizuje také on, je vyhodnocování takto definovaných filtrů pouze v jeho kompetenci. DataSet na klientské straně sám rozezná pouze dva případy a to, když je použit shodný filtr opakovaně nebo pokud již byla data tabulky stažena kompletně. Oba případy jsou signálem, že pro daný požadavek na data již není třeba stahovat nic nového. Ostatní typy porovnání filtrů jsou na tvůrci aplikace.

DataSet vystavuje delegáta `CompareFilters`, který, pokud je nastaven, je volán vždy, když se mají porovnat dva filtry, jež DataSet nedokáže sám vyhodnotit. Aplikace mu pak přes výčet (enum) `FiltersCompareResult` vrátí výsledek tohoto porovnání. Možné výsledky ukazuje Obr. 35.



Obr. 35 – Možné typy výsledků porovnání filtru dat již stažených a požadovaných

V 1. případě je tedy třeba stáhnout vše. Ve 2. pouze část dat, čili DataSet k filtru přidá i seznam ID již stažených záznamů zapsaný v mini-jazyku (viz kap. 4.12.1, str. 148), aby se tyto záznamy znovu nestahovaly. Ve 3. případě, kdy není třeba stahovat nic, je tento požadavek na data rovnou vyhodnocen jako splněný, bez nutnosti kontaktovat server. Poslední 4. případ je pak zpracován stejně jako 2., ovšem pro příště je zaznamenáno, že data tabulky jsou již stažena kompletně a nebude se pro ni již nic dalšího stahovat, což je výhodné například u číselníků.

Filtry se tedy ve všech případech vyjma 3. zasílají ve svém nezměněném znění na server, kde jsou po deserializaci požadavku předány této části aplikace, aby načetla data pouze podle nich vymezená. Ve 2. a 4. případě je navíc přiložen i seznam ID záznamů (jako `IEnumerable<int>`), které již není potřeba stahovat.

Odložená data

Pole některých tabulek mohou obsahovat rozsáhlejší data s proměnlivou délkou. Např. RDBS Firebird podporuje datový typ BLOB pro ukládání binárních dat nebo velmi rozsáhlých textů, které nelze jednoduše ukládat v položkách některého jiného standardního datového typu. [86]

Takováto data není vhodné přenášet hromadně pro více záznamů najednou, jednak kvůli jejich datové objemnosti a také vzhledem k tomu, že se v datových přehledech (tabulkách) stejně přímo nezobrazují (např. QML data otázky). Obvykle se zobrazí až při požadavku detailního výpisu jednotlivých záznamů, čili až pak má smysl tato data stáhnout a to pouze pro jeden konkrétní záznam.

Takovéto typy hodnot se dají rozlišit v atributu vlastnosti při definici třídy pomocí dalšího nepovinného parametru. DataSet je pak nestahuje při požadavku na hromadná data, ale pouze na zvláštní vyžádání. To se musí týkat jednoho objektu určeného ID a obsahovat seznam položek tohoto druhu hodnot (sloupců tabulky), které se mají stáhnout (viz Kód 27 na str. 134). Zároveň je u každého objektu zvlášť evidováno, které položky odložených dat již byly staženy, aby je bylo možné rozlišit od prázdných (NULL), byť stažených dat, a nestahovat je tak zbytečně znovu.

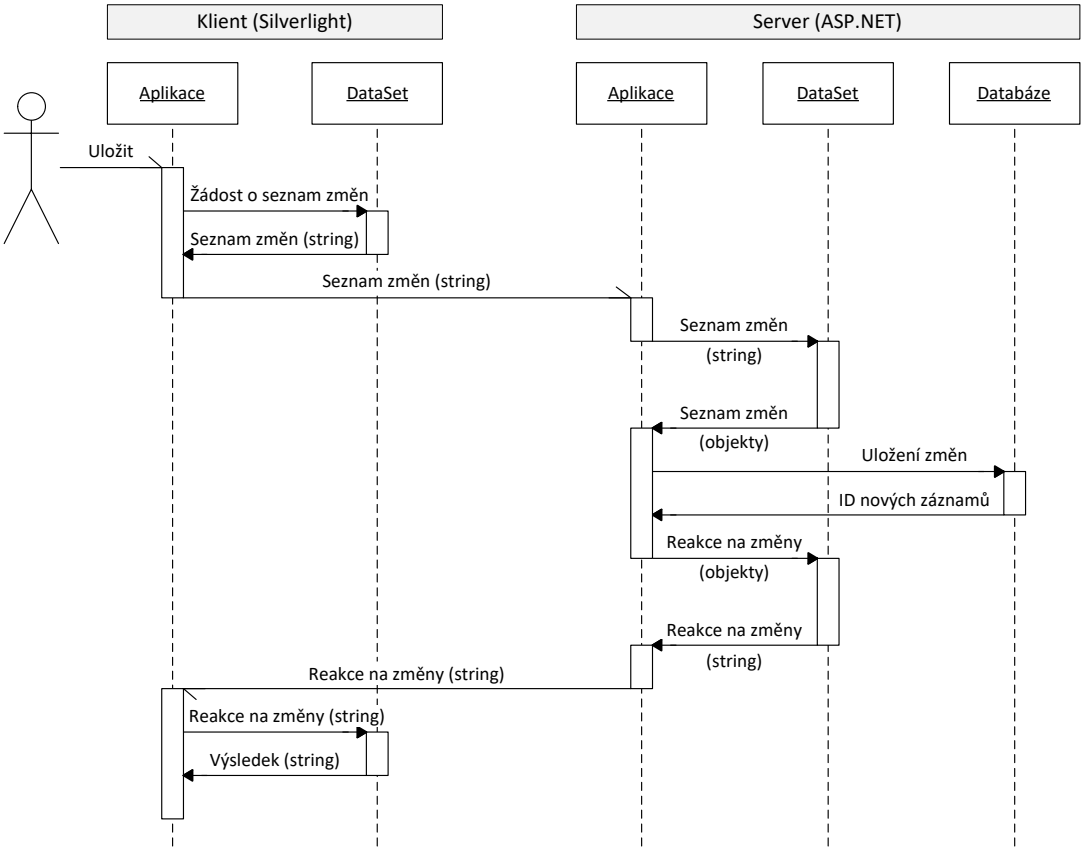
4.10.3 Ukládání změn

Veškeré změny v datech se znamenávají a evidují ve zvláštním seznamu. V případě opakovaných změn týchž dat je samozřejmě toto rozpoznáno a evidována je pouze poslední změna, s přihlédnutím k jejímu typu. Ty jsou klasicky tři (seřazeno dle priority změn):

- **Delete** – Je-li datový záznam vymazán, předchozí změny netřeba dále evidovat, stačí si uchovat pouze ID tohoto záznamu. Byl-li záznam předtím přidán a dosud neuložen, pak je ze seznamu změn odstraněn úplně.
- **Insert** – Nově vložený záznam vždy na server odesílá všechny své hodnoty záznamu, i když je později (před odesláním) ještě upraven.
- **Update** – Při změně dat záznamů se eviduje, které položky záznamu byly změněny a pouze jejich nové hodnoty se odesílají na server.

Obr. 36 ukazuje průběh procesu ukládání změn na server. V okamžiku kdy si uživatel vyžádá uložení provedených změn na server, aplikace si od DataSetu nechá sestavit serializovaný seznam změn. Ten odešle na server, kde je předán k deserializaci webové části DataSetu. Objekt, který vrátí, obsahuje „srozumitelný" seznam změn, jež aplikace uloží do databáze. Tato operace by měla být prováděna v transakci [103], aby v případě chyby nedošlo k neúplné změně dat. I tu ale samozřejmě může aplikace podporovat. V této fázi lze zároveň kontrolovat i případné konkurenční změny záznamu jiným uživatelem např. prostřednictvím časové známky.

V průběhu ukládání dat, případně po potvrzení transakce (záleží na systému připojení k databázi), je do objektu seznamu změn u nově vkládaných objektů doplňováno jejich databází nově přidělené ID. Kromě něho je třeba také pokaždé potvrdit, že uložení změny proběhlo v pořádku, případně uvést chybové hlášení, které při pokusu o uložení vzniklo.



Obr. 36 – Sekvenční diagram ukládání změn dat přes DataSet

Objekt seznamu změn doplněný o tyto reakce na změny je předán opět DataSetu k serializaci a výstupní řetězec je odeslán zpět klientské aplikaci. Ta reakční řetězec předá své verzi DataSetu, která vrátí buď prázdný řetězec na důkaz toho, že vše proběhlo v pořádku, nebo přehledně sepsaná chybová hlášení, která během procesu vznikla.

4.10.4 Shrnutí

DataSet pro Silverlight je užitečná pomůcka pro přenos dat mezi aplikacemi Silverlightu a serverem. Snaží se minimalizovat spojení se serverem a tím jednak snížit jeho zatížení, ale i zrychlit práci s aplikací, čímž zvyšuje uživatelský komfort při práci s ní. Na straně serveru přijímá data ve standardní relační podobě a na klientské straně je překládá a spravuje jako objekty tříd definovanými tvůrcem aplikace. Definice vztahů těchto objektů s databázovými prvky je přitom velmi snadná, neredundantní a přehledná.

V kombinaci s autentizačním protokolem (viz kap. 4.7, str. 91) a šifrováním je pak DataSet pro Silverlight bezpečným a snadno použitelným nástrojem, který umožní tvůrcům cloud aplikací soustředit se více na vývoj aplikační logiky, bez nutnosti řešit problémy spojené s přenosem a zabezpečením dat prostřednictvím veřejné sítě internet.

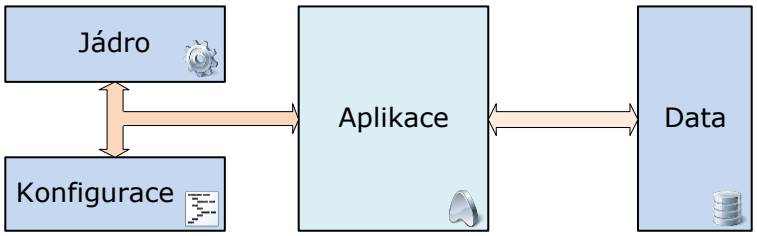
Základní třídy `SilverlightDataSet` a `WebDataSet` lze snadno začlenit do projektu a nastavit jejich použití. Mohou tak být nápomocny při vývoji aplikací, které pracují s centrálně ukládanými daty. Postupy, které byly pro implementaci těchto tříd použity, lze přitom jednoduše implementovat i v jiných prostředích a jazycích a zároveň mohou být inspirací pro další rozvoj v oblasti přenosu dat v cloud computing RIA aplikacích.

4.11 Konfigurační frameworky [◎ IV.3]

Nedílnou součástí tohoto systému je i administrační rozhraní umožňující sestavování a nastavování testů, správu otázek, uživatelů a skupin, vyhodnocování výsledků ze zkoušení apod. (viz kap. 4.3.2, str. 72). Při tvorbě této části však nebyl použit klasický postup, při němž se vytváří každé okno (stránka) aplikace zvlášť, ale vzhledem k vícenásobně se opakujícím typů oken byl vytvořen konfigurační framework, jež následně umožnil podstatnou část administrace vytvářet pouze pomocí konfiguračních dat v jednoduchém XML souboru (např. viz [104]).

Framework je softwarová struktura, která pomáhá při vývoji jiného softwarového produktu. Cílem takovéto struktury je převzetí typických problémů, s kterými se při řešení častěji potýkáme a plně je zautomatizovat. Vývojář se nemusí zdržovat zbytečnou režií kolem známých a už několikrát řešených věcí a může se plně věnovat implementaci svého konkrétního problému. [105 str. 8]

Konfigurační frameworky vychází z myšlenky, že není třeba programovat veškerý obsah celého systému, ale pouze framework (jádro), který bude jednotlivá okna aplikace generovat dynamicky na základě konfiguračních dat (viz Obr. 37). Ta mohou být uložena v krátkých souborech nebo databázi. Strukturu aplikace tak lze měnit bez nutnosti její rekompilace, aktualizace a dokonce i jejího úplného vypnutí.



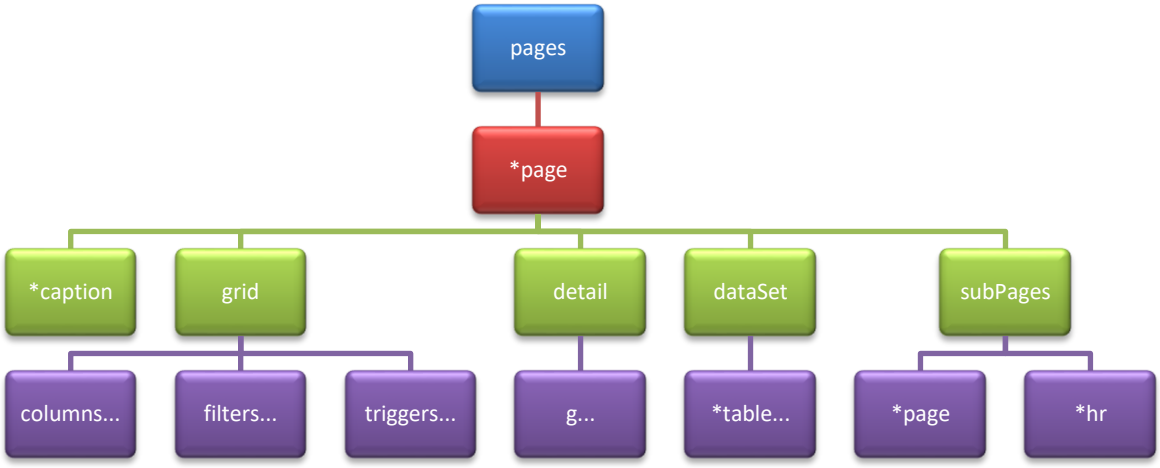
Obr. 37 – Schéma konfiguračního frameworku

Jako příklad takovéhoto frameworku zde bude popsáno právě řešení pro administraci univerzálního testovacího rozhraní. Tohoto principu však lze výhodně využít v mnoha různých typech projektů, kde se vícekrát opakují tytéž typy oken. Klasickým případem mohou být třeba desktopové informační systémy, jež se mohou sestávat i z mnoha desítek oken, pouze však několika málo druhů (nejčastěji přehled a detail).

4.11.1 Struktura konfiguračního XML

Zápis konfiguračních dat, jež definují jednotlivá okna celého systému, je možné provést mnoha způsoby. Údaje každého okna lze např. uchovávat vez zvláštním souboru, uložit je do databáze jako jednotlivé záznamy nebo je evidovat i v rámci jednoho souboru. Vzhledem k menšímu rozsahu této aplikace byl zvolen poslední způsob. Obr. 38 ukazuje základní strukturu tohoto konfiguračního XML souboru od kořenového `<pages>`.

V tomto schématu hvězdička (*) před názvem elementu znamená, že na daném místě může být element uveden opakovaně vícekrát, tři tečky (…) za názvem elementu naznačují jeho další, ve schématu nezachycené větvení. Z elementů `<grid>` a `<detail>` může být použit vždy pouze jeden, a to `<grid>` u přehledů a větvených přehledů a `<detail>` u detailních formulářů. Existují i speciální typy oken, jež pak mohou mít i zcela odlišnou strukturu (např. okno pro vyzkoušení otázky či testu, viz str. 143). Blíže všechny tyto typy budou rozebrány v dalším textu.



Obr. 38 – Struktura konfiguračního XML definujícího okna aplikace

Definici jednotlivých oken lze rozdělit na tyto údaje:

- Jaká data jsou pro dané okno zapotřebí
- Jak má okno vypadat (rozložení vizuálních komponent)
- Do kterých dalších oken se z tohoto lze dostat
- Specifické chování při určitých událostech

4.11.2 Parametry

Parametry jsou hodnoty předávané mezi jednotlivými okny. Pokud například chceme z přehledu uživatelů otevřít podpřehled s výsledky pouze zvoleného uživatele, je třeba tomuto podpřehledu předat identifikátor (ID) vybraného uživatele, aby mohl vyfiltrovat a zobrazit pouze požadovaná data. Existují též parametry globální (např. IdOrganizace, jakožto identifikátor organizace, jejíž dat se administrace provádí), které framework automaticky předá každému nově vytvářenému oknu. Předávání ostatních parametrů pak již závisí na jednotlivých oknech.

4.11.3 Načítání dat

Pro načítání dat byl použit princip popsaný v kapitole 4.10 (str. 125). Jde o vlastní původní komponentu, která zprostředkovává komunikaci klientské aplikace se serverem. Na klientské straně udržuje v paměti data strukturovaná do objektů a pomocí serializované komunikace je dokáže efektivně a neredundantně synchronizovat s centrální databází na serveru. Klientské aplikaci tedy pouze stačí předložit požadavek na data a pak už jen vyčkat potvrzení, že jsou připravena.

Tato komponenta je sice vytvořena univerzálně a lze ji použít v libovolném projektu, ovšem disponuje i pomocnými metodami, které jsou přímo určeny pro podporu konfiguračních frameworků. Konkrétně jde o možnost sestavování objektového požadavku na data z textového řetězce či XML. Textový řetězec používá vlastní mini-jazyk pro jednodušší dotazování se na data, XML verze pak umožňuje sestavovat i komplikovanější dotazy včetně podpory parametrů.

```
A_QUESTIONS(ID_GROUP=%IdTest%);A_RESULTS(ID_TEST=21);A_TESTS(*);A_SOLVINGS
```

Kód 26 – Ukázka požadavku na data čtyř různých tabulek, první dvě filtrované

Požadavky na jednotlivé tabulky se oddělují středníkem a za databázovým názvem tabulky může v kulatých závorkách následovat filtr na data, v jehož obsahu lze použít i parametry (jsou uvedené mezi znaky procent, např. `%parametr%`). Podmínky filtru lze skládat pomocí operátorů `&` (and) a `|` (or). Hvězdička či nic znamená, že se mají stáhnout všechny záznamy dané tabulky.

```
A_QUESTIONS[%Id%]:DATA,NOTE;A_RESULTS[1]:TEST_DATA,RESULT;A_TESTS[1]:*;A_SOLVINGS[1]
```

Kód 27 – Ukázka požadavku na odložená data jednoho záznamu ze čtyř různých tabulek

Hranaté závorky jsou používány při stahování dat jednoho konkrétního záznamu a uvozují jeho primární klíč ID. Ten může být určen parametrem či hodnotou. Stahování jednoho konkrétního záznamu se provádí při zobrazování jeho detailu a pouze tímto způsobem lze stáhnout tzv. „odložená data" (viz str. 130). To jsou hodnoty, které bývají datově rozsáhlejší (např. dlouhý popis, XML data apod.) a hromadně se pro více záznamů najednou nestahují. To je jednak kvůli snížení objemu přenášených dat a také proto, že se v hromadných přehledech kvůli své rozsáhlosti ani stejně nezobrazují. Jelikož ani v detailním zobrazení nemusí být vždy veškerá tato data zobrazena, je možné jejich výčet konkretizovat v rámci dotazovacího mini-jazyka za dvojtečku, případně uvést hvězdičku (nebo nic) pro stažení veškerých odložených hodnot. Dotazy na konkrétní záznamy i na hromadná data lze samozřejmě kombinovat i v rámci jednoho dotazovacího řetězce.

Tyto dotazy lze zapsat buď do atributu `data` elementu `<dataSet>`, nebo je též možné je podrobněji rozepsat do více XML podelementů (viz Kód 28).

```
<dataSet>
  <table name="A_RESULTS" refresh="1">
    <filter param="IdOrganization">
      ID_LIMIT.ID_PERIOD.ID_ORGANIZATION.ID=%IdOrganization%</filter>
    <filter param="IdUser">ID_USER.ID=%IdUser%</filter>
    <filter param="IdTest">ID_TEST.ID=%IdTest%</filter>
    <filter param="IdLimit">ID_LIMIT.ID=%IdLimit%</filter>
  </table>
  <table name="A_USERS">
    <filter param="IdOrganization">
      Contains:Organizations~ID_ORGANIZATION.ID=%IdOrganization%</filter>
    <filter param="IdUser">ID=%IdUser%</filter>
  </table>
  <table name="A_TESTS">
    <filter param="IdOrganization">
      Contains:Organizations~ID_ORGANIZATION.ID=%IdOrganization%</filter>
    <filter param="IdTest">ID=%IdTest%</filter>
  </table>
  <table name="A_COMPUTERS" />
  <table name="A_TEST_LIMITS">
    <filter param="IdLimit">ID=%IdLimit%</filter>
    <filter param="IdTest">ID_TEST.ID=%IdTest%</filter>
  </table>
  <table name="A_PERIODS" refresh="1">
    <filter param="IdTest">
      Contains:Limits~ID_TEST.ID=%IdTest%|Contains:LimitsResults~ID_TEST.ID=%IdTest%</filter>
    <filter param="IdLimit">
      Contains:Limits~ID=%IdLimit%|Contains:LimitsResults~ID=%IdLimit%</filter>
  </table>
  <table name="A_ORG_GROUPS">
    <filter param="IdOrganization">ID_ORGANIZATION.ID=%IdOrganization%</filter>
  </table>
</dataSet>
```

Kód 28 – Ukázka definice požadavků na data pro přehled výsledků

Při podrobnějším rozepsání do XML se požadavky na jednotlivé tabulky udávají do elementu `<table>`, kde opět atribut `data` může obsahovat podrobný dotaz v popsaném mini-jazyku. Lze též ovšem pouze do atributu `name` uvést databázový název tabulky a filtry rozepsat do dalších podelementů `<filter>`. Atribut `refresh` signalizuje, která data se mají znovu načíst ze serveru při zavolání funkce obnovy dat z daného okna.

Jednotlivé části filtru se skládají s operátorem `and`. Atribut `param` v elementu `<filter>` nejen signalizuje, které parametry bude v dotazu třeba nahradit jejich hodnotami, ale zároveň určuje, že nebude-li daný parametr okno znát, nebude tato podmínka do filtru vůbec zahrnuta. To je zároveň hlavní rozdíl mezi XML zápisem a dotazovacími řetězci, kdy jsou vždy použity všechny podmínky filtru. Díky tomu lze totéž okno spouštět se zcela odlišným obsahem. V tomto případě (viz Kód 28) jde o přehled výsledků ze zkoušení, které mohou být pouze za daný test, pro určitého uživatele, pro danou skupinu uživatelů, pro konkrétní spuštění testu apod., popř. jejich kombinace (např. „výsledky uživatele A z testu B").

4.11.4 Vzhled oken

Ve většině systémů obvykle existuje pouze malá skupina typů oken, která pokrývá drtivou většinu systému. Nejčastěji to jsou přehledy a detaily. Obdobný postup však lze aplikovat i na kterékoli další typy oken.

Typ okna, které je definováno v XML elementu `<page>`, určuje jeho atribut `type`. V tomto frameworku jsou možnosti následující.

- `grid` - přehled
- `tree` – větvený přehled
- `detail` – detailní formulář
- `designFrame` – design otázky
- `testerFrame` – zkouška testu
- `upload` – nahrávání souborů

Přehledy

Přehledy jsou okna, obsahující obvykle data určité databázové tabulky, propojení tabulek či pohledu, zobrazená ve formě tabulky (gridu), tedy ve sloupcích a řádcích. Data v tabulce by mělo být možné libovolně řadit, filtrovat, seskupovat, sumarizovat, prohledávat apod., což obvykle zajišťuje přímo použitá komponenta (v tomto případě DXGrid viz [w13]). Pro zobrazení více podrobností určitého záznamu (řádku) a jejich úpravu pak slouží okno typu detail (viz násl. kap. na str. 140), které je třeba otevřít s parametrem Id, což je ID záznamu vybraného v tabulce (přehledu). Kromě detailu může být vhodné pro zvolený záznam umožnit otevřít i jiná okna s jemu podřízenými podpřehledy (např. z přehledu uživatelů otevřít přehled výsledků zvoleného uživatele).

Pro definici přehledu obvykle stačí uvést, které sloupce má obsahovat a jak má jejich data reprezentovat. Například čísla mohou mít určitý formát zobrazení (měna, procenta, desetinná, celá), stejně tak data a časy, hodnoty typu boolean (ano/ne) se mohou zobrazovat jako zaškrtávací políčka apod.

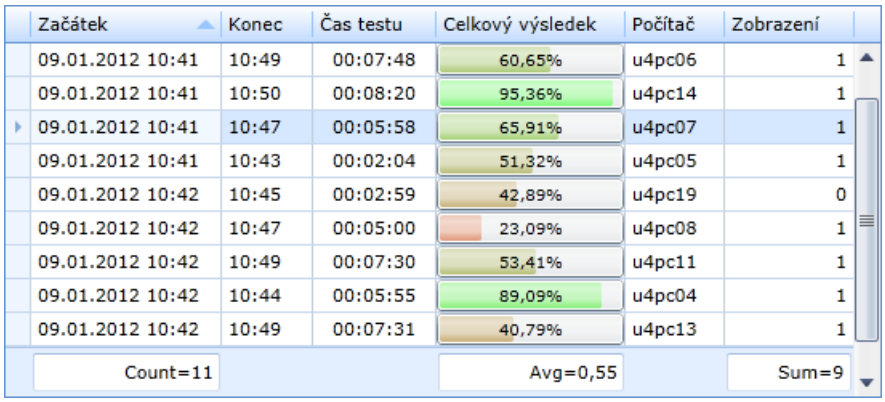
```
<grid table="Result">
 <columns>
  <col name="TestName" caption="Test" hideIfParam="IdTest" />
  <col name="UserName" caption="Zkoušený" hideIfParam="IdUser" />
  <col name="PeriodName" caption="Perioda" visible="0" />
  <col name="LimitGroupName" caption="Skupina" visible="0" />
  <col name="Start" caption="Začátel" formatString="dd.MM.yyyy HH:mm" sort="0D" />
  <col name="End" caption="Konec" formatString="HH:mm" />
  <col name="TestTimeSpan" caption="Čas testu" formatString="HH:mm:ss" />
  <col name="Score" caption="Dosažený výsledek" formatString="#0.00%" />
  <col name="ScoreBonus" caption="Penalizace" formatString="#0.00%;-#0.00%; " ro="0" />
  <col name="ScoreFinall" caption="Celkový výsledek" edit="Score" />
  <col name="ComputerName" caption="Počítač" ro="0" />
  <col name="IpStart" caption="IP" visible="0" />
  <col name="ResultsShowCount" caption="Zobrazení" />
  <col name="Note" caption="Poznámka" edit="Memo" ro="0" />
  <col name="IsArchived" caption="Archiv" ro="0" />
 </columns>
 ...
</grid>
```

Kód 29 – Ukázka definice sloupců pro přehled výsledků

Nastavení jednotlivých sloupců tedy určují atributy elementů `<col>`, jejichž seznam následuje:

- `name` – název vlastnosti objektu, jejíž hodnota se bude v daném sloupci zobrazovat.
- `caption` – je srozumitelným nadpisem v záhlaví sloupce.
- `formatString` – určuje formátovací řetězec pro netextová data. Tyto řetězce v tomto případě přímo podporuje jak programovací jazyk (C#) tak i použitá komponenta gridu, tudíž ji stačí pouze nastavit příslušnému sloupci a data budou zobrazována v daném formátu.
- `ro` – určuje, je-li daný sloupec pouze pro čtení (read-only, 1 = ano – výchozí hodnota) či umožňuje přímou editaci v rámci přehledu (0 = ne).
- `edit` – určuje typ editoru, který je použit nejen při přímé editaci dat, ale umožňuje i jejich správné formátování pro pouhé zobrazení. Není-li uveden přímo, je odvozen na základě datového typu vlastnosti ve sloupci zobrazované. Díky tomu lze například použít rozdílné editory pro tytéž datové typy (např. pro textový řetězec jednořádkový či víceřádkový editor „Memo“) a zároveň si případně vytvořit i vlastní editory a zobrazovače dat (např. „Score“).
- `sort` – umožňuje nastavit výchozí řazení dat v tabulce podle vybraných sloupců. První znak určuje prioritu řazení a druhý (poslední) směr (A – ascending – vzestupně, D – descending – sestupně).
- `visible` – definuje, je-li sloupec ve výchozím stavu přehledu zobrazen či skryt. Skryté sloupce pak může uživatel v případě zájmu ručně zobrazit, popřípadě skrýt ty které nepotřebuje.
- `hideIfParam` – podmiňuje skrytí sloupce existencí parametru okna. Je tak možné např. v přehledu výsledků skrýt sloupec s názvem testu, jsou-li zobrazeny výsledky pouze jednoho testu.

Tyto atributy lze samozřejmě pro jiný systém definovat odlišně, případně doplnit o podporu jakýchkoli dalších.



Začátek	Konec	Čas testu	Celkový výsledek	Počítač	Zobrazení
09.01.2012 10:41	10:49	00:07:48	60,65%	u4pc06	1
09.01.2012 10:41	10:50	00:08:20	95,36%	u4pc14	1
09.01.2012 10:41	10:47	00:05:58	65,91%	u4pc07	1
09.01.2012 10:41	10:43	00:02:04	51,32%	u4pc05	1
09.01.2012 10:42	10:45	00:02:59	42,89%	u4pc19	0
09.01.2012 10:42	10:47	00:05:00	23,09%	u4pc08	1
09.01.2012 10:42	10:49	00:07:30	53,41%	u4pc11	1
09.01.2012 10:42	10:44	00:05:55	89,09%	u4pc04	1
09.01.2012 10:42	10:49	00:07:31	40,79%	u4pc13	1
Count=11			Avg=0,55		Sum=9

Obr. 39 – Ukázka přehledu Výsledky testu (některé sloupce jsou pro přehlednost skryty)

Obecně by mělo platit, že rozsáhlejší zápisy mají i svou zkratku a výchozí hodnota neuvedených atributů pokrývá nejvyšší možný podíl případů.

Filtry

Přehledy načítají data z určité databázové tabulky, a ač je pro každý z nich v sekci `<dataSet>` přesně vymezeno, která pouze jsou pro něj potřeba, tento požadavek se již při jejich zobrazování neuplatňuje. Tím pádem sice nehrozí, že by některá data při zobrazení přehledu chyběla, ale může dojít k tomu, že tam budou některá navíc (např. zobrazí-li se nejprve přehled výsledků uživatele A a pak přehled výsledků uživatele B, budou ve druhém přehledu zahrnuty výsledky obou uživatelů, neboť čerpají z téhož datového zdroje, který na klientské straně dále nefiltrují). Z tohoto důvodu je u přehledů sekce `<filters>`, jež umožňuje tyto filtrace nad daty staženými na klientské stanici definovat (viz Kód 30).

```
<filters>
  <filter param="IdUser">IdUser=%IdUser%</filter>
  <filter param="IdTest">IdTest=%IdTest%</filter>
  <filter param="IdLimit">IdLimit=%IdLimit%</filter>
</filters>
```

Kód 30 – Ukázka definice filtrů dat pro přehledy na klientské straně

Jednotlivé filtry se zapisují do elementů `<filter>`, jejichž hodnota je podmínkou filtru. Atribut `param` určuje, který parametr podmínka používá a tedy i vyžaduje. Není-li přehledu tento parametr při jeho vytváření předán, nebude ani daný filtr aplikován. Podmínky jednotlivých filtrů se slučují operátorem `and`.

Triggry

Element `<triggers>` je možné použít pouze u přehledů, neboť jen z nich se obvykle zakládá nový záznam (byť je posléze otevřený v detailu). Je-li tedy takto vytvořen nový záznam, je možné automaticky přednastavit některé z jeho hodnot. O to se stará právě sekce `<triggers>`.

```
<triggers>
  <insert property="IdTest" param="IdTest" />
  <insert property="IdGroup" param="Id" only="file" />
</triggers>
```

Kód 31 – Ukázka trigrů z větveného přehledu otázek v testu

Jako podelementy lze použít pouze `<insert>`, jež jsou zpracovávány při vkládání nového záznamu. Ty mohou obsahovat následující atributy.

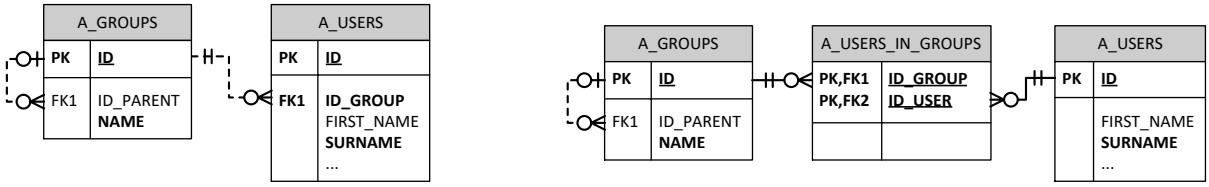
- `property` – název vlastnosti objektu přehledu, do které bude nastavena hodnota
- `param` – název parametru okna, jehož hodnota bude vlastnosti nastavena
- `value` – konstantní hodnota pro nastavení („now" nastaví aktuální datum a čas)
- `new` – pouze pro vlastnosti typu datový objekt – vytvoří do nich tento objekt (název jeho třídy je hodnotou atributu)
- `only` – omezení platnosti na složky či položky (pouze pro větvené přehledy, viz níže)

V případě atributu `new`, kdy se do vlastnosti vytváří nový objekt se element `<insert>` může dále větvit na další podelementy `<insert>`, jež pak nastavují hodnoty pro vlastnosti tohoto nového objektu.

Větvené přehledy

Větvené přehledy jsou přehledy zobrazující data ve stromové (hierarchické) struktuře. Např. skupiny uživatelů + uživatelé.

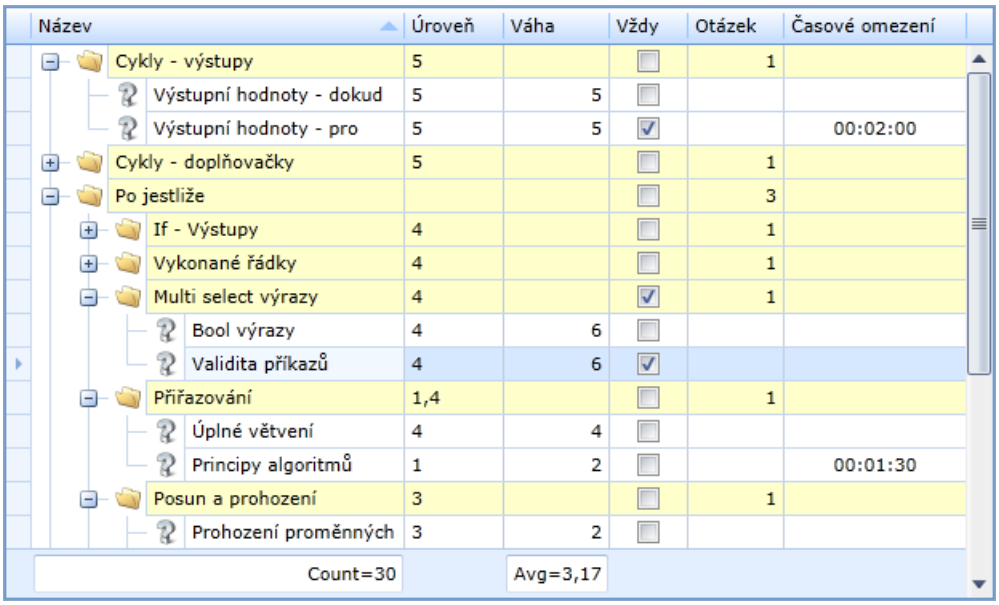
Ve větveném přehledu se rozlišují dva typy položek – složka a položka. Složky mají minimálně tři hodnoty: ID, id odkazující na předka a název. Díky prvním dvěma lze data snadno uspořádat do hierarchické struktury[78] a název umožňuje jejich jednotné přehledné vyobrazení. Položky pak mohou být do složek zařazeny dvěma způsoby: přímo (1:n, viz Obr. 40 vlevo) a pomocí vazební tabulky (m:n, viz Obr. 40 vpravo). V prvním případě bude každá položka v přehledu právě jednou, ve druhém tam může být i vícekrát, popř. ani jednou.



Obr. 40 – Datový model zařazení uživatelů ve skupinách vazby 1:n (vlevo) a m:n (vpravo)

Vzhled zvoleného okna velmi záleží na použité komponentě pro zobrazování větveného přehledu. Základní komponenta ze Silverlight Toolkit [w63] totiž umí zobrazovat pouze jednu hodnotu (název) a nikoli ucelený sloupcový přehled. V tomto případě byla použita komponenta od DevExpress [w14], jež umožňuje využít plnohodnotný tabulkový přehled včetně větvené struktury (viz Obr. 41).

[78] složky první úrovně (kořenové) mají ID předka null, v číselném (int) vyjádření nula



Obr. 41 – Ukázka větveného přehledu otázek testu Algoritmy

Zápis rozložení je téměř totožný se zápisem klasického přehledu (viz Kód 32), tedy rozdělení do sloupců, jejich vazba na data, nadpis, editor atd. Navíc je zde však možnost přidat atribut `only`, a do něho uvést hodnotu „folder" nebo „file". Tato vlastnost vymezuje použití pouze pro určitý typ položky, tedy složku (folder) či položku (file). Hodnota ve sloupci u druhého typu položky pak nelze editovat a zůstává vždy prázdná.

```
<columns>
  <col name="Name" caption="Název" ro="0" sort="0A" />
  <col name="Level" caption="Úroveň" ro="0" />
  <col name="Weight" caption="Váha" ro="0" />
  <col name="IsAllways" caption="Vždy" ro="0" />
  <col name="IsMix" caption="Míchat" ro="0" only="folder" />
  <col name="QuestionCount" caption="Otázek" ro="0" only="folder" />
  <col name="IsTogether" caption="Pohromadě" ro="0" only="folder" />
  <col name="TimeLimtSpan" caption="Časové omezení" ro="0" only="file" />
  <col name="Index" caption="Pořadí" ro="0" />
</columns>
```

Kód 32 – Ukázka definice sloupců ve větveném přehledu otázek testu

Dalším rozdílem v definici větveného přehledu je nutnost místo jednoho datového zdroje (názvu třídy objektu) definovat dva a vztahy mezi nimi. V elementu `<grid>` se tedy definují tyto další atributy.

- `folder` – název třídy s daty pro složky (např. „QueGroup")
- `file` – název třídy s daty pro položky (např. „Question")
- `fileIdParent` – název vlastnosti položky s identifikátorem skupiny (např. „IdGroup")
- `fileName` – název vlastnosti položky s jejím názvem (např. „Name")
- `fileIcon` – název ikony položky
- `canMoveFilesToRoot` – povolení nebo zákaz přesunu položek mimo všechny skupiny (do kořene)
- `treeListItemSubType` – název třídy zapouzdřující oba typy hodnot do jedné

Složky, na rozdíl od položek, mají standardizované některé vlastnosti, neboť jsou odvozeny vždy od společného předka – třídy `TreeBase` – díky čemuž v jejich specifikaci není třeba uvádět všechny vazby, jako je tomu u položek. Konkrétně jde o vlastnosti s odkazem na předka (IdParent), název (Name) a ikonu.

Komponenty zobrazující hierarchická data obvykle přijímají výhradně jeden datový zdroj, navíc obsahující pouze objekty téže třídy. Tato třída navíc musí být u všech objektů zcela shodná, nikoli zděděná. To je při kombinování složek a položek problém, neboť je nelze použít přímo.

Pro řešení byla použita modifikace návrhového vzoru Decorator [106 stránky 22-35]. Byla vytvořena obalová třída, jež uchovává vždy pouze referenci na zapouzdřovaný objekt a obsahuje vlastnosti odkazující se na jeho vlastnosti, jež musejí být pro platné zobrazení v přehledu společné pro složku i položku. Zároveň si je tento objekt vždy přesně vědom, který z těchto dvou typů zapouzdřuje, a podle toho i obsluhuje jeho složky.

Detaily

V oknech zobrazujících detailní informace jednoho záznamu, již není dominantním prvkem tabulka, ale formulář, skládající se obvykle s popisků a editorů. Ty by měly být vhodně (vzhledně, logicky a intuitivně) rozmístěny na ploše okna, aby jejich vyplňování či úprava byly pro uživatele co nejsnazší.

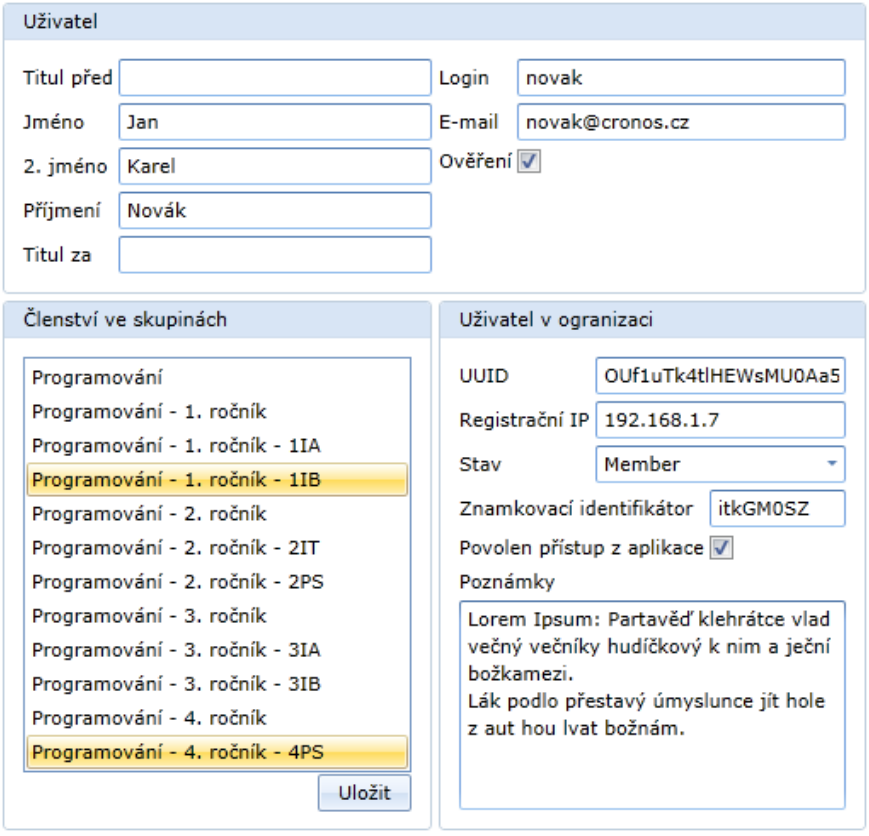
```
<detail table="User">
  <g>
    <g>
      <g o="h" caption="Uživatel">
        <g>
          <itm name="TitleBefore" caption="Titul před" />
          <itm name="FirstName" caption="Jméno" />
          <itm name="MiddleName" caption="2. jméno" />
          <itm name="Surname" caption="Příjmení" />
          <itm name="TitleAfter" caption="Titul za" />
        </g>
        <g>
          <itm name="Login" caption="Login" ro="1" />
          <itm name="Email" caption="E-mail" ro="1" />
          <itm name="IsEmailVerify" caption="Ověření" ro="1" />
          <itm name="IsArchived" caption="Archiv" />
        </g>
      </g>
      <g o="h">
        <g caption="Členství ve skupinách">
          <itm name="Groups" edit="List" source="OrgGroup" labelPos="top" />
        </g>
        <g subObject="OrgUser" caption="Uživatel v ogranizaci">
          <itm name="Uuid" caption="UUID" ro="1" />
          <itm name="Ip" caption="Registrační IP" ro="1" />
          <itm name="OrgState" caption="Stav" />
          <g labelGrid="local">
            <itm name="MarkId" caption="Znamkovací identifikátor" />
            <itm name="IsAppTrust" caption="Povolen přístup z aplikace" />
          </g>
          <itm name="Note" caption="Poznámky" edit="Memo" vAlign="stretch"
               labelPos="top" />
        </g>
      </g>
    </g>
  </g>
</detail>
```

Kód 33 – Ukázka definice formuláře pro detail uživatele

Rozdělení do logických celků je zde realizováno pomocí skupin (group), které v zápisu rozložení formuláře zastupuje element `<g>`. Ten může mít několik atributů a to `o`, v němž se definuje orientace řazení prvků skupiny a to buď pod sebe (v – vertikálně, výchozí hodnota) nebo vedle sebe (h – horizontálně). Druhým atributem je `caption`, jež umožňuje nastavit nadpis skupiny. Pokud nadpis není uveden, nezobrazuje se ani záhlaví a orámování skupiny, což může sloužit například pro vizuální řazení prvků do více sloupců v rámci jednoho nadřazeného rámečku. Další atribut `subObject` umožňuje vymezit datový kontext[79] skupiny na vlastnost editovaného objektu, jež je sama objektem s vlastními vlastnostmi (viz Příloha 14). Veškeré položky této skupiny se pak již odkazují na vlastnosti tohoto podobjektu.

Atribut `labelGrid` v elementu skupiny `<g>` umožňuje nastavit hodnotu „local", která vyřadí položky této skupiny z globálního systému zarovnávání formuláře a veškeré její položky zarovná zvlášť. Ignoruje tedy rozměry popisků z ostatních skupin a naopak ostatní skupiny ignorují tuto. Je-li potřeba takto ošetřit pouze jedinou položku v rámci některé skupiny, stačí ji obalit elementem `<g>` a neuvést jeho nadpis (`caption`).

Celý detail je zároveň vždy jedna hlavní skupina, jejíž obsah se vytváří rekurzivně.



Obr. 42 – Ukázka detailu uživatele

Jednotlivé položky formuláře se tedy skládají z popisku a editačního prvku. Zapisují se do elementu `<itm>` (item – položka) a mohou mít následující atributy.

- `name` – název vlastnosti objektu, který položka zobrazuje a umožňuje upravovat
- `caption` – popisek před (popř. nad) položkou vysvětlující její význam

[79] datový kontext (DataContext) je vlastností kontejneru (Panel) sdružujícího editační položky a určuje objekt, s jehož vlastnostmi jsou tyto položky provázány a popř. jimi editovány [125 str. 77]

- `edit` – umožňuje změnit editační prvek (viz níže), standardně je totiž volen automaticky podle datového typu vlastnosti objektu
- `labelPos` – určuje, kde bude umístěn popisek položky, jestli před ní (left) nebo nad ní (top)
- `formatString` – určuje masku pro zobrazení a editaci určitých typů dat (datum, čas, čísla apod., viz *formatString* v přehledech)
- `ro` – určuje, zda položka je či není (výchozí hodnota) pouze pro čtení (read-only)
- `width` – umožňuje nastavit editačnímu prvku konkrétní šířku (aby např. prvek pro editaci čísla nebyl zbytečně dlouhý)
- `height` – umožňuje nastavit editačnímu prvku konkrétní výšku (vhodné např. pro editory delších textů „Memo")
- `minWidth` – umožňuje nastavit editačnímu prvku minimální šířku, pod kterou se nezmenší, i kdyby se kvůli tomu formulář nevešel do rozměrů okna a musel se tak zobrazit horizontální posuvník
- `minHeight` – umožňuje nastavit editačnímu prvku minimální výšku
- `align` – horizontální zarovnání obsahu editačního prvku
- `vAlign` – vertikální zarovnání editačního prvku; umožňuje nastavit hodnotu „stretch", která způsobí jeho roztahování do výšky, ve volném prostoru okna
- `hAlign` – horizontální zarovnání editačního prvku
- `font` – název písma pro editační prvek (vhodné měnit např. pro přímou editaci XML kódu)
- `fontSize` – velikost písma pro editační prvek

Některé položky se specifickými editory pak mohou mít i další atributy. K dispozici jsou následující editory.

- `Text` – krátký jednořádkový text (standardní pro *string*, např. jméno)
- `Memo` – dlouhý víceřádkový text (např. popis)
- `Integer` – celé číslo (standardní pro *int*, např. počet)
- `Float` – desetinné číslo (standardní pro *float* a *decimal*, např. body)
- `Check` – zaškrtávací políčko (standardní pro *bool*, např. povolit přístup – ano/ne)
- `DateTime` – datum, popř. datum a čas, podle formatStringu (standardní pro *DateTime*, např. zahájení testu)
- `TimeSpan` – čas (standardní pro *TimeSpan*, např. délka řešení testu)
- `Combo` – výběrová nabídka, vyžaduje další nastavení `source` pro určení zdroje dat a `display` pro určení vlastnosti, jejíž text se bude v nabídce zobrazovat (standardní pro datové objekty, např. výběr skupiny, které se zadává test)
- `Enum` – stejné jako Combo, pouze hodnoty na výběr se automaticky sestaví ze všech možných hodnot výčtu (standardní pro *enum*, např. stav uživatele v organizaci)
- `Picture` – výběr obrázku ze seznamu importovaných obrázků (např. ikona testu)
- `PicturePreview` – náhled obrázku zmenšeného na zadané rozměry nebo do velikosti okna (např. náhled obrázku u položky seznamu importovaných zdrojů)

- `List` – specifický editor zobrazující seznam položek jiného zdroje, které se dají kliknutím označovat a vzniká tak vazba m:n mezi editovaným objektem a objekty daného zdroje (např. členství uživatele ve skupinách)
- `TestSettings` – vlastní specifický editor pro editaci nastavení testu (blíže bude popsán v kap. 4.11.5, str. 145)

Speciální typy oken

Speciální typy oken, na rozdíl od těch předchozích, byly vytvořeny pro jeden konkrétní účel a nejsou používány opakovaně pro různá data, proto budou zmíněny jen okrajově.

DesignFrame je okno určené pro vytváření, a testování jedné konkrétní otázky. Pracuje s XML kódem, který transformuje do finálního QML, z něhož dokáže spustit plně funkční otázku ve fiktivním testu s definovaným nastavením.[80]

TesterFrame je okno, jež je pouze prostředníkem mezi testovacím rozhraním aplikace (viz kap. 4.3.1, str. 70, podobně jako přístup prostřednictvím API, viz Obr. 17). To se otevírá v rámci okna administračního systému s plnohodnotným testem. Zde si lze odzkoušet především nastavení testu. Test je možné spouštět jak přímo pro test s jeho obecným nastavením, tak i pro jeho jednotlivá spuštění a to s jejich specifickým nastavením. V obou případech je spuštění ryze testovací, funkční i mimo vymezení časové dostupnosti testu a bez ukládání výsledků.

Upload je okno určené pro nahrávání multimediální souborů (např. obrázků) na server.

Pro případné další potřeby není problém kdykoli přidat další okno se specifickými vlastnosti, ale jak je patrné, takových je skutečně minimum a majoritní část aplikace pokrývají okna s univerzálním použitím.

Podokna

Základní okna, jako je např. přehled testů lze otevírat přímo z hlavní nabídky[81]. Ostatní přehledy se pak otevírají až z těchto oken jako jejich podokna. Ty zároveň přejímají parametry z oken, ze kterých byly spuštěny. Obvykle jde o identifikátor (ID) některého ze záznamů (objektů), pro nějž se má otevřít jeho detail či podpřehled[82].

```
<subPages>
  <page name="TestDetail" caption="Nový test" params="Id:ID;TestName:Name" for="new" />
  <page name="TestDetail" caption="Detail testu" params="Id:ID;TestName:Name" />
  <hr />
  <page name="TestQuestionsGroups" caption="Otázky testu"
        params="IdTest:ID;TestName:Name" />
  <page name="TestLimits" caption="Limity testu" params="IdTest:ID;TestName:Name" />
  <page name="Results" caption="Výsledky" params="IdTest:ID;TestName:Name" />
  <hr />
  <page name="TesterFrame" caption="Zkusit test" params="IdTest:ID;TestName:Name" />
</subPages>
```

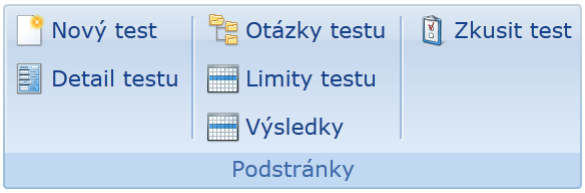
Kód 34 – Zápis podoken přehledu testů

[80] editor otázek si lze přímo vyzkoušet na [w12].
[81] okna, která je možná spouště přímo z hlavní nabídky (nejen tedy jako podokna) mají v elemntu `<page>` atribut `main` nastaven na hodnotu 1
[82] podpřehled – přehled obsahující pouze záznamy vztažené k určitému nadřazenému objektu (např. výsledky pouze určitého uživatele)

Údaje o jednotlivých podoknech, které lze otevřít z okna aktuálního, se zapisují pod element `<subPages>` do elementů `<page>` (viz Kód 34). Může se zde vyskytovat i element `<hr>`, který do nabídky tlačítek odkazujících se na jednotlivá podokna přidá separátor (oddělující čáru), čímž je lze rozdělit na tematické skupiny (viz Obr. 43). Element `<page>` pak obsahuje následující atributy.

- `name` – kódový název podokna, identifikující element v hlavní větvi (atribut `name`), jehož data se pro stránku načtou
- `caption` – zobrazený název okna, resp. popisek na tlačítku, který jej otvírá
- `params` – parametry předávané podoknu ve speciálním formátu (viz níže)
- `for` – hodnota „new" znamená, že dříve než se toto okno otevře, vytvoří se nový záznam (objekt) stejného typu jako je zdroj aktuálního přehledu či detailu a právě on a jeho výchozí vlastnosti budou předány otevíranému podoknu

Atribut `params` tedy umožňuje uvést parametry, jež se předají novému podoknu. Ty se zapisují v párech oddělených středníkem, přičemž párové dvojice se oddělují dvojtečkou. První z každé dvojice určuje název parametru, pod kterým si jej bude moci podokno načíst a druhý je název vlastnosti aktuálního objektu[83], z níž bude předávaná hodnota načtena.



Obr. 43 – Ukázka tlačítek pro otevření podoken přehledu testů

Nadpisy

Jedním z atributů elementu `<page>` je `caption`, udávající uživatelský název okna (např. „Výsledky"). Ten se však může různit, podle dalších omezení, které oknu udávají vstupní parametry (např. „Výsledky uživatele Jan Novák"). Z tohoto důvodu mohou všechna okna využít elementy `<caption>`, které jim umožňují definovat tento název v závislosti na dostupnosti jednotlivých parametrů.

```
<page name="Results" caption="Výsledky" type="grid" main="0">
  <caption caption="Výsledky uživatele %UserName% z testu %TestName%"
           param="UserName,TestName" />
  <caption caption="Výsledky uživatele %UserName%" param="UserName" />
  <caption caption="Výsledky testu %TestName%" param="TestName" />
  ...
</page>
```

Kód 35 – Ukázka použití elementů **`<caption>`** pro definici uživatelského názvu okna v závislosti na jeho vstupních parametrech

Hlavním atributem elementu `<caption>` je tedy stejnojmenný `caption`, jež udává uživatelský název. Ten smí obsahovat parametry (slova z obou stran uvozená znakem procenta %), které budou nahrazeny hodnotou daného parametru. To, který z elementů `<caption>` se

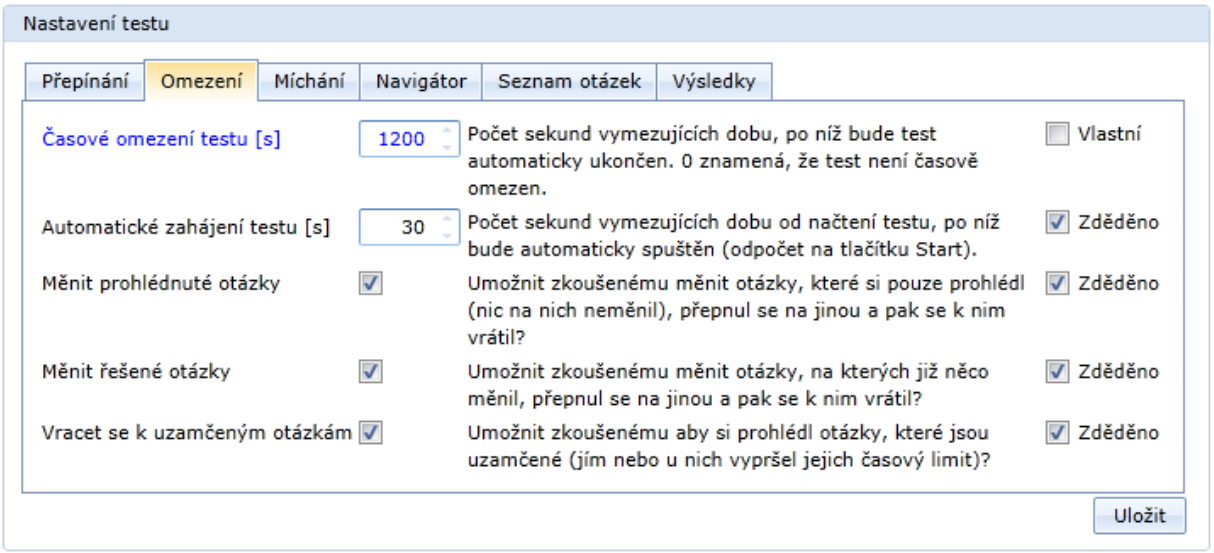
[83] aktuální objekt – v přehledech je to objekt (záznam) v přehledu aktuálně zvolený (označený), v detailech je to ten, jehož data jsou načtena

použije pro název okna, závisí v první řadě na přítomnosti všech vstupních parametrů uvedených v atributu `param` (je-li jich více, jsou odděleny čárkou) a za druhé na jeho pořadí. Vyhovovalo-li by tedy více elementů, pro volbu názvu bude použit prví z nich.

4.11.5 Konfigurace přímo v kódu pomocí atributů

Jedním ze zvláštních typů editorů je nastavení testu (*TestSettings*, viz Obr. 44), který umožňuje definovat jednotlivé hodnoty nastavení testu[84]. Těchto nastavení je více, jsou různého typ (ano/ne, číslo, text, výběr apod.) a pro přehlednost jsou rozděleny do skupin. Zároveň existuje několik stupňů téhož nastavení, z nichž vyšší překrývá nižší (viz Obr. 15 na str. 67).

Formulář pro toto nastavení mohl být vytvořen několika způsoby. Jedním z nich je pomocí výše popsaného způsobu vytváření formulářů detailu v rámci konfiguračního XML. Vzhledem k jeho rozsáhlosti a především potřebě jeho opakovaného použití v různých oknech ovšem tento způsob zvolen nebyl. Další možností bylo vytvoření zvláštní uživatelské komponenty (*UserControl*), s pevně předdefinovanými položkami a tu používat pro editaci nastavení jako celku. Vzhledem k tomu, že nastavení je ukládáno ve formátu XML jako jedna datová položka, je tato možnost vhodná i z tohoto hlediska. Tento způsob však byl použit jen z části.



Obr. 44 – Ukázka editoru nastavení testu

Komponenta pro editaci nastavení testu byla vytvořena univerzálním způsobem a to tak, že se veškerý její obsah automaticky generuje na základě konfiguračních dat. Tato data však nejsou v externím XML souboru, ale tentokrát jsou včleněna přímo do kódu aplikace. Přesněji řečeno, hodnoty potřebné pro automatizované generování tohoto editoru jsou uvedeny jako programové atributy [100 str. 449] u definice tříd a jejich vlastností, jež jsou používané pro uchovávání načtených datových objektů v paměti během editace. Tento postup vychází z metod tzv. aspektově orientovaného programování (viz [107]), kdy jsou rutinní části kódu separovány a zapouzdřeny mimo jeho hlavní část a sami se aktivují pouze na základě určitých příznaků [108].

[84] funkčnost editoru nastavení testu lze vyzkoušet na [w21]

```
[XmlClass("Omezení")]
public class LimitsSettings : DepenencyObjectBase
{
  private double testTimeLimit = 0;       // Výchozí hodnota
  [XmlProperty("Časové omezení testu [s]", "Počet sekund vymezujících dobu, …")]
  public double TestTimeLimit
  {
    get { return testTimeLimit; }
    set { testTimeLimit = value; PropertyChangedHandler("TestTimeLimit"); }
  }
  private bool canChangeSolved = true;    // Výchozí hodnota
  [XmlProperty("Měnit řešené otázky", "Umožnit zkoušenému měnit otázky, …")]
  public bool CanChangeSolved
  {
    get { return canChangeSolved; }
    set { canChangeSolved = value; PropertyChangedHandler("CanChangeSolved"); }
  }
  ...
}
```

Kód 36 – Ukázka kódu třídy **`LimitSettings`**, obsahující konfigurační atributy

Přímo v kódu se tedy třídám a vlastnostem přiřadí jejich popisky, popř. poznámky, které se posléze zobrazují v editačním formuláři. Ten se generuje metodou, jíž se na vstupu předá informace o typu hlavní třídy `TestSettings`, která má pouze 6 vlastností, což jsou skupiny jednotlivých nastavení. Ty se pomocí reflexe [101 str. 489] zkontrolují a obsahují-li jejich typy (třídy) atribut `XmlClass` (viz Příloha 15), je pro ně ve formuláři vytvořena vlastní záložka s popiskem udaným tímto atributem. Na obsahovém panelu pod touto záložkou se pak generují jednotlivé položky propojené s vlastnostmi daného objektu metodou Data Binding[85] [109 str. 275]. Popisek a poznámka jsou převzaty z atributu `XmlProperty` a druh editoru položky je zvolen na základě datového typu každé vlastnosti.

Zaškrtávací políčka „Zděděno" či „Vlastní" vpravo informují o původu každé hodnoty. Je-li hodnota zděděna (zaškrtnuta), pak se do XML informace o jejím nastavení vůbec neukládá a je převzata z obecnějšího nastavení. Je-li hodnota vlastní (políčko není zaškrtnuto, je u něj napsáno „Vlastní" a její popisek je modrý), pak se do XML s nastavením ukládá. Po opětovném zaškrtnutí se položce znovu nastaví hodnota, jež má nastavenou její předek.

Určení hodnoty jednotlivých položek nastavení probíhá tak, že se každá z nich zkouší načíst postupně z XML uložených na různých úrovních (viz Obr. 15 na str. 67). Je-li hodnota v některém z XML nalezena, v jejím dalším vyhledávání se nepokračuje a přechází se k určení hodnoty položky další.

4.11.6 Lokalizace

V konfiguračním XML jsou kromě obecných informací o datech, editačních položkách, vazbách apod. uloženy i textové popisky v konkrétním jazyce. Totéž platí i o konfiguraci přímo v kódu. To by mohlo vadit v případě, že systém má být více jazyčný, tj. lokalizovaný pro více řečí různých národností.

[85] podobnou funkci automatického generování formuláře na základě vlastností třídy poskytuje také komponenta DataFrom z rozšíření Silverlight Toolkit [w63], viz [125 stránky 78-81]

Řešením může být, že místo konkrétních popisků by se uváděly pouze klíčové řetězce, odkazující buď klasicky na položky v tzv. *resources* [w40], nebo jiném externím zdroji [110]. Až z nich by se pak v případě potřeby čerpalo konkrétní znění každého popisku, přičemž příslušný soubor se zdroji by byl dosazen dle uživatelem zvoleného národnostního nastavení.

4.11.7 Shrnutí

Konfigurační frameworky jsou alternativou ke klasickým strukturálně či v současnosti nejrozšířenějším objektově programovaným frameworkům. V praxi se v čisté formě pro desktopové aplikace zatím příliš nevyužívají, ač je jejich potenciál obrovský. Skýtají totiž řadu výhod, jako je např. iterativní vývoj, nasazování nových modulů bez nutnosti reinstalace na jednotlivých stanicích, distribuce nových verzí nenarušující provoz, snadný vývoj a servis atd.

Ve sféře informačních systémů by tento přístup mohl být velmi užitečný, a to jak pro jejich tvůrce, tak i správce a uživatele. Je-li zapotřebí např. v některém z oken desktopové aplikace změnit popisek nebo přidat, přemístit či ubrat položku, není zapotřebí složité, pracné, administrativně náročné a na testování zdlouhavé vytváření a distribuce nové verze celé aplikace či její knihovny. Stačí pouze upravit příslušný XML element uložený např. v globální systémové databázi a všem klientským stanicím se tento zásah projeví ihned při příštím otevření okna, aniž by museli aplikaci vypínat. Přitom k tomuto zásahu stačí prostý textový editor či jednoduché administrační webové rozhraní a není zapotřebí žádného složitého, výpočetně náročného vývojového prostředí ani kompilátoru.

Tutéž okamžitou aktualizaci umožňují samozřejmě už ze své podstaty on-line webové systémy vytvořené v HTML, nicméně i zde se může použití konfiguračního frameworku zhodnotit. Ve větších systémech sice bývá zvykem vytvořit určitý framework, který např. na všech stránkách standardizuje vzhled a rozložení, ovšem obvykle je zapotřebí layout prvků a jejich funkcionalitu na každé stránce programově řešit zvlášť, byť jen voláním globálních metod. Tato zde uvedená úplná abstrakce celého uživatelského rozhraní od kódu určujícího chování systému, oken (stránek) a prvků je sice náročnější na úvodní implementaci, avšak mnohonásobně se vyplatí již od středně rozsáhlých aplikací, při dalším vývoji i následné údržbě.

Pokud by byl takovýto systém postavený na konfiguračním frameworku, mohlo by to jeho autorům i uživatelům přinést značné výhody, jako např. malá velikost aplikace, nízká paměťová náročnost, jednoduchá instalace, snadná aktualizace i za běhu, údržbu zvládne i laik apod. Pro zjednodušení generování konfiguračních dat navíc není problém vytvořit i jejich vlastní editor typu WYSIWYG.

Pro nasazení na webu existují také již mnohá hotová řešení (např. DotNetNuke [w18], Joomla [w39], Drupal [w19] a v podstatě i Moodle [w48] a další), využívající podobného principu. Obvykle jsou dodávána s redakčním systémem, který kompletně zapouzdřuje správu konfiguračních dat do uživatelsky přívětivějších formulářových editorů. Ne vždy jsou však tyto systémy vhodné pro všechny konkrétní účely. V oblasti uživatelsky přívětivějších těžkých klientů (desktopových aplikací) jsou pak konfigurační frameworky spíše výjimkou. Ne jinak tomu je i u bohatých internetových aplikací využívajících výhody těžkých klientů při architektuře klientů lehkých, jako je Flash či v tomto případě použitý Silverlight. Jak zde bylo předvedeno, vytvořit vlastní zcela originální framework pro libovolné konkrétní potřeby není nikterak složité a vzhledem k získaným pozdějším úsporám se to ve spoustě případů vyplatí.

4.12 Mini-jazyky

V průběhu vývoje univerzálního testovacího prostředí bylo v různých etapách nezbytné řešit způsoby zápisu nejrůznějších typů údajů. Standardní formy zápisů přitom nebyly vždy zcela nejvhodnější nebo dokonce neexistovaly vůbec. Proto byla vytvořena celá řada „mini-jazyků" pro tyto různé případy. Obecně lze říci, že u všech byly kladeny požadavky na jejich přehlednost, stručnost, srozumitelnost pro člověka a rychlost jejich zpracování.

Některé z nich budou v této části představeny, neboť mohou být inspirací pro ostatní, jak z hlediska přímého použití, tak i ve smyslu vytváření zcela nových mini-jazyků pro nejrůznější účely v podobném duchu.

Pro jednotlivé mini-jazyky popisované v této části vždy nejprve uvedeme do problému, pro jehož řešení byl vytvořen. Následně bude mini-jazyk podrobněji popsán a existují-li jeho alternativy, porovnáme jej s nimi a zhodnotíme jeho přednosti.

4.12.1 Seznamy číselných identifikátorů [◎ III.3.B]

Seznamy číselných identifikátorů jsou celkem často používanou součástí různých, většinou uzavřených systémů, proto nelze s jistotou tvrdit, že navržený způsob zkrácení jejich zápisu již není používán někým jiným, nicméně i přes to může jeho představení být pro jiné přínosem.

V administračním rozhraní (viz kap. 4.3.2, str. 72) jsou veškerá data načítána ze serveru „Just In Time", tedy právě když jsou zapotřebí a zároveň pouze ta, která jsou třeba (viz kap. 4.10.2, str. 127). Může se však stát, že při delší práci jsou ze serveru žádána data, stažená již dříve v jiném okně (viz kap. 4.11, str. 132). Ta je samozřejmě zbytečné stahovat znovu. V případě, že je žádána tatáž sada dat jako před tím (filtr vymezující jejich množinu je shodný nebo podmnožinou toho předchozího, viz Obr. 35 na str. 129) se samozřejmě komunikace se serverem ani nenavazuje. Vymezuje-li však nový filtr data, jejichž podmnožina již mohla být stažena, je zapotřebí uvést výčet těchto dat, která již nejsou znovu požadována. Toto lze vymezit buď dříve použitým filtrem či konjunkcí filtrů, popř. u individuálně stahovaných záznamů výčtem jejich ID.

Výčet ID však může obsahovat v některých případech i seznam tisíce číslic, která všechna vypisovat např. zvlášť do elementů XML může značně navýšit velikost datového požadavku odesílaného na server.

Číselné hodnoty ovšem nemusí být nutně rozepisovány do jednotlivých elementů, byť by to tak z objektového a XML hlediska bylo nejkorektnější. Lze je totiž zapsat v rámci jedné hodnoty, ať již elementu či atributu, a to jako textový řetězec čísel oddělených např. čárkami. A i tento zápis je možné efektivně zkrátit. Vychází se z toho, že číselná ID datových záznamů jsou vždy kladná a, s výjimkou vymazaných či pro daného uživatele nedostupných záznamů, po sobě jdoucí.

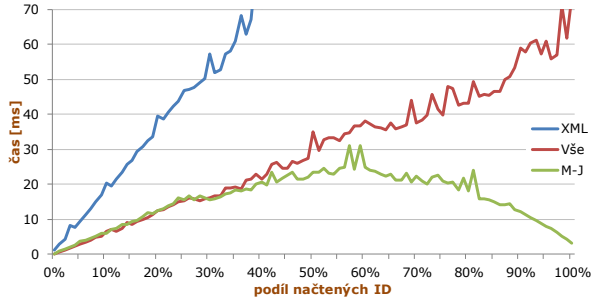
Nejprve je nezbytné tyto hodnoty seřadit. Pak lze snadno identifikovat sekvence hodnot jdoucích bezprostředně po sobě (např. „4,5,6,7,8") a ty nahradit zkráceným zápisem jejich rozsahu (např. „4-8"). Tím dojde k úspoře délky v závislosti na počtu členů takovéto sekvence, přičemž zkrácení je dosaženo již u tří členů. Tyto zkrácené zápisy se pak zapíší mezi hodnoty nespadající do žádné sekvence v seznamu odděleném čárkami. Například zápis seřazeného seznamu ID „1,2,3,4,6,8,9,10,11" se zkrátí na „1-4,6,8-11".

Rychlost práce se seznamy číselných identifikátorů

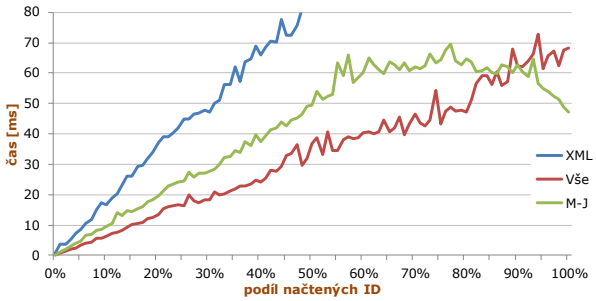
Následující grafy porovnávají rychlosti a velikost různých typů přípravy seznamu ID hodnot na datový přenos. Zakódování (viz Graf 24) tedy probíhá přepisem seznamu číselných hodnot (`List<int>`) na textový řetězec. Dekódování (viz Graf 25) pak z tohoto textového řetězce znovu zpětně vytvoří číselný seznam. Pro porovnávání byly použity tři různé metody:

- **XML** – zápis ID do XML, každé z nich jako, jako samostatný element (např. `<ids><id>1</id><id>2</id>…</ids>`), výstupní řetězec je bez *white-spaces*
- **Vše** – zápis všech ID do prostého seznamu bez jeho zkracování (např. „1,2,3…“)[86]
- **M-J** – zápis ve zde popsaném mini-jazyku

Měření bylo provedeno tak, že z po sobě jdoucí sekvence celých čísel od 1 do 10 000 bylo postupně náhodně vybíráno 0 až všech 10 000 hodnot, s krokem 100 (tj. 0, 100, 200, …, 9 900, 10 000). Kolik procent z těchto ID bylo vybíráno, ukazuje u grafů osa X. Z důvodu minimalizace náhodné složky byl pro každý vybíraný počet pokus 10x zopakován a pro graf pak použit průměr z těchto deseti hodnot. Parametry testovacího stoje uvádí Příloha 25.



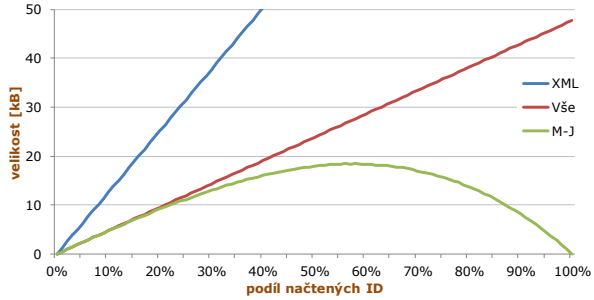
Graf 24 – Rychlost zakódování ID



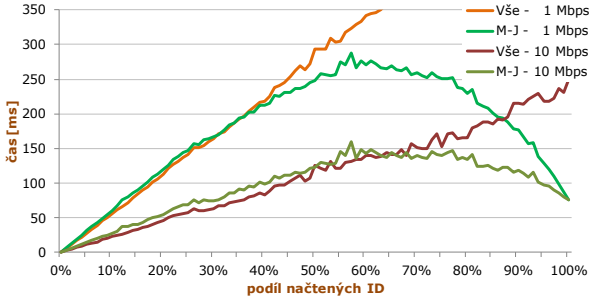
Graf 25 – Rychlost dekódování ID

První graf (Graf 24) ukazuje téměř lineární vzestup výpočetní (časové) náročnosti kódování ID do XML i do kompletního seznamu odděleného čárkami. Algoritmus provádějící zápis do mini-jazyka sice ve své první třetině roste podobně jako zápis kompletního seznamu, ale dále je již jeho průběh parabolický. Dle regresní funkce jednotlivých křivek překoná v rychlosti kódování mini-jazyk kompletní zápis od 35%, přičemž vrchol paraboly této křivky (M-J) je při 55% načítaných hodnot z jejich celkového počtu.

Při dekódování (Graf 25) je opět nejpomalejší zpracování XML zápisu, ovšem vzhledem k nutnosti kontrole přítomnosti pomlčky v každém členu je až do 91% dekódování řetězce mini-jazyka pomalejší (tentokrát roste až do 78%), než kompletní výpis všech požadovaných ID.



Graf 26 – Výsledná velikost

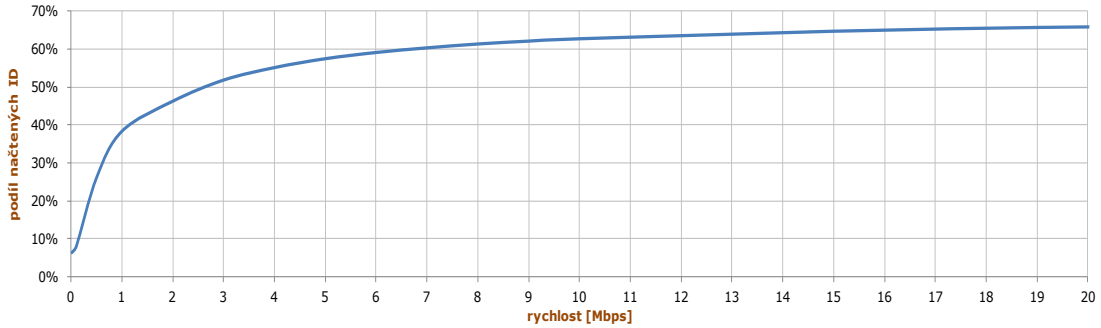


Graf 27 – Celková rychlost

[86] kompletní výpis všech hodnot oddělených čárkou by odpovídal zápisu do formátu JSON jako pole hodnot

Co se týče datové objemnosti zakódovaného řetězce (viz Graf 26), v tomto případě je zápis v mini-jazyku vždy nejkratší, přičemž jeho maximální průměrné délky 18 kB je dosaženo v 57%.

Celková rychlost (viz Graf 27) byla vypočtena jako součet času potřebného pro zakódování, odeslání zakódovaných dat[87] přes internet (pro porovnání ve dvou různých rychlostech: 1 Mbps a 10 Mbps) a dekódování. V tomto grafu byla již vynechána metoda kódování do XML, jelikož ve všech třech předchozích případech byla vždy zřetelně nejpomalejší, tudíž by ani zde neměla šanci uspět. Z grafu je patrné, že z hlediska celkové rychlosti zpracování je použití mini-jazyka výhodnější použít až od určitého podílu již načtených ID a určité rychlosti spojení se serverem. Z regresního modelu obou křivek byl pro různé rychlosti určen podíl načtených ID, kde se tyto dvě křivky (Vše a M-J) protínají a z těchto hodnot byla vytvořena následující křivka (viz Graf 28).



Graf 28 – Křivka určující hranici kombinací rychlosti a podílu načtených ID, od které je rychlejší použít mini-jazyk (nad křivkou)

Mini-jazyk je tedy výhodnější použít při kombinacích rychlosti a podílu již načtených ID, které se nacházejí nad touto křivkou, vypsat seznam všech ID je pak výhodnější používat při kombinacích pod touto křivkou (viz Graf 28). Metodou nejmenších čtverců byl tento vztah vyjádřen nejvíce odpovídající racionální lomenou kvadratickou rovnicí[88] (viz Vzorec 35), kde **x** je rychlost spojení v Mbps a **y** relativní podíl posílaných ID z jejich celkového počtu. Na jeho základě by se program měl automaticky rozhodovat, kterou metodu kdy použije.

$$y = \frac{0{,}649x^2 + 0{,}367x + 0{,}066}{x^2 + x + 1}$$

Vzorec 35 – Výpočet podílu načtených ID, od kterého je výhodnější použít mini-jazyk než výpis všech ID

Nutno dodat, že pokus byl realizován se zcela náhodnými hodnotami v rozsahu od 1 do 10 000. Většina čísel tedy byla čtyřciferných, tzn., že jejich zápis zabíral 4 byty, tj. 32 bitů, což je i rozsah hodnoty integer (Int32). Při vyšších (víceciferných) hodnotách by pak bylo výhodnější tyto hodnoty nezapisovat v textovém formátu, ale posílat je jako datový proud.

[87] velikost dat byla navýšena o 50%, což je průměrný přírůstek na režii, při posílání dat prostřednictvím SOAP přes HTTP [127 str. 43]
[88] pro tento výpočet byl použit software *Metoda nejmenších čtverců* [w43]

4.12.2 Intervaly [◎ II.3.C]

U testových otázek může jejich autor definovat i zpětnou vazbu, tj. informaci, která se zkoušenému zobrazí po uzamčení otázky či ukončení celého testu (viz kap. 4.1.8, str. 64). Tato informace zkoušenému obvykle sděluje, kde a proč udělal chybu a jak a proč to mělo být správně. Tato sdělení mohou mít více verzí, z nichž se zobrazí pouze ty, jež odpovídají určitým definovaným podmínkám, závislých obvykle na tom, zdali zkoušený odpověděl správně, chybně nebo otázku či její část neřešil vůbec. Je však také možné rozlišovat i jiné případy např. v závislosti na získaném bodovém skóre nebo na číselných hodnotách zadaných uživatelem.

Takovéto hodnoty, na rozdíl od ID, samozřejmě mohou obsahovat i záporná a desetinná čísla, takže již není možné vystačit si pouze s výčtem takovýchto hodnot (např. $\{1,2,3,4,5\}$), byť bez ohledu na délku takovéhoto zápisu. U rozsahů je třeba rozlišovat intervaly uzavřené (např. $\langle 1,5\rangle$), otevřené (např. $(1,\ 5)$) a polootevřené (např. $\langle 1,5)$ nebo $(1,\ 5\rangle$) a umožnit použití i intervalů jednostranně nekonečných (např. $(-\infty,\ 5)$ nebo $(5,\ +\infty)$). Řešení tedy musí v jednotném a přehledném stylu kombinovat všechny tyto možnosti, samozřejmě bez použití neobvyklých znaků jako je nekonečno („∞“).

Konkrétní přesné hodnoty lze tedy vypisovat stejným způsobem, jako tomu bylo u ID (např. „1,2,3,4,5“). Jako desetinný oddělovač je pak použita tečka (např. „1,1.5,2,2.5,3“). S ohledem na záporná čísla byl u zápisu rozsahů znak pomlčky (mínus „-“) nahrazen znakem vlnovky („~“). Díky tomu lze tedy relativně přehledně zapisovat i záporné rozsahy (např. „-5~-1“). Pro rozlišení uzavřených intervalů (spadá-li hraniční číslo rozsahu mezi uvedený výčet) byl použit znak plus („+“) obecně spojený s výrazem „včetně“, který se zapíše na příslušný konec (příp. oba) zápisu rozsahu (např. „1~5+“ je $(1,\ 5\rangle$, „+1~5“ je $\langle 1,5)$ a „+1~5+“ je $\langle 1,5\rangle$).

Intervaly obsahující nekonečno se mohou při seřazených hodnotách vyskytovat pouze na začátku (-∞) nebo na konci (+∞). Pro naznačení neomezené dolní či horní hranice intervalu je tedy možné, místo zápisu znaku pro nekonečno („∞“), tuto hodnotu vynechat zcela (např. „~0“ pro záporná čísla nebo „0~“ pro kladná). Prázdný interval $\{\}$ a neomezený interval $(-\infty,\ +\infty)$ není třeba řešit, protože v tomto případě by nemělo smysl je používat. Položku omezenou podmínkou, která neplatí nikdy, není třeba vůbec uvádět, a naopak položku omezenou podmínkou, která platí vždy, není třeba vůbec omezovat.

```
Matematický zápis:      (−∞,−5⟩ ∪ {−1, 1, 1.5, 2} ∪ ⟨5, 7.3) ∪ {9} ∪ (12,+∞)
Zápis v mini-jazyku:    ~-5+,-1,1,1.5,2,+5~7.3,9,12~
```

Kód 37 – Porovnání intervalů v matematickém zápisu a uvedeném mini-jazyku

4.12.3 Maska IP adres [◎ IV.2.D]

Vytvořené testy nejsou automaticky dostupné ostatním uživatelům pro spuštění, ale jejich autor musí ještě povolit jejich otevření. To se může vztahovat na určitého uživatele či skupinu uživatelů, na určitý datum a čas (blíže probráno v kap. 4.12.8 na str. 158) a také na určitou IP adresu nebo rozsah adres.

Určení IP adres, ze kterých pouze bude možné test spustit se provádí v klasickém čtyřsložkovém formátu (např. 90.80.170.3), přičemž místo kterékoli části lze uvést hvězdičku (např. „90.80.170.*“), díky čemuž jsou do výběru zahrnuty veškeré IP adresy shodné pouze v ostatních

částech. Jedná se tedy o uživatelsky přehlednější způsob definice rozsahu povolených IP adres, než je klasický zápis s maskou podsítě („255.255.255.0“). Kromě hvězdičky mohou jednotlivé části obsahovat i výčet konkrétních hodnot oddělených čárkou, nebo i rozsah oddělený pomlčkou, jako je tomu u zápisu rozsahu ID (např. „90.80.170.5,8,10,20-30,50-60“). Je-li třeba vypsat více konkrétních adres, stačí je oddělit středníkem (např. „90.80.170.*;90.80.171.50-60“). Pokud je nutné naopak některou z IP adres z povolení vyloučit, stačí před ni v tomto výčtu uvést znak „x“ (např. „x90.80.171.*“).

4.12.4 Barevné přechody [◎ II.2.C]

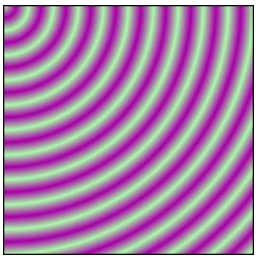
Podmnožina jazyka QML, označovaná QML-graphics, umožňuje vykreslovat nejrůznější vektorovou grafiku (např. čáry, obdélníky, elipsy, polygony apod.). Obrysy a výplň těchto objektů mohou mít nastaveny různé barvy a to nejenom souvislé plochy, ale i barevné přechody.

Podporovány jsou přechody lineární a radiální. U lineárních se určuje vektor, v jehož směru pak k barevným přechodům dochází, u radiálních (kruhových) poloměr (zvlášť horizontální a vertikální – *RadiusX*, *RadiusY*), poloha středu (*Center*), bod, z něhož kruhy vychází (*GradientOrigin*) a styl pokračování přechodů za hranicí kruhu (*SparedMethod*, což lze nastavit též u přechodů lineárních). U obou těchto druhů přechodů se dále definují tzv. stopy, tj. barvy, mezi kterými se přechody odstupňují a jejich relativní pozice na ploše rozsahu (0 = začátek, 1 = konec, čili např. 0.5 = střed).

V jazyce XAML se všechny tyto varianty zapisují do zvláštních XML elementů, kde se jednotlivé vlastnosti uvádějí v atributech a stopy v elementech (viz Kód 3).

```
<RadialGradientBrush RadiusX="0.05" RadiusY="0.05"
   Center="0,0" GradientOrigin="0,0" SpreadMethod="Reflect">
  <GradientStop Color="#FFAAFFAA" Offset="0" />
  <GradientStop Color="#FFAA00AA" Offset="0.49" />
  <GradientStop Color="#FFAA00AA" Offset="0.51" />
  <GradientStop Color="#FFAAFFAA" Offset="1" />
</RadialGradientBrush>
```

Kód 38 – Ukázka zápisu radiálního přechodu v XAML



Obr. 45 – Výsledek z Kód 38

Podobná formu zápisu se provádí i v jazyce SVG (viz Kód 39).

```
<radialGradient id="kruhy" r="5%"
    cx="0" cy="0" fx="0" fy="0" spreadMethod="reflect">
  <stop style="stop-color: #AAFFAA;" offset="0%" />
  <stop style="stop-color: #AA00AA;" offset="49%" />
  <stop style="stop-color: #AA00AA;" offset="51%" />
  <stop style="stop-color: #AAFFAA;" offset="100%" />
</radialGradient>
```

Kód 39 – Ukázka zápisu radiálního přechodu v SVG

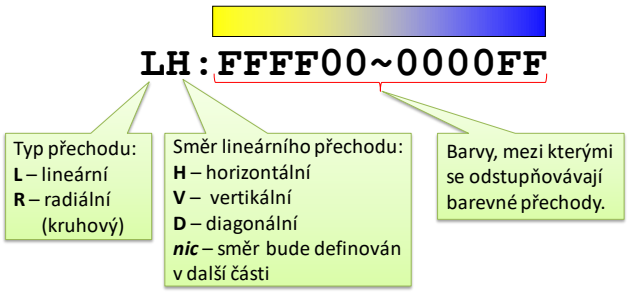
Takovýto zápis je vcelku srozumitelný a QML podporuje jeho obdobu (viz Kód 40). Je však zároveň i dosti zdlouhavý, a ač je tedy sám o sobě přehledný, celkovou přehlednost a čitelnost QML kódu tak snižuje. Jinou možností, jak definovat libovolně komplikovaný barevný přechod je zápis

pomocí mini-jazyka, tj. jako jeden souvislý textový řetězec. Čitelnost tohoto zápisu přitom není o tolik nižší, ovšem celkové přehlednosti kódu značně prospěje.

```
<background type="radial" center="0,0" origin="0,0"
    radiusX="0.05" radiusY="0.05" spread="Reflect">
  <stop color="AAFFAA" offset="0" />
  <stop color="AA00AA" offset="0.49" />
  <stop color="AA00AA" offset="0.51" />
  <stop color="AAFFAA" offset="1" />
</background>
```

Kód 40 – Ukázka zápisu radiálního přechodu v QML (elementy)

Aby bylo možné používat identifikátory náhodných barev, které jsou popsané v kapitole 4.5 (např. „P-1-2-3“), není v tomto mini-jazyku již nikde více použita jako oddělovač pomlčka. Znakem, oddělujícím hlavní části zápisu definice barevného přechodu je v tomto případě dvojtečka („:“). Není-li tato v zápisu použita, jedná se o jednobarevnou plochu (*SolidColor*). Vyskytuje-li se však dvojtečka v řetězci, pak je v jeho první části nejprve určeno, o jaký typ přechodu se jedná. Je-li prvním znakem „L“ jde o přechod lineární, pokud je to „R“ jde o přechod radiální.

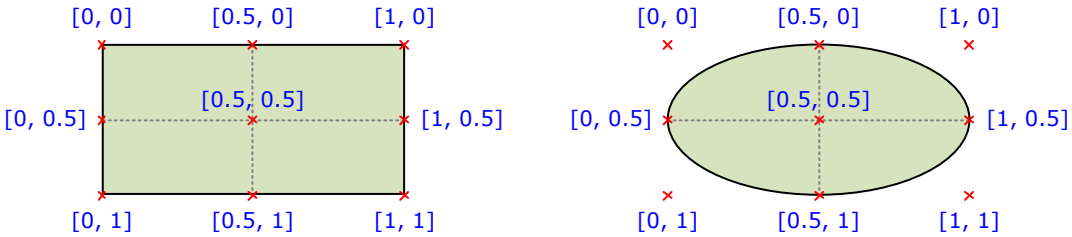


Obr. 46 – Schematická ukázka zápisu lineárního barevného přechodu

Lineární přechody, na rozdíl od radiálních, mohou v této první části mít ještě druhý znak, který umožňuje použít některý ze standardních směrů barevných přechodů. Možnosti jsou „V“ pro vertikální směr (shora dolů), „H“ pro horizontální (zleva doprava, viz Obr. 46) a „D“ pro diagonální (z levého horního rohu do pravého dolního). Zapsání tohoto znaku jako malého písmena pak obrací směr (např. „h“ bude horizontální přechod zprava doleva), v případě diagonály („d“) bude použita místo hlavní diagonály vedlejší (z pravého horního rohu do levého dolního).

Poslední možností je tento druhý znak neuvádět vůbec a definovat libovolný směr v další části zápisu (za dvojtečkou), pomocí souřadnic dvou relativních bodů[89], oddělených vlnovkou (např. „`0.5,0~0.5,1`“, značí vektor směrující z bodu [0.5, 0] do bodu [0.5, 1], což je ekvivalentní vertikálnímu přechodu „V“). V další nepovinné části lze definovat hodnotu pro vlastnost *SparedMethod*, stejným způsobem, jako je tomu u radiálních přechodů (viz níže). V poslední části zápisu jsou pak uvedeny stopy barevných přechodů.

[89] relativní souřadnice určují polohu bodu v ploše vhledem k její celkové šířce a výšce, tzn., že souřadnice [0, 0] značí bod vybarvované plochy, který je nejvíce vlevo nahoře a souřadnice [1, 1] bod, který je nejvíce vpravo dole, viz Obr. 47



Obr. 47 – Ukázka pozic s relativními souřadnicemi obdélníka a elipsy

U radiálních přechodů jsou výchozí hodnoty nastaveny na nejpoužívanější kombinaci, tj. poloměry přes celou plochu (1 a 1), poloha středu i bodu, z něhož kruhy vychází uprostřed [0.5, 0.5] a styl pokračování přechodů za hranicí kruhu je souvislá barva poslední stopy (*Pad*). Je-li třeba některou z těchto hodnot změnit, je to možné v kterékoli samostatné nepovinné části zápisu (resp. druhé až předposlední). Jejich pořadí není nijak určeno, která část definuje kterou hodnotu rozhoduje první znak jejího zápisu, jež je shodný s prvním písmenem názvu vlastnosti, kterou mění (R – Radius, C – Center, G – GradientOrigin, S – SparedMethod).

Při nastavování poloměru může za identifikátorem „R" rovnou následovat číslo, což by změnilo horizontální (*RadiusX*) i vertikální (*RadiusY*) poloměr na stejnou relativní hodnotu. Je-li třeba změnit pouze jeden z nich, nebo každý na jinou hodnotu, je třeba před toto číslo uvést identifikátor osy, ve které se poloměr nastavuje (např. „Rx0.4", „Ry0.8" nebo „Rx0.4y0.8"). Souřadnice středu a výchozího bodu pro kruhovou soustřednost se nastavují stejně (rozdílný je pouze první znak), tj. přímým vypsáním souřadnic bodu (např. „C0.7,0.3" pro střed a „G0.3,0.3" pro počátek). Možné hodnoty pro vlastnost *SparedMethod* jsou tři (pad, reflect a repeat) a její hodnotu je zde třeba vypsat celou (např. `Sreflec`). V poslední části zápisu jsou i zde uvedeny stopy barevných přechodů.

Poslední část zápisu je tedy u lineárních i radiálních přechodů společná a určuje stopy přechodu. Jejich definice je v obou případech naprosto totožná. Stopy jsou vždy určeny relativní pozicí a barvou. Jednotlivé stopy se v tomto zápisu oddělují vlnovkou („~"). Jejich pozici není povinné uvádět. Pokud nebude uvedena, dopočítá se sama jako poměr indexu stopy v jejich seznamu (index je počítán od nuly) ku jejich celkovému počtu **n** zmenšenému o 1 (tj. *offset* $= \frac{i}{n-1}$, čili první = 0, poslední = 1, prostřední u tří = 0.5 atd.). Je-li třeba pro některé stopy použít jiné pozice než tyto pravidelně odstupňované, je možné je zapsat před identifikátor barvy (ať již náhodné „`P-1-2-3`" či konstantní „`FFAABBCC`") a oddělit ji znakem svislé čáry („|").

```
R:R0.1:C0,0:G0,0:Sreflect:AAFFAA~0.49|AA00AA~0.51|AA00AA~AAFFAA
```

Kód 41 – Ukázka zápisu radiálního přechodu v QML (mini-jazyk)

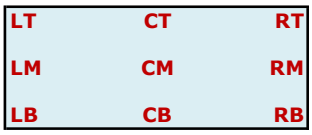
Kód 41 ukazuje zápis stejného radiálního barevného přechodu, jako ukazoval Kód 38 v XAML, Kód 39 v SVG a Kód 40 v QML rozepsaném do elementů (viz Obr. 45). Datová objemnost (zápis v tomto formátu má pouze 63 znaků) je v tomto případě pouze 20% oproti zápisu v XAML (316 znaků) a 21% oproti SVG (302 znaků), white-spaces nejsou počítány (další podobná porovnání viz Příloha 16). Přehlednost celého kódu se navíc díky stručnosti zápisu zvyšuje a čitelnost zápisu samotného není o mnoho nepřehlednější než v předchozích případech.

4.12.5 Poloha hodnotící ikony [◎ II.3.C]

Jak již bylo zmíněno, po uzamčení otázky nebo ukončení celého testu se může zobrazit jeho rozbor (viz kap. 4.1.8, str. 64). V něm se u jednotlivých aktivních prvků v každé otázce automaticky zobrazí ikona oznamující jejich základní rozdělení na tři stavy: správná volba, špatná volba a žádná volba, jež je však příčinou bodové ztráty, resp. nepřičtení bodů, které být přičteny mohly (viz Obr. 13 na str. 65). Tyto ikony se samozřejmě zobrazují co nejblíže aktivnímu prvku, který hodnotí, ovšem jejich pozici, vzhledem k neomezené variabilitě vzhledu každé z otázek, nelze paušalizovat. Jejich pozici tedy určuje autor testu.

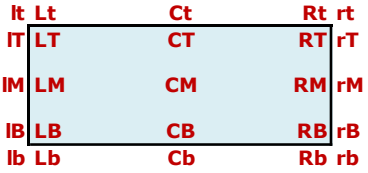
Pro různé grafické útvary se tedy hodí různá pozice této hodnotící ikony, s přihlédnutím ke grafice jejich obsahu a okolí. Jedna z obvyklých praxí je odškrtnutí nebo křížek značící chybnou odpověď uvádět v prvém dolním rohu. Podle okolí či obsahu prvku se pak určuje, zdali jej umístit uvnitř nebo vně jeho ohraničení (je-li nějaké). Stejně dobře však může posloužit i kterýkoli jiný roh či strana objektu, je-li to k okolnímu obsahu vhodnější.

Za nejběžnější lze předpokládat klasické pozice horizontální vlevo (Left), uprostřed (Center) a vpravo (Right) a vertikální nahoře (Top), uprostřed (Middle) a dole (Bottom). První písmena anglických názvů všech těchto pozic jsou různá a tak je lze použít pro dvouznakový zápis kombinace pozice ikony, čímž se získává 9 možných pozic (viz Obr. 48).



Obr. 48 – Ukázka devíti základních pozic v obdélníku

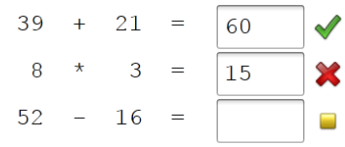
Obsah hodnoceného prvku otázky jej však může vyplňovat celý a naopak jeho okolí obsahovat volná místa, načež by mohlo být v některých případech vhodné umístit ikonu vně tohoto prvku. Je-li pro některé zarovnání (mimo středového) tento požadavek, stačí příslušné písmeno zapsat malé (např. „rB“). Tímto způsobem je možné zapsat již 25 pozic (viz Obr. 49).



Obr. 49 – Ukázka pětadvaceti rozšířených pozic v a vně obdélníku

Ne vždy je žádoucí, aby ikona byla umístěna až těsně k hranici objektu, který hodnotí, nebo naopak může být vhodné ji posunout právě na tuto hranici. Je tedy třeba umožnit rozšíření zápisu o takové posunutí. Za tímto účelem stačí za znak příslušné pozice napsat číslo (ať již kladné či záporné), jež značí počet obrazových bodů (pixelů), o nějž se má ikona v daném směru posunout (např. „r3B“ nebo „rB-4“, popř. i „r3B-4“). Pokud je třeba ji v obou směrech (horizontálním i vertikálním) posunout stejně, lze toto číslo napsat hned na začátek a není třeba jej pak znovu uvádět u každé z pozic (např. „-3rB“).

Tímto způsobem lze tedy dosáhnout v podstatě libovolné pozice ikony uvnitř i vně prvku. Přitom nejobvyklejší kombinace mají logický a dobře zapamatovatelný způsob zápisu a i na ty méně obvyklé se dá velmi snadno zvyknout.



Obr. 50 – Ukázka hodnotících ikon s polohou „r-6M“

V případě, že obsah otázky nedovoluje vhodné umístění hodnotící ikony na žádnou z možných pozic ani v okolí prvku (např. při doplňovaní slov v textovém odstavci), je možné na místo její pozice napsat hodnotu „color“ a místo ikony se celý prvek překryje obdélníkem s barvou pozadí symbolizujícím příslušný výsledek (jako u semaforu: zelená = správně, červená = špatně, oranžová = neřešeno ač řešeno mělo být). Průhlednost tohoto barevného obdélníka je animovaná, tzn., že se postupně zobrazuje až do 80% viditelnosti a pak opět mizí až na 0%, přičemž jeden tento cyklus trvá 2 sekundy.

4.12.6 Cíle, položky, body [◎ II.2.C]

Zápis bodového ohodnocení jednotlivých kombinací aktivních prvků cílů a položek (viz str. 45) lze provést nejen jako rozpis do samostatných XML elementů pro každou možnou kombinaci (viz str. 55), ale je to možné i v rámci jednoho řetězce pomocí mini-jazyka. Ten se snaží pokrýt nejobvyklejší případy hodnocení a i ty umožňuje zaznamenat ve zkratkách.

Nejčastěji bývá právě jeden cíl správný pro právě jednu položku. Ty jsou navíc díky XSLT šablonám (viz kap. 4.1.5, str. 47) obvykle označovány identickými identifikátory (např. položka s ID = 1 patří do cíle s ID = 1) a zároveň bývají nejčastěji bodovány všechny stejně, tedy např. 1 bodem. Otázka s šesti cíli a položkami pak vyžaduje do sekce s vyhodnocením `<scores>` zapsat 6 elementů, každý se třemi atributy. Mini-jazyk ovšem pro tyto případy zápis razantně zkracuje.

Zápis lze provést buď do samostatných elementů `<target>` jako v předchozím případě, ovšem za použití již pouze jednoho atributu `data`. Ten může však být i rovnou v hlavním elementu pro hodnocení cílů `<targets>` a zastoupit tak veškerý jeho obsah bez nutnosti rozepisování do dalších elementů. Ty jsou pak nezbytné pouze v případě, že jeden cíl může obsahovat více než jen jednu bodovanou položku, přičemž oba způsoby zápisu lze kombinovat.

Pro nastavení jedné kombinace cíl-položka je tedy zapotřebí tří údajů: ID cíle, ID položky a skóre, jež jejich kombinace přináší. Do atributu `data` se tedy zapíší tyto údaje v tomto pořadí za sebe, oddělené čárkou (např. „3,3,1“). Pro nejčastější případ, kdy je jeden správný pár cíl-položka bodován jedním bodem, lze poslední z hodnot vynechat a zapsanému páru tak bude dosazen právě jeden bod (např. „3,3“ je tedy totéž co „3,3,1“). Pro další obvyklý způsob, kdy správná kombinace cíle a položky bývá označena i stejným ID dokonce stačí uvést pouze ID jednoho z nich a druhé se dosadí samo (např. „3“ je tedy totéž co „3,3,1“).

Tyto jeden až tří členné kombinace se tedy zapisují do atributu `data` individuálních elementů `<target>`. V případě definice přímo v elementu `<targets>` je zápis totožný, pouze s tím, že obsahuje těchto trojkombinací více v jednom, přičemž jednotlivé kombinace se od sebe oddělují středníkem (např. „1,2,2;2,1,1.5;3,3,-1“ nebo jen „1;2;3;4;5“).

4.12.7 Úrovně [◎ IV.2.B]

Jednotlivé testy se sestavují z otázek, které je však možné používat i ve více testech. V rámci jednoho testu se jemu přiřazené otázky mohou třídit do skupiny, které kromě přehlednosti umožňují i další nastavení, jako je např. váha pro přepočet přírůstku skóre k celkovému testu za otázky vybrané z dané skupiny, nebo počet otázek, které pouze se mají z této skupiny do testu vybrat. Jedním z takovýchto parametrů, který se dá nastavit jak u skupin, tak i u samotných otázek je „úroveň". Ta umožňuje tvůrci testu rozlišit jednotlivé otázky či jejich skupiny pomocí libovolného značení do kategorií, jež mohou být i napříč skupinami. Může jít tedy třeba o tematické uspořádání (např. v matematice malá násobilka, velká násobilka, dělení apod.), ale i jakékoli jiné, přičemž každý objekt (otázka či skupina) může spadat do libovolného množství takovýchto kategorií. Tato kategorizace má výhodu v tom, že při každém spuštění testu lze definovat množinu kategorií, ze kterých pouze se mají otázky vybírat, popř. v jakém množství, čímž ovšem nejsou postižena nastavení jednotlivých skupin.

Vše je nastavitelné velmi intuitivně, a to tak, že u každé otázky i skupiny je možnost zadat libovolný textový řetězec do kolonky „úroveň" (level). Názvy úrovní si tvůrce testu může volit zcela libovolně, vyjma několika rezervovaných znaků, takže např. názvy jako „1", „A", „A1", „nasobilka" apod. jsou možné. Spadá-li objekt do více kategorií, lze je do označení úrovně vypsat všechny, oddělené čárkou (např. „A1,B1,C"). Za definici úrovně se počítá i hodnota null či prázdný řetězec (oboje je ekvivalentní), což je vždy výchozí hodnota, takže dokud zadavatel testů nezačne nějak objekty rozlišovat, jsou do každého testu povoleny vždy všechny otázky.

Zmíněných speciálních znaků, vyjma čárky, je pak využíváno výhradně až při zadávání testu, pro snazší definici výčtu povolených kategorií. Tento výčet se totiž taktéž zadává jako textový řetězec. Konkrétně jde o tyto znaky: *, ?, -, ~, #, +.

- Hvězdička (*) je zástupný znak, jež sám o sobě zahrne do výběru úplně všechny otázky, nehledě na jejich kategorie (podobný význam má tento znak i v SQL, kde znamená „vyber všechny sloupce" [111 str. 131]). Tento znak však lze zapsat i v kombinaci s jiným znakem, což pak tvoří masku pro porovnání názvů kategorií s tím, že hvězdička značí libovolnou část textového řetězce, tj. nic až jakkoli dlouhý sled znaků (např. „A*" vybere „A", „A1", „A2" i „A123").
- Otazník (?) má podobný význam jako hvězdička, ale zastupuje pouze (přesně) jeden, byť libovolný znak (např. „A?" vybere „A1" a „A2", ale nikoli „A" ani „A123").
- Mínus (-) na začátku názvu úrovně umožňuje z výběru naopak některou z kategorií vyloučit (např. „*,-A2" vybere všechny kategorie, mimo „A2"). Spadá-li však např. otázka do více kategorií, z nichž jedna je takto zakázána a druhá povolena, bude do výběru zařazena (např. „A1,-A2" povolí pro výběr otázku s označením „A1,A2").
- Vlnovka (~) má stejnou funkci jako mínus (-), ovšem takto označené kategorie z výběru vylučuje absolutně, i kdyby byl daný objekt v jiných povolených kategoriích (např. „A1,~A2" nepovolí žádnou otázku, jež spadá do úrovně „A2", čili ani „A1,A2").
- Křížek (#) umožňuje omezit počet otázek, které mohou být z dané kategorie vybrány (např. „A1#3" povolí vybrat z kategorie „A1" maximálně 3 otázky).
- Plus (+) je pouze doplňkem pro křížek a je-li tento znak napsán za číslem udávajícím početní omezení, pak není tento počet maximálním možným, ale přesným určení počtu otázek pro danou kategorii (např. „A1#3+" předepisuje do testu vybrat právě tři otázky z kategorie „A1").

Úroveň výběru	Úroveň otázek								
	1	2	1,2	A1	A2	ABC	1,A1,2	A2,3	A1,A2,ABC
*	X	X	X	X	X	X	X	X	X
?	X	X	X				X	X	
1,2	X	X	X				X		
A*				X	X	X	X	X	X
A?				X	X		X	X	X
2		X							
A1,A2				X	X		X	X	X
*,-A?	X	X	X			X	X	X	X
*,~A?	X	X	X			X			
A*,-A?						X			X

Tab. 27 – Ukázka vlivu různých filtrů (řádky) na úrovněmi označené otázky (sloupce)

4.12.8 Časování [◎ IV.2.E]

Určité události je v některých případech nezbytné omezit pouze na určitý čas. Například může jít dostupnost určitého testu nebo o členství uživatelů ve skupinách. V nejjednodušším případě lze takovouto dobu vymezit pouze dvěma daty (a časy), čili hodnotami od – do. V případě hodnocených opakovacích testů je však obvyklé, že je třeba jejich dostupnost omezit nikoli pouze na vymezené období, ale i na periodicky se opakující hodiny, např. v týdenních či čtrnáctidenních intervalech (při cvičení ve škole). Není však vyloučena i jiná perioda opakování. Zároveň je třeba mít možnost vyloučit z vymezeného času i určitá období, např. svátky, prázdniny, ale i jiné, dlouhodobě dopředu neplánované výpadky. Důležitá je i možnost omezit přístup nejen časově, ale také na určitá stanoviště, konkrétně třeba pouze na určité IP adresy. Takto nadefinované podmínky přitom musí být uložitelné do databáze a zároveň i rychle čitelné a v rámci SQL.

Pro snadný a přehledný zápis podmínek omezení časové dostupnosti byl vytvořen jednoduchý jazyk na bázi XML, kombinující v hodnotách atributů i textové mini-jazyky pro zápis časových rozsahů. Ten specifikuje čtyři elementy:

- add
- remove
- only
- repeat

Element `<add>` přidává do seznamu povolených období určitý časový rozsah, element `<remove>` naopak do povolených období přidává zakázané intervaly. Druhé dva elementy mohou být uvedeny pouze jako podřízené dvěma předchozím. Element `<only>` omezuje platnost nadřazeného elementu (`<add>` nebo `<remove>`) na více specifikované období, element `<repeat>` umožňuje nadřazenému časovému intervalu nastavit periodické opakování.

```
<add date="01.09.2011-30.06.2012">          <!-- Celý školní rok 2011/12 -->
  <only week="o1" time="08:35~20m" />       <!-- jen liché pondělky 08:35-08:55 -->
  <only week="e2" time="09:40~20m" />       <!-- nebo sudé úterky   09:40-10:00 -->
  <remove date="23.12.2011-03.01.2012" />   <!-- ale bez Vánočních svátků. -->
</add>
```

Kód 42 – Ukázka kombinace elementů **`<add>`**, **`<only>`** a **`<remove>`**

Všechny tyto elementy, s výjimkou `<repeat>`, mohou obsahovat následující tři atributy:

- date
- time
- week

Atribut `date` může obsahovat několik typů hodnot:

- konkrétní datum (např. 22.06.2012) – platí jen pro tento den
- datový rozsah
  - určený dvěma daty (např. „01.09.2011-30.06.2012“) – není-li uveden čas, zahrnuty jsou i celé dny počátečního a koncového data (tj. „01.09.2011 00:00-30.06.2012 23:59:59“)
  - určený dvěma daty s časem (např. „14.05.2012 13:05-14.05.2012 14:50“)
  - určený počátečním datem a časem a dobou trvání (např. „14.05.2012 13:05~45m“)
  - otevřený rozsah, určený jen počátkem nebo koncem (např. „01.09.2011-“ nebo „-30.06.2012“)
- datový výčet (např. „07.05.2012;14.05.2012;21.05.2012“)
- kombinace výčtu a rozsahů (např. „07.05.2012;14.05.2012 13:05~45m“)
- pouze dobu trvání (např. „~1h“) – nutno doplnit o podřízený element `<repeat>`, jež určí počátky této doby (viz Kód 46)

V uvedených zápisech se datum uvádí ve formátu „dd.MM.yyyy“, datum s časem pak ve formátu „dd.MM.yyyy HH:mm:ss“, přičemž sekundy není nezbytné uvádět. Středník („;“) v zápisech odděluje jednotlivé hodnoty výčtu, znak pomlčky (mínus „-“) je používán jako oddělovač rozsahů určených počátkem a koncem a vlnovka („~“) u rozsahů definovaných počátkem a dobou trvání. Ta je k počátku přičtena přesně na sekundu (např. označuje-li „10:00~1h“ interval X, pak platí, že 10:00:00 <= X < 11:00:00). Doby trvání jsou specifikovány hodnotou (u desetinných čísel je oddělovačem tečka) a jednotkou, které jsou tyto:

- s – sekundy
- m – minuty
- h – hodiny
- d – dny
- w – týdny
- M – měsíce (např. viz Kód 43)
- y – roky

```
<add date="01.02.2012~1M" />
```

Kód 43 – Interval zahrnující celý únor 2012 (od 1.2.2012 00:00:00 do 1.3.2012 00:00:00)

Pro atribut `time` je postup obdobný, hodnoty se však týkají již pouze času. Ten je zapisován ve formátu „HH:mm:ss“, kde hodiny jsou ve 24 hodinovém formátu (00-23) a sekundy není povinné uvádět. Časové omezení se uvádí pouze v rozsazích (např. „13:00-14:00“ nebo „13:00~1h“), popř. jako výčet rozsahů (např. „13:00-14:00;16:00~1h“).

Je-li v elementu uveden atribut `date` i `time` současně, pak musí platit obě vymezení, aby bylo rozmezí povoleno (add) či zamítnuto (remove). Tím lze omezit datový rozsah pouze na určité hodiny každého dne (např. viz Kód 44).

```
<add date="14.06.2012~1w" time="13:00~1h;16:00~1h" />
```

Kód 44 – Vymezení dvou hodinových intervalů každý den po dobu jednoho týdne

Atribut `week` umožňuje uvést výčet nebo rozsah dnů v týdnu v číselném formátu (tj. 1 = pondělí, 7 = neděle) a dále tak vymezit platnost intervalu ve vztahu k týdennímu cyklu (např. „1;3;5" nebo „1-5"). Je-li třeba navíc rozlišit lichý a sudý týden, lze před tato čísla dnů v týdnu uvést znak určující paritu týdne, konkrétně „o" (odd) pro lichý a „e" (even) pro sudý (např. „o1;e2;3" – pondělí lichého týdne, úterý sudého a každá středa, též viz Kód 45). Číslo týdne v roce je stanoveno dle normy ISO 8601, tj. že prvním týdnem roku je ten, jehož čtvrtek je v daném roce [112].

```
<add date="01.01.2012~1y" week="1-3;o4-5;e6" />
```

Kód 45 – Celý rok 2012 pondělí až středa, plus čtvrtek a pátek v liché týdny a sobotu v týdny sudé

Element `<repeat>` umožňuje pro povolení či omezení období nastavit opakování[90]. Jeho atribut `type`, určuje typ opakování, odvozený opět od standardních časových jednotek (second, minute, hour, day, week, month, year). Všechny typy opakování mají dále společné atributy `start` (nastavuje počáteční datum popř. i čas opakování), `period` (určuje číselnou hodnotu periody opakování, např. 7 u day znamená opakování každých 7 dní) a konec. Konec může být určen několika způsoby a to počtem opakování (atribut `count` obsahující celé číslo, např. viz Kód 46) nebo konečným datem (atribut `end`). Není-li uveden ani jeden z těchto možných atributů určujících konec opakování, jedná se o opakování nekonečné.

```
<add date="~5m">
  <repeat type="hour" period="12" start="25.02.2012 08:00" count="14" />
</add>
```

Kód 46 – Pět minut v 8 ráno a večer (20:00), počínaje 25.2.2012, v počtu čtrnácti opakování (tj. 7 dní)

Typy opakování week, month a year dále umožňují další pro ně specifická nastavení. U týdenního opakování jsou to dny v týdnu, jejichž číselný výčet či rozsah se uvádí do atributu `weekDay` (např. viz Kód 47). Vzhledem k určenému počátku a periodě opakování se oproti zápisu dnů v týdnu přímo do hlavního elementu (add, remove či only) již nerozlišuje lichý a sudý týden a dny jsou tedy vždy pouze čísly (např. „1;3-5").

```
<add time="18:00~2h">
  <repeat type="week" weekDay="1;4" period="4" start="01.01.2012" count="10" />
</add>
```

Kód 47 – Každý čtvrtý týden v pondělí a čtvrtek od 18:00 na 2 hodiny po 10 týdnů počínaje 1.1.2012

[90] některé možnosti opakování byly z části inspirovány opakováním událostí v aplikaci Microsoft Outlook [128 str. 88]

Měsíční opakování umožňuje vztáhnout jej k určitému dnu nebo týdnu měsíce. Pro den či dny stačí uvést jejich výčet nebo rozsah (např. „1-3;12-15“) do atributu `day`. Ve výčtu lze též použít znak „L“ (last), což značí poslední den v měsíci. Je-li potřeba počítat od konce, stačí za „L“ uvést o kolik dní před koncem měsíce se jedná (např. „L1“ je předposlední den v měsíci). U týdnů v měsíci se určuje den v týdnu a zároveň kolikátý již je v daném měsíci (např. druhá neděle v měsíci). Kombinace těchto dvou hodnot se zapisují formou výčtu do atributu `weekDay` jako dvojciferná čísla tak, že první cifra značí, kolikátý týden v měsíci to již má být a druhá cifra den v týdnu (např. druhá neděle = „27“). I zde lze místo první cifry uvést písmeno „L“, což i v tomto případě značí poslední den týdne v měsíci (např. poslední středa = „L3“, též viz Kód 48).

```
<add date="~1d">
  <repeat type="month" weekDay="L7" period="2"
          start="01.01.2012" end="31.05.2012" />
</add>
```

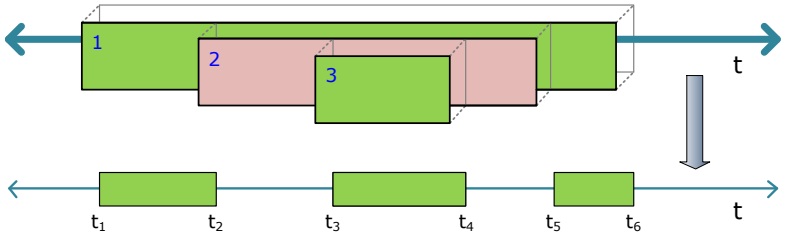
Kód 48 – Celý den každou poslední neděli v každém druhém měsíci mezi lednem a květnem 2012

Roční opakování lze vztáhnout ke dnům a měsícům. U dnů se opět jedná o jejich výčet či rozsahy v atributu `day`, přičemž se jedná o dny v roce, tj. 1-365 (v přestupném roce pak 1-366). Den může být určen i datem (nebo výčtem či rozsahy dat) v atributu `date` a to ve formátu „dd.MM“ (např. „24.12“ nebo „01.07-31.08“). Roční opakování v určitých měsících a jejich dnech se zapisuje v podobném formátu, jako tomu bylo u měsíčního opakování a atributu `weekDay`. Zde se však tyto dny zapisují do atributu `monthDay` a jsou doplněny o čísla měsíců, oddělených čárkou (např. „LD,3;27,5“ – poslední den března a druhá neděle v květnu, nebo též viz Kód 49).

```
<add date="~1d">
  <repeat type="year" monthDay="LD,2" period="4" start="01.01.0000" />
</add>
```

Kód 49 – Celý den každého posledního února na přestupný rok

Jednotlivé element s pravidly se zapisují do hlavního kořenového elementu buď pod sebe (na stejnou úroveň XML hierarchie) nebo se do sebe vnořují. Pravidla na stejné úrovni se zpracovávají postupně tak, že později zapsané pravidlo (element) má vyšší platnost („překrývá“) dříve zapsané pravidlo. Lze tak např. povolit určitý rozsah, z něho část zakázat a z této zakázané části opět podčást povolit (např. viz Obr. 51 a Kód 50).



Obr. 51 – Ilustrační schéma skládání pravidel (viz Kód 50)

```
<add date="t1-t6" />
<remove date="t2-t5" />
<add date="t3-t4" />
```

Kód 50 – Zápis k Obr. 51

Vnořené elementy oproti tomu doplňují a upřesňují nadřazený element na téže úrovni platnosti (např. viz Kód 51). Hloubka vnoření přitom není nijak omezena. Podelementů může být zároveň i více na téže úrovni, pak se spojování jejich platnosti řídí stejným postupem jako v kořenovém elementu.

```
<add date="t1-t6">
  <remove date="t2-t3" />
  <remove date="t4-t5" />
</add>
```

Kód 51 – Zápis s ekvivalentním výsledkem jako Kód 50, pomocí dvouúrovňových elementů

Kromě časového vymezení ovšem může být přístup omezen i jinými parametry. Například již výše zmíněnou IP adresou nebo u testů počtem pokusů pro jednotlivé časové intervaly. Za tímto účelem je možné doplnit libovolný element atributem `params` a v něm uvést tato další omezení ve formátu „klíč:'hodnota';" (např. „`pokusu:'3';ip:'192.168.1.*'`"). Výhodou parametrů je, že mohou být zcela libovolné, při zpracování časových kritérií se pouze eviduje jejich přítomnost a jejich obsah se neřeší. V případě potřeby rozhodnutí platnosti některého z elementů jsou však tyto parametry předány k vyhodnocení aplikaci, jež s tímto modulem pracuje.

Proces vyhodnocení takovéhoto zápisu omezení není zcela triviální operací a vyžaduje určitý výpočetní čas. Není proto zcela vhodný například pro filtraci pomocí SQL dotazů. Také ovšem není efektivní z databáze načíst a programově kontrolovat veškeré záznamy, je-li dané pravidlo aktuálně platné či nikoli. Tento problém byl vyřešen tak, že pravidlo je vyhodnocováno již v okamžiku jeho ukládání do databáze v textovém formátu (XML). Při tom se vyhodnotí, je-li pro danou chvíli pravidlo platné či nikoli a zároveň se i určí časový okamžik, kdy dojde k nejbližší změně tohoto stavu, a obě tyto hodnoty se uloží do zvláštních sloupců databázové tabulky. Při vyhledávání platných záznamů se pak hledají ty s pozitivním příznakem platnosti nebo ty, jejichž datum a čas změny platnosti je menší než aktuální. Pouze pro ně se pak znovu vyhodnotí jejich aktuální platnost a nalezne datum a čas další změny a tyto hodnoty v databázi aktualizují.

Výpočetní náročnost při prohledávání aktuálně dostupných záznamů se tak velmi snižuje na nezbytné minimum, přičemž veškeré výpočty jsou prováděny pouze pro prvního uživatele a ostatní již pracují jen s jím vypočtenými hodnotami. V případě použití dodatečných parametrů, konkrétně u spuštění testů s omezením přístupu dle IP a počtu pokusů na test, jsou jednotlivé hodnoty parametrů taktéž rozepsány do zvláštních polí databázové tabulky a mohou tak být i přesto rychle a snadno vyhodnoceny.

4.12.9 Absolvování testu [◎ IV.2.F]

Pro jednotlivá otevření testů je možné nastavit podmínky, které je třeba splnit k tzv. absolvování testu. To může být např. podmínkou pro získání nějaké známky, připuštění ke zkoušce apod. Absolvování testu přitom nemusí být pouze na základě jeho složení nad určitý počet procent, ale mělo by být možné jej definovat složitěji, v závislosti na výběru nejlepších či průměrných výsledků získaných za určité časové období, popř. období po sobě jdoucích. Vzhledem k rozsáhlým možnostem míchání otázek (vnějšího i vnitřního) tak může být vyloučena náhoda a relevantně prokázána orientace v daném učivu. Aby bylo možné tyto podmínky absolvování testu dostatečně flexibilně pro různé případy specifikovat, byl pro jejich definici vytvořen textový mini-jazyk.

Výstupem hodnocení výsledků testů by v tomto případě měla být logická hodnota (ano/ne), proto je zápis kritérií v podstatě podmínkou, obsahující klasické operátory porovnání `<`, `>`, `>=`, `<=`, `=`, `!=` a spojování více podmínek `&` (and), `|` (or). Pro souhrnné výpočty výsledků z více testů je možné použít agregační funkce `min` (minimum), `max` (maximum), `avg` (průměr) a `count` (počet), podobně jako v SQL (viz [111 str. 76]). K dispozici je i cyklus `for`. Pro určení období, za které lze výpočty provádět, jsou podporovány jednotky `last`, `hour`, `day`, `week`, `month` a `year`.

```
funkce( [ [jednotka[ [-X] | [,rozsah] ], ] [#počet,] [podmínka] ] | [pole] )
```

Kód 52 – Struktura zápisu agregační funkce

Kód 52 ukazuje strukturu obecného zápisu agregační funkce. Ta tedy může mít žádný až čtyři parametry. Žádný z nich však není potřeba v případě, že je funkce použita v cyklu, neboť pak do ní je dosazeno pole hodnot z časové jednotky (hodiny, dne, týdne, …) odpovídající aktuálnímu kroku cyklu. Není-li funkce použita v cyklu, tak by jejím prvním parametrem měla být jednotka a druhým rozsah, za který se agreguje (např. `avg(day,3)` vrací průměrný výsledek za poslední 3 dny). Výpočet se provádí vždy od aktuálního okamžiku (posledního testu) při ukládání jeho výsledku, neboť pouze poslední test vždy může být tím, který dovrší splnění kritérií absolvování testu.

Pokud rozsah není uveden, je výpočet vztažen k aktuálně probíhající jednotce (např. `avg(day)` vrací průměrný výsledek z dnešního dne, jehož počátkem je půlnoc). Je-li třeba takto počítat s dřívější než aktuální jednotkou, lze přímo za ní napsat znak mínus a počet jednotek, o které se má posunout (např. `avg(day-1)` vrací průměrný výsledek za celý včerejší den).

Speciální jednotkou je `last`, která není omezena časově, ale vrací výsledek posledního testu. Je-li ve funkci zadán její rozsah např. jako X, pak se počítá s X posledními testy (např. `avg(last,3)` vrací průměrný výsledek ze tří posledních testů). Pokud je od hodnoty `last` odečteno číslo, pak je výsledkem jedna hodnota konkrétního testu, jež se dá i přímo porovnávat (např. `last>=last-1` je podmínka, že výsledek posledního testu musí být stejný nebo lepší než výsledek testu předposledního).

Posledním parametrem funkcí může být podmínka, která umožňuje omezit agregační výpočty pouze na testy, jejichž výsledek splňuje určitou vlastnost. Pro porovnání výsledku testu v takovéto podmínce slouží klíčové slovo `score` (např. `count(day,score>=0.5)` vrací počet dnešních testů napsaných na minimálně 50%).

U funkcí `min` a `max` lze použít ještě další parametr (před podmínkou), kterým je počet. Ten, aby se odlišil od ostatních parametrů, začíná znakem `#` (křížek), za nímž následuje celé číslo. Je-li tento parametr zadán, pak funkce nevrací jednu hodnotu, ale pole hodnot, jejichž maximální délka je právě vymezena tímto počtem (např. `max(day,#3)` vrací pole tří hodnot – tří nejlepších výsledků dnešního dne; pokud bylo dnes napsáno méně testů, bude i toto pole kratší). Takovéto pole hodnot lze poté dále agregovat, tzn. použít jej jako parametr do jiné funkce (např. `avg(max(day,#3))` vrací průměrnou hodnotu ze tří nejlepších výsledků dnešního dne).

```
for(jednotka, počet, podmínka)
```

Kód 53 – Struktura zápisu cyklu

Cyklus `for`, jehož obecnou strukturu ukazuje Kód 53, umožňuje opakovaně ověřit podmínku pro více jednotek zvlášť. Parametr **jednotka** tedy určuje krok cyklu, **počet** říká, jak moc zpětně má testování podmínky probíhat a **podmínka** udává testované kritérium vracející logickou hodnotu. Aby byl výsledek cyklu pozitivní (true), musí být pozitivní i výsledek podmínky v každém kroku cyklu. V rámci podmínky lze používat veškeré agregační funkce, v nichž již nebude udávána jednotka ani rozsah, neboť tím jsou vždy výsledky testů za jednotku aktuálního kroku cyklu (např. `for(day,3,avg>=0.5)` testuje, byl-li průměrný výsledek v každém z posledních tří dnů alespoň 50%; nebo `for(last,3,score>=0.6)` testuje, byly-li výsledky z posledních tří testů alespoň 60%).

Popis podmínky	Zápis podmínky
Splnit test alespoň na 80%	`last>=0.8`
Třikrát splnil test alespoň na 80%	`count(score>=0.8)>=3`
Opakovaně po sobě třikrát splnit test alespoň na 80%	`min(last,3)>=0.8`
Průměrný výsledek tří posledních pokusů musí být alespoň 80%	`avg(last,3)>=0.8`
Průměrný výsledek za poslední 3 dny musí být alespoň 80%	`avg(day,3)>=0.8`
Tři nejlepší výsledky z posledního týdne musí mít průměr min. 80%	`avg(max(week,#3))>=0.8`
Průměrný výsledek z testování v rámci jednoho dne musí 5 dnů po sobě být alespoň 80%	`for(day,5,avg>=0.8)`
Průměr ze tří nejlepších výsledků každého dne musí 5 dnů po sobě být alespoň 80%	`for(day,5,avg(max(#3))>=0.8)`
Alespoň tři výsledky z každého dne byly 5x po sobě musí být minimálně na 80%	`for(day,5,count(score>=0.8)>=3)`
Nejlepší výsledek dne musí být 3 dny po sobě lepší nebo stejný než dne předchozího	`max(day)>=max(day-1)&` `max(day-1)>=max(day-2)`

Tab. 28 – ukázky zápisu podmínek různých kritérií pro absolvování testu

Vyhodnocení, byl-li test absolvován či nikoli se automaticky provádí po dokončení každého testu. Pokud jsou kritéria absolvování splněna, je příznak signalizující tuto informaci uložen (pro konkrétního uživatele a otevření testu) a případné další testové pokusy jej již nezmění.

Kritéria absolvování testu lze kombinovat i s časovým omezením jeho dostupnosti včetně počtního omezení jeho opakování v rámci jedné periody (viz kap. 4.12.8, str. 158). Díky tomu je možné i nastavit takové podmínky, jež by motivovaly uživatele k častějšímu opakování dané látky (byť i během testu), a ten si ji tak lépe a dlouhodoběji zapamatoval.

4.12.10 XML data [◎ III.4.D]

Testy disponují širokými možnostmi nastavení (viz kap. 4.2.1, str. 67). Tato nastavení se určují na několika úrovních. Existují jeho výchozí hodnoty, ty lze ovšem změnit definicí, byť jen částečnou, jiných hodnot pro daný test. Pro každé své spuštění pak může test mít libovolné hodnoty nastavení dále upraveny dle potřeby. Při spouštění testu přes API (viz kap. 4.3.3, str. 72) je předpoklad, že test může být zpřístupněn dlouhodoběji a hostující aplikace by mohla mít potřebu nastavení dále upravovat. Proto i ona může stávající kombinaci nastavení měnit svými hodnotami (viz Obr. 15 na 67). API komunikace je v základu realizována přes parametry URL a proto bylo zapotřebí do něho umožnit zařadit XML v nějaké přijatelné formě. URL by přitom mělo být co nejkratší a obsahovat pouze základní textová data, navíc bez znaků jako je „=“, „&“, zalomení řádků, mezery, diakritika apod.

XML zápis (viz Kód 54) samozřejmě může být reprezentován bez tzv. „white spaces“, neboli mezer a zalomení řádků, jež nejsou součástí dat, ale pouze formátují XML elementy do pro člověka přehledné podoby. Znaky jako „=“ a mezery by se pak daly zakódovat pomocí metody `UrlEncode`, která je nahradí jejich hexadecimálním zápisem (např. mezeru zapíše jako „%20“) [42 str. 139]. Tím by se však délka celého zápisu mohla značně narůst, nehledě na v XML typické „upovídané“ uzavírací tagy.

```
<settings>
  <navigator stateForAllowTestExit="Nothing" showScore="0" />
  <limits testTimeLimit="960" canChangeSolved="1" />
  <mixing mixQuestions="1" questionsCount="6" />
</settings>
```

Kód 54 – Ukázka standardního zápisu nastavení testu v XML

Jinou možností je použít zápis typu JSON. Ten oproti XML místo uzavíracích tagů, obsahujících znovu celý název elementu, používá pouze složené závorky ohraničující platnost „elementu“. Názvy elementů JSON uvádí v uvozovkách (některé konvertory je nevyžadují), přičemž standardně atributy od elementů nerozlišuje. Některé konvertory tak pro jejich rozpoznání přidávají před jejich název pomlčku (např. [w82]). Dvojtečkou se oddělují názvy elementů od jejich hodnoty, která se dle potřeby a typu uvádí samostatně (čísla), v uvozovkách (textové řetězce), popř. v hranatých závorkách (pole hodnot). Jednotlivé elementy na stejné úrovni se pak oddělují čárkami (viz Kód 55).

```
{
  "settings": {
    "navigator": { "-stateForAllowTestExit": "Nothing", "-showScore": 0 },
    "limits": { "-testTimeLimit": 960, "-canChangeSolved": 1 },
    "mixing": { "-mixQuestions": 1, "-questionsCount": 6 }
  }
}
```

Kód 55 – Ukázka zápisu nastavení testu v JSON

Tento zápis by pro tyto účely byl jistě vhodnější, nicméně i tak by mohl být ještě zestručněn a oproštěn od některých nevhodných znaků (např. uvozovky).

Konfigurace testů naštěstí obsahuje pouze hodnoty jednoduchých datových typů, tj. boolean (logická hodnota ano/ne), integer (celá čísla), dobule (desetinná čísla) a enum (výčet). Pro textové hodnoty by pak bylo třeba řešit případy, jako např. zalomení řádku, nepovolené znaky apod., což by však mohlo jít obdobně jako v XML (použitím hexadecimálního vyjádření pro tyto znaky, např. pomocí již zmíněné metody `UrlEncode`), nebo jako v JSON, kde mají tyto znaky speciální dvouznakový kód začínající obráceným lomítkem (např. `\"` je uvozovka, `\t` tabulátor, `\n` zalomení řádku atd.).

Veškeré parametry nastavení testu jsou uloženy pouze v XML atributech, takže nebylo ani nezbytné rozlišovat hodnoty elementů. Krom toho se touto cestou přenáší pouze změny v konfiguraci, nikoli konfigurace celá, načež lze předpokládat i kratší, nebo dokonce nulovou délku tohoto URL parametru.

Obdobně jako u JSON zápisu bylo použito závorkové vymezení rozsahu elementu. Tím byly zároveň opatřeny i atributy (resp. jejich hodnoty), s tím rozdílem, že pro elementy byly použity závorky kulaté a pro atributy hranaté. Díky tomuto rozdílu je vždy hned poznat, patří-li daný název elementu či atributu (podle toho, jaká závorka za ním následuje) a zároveň je i oddělen od toho následujícího (uzavírací závorkou). Nejsou tak za potřebí žádné další znaky pro vymezování ani oddělování názvů i hodnot, jako např. „<", „>", „=", uvozovky, mezery, dvojtečky, čárky apod. (viz Kód 56)

Pro zápis polí, na rozdíl od XML není nezbytné každou z jeho hodnot znovu uvádět do zvláštního elementu (např. „`<ids><id>1</id><id>2</id>…</ids>`"), ale podobně jako v JSON pouze jeho název a výčet hodnot mezi složenými závorkami (např. „`ids(id{1,2,…})`").

```
settings(navigator(stateForAllowTestExit[Nothing]showScore[0])limits(testTimeLimit[
960]canChangeSolved[1])mixing(mixQuestions[1]questionsCount[6]))
```

Kód 56 – Ukázka zápisu nastavení testu v mini-jazyku

Takovýto zápis, není-li příliš dlouhý, je přitom dobře čitelný i pro člověka. Zároveň je dosaženo maximální úspory délky zápisu, aniž by došlo ke ztrátě jakýchkoli informací a bez nutnosti slovníkové komprese. Konkrétní délky zápisu pro několik pokusných dat uvádí Tab. 29.

Popis nastavení	Počet		Délka			Úspora	
	Elementy	Atributy	XML	JSON	M-J	XML	JSON
Nastavení hodnoty v jedné sekci	1	1	59	54	44	25,4%	18,5%
nastavení hodnoty v každé sekci	6	6	195	200	160	17,9%	20,0%
Prezentovaná část nastavení testu	3	6	176	180	147	16,5%	18,3%
Nastavení omezení přepínání oken	1	8	191	193	162	15,2%	16,1%
Nastavení seznamu otázek	1	13	256	263	217	15,2%	17,5%
Kompletní nastavení celého testu	6	47	1 019	1 067	909	10,8%	14,8%
						16,8%	**17,5%**

Tab. 29 – Tabulka porovnávající délky jednotlivých druhů zápisů (XML, JSON a představený mini-jazyk) pro různá nastavení testu

Tab. 29 porovnává některé případy nastavení testu zaznamenaná v různých zápisech. Jak je patrné, navržený mini-jazyk je ve všech případech nejkratší. Poměrná úspora oproti zápisu do XML i JSON je přitom vyšší u kratších zápisů, tj. u nižšího počtu nastavovaných položek, což je právě případ, pro který je tento mini-jazyk navržen.

Pro jednoduché používání byly vytvořeny metody `XmlToStr` a `StrToXml`, které kódují/dekódují XML data (typu `XElement`) na textový řetězec (`string`) tohoto formátu a opačně. Díky tomu je z programátorského hlediska veškerá práce s takto přenášenými daty velmi jednoduchá a přitom efektivní.

4.12.11 Obecné zásady pro pomocné mini-jazyky

Jak tedy takový mini-jazyk navrhnout? Vždy záleží na tom, co je jeho cílem. Např. u barevných přechodů (kap. 4.12.4) to byla definice některého z podobjektů třídy *Brush*. Nebo u ID (kap. 4.12.1) byl výstupem seznam celých čísel (`IList<int>`), což vyžadoval používaný O-R mapovací systém[91], aby se tento seznam dal přímo uvést do filtru dat načítaných z databáze („`ID not in (…)`"). V případě časování (kap. 4.12.8) a u absolvování testu (kap. 4.12.9) šlo o srozumitelný zápis podmínek, jejichž platnost se testovala.

Určité části jednotlivých zápisů mohou mít pevnou strukturu s daným pořadím a významem jednotlivých znaků (jako např. určité typy EDI formátu, viz [113]). Tyto sekvence, by však neměly být příliš dlouhé, neboť jsou velmi křehké, tj. náchylné na chyby. Jejich znaky, na rozdíl od např. XML, již nejsou doplněny žádnými identifikátory, tudíž se s jejich délkou i exponenciálně snižuje přehlednost pro uživatele a v případě nedodržení předepsané struktury již ani algoritmus zpracovávající daný řetězec, nedokáže dekódovat žádnou jeho další část. V představených mini-jazycích se přitom více jak dvouznakové sekvence (např. u barevných přechodů LH: Lineární Horizontální směr, nebo u period XY: X-tý den v Y-tém týdnu měsíce) bez separátorů (čárky, dvojtečky, vlnovky, …) či identifikátorů (např. první znak v částech barevných radiálních přechodů), až na obecně známé zápisy, jako je např. hexadecimální zápis barvy (např. #FF0066CC), právě z těchto důvodů nevyskytovaly.

Oddělovače jednotlivých částí jsou užitečné nejen pro zpřehlednění zápisu uživatelům, ale usnadňují i jejich programové zpracování. Jsou-li totiž separátory vhodně nastaveny, lze použít metodu `Split`[92], někdy i opakovaně, a implementaci algoritmu zpracovávajícího daný mini-jazyk tak značně zjednodušit. Např. seznam ID stačí rozdělit podle čárek a vzniklé pole hodnot převést do seznamu. Narazí-li se přitom na hodnotu obsahující znak mínus, rozdělí se tento podřetězec podle něj a do seznamu se v cyklu přidají hodnoty od čísla daného prvním členem vzniklého pole do čísla daného členem druhým. Obdobně je tomu i u barevných přechodů, kde se nejprve řetězec rozdělí na části podle dvojtečky, a jednotlivé části se zpracují samostatně, přičemž ta poslední (se stopami) se dále rozdělí, nejprve podle vlnovky a každá ze vzniklých částí během jejich zpracování ještě podle svislé čáry („|" vyskytuje-li se v nich). Zpracovávající algoritmy se tak skládají téměř výhradně z cyklické kontroly jednotlivých elementárních částí, popř. pracují rekurzivně, jako je tomu v tzv. metodě „rozděl a panuj" (např. viz [114]).

O něco náročnější je implementace zpracování mini-jazyků obsahujících vymezené oblasti, obvykle závorkami (např. absolvování testu či zkrácený zápis XML dat). Zde je zapotřebí řetězec zpracovávat postupně znak po znaku. Pro určení oblasti mezi závorkami lze buď použít regulární výrazy, nebo vlastní implementaci, která je mnohdy i méně výpočetně náročná, i v případě možnosti vícero vnořených závorek téhož typu. V tomto případě lze s výhodou využít rekurzivního způsobu zpracování.

[91] jako O-R (objektově relační) mapovací systém byl použit XPO [w53]
[92] metoda `Split` rozdělí textový řetězec na pole řetězců podle zadaného dělícího znaku (např. čárka)

Při návrhu mini-jazyka taktéž platí, že defaultní hodnoty[93] by měly pokrývat co možná nejvyšší procento případů a pro ostatní vícečetné případy by měly existovat zkratky (zkrácené zápisy). Je-li zkratek příliš, není na škodu, podporuje-li systém i více druhů zápisů téhož (např. u barevných přechodů lze zápis provést jak v mini-jazyku, tak i v plně větveném QML).

V určitých případech může být též žádoucí, aby daný mini-jazyk umožňoval další rozvoj (např. přidáním nových částí do jeho zápisu), při zachování zpětné kompatibility. Pak by každá část zápisu měla být striktně identifikovatelná a algoritmus zpracovávající daný řetězec by tyto neznámé části ignoroval. Jinou možností je číslování verzí, jež by byla součástí zápisu, přičemž je-li verze vyšší než ta, kterou program dokáže zpracovat, ohlásí tuto skutečnost dříve, než se zpracováním začne, popř. nabídne svůj update.

Shrnutí

Vlastní mini-jazyky mohou pomoci vyřešit řadu úloh, na které dosud standardizovaná forma zápisu neexistuje, nebo je v daném případě příliš složitá či nepřehledná, popřípadě řeší ty úlohy, které dosud nikdo dříve neřešil. Ať se již jedná o mini-jazyky zadávané uživatelem (člověkem), či na vstupu i výstupu zpracovávané výhradně strojově, jejich struktura by měla být vždy dobře promyšlená, přehledná ale i rychle a snadno výpočetně zpracovatelná.

Mini-jazyky zde představené nechť jsou inspirací nejen pro jejich použití v jiných projektech, ale i jako směr, kterým lze vést řešení nových problémů, na něž během vývoje nejrůznějších systémů jejich tvůrci narážejí.

[93] defaultní, neboli výchozí hodnoty, jsou použity pro parametry v případě, že tyto nejsou v zápisu nastavení vůbec uvedeny

5 NASAZENÍ V PRAXI

V této části bude prezentováno několik příkladů praktického využití univerzálního testovacího prostředí. Kromě nich, byl tento systém představen i na soutěži a konferenci eLearning 2011 (viz [ap-8]), kde získal cenu odborné poroty za inovativní produkt (viz Příloha 17).

5.1 Přímé testování

[◎ III.4.A]

Testovací systém je již od první funkční zkušební verze z října 2010 nasazen při výuce programování na střední škole Podorlické vzdělávací centrum v Dobrušce. Zde byl použit jako hodnotící nástroj a to, díky své automatizaci při opravování, pro krátký opakovací test téměř v každé hodině. Počty platných známkovaných testů, které byly jeho prostřednictvím se studenty v jednotlivých pololetích na této škole napsány, ukazuje Tab. 30.

Školní rok	Pololetí	Počet testů
2010/11	1.	1 041
	2.	873
2011/12	1.	1 429
	Celkem	3 343

Tab. 30 – Počty uskutečněných platných známkovaných testů při přímém použití aplikace

5.2 LMS

[◎ III.4.B]

Univerzální testovací prostředí bylo také úspěšně integrováno do systémů LMS Moodle a částečně i Blackboard.

5.2.1 Moodle

Na střední škole Podorlické vzdělávací centrum v Dobrušce je pro podporu výuky používán LMS Moodle [w48]. Do něho byla metodou volného přístupu (viz str. 75) integrována řada kontrolních otázek shrnujících některá témata z předmětu programování (viz Příloha 18).

Moodle je open-source LMS, díky čemuž nebyl problém do něho v jazyce PHP vytvořit i vlastní modul [115 str. 59], který umožňuje pro své jednotlivé uživatele uchovávat jejich identifikátory z testovacího systému a plně je tak zabezpečenou metodou prostřednictvím API (viz kap. 4.3.3, str. 72) propojit s jejich účty na serveru s testy. Takto mohou být jejich výsledky z testů pro samostatné opakování (viz Příloha 19) jmenovitě evidovány a díky tomu při jejich příštím sestavování použita metoda cíleného výběru otázek (viz kap. 4.3, str. 70) a zároveň i definovány podmínky absolvování těchto testů (viz kap. 4.12.9, str. 162). V případě hodnocených testů lze v Moodle archivovat jejich výsledné skóre či známky.

5.2.2 Blackboard

Univerzální testovací prostředí lze v módu volného přístupu (viz str. 75) velmi snadno propojit i s LMS Blackboard, používaném na Univerzitě Hradec Králové. Krátké nehodnocené testy pro samostatné opakování látky byly zařazeny např. do kurzu Psychologie I (viz Příloha 20) pomocí HTML objektu iFrame (viz Kód 57).

```
<iframe width="99%" height="610" style="min-width: 572px; min-height: 607px;"
src="http://www.alltest.eu/TestFree.aspx?free=Lbc9d83ca02794bfbb1e6e97be06ee219&
title=Skládačka"></iframe>
```

Kód 57 – Ukázka HTML kódu pro vložení jednoduchého testu do LMS Blackboard

Blackboard je komerční LMS s uzavřeným kódem [116 str. 36], což znesnadňuje implementaci jeho zabezpečeného propojení s jinými systémy. Poskytuje sice API pro Javu, dostupné na administrátorské úrovni [117 str. 553], ale toto řešení, podobně jako u modulu pro Moodle (viz kap. 5.2.1), by opět bylo použitelné pouze pro tento jediný LMS.

Blackboard však podporuje také standard SCORM [w61], implementace jehož rozhraní (např. viz [118]) do testovacího prostředí by plnou integraci mohla zajistit [119]. Tento standard navíc podporuje i řada dalších LMS, díky čemuž by následně mělo být možné testovací prostředí s plným zabezpečením bez potřeby dalších úprav používat v každém z nich.

5.3 Externí webové aplikace [◎ III.4.B]

Univerzální testovací prostředí je multiplatformní a plně funkční ve všech nejpoužívanějších webových prohlížečích jako je Internet Explorer (viz Příloha 7), Firefox (viz Příloha 18), Google Chrome (viz Příloha 6), Opera (viz Příloha 22) a také Safari přímo pod operačním systémem Mac OS X (viz Příloha 21).

Kromě systémů pro podporu e-learningu lze testovací prostředí propojit v podstatě s libovolnou webovou stránkou či aplikací. V této části budou dva takové případy stručně představeny.

5.3.1 Web Algoritmy

Program Algoritmy byl vytvořen v roce 2005 [120] pro podporu výuky algoritmizace, datových struktur a základů programování obecně, prostřednictvím jednoduchého pseudokódu. Jen během zimního semestru 2010/11 (2.9.2010 – 6.2.2011) přibylo 526 jeho nových uživatelů [ap-6].

Na webové stránky o tomto projektu [w55] byl přidán krátký test, kde si mohou uživatelé programu ověřit své znalosti z algoritmizace (viz Příloha 22). V tomto případě je k testovacímu systému přistupováno plně zabezpečeným spojením, přičemž kontrolní součty parametrů URL jsou realizovány na straně serveru v jazyce PHP.

5.3.2 Elektronická učebnice

Pro učebnici angličtiny [121] byla vytvořena i její elektronická on-line verze, která je dostupná na [w22]. Oproti tištěné verzi disponuje tato řadou interaktivních prvků spouštěných klinutím přímo na příslušné místo v knize. Patří mezi ně přehrávání audio příloh, animace lingvistických postupů a také testování.

Testy jsou spouštěny ve vlastním okně prohlížeče (viz Příloha 23), prostřednictvím šifrované a skryté verze API. Díky úvodní registraci uživatelů pro přístup k plné verzi knihy mohou být testovacímu systému předávány jejich identifikátory, prostřednictvím kterých jsou pro každého evidovány jejich výsledky a ty mohou být v některých případech zohledněny při následném výběru otázek do opakovaně řešených testů (viz kap. 4.3, str. 70 a kap. 4.7, str. 91).

5.4 Další oblasti využití

Univerzální testovací prostředí má velký potenciál pro široké nasazení v nejrůznějších oblastech. Některé z nich jsou uvedeny v následujícím seznamu.

- Školy
  - Zkoušení (průběžné, opakovací, „pětiminutovky", …)
  - Výuka na interaktivní tabuli
  - Zápočty, zkoušky
  - Přijímací řízení
  - Státní maturity
  - LMS modul
- Veřejný sektor
  - Autoškoly (soukromé opakování i závěrečný test)
  - Rekvalifikační testy
- Firmy
  - Přijímací pohovory
  - Psychotesty
  - Testy způsobilosti
  - Periodické zkoušky (periodické ověřování znalostí nezbytných k výkonu některých povolání)
- Web
  - Podpora tutoriálů
  - Ankety
  - Dotazníky
  - Zábavné testy
  - IQ testy
  - Vědomostní soutěže
  - Captcha

Kromě uplatnění hlavního cíle této práce, lze dále využívat také její dílčí publikované výstupy a to jak pro přímé použití, tak i pro další výzkumy v oblastech, jimiž se zabývají.

5.4.1 Desktopové aplikace

Testovací prostředí lze kromě webových aplikací snadno integrovat i do aplikací desktopových (např. ilustrační ukázka fiktivní aplikace viz Příloha 24). Při vývoji např. v .NET Frameworku (Windows Forms) je možné tento postup realizovat díky standardnímu ovládacímu prvku WebBrowser, pro jehož ovládání stačí pouze nastavit správnou URL adresu [122 str. 459].

Na rozdíl od prohlížeče tak lze se serverem komunikovat nejen zabezpečenými, ale také uživateli zcela skrytými URL. Aplikaci je navíc možné doplnit o řadu dalších nástrojů a funkcí, které plugin prohlížeče z bezpečnostních důvodů neumožňuje (např. plnohodnotný fullscreen, komunikaci s jinými servery v internetu apod.)

6 DALŠÍ ROZVOJ

Univerzální testovací prostředí je ve stávajícím stavu schopno plnohodnotného provozu a v praxi je také reálně používáno (viz kap. 5, str. 169). I přesto jej však čeká další rozvoj, který bude probíhat mimo rámec disertační práce i nadále. Ve většině případů se však již nejedná o výzkumné, ale spíše implementační cíle. Pro další rozvoj je pak pozitivní, že XML (a tím pádem i QML) je jazyk, který umožňuje jeho další rozvoj při zachování plné zpětné kompatibility, takže data vytvořená nyní budou funkční i v dalších verzích.

Testy i jednotlivé otázky lze jednoduše vkládat do libovolných webových stránek (viz kap. 5.3.1, str. 170) a aplikací (viz kap. 5.3.2, str. 170) prostřednictvím rámce iFrame, popř. samostatného rámce. To je možné i v případě LMS (viz kap. 5.2.2, str. 169), ovšem zde bývá žádoucí, kromě zpřístupnění testu, umožnit také uložení výsledků zpět do těchto systémů. Lze sice vytvořit modul, který by tuto funkci obstarával (viz kap. 5.2.1, str. 169), ale to je možné pouze u některých otevřených LMS, přičemž u každého z nich je třeba postupovat individuálně. Aby tato integrace byla v případě LMS, což je v podstatě jedno z hlavních cílových prostředí, více automatizovaná, bude implementována podpora standardu **SCORM**. Objekty s tímto rozhraním pak umožňuje importovat většina stávajících LMS na uživatelské úrovni.

Testovací prostředí je možné používat i zcela samostatně (viz kap. 5.1, str. 169). Uživatelé se do systému zaregistrují pod svými jmény, správce jim umožní přístup do jím spravované organizace (viz kap. 4.3.2, str. 72) a následně mohou pod svým jménem a heslem spouštět aktuálně zadané testy (viz kap. 4.3.1, str. 70) a posléze si i prohlížet své výsledky (viz. kap. 4.1.8, str. 64). Proces přihlašování, popř. i samotné registrace, by přitom mohl být zjednodušen implementací protokolu **OAuth**[94] nebo **OpenID** [w52], jež by umožňovaly autentizaci uživatelů prostřednictvím třetích stran (např. MojeID [w44], Facebook [w25], Google [w27] atd.), přičemž rozhraní OAuth podporují i některé LMS (např. Moodle, viz [w46]). Mimo používání externích služeb pro autentizaci při vstupu do testovacího systému, by také neměl být v případě OAuth problém zavést i službu opačnou, tzn. umožnit autentizaci uživatelů registrovaných v testovacím systému aplikacím třetích stran, samozřejmě výhradně se souhlasem uživatele.

Aplikační rozhraní pro zpřístupňování testů různým webovým (viz kap. 5.3, str. 170) i desktopovým (viz kap. 5.4.1, str. 171) aplikacím již testy podporují (viz kap. 4.3.3, str. 72). V další fázi by mělo být přidáno i **administrační API**, které by umožňovalo také přístup pro kompletní správu systému prostřednictvím autentizovaných zpráv zasílaných a přijímaných externí aplikací. Otevřela by se tak cesta pro vznik nezávislých aplikací třetích stran, jež by mohly obsáhnout celkovou administraci systému, včetně např. editoru otázek, šablon apod. Mohlo by tak vzniknout i určité mikro tržní prostředí, zahrnující nejen takovéto aplikace, ale i tvorbu šablon, otázek a testů dle zadání, školení a tvorbu manuálů na jejich používání apod.

Testové otázky se sestavují ve speciálním jazyce QML (viz kap. 4.1, str. 39). Ten není nijak náročný (obtížností jej lze přirovnat např. k HTML) a lze jej editovat v libovolném textovém editoru. Podporuje však celou řadu nejrůznějších funkcí, jejichž správné použití může mnoha uživatelům činit potíže. Ne všichni jsou pak také zvyklí vytvářet grafické a funkční celky pomocí přímého zápisu v kódu. Pro tyto účely by měl vzniknout **WYSIWYG editor** otázek, umožňující sestavování QML

[94] OAuth – *otevřený protokol umožňující bezpečnou API autorizaci jednoduchou a standardní metodou z desktopových i webových aplikací* [w50]

na grafické uživatelské úrovni. Užitečná by mohla být i podpora editace některých základních typů otázek využívajících šablon (viz kap. 4.1.5, str. 47).

Vytvořené testy je sice možné sdílet napříč jednotlivými organizacemi, avšak pro jejich volnější distribuci bude přidána podpora **exportu** a **importu** nejen jednotlivých otázek, ale i celých testů, šablon (viz kap. 4.1.5, str. 47) a v rámci archivace i protokolů ze zkoušení (viz kap. 4.2.3, str. 68). Export bude možný jak v otevřeném XML formátu, tak i v šifrou zabezpečené formě (viz kap. 4.9, str. 113).

Proces interního míchání otázek (viz kap. 4.1.6, str. 48) je díky několikanásobné potřebě parsování QML zápisu a XSL transformaci (viz kap. 4.1.5, str. 47) značně výpočetně náročnou operací. I když se jedná pouze o stovky milisekund, při hromadném požadavku na server v jednom okamžiku by mohlo dojít k jeho nadměrnému zatížení a znatelnému zpomalení odezvy. Proto by měla být tato část (viz kap. 4.2.2, str. 68) **generování testu přesunuta na klientskou aplikaci**, načež by server pouze vybral otázky pro test (viz kap. 4.3, str. 70), odeslal je spolu s podklady (viz kap. 4.7, str. 91) klientské aplikaci a následně přijal, uložil a potvrdil výsledek vygenerovaný aplikací.

Testové otázky mohou obsahovat vlastní vektorovou grafiku (viz kap. 4.1.1, str. 39) a podporují také vkládání rastrových obrázků (viz str. 41), včetně možnosti jejich animace (viz kap. 4.1.2, str. 43). V další fázi by měla být přidána i podpora pro multimediální obsah, tzn. **audio** a **video**. Soubory těchto formátů by přitom mělo být možné nejenom vkládat na server s testy, ale spouštět je i z externích zdrojů (např. YouTube [w84]). V takovém případě však bude nezbytné před každým testováním znovu ověřit jejich dostupnost, aby kvůli výpadku cizího serveru nedošlo k narušení průběhu testování.

Aby mohl být systém využíván globálně i mimo česky mluvící uživatele, bude jeho testovací (viz kap. 4.3.1, str. 70) a administrační (viz kap. 4.3.1, str. 70) rozhraní **lokalizováno** do anglického jazyka, s možností následného přidání podpory jazyků dalších.

Kromě realizačních cílů nastíněných v této kapitole, existuje samozřejmě ještě mnoho dalších. Prioritou však bude zpřístupnění testovacího prostředí dalším organizacím na administrátorské úrovni a následný rozvoj dodatečných nadstaveb a funkcí pak již může být pozvolný.

7 ZÁVĚR

Disertační práce se zabývá oblastí elektronického testování, z níž bylo nejprve představeno a zhodnoceno několik stávajících systémů pro různé druhy testování. Výsledky dotazníkového šetření upozornily na nedostatky těchto systémů z uživatelského hlediska a spolu s jejich některými nepopiratelně kvalitními vlastnostmi se staly podkladem pro návrh a postupnou realizaci univerzálního testovacího prostředí. To se snaží oprostit od nedostatků již existujících systémů, rozšířit a obohatit funkcionalitu o nové žádané funkce a zároveň reagovat na současné moderní trendy v oblasti vývoje aplikací. Testovací prostředí bylo vytvořeno jako bohatá internetová aplikace v technologii Silverlight, dodržující základní principy cloud computingu.

Dílčí problémy, jež vývoj testovacího prostředí přinesl, byly řešeny převážně originálními postupy, založenými na soudobých poznatcích a výzkumech. Principy těchto subsystémů byly zároveň navrženy a popsány tak, aby jejich poznatky mohly být aplikovatelné i mimo cílové nasazení v tomto projektu a sloužily jako výchozí bod či rámcová metodika pro vývoj nebo výzkum podobně zaměřených systémů a oblastí. Patří mezi ně následující výsledky.

- Kategorizovaný **přehled existujících testovacích systémů** (viz kap. 3.1) s podrobně popsanými funkcemi, výhodami, nevýhodami, nedostatky a doporučeními.
- Dotazníkový **průzkum** (viz kap. 3.2) mezi uživateli elektronických testovacích systémů, zaměřený na míru a spokojenost s jejich užíváním, včetně přehledu požadavků na stávající a nově navrhované funkce, odhalující nedostatky těchto systémů.
- Neomezená rozmanitost otázek, podporující tvůrčí potenciál autorů testů vedla k vytvoření vlastního **jazyka pro tvorbu otázek** QML (viz kap. 4.1), umožňujícího kreslení základních prvků vektorové grafiky, správu a vkládání rastrových obrázků, transformace a animace libovolných objektů, apod. Náhodné prvky dovolují nastavit interní míchání přímo v rámci jednotlivých otázek. Ty díky tomu mohou být používány opakovaně bez rizika automatizované interpretace řešení zkoušeným, aniž by látce v otázce zahrnuté rozuměl. Aktivní prvky poskytují zkoušeným plnou interaktivitu, jejímž prostřednictvím mohou na otázky zábavnou formou odpovídat a vše je přitom kompletně automaticky vyhodnotitelné fuzzy přístupem. Rozsáhlejší definice otázek lze zároveň snadno minimalizovat použitím šablon a XSL transformací.
- Univerzální testovací prostředí disponuje několika **rozhraními** (viz kap. 4.3) pro přístup různých typů uživatelů či externích aplikací. Přímé použití pro zkoušení zajišťuje testovací rozhraní a správcům je k dispozici část administrační. Aplikační rozhraní systému pak umožňuje veškerou funkčnost a interaktivitu integrovat do dalších aplikací třetích stran, ať se již jedná o prostou webovou prezentaci, webový portál, LMS či dokonce aplikaci desktopovou.
- V rámci automatického vyhodnocování výsledků, konkrétně otevřených textových odpovědí, bylo pro různé případy otestováno, změřeno a vysvětleno několik algoritmů pro vyhodnocení míry **podobnosti textových řetězců** (viz kap. 4.4). Měření se týkala jak výsledků jednotlivých porovnání, tak i časové náročnosti na jejich výpočet. Výstupní přehledy mohou sloužit při výběru optimálního algoritmu pro konkrétní případy užití.

- Navržený postup při **míchání otázek** (viz kap. 4.6) pro sestavování periodicky zadávaných opakovacích testů umožňuje usměrňovat náhodnostní složku tak, aby byla maximalizována jejich použitelnost. Stejného principu přitom lze použít nejen pro implementace budoucích testovacích aplikací, ale může být také inspirací pro další rozvoj teorií výběrových funkcí.
- Pro uchování a vyhodnocování náhodných hodnot uvnitř složitých struktur, jež mohou tvůrci do otázek zanést, bylo použito kódování prostřednictvím **charakteristických textových řetězců** (viz kap. 4.7), které v tomto případě umožnily pokročilou práci s náhodnými prvky a jejich ovlivnění pro generování více různých, než jen čistě náhodných hodnot. Popsaný princip má však daleko širší potenciál využití a nastíněna byla nejen řada variant na další modifikace tohoto algoritmu pro různé případy, ale také několik dalších oblastí jeho možného využití v praxi.
- Systém **autentizace** (viz kap. 4.8) uživatelů umožňuje bezpečné ověřování identit pro aplikace komunikující se serverem prostřednictvím internetu, s vyloučením podvržených zpráv útočníkem typu man-in-the-middle, i při jinak nechráněné komunikaci. Implementace tohoto postupu je přitom velmi snadná a zpomalení komunikace, které přináší je přijatelně nízké.
- Popsaný **šifrovací algoritmus** (viz kap. 4.9) vychází z principu Vernamovy dokonalé šifry a lze jej používat jako alternativu k běžně rozšířeným šifrám např. na ochranu archivů a souborů pro dlouhodobou úschovu. Jeho výhodou je snadná implementace do libovolného jazyka a bezpečnost. Nevýhodou je nižší rychlost, která je však přímo determinována použitým hashovacím algoritmem, jež může být v budoucnu zrychlen.
- Princip efektivního **předávání dat** v internetu (viz kap. 4.10) automaticky mapuje relační data na objektová a zpět, přičemž jsou server a klientská aplikace spojeni pouze asynchronně prostřednictvím veřejné sítě. Zvolená metoda se oproti jiným běžně používaným postupům snaží automatizovaně vyhodnocovat požadavky na data tak, aby byla minimalizována potřeba spojení se serverem a díky tomu sníženo jeho zatížení, zrychlena práce s aplikací a tím pádem i zvýšen uživatelský komfort při práci s ní. Definice vztahů mapovaných objektů s databázovými prvky je přitom velmi snadná, neredundantní a přehledná. Výsledný subsystém je nejen bezpečným a snadno použitelným nástrojem, který může umožnit dalším tvůrcům cloud aplikací soustředit se více na vývoj aplikační logiky, bez nutnosti řešit problémy spojené s přenosem a zabezpečením dat, ale použité postupy lze jednoduše implementovat i v jiných prostředích a jazycích a zároveň mohou být inspirací pro další rozvoj v oblasti přenosu dat v cloud computing RIA.
- **Konfigurační frameworky** (viz kap. 4.11) jsou alternativou ke klasickým strukturálně či nejrozšířeněji objektově programovaným frameworkům. Popsaný postup prezentuje jejich elegantnost a výhody využití, mezi které patří plný iterativní vývoj, jednoduchá instalace, distribuce a nasazování nových verzí či modulů nenarušující provoz a bez nutnosti reinstalace na jednotlivých stanicích, snadný vývoj a servis, malá velikost aplikace, nízká paměťová náročnost, údržbu zvládne i laik v prostém textovém editoru apod. Uplatnění tohoto přístupu by mohlo být zejména v oblasti RIA a desktopových informačních systémů.

- V další části bylo prezentováno několik nově vytvořených **mini-jazyků** (viz kap. 4.12), pro různé případy užití, včetně konkrétních, měřením podložených porovnání s jejich stávajícími alternativami, existují-li. Názorně bylo prokázáno, že vlastní mini-jazyky mohou pomoci vyřešit nebo alespoň značně zjednodušit řadu úloh, na které dosud standardizovaná forma zápisu neexistuje, případně je v daném případě příliš složitá či nepřehledná. Výsledná doporučení kladou důraz především na to, aby jejich struktura byla vždy dobře promyšlená, uživatelsky přehledná, ale i rychle a snadno výpočetně zpracovatelná. Představené mini-jazyky mohou být inspirací nejen pro jejich přímé použití v jiných projektech, ale i jako směr, kterým lze vést řešení některých nových problémů.

Všechny cíle stanovené ve 2. kapitole tak byly beze zbytku úspěšně splněny. Díky tomu vzniklo nové univerzální testovací prostředí použitelné v nejrůznějších oblastech a soubor originálních algoritmizací úloh spojených s elektronickým testováním.

Předposlední kapitola (viz kap. 5) prezentuje aktuální případy nasazení testovacího prostředí v praxi, a to jak z hlediska samostatného používání, tak i jeho integrace do dalších systémů prostřednictvím aplikačního rozhraní. Navržena je také řada dalších oblastí, kde by mohlo být testovací prostředí použito. Poslední část (viz kap. 6) se pak zabývá dalším rozvojem systému do budoucna, který bude probíhat mimo rámec disertační práce i nadále.

8 SEZNAM POUŽITÉ LITERATURY

1. **Pokharel, M. a Park, J. S.** Cloud Computing: Future solution for e-Governance. *Proceedings of the 3rd international conference on Theory and practice of electronic governance.* Tallinn, Estonia : ACM, 2009. ISBN 978-1-60558-663-2.

2. **Čermák, I.** Could computing - výzva nebo hrozba. *Internet, bezpečnost a konkurenceschopnost organizací 2011.* Zlín : Univerzita Tomáše Bati ve Zlíně, 2011. stránky 66-71. ISBN 978-80-7454-012-7.

3. **Krčmář, P.** SSL není bezpečné, ukazuje případ Comodo. *Root.cz.* [Online] 4. duben 2011. [Citace: 4. duben 2011.] http://www.root.cz/clanky/ssl-neni-bezpecne-ukazuje-pripad-comodo/.

4. **Květoň, K.** Stav a perspektivy LMS/CMS. *Fenomén e-learningu v současném vzdělávání.* Praha : Econ Publishing, 2003. ISBN 80-86433-20-X.

5. **Goldberg, M. W.** WebCT and first year: student reaction to and use of a web-based resource in first year Computer Science. *ACM SIGCSE Bulletin.* New York : ACM, 1997. Sv. 29, 3. ISBN 0-89791-923-8.

6. **Malý, F.** Distribuované virtuální výukové prostředí. *Disertační práce.* Hradec Králové : Univerzita Hradec Králové, 2009.

7. **Milková, E.** Optimization of students' study habits using on-line testing. *International Conference on WEB Information Systems and Technologies (WEBIST 2008).* Funchal : Madeira-Portugal, 2008. stránky 298-303. ISBN 978-989-8111-27-2.

8. **Cole, J. a Foster, H.** *Using Moodle: Teaching with the Popular Open Source Course Management System.* 2nd Edition. Sebastopol : O'Reilly Media, 2008. ISBN 978-0-596-52918-5.

9. **Corbera, F., a další.** Development of a New MOODLE Module for a Basic Course on Computer Architecture. *ACM SIGCSE Bulletin.* New York : ACM, 2008. Sv. 40, 3. ISBN 978-1-60558-078-4.

10. **Rößling, G. a Kothe, A.** Extending Moodle to Better Support Computing Education. *ACM SIGCSE Bulletin - ITiCSE '09.* New York : ACM, 2009. Sv. 41, 3. ISBN 978-1-60558-381-5.

11. **Kružík, M.** Import testů do Moodle. *Multimediální příloha Sborníku příspěvků z konference a soutěže eLearning 2009.* Hradec Králov : Gaudeamus, 2009. ISBN 978-80-7041-971-7.

12. **Chudá, D., Trochanová, H. a Hlaváč, J.** Kritériá testovacích modulov a testov v e-learningu. *Sborník příspěvků ze semináře a soutěže eLearning 2006.* Hradec Králové : Gaudeamus, 2006. stránky 226-231. ISBN 80-7041-416-2.

13. **Brandejsová, J., Brandejs, M. a Novotný, G.** Na Masarykově univerzitě opravují písemky počítače. *6. ročník konference Alternativní metody výuky 2008.* Hradec Králové : University of Ostrava, 2008. ISBN 978-80-7041-454-5.

14. **Brandejs, M., Brandejsová, J. a Lunter, Ľ.** Využití nástrojů elektronického zkoušení IS MU při přijímacích a státních závěrečných zkouškách. *UNINFOS 2010.* Trnava : CIS TU v Trnave, EUNIS Slovensko, 2010. ISBN 978-80-8082-407-5.

15. **Brandejsová, J.** Skenováním písemek se zjednodušuje zkoušení. *muni.cz - měsíčník Masarykovy univerzity.* únor : Masarykova univerzita, 2006. ISSN 1801-0806.

16. **Hrehová, S.** Testy v edukačnom procese. *Alternativní metody výuky 2008.* Hradec Králové : Gaudeamus, 2008. ISBN 978-80-7041-454-5.

17. **Zatloukal, K. a Ulrich, M.** Hotpotatoes. *Alternativní metody výuky 2007.* Praha : Přírodovědecká fakulta Univerzity Karlovy v Praze, 2007. ISBN 978-80-7041-129-2.

18. **Anaya, K., a další.** Designing Self-Test Exercises in WebCT. *Advances in Technology-Based Education: Toward a Knowledge-Based Society.* 2003. stránky 1680-1684.
19. **Šemeláková, Ľ.** Vytvorenie všeobecnej šablóny pre podporu tvorby e-vzdelávacích materiálov v prostredí XHTML editora eXe. *Alternativní metody výuky 2007.* Praha : Přírodovědecká fakulta Univerzity Karlovy v Praze, 2007. ISBN 978-80-7041-129-2.
20. **Indzhov, H., Sokolova, M. a Totkov, G.** A software frame for modelling and runtime control of adaptive testing. *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies.* Sofia, Bulgaria : ACM, 2010. ISBN 978-1-4503-0243-2.
21. **Plháková, A.** *Dějiny psychologie.* Praha : Grada Publishing, 2006. str. 328 s. ISBN 978-80-247-0871-3.
22. **Ebbinghaus, H.** *Memory: A Contribution to Experimental Psychology.* [překl.] H. A. Ruger a C. E. Bussenius. New York : Teachers College, Columbia University, 1885.
23. **Baddeley, A.** *Vaše paměť.* [překl.] R. Kamenická. Brno : Jota, 1999. str. 335. ISBN 80-7242-046-1.
24. **Kassin, S.** *Psychologie.* [překl.] D. Brejlová, V. Balaštíková a H. Šolcová. Brno : Computer Press, 2007. ISBN 978-80-251-1716-3.
25. **Pimsleur, P.** A Memory Schedule. *The Modern Language Journal.* Boston : Blackwell Publishing, 1967. Sv. 51, č. 2, stránky 73-75. ISSN 00267902.
26. **Kowalski, J.** Forget about forgetting. *Super Memory: Forget about forgetting.* [Online] 1994. [Citace: 20. únor 2011.] http://www.supermemo.com/articles/kowal.htm.
27. **Brandejs, M., a další.** Inteligentní dril: studenti méně opakují a více si pamatují. *7. ročník konference Alternativní metody výuky 2009.* Hradec Králové : Gaudeamus, 2009. ISBN 978-80-7041-515-3.
28. **Wozniak, P., A.** Theoretical aspects of spaced repetition in learning. *Super Memory: Forget about forgetting.* [Online] 1990-2000. [Citace: 16. březen 2011.] http://www.supermemo.com/articles/theory.htm.
29. **Švarcová, I.** *Základy pedagogiky.* 2. vydání. Praha : Vysoká škola chemicko-technologická v Praze, 2008. str. 315. ISBN 978-80-7080-690-6.
30. **Kerfoot, P. B., a další.** Randomized, Controlled Trial of Spaced Education to Urology Residents in the United States and Canada. *The Journal of Urology.* Boston : American Urological Association, 2007. Sv. 177, 4, stránky 1481-1487. PMID 17382760.
31. **Pavlik, P. I. a Anderson, J. R.** Using a Model to Compute the Optimal Schedule of Practice. *Journal of Experimental Psychology: Applied.* Washington : American Psychological Association, 2008. Sv. 14, č. 2, stránky 101-117. ISSN-1076-898X.
32. **Botlík, J.** Elektronické testy pro výuku informatiky. *7. ročník konference Alternativní metody výuky 2009.* Hradec Králové : Gaudeamus, 2009. ISBN 978-80-7041-515-3.
33. **Birčáková, L. a Birčák, M.** Program na testovanie pomocou PC. *7. ročník konference Alternativní metody výuky 2009.* Hradec Králové : Gaudeamus, 2009. ISBN 978-80-7041-515-3.
34. **Lammarsch, T., a další.** A Comparison of Programming Platforms for Interactive Visualization in Web Browser Based Applications. *International Conference Information Visualisation.* Washington DC, USA : IEEE Computer Society, 2008. ISBN 978-0-7695-3268-4.
35. **Harold, E. R.** *XML 1.1 Bible.* 3. Indianapolis : Wiley Publishing, 2004. str. 1054. ISBN 0-7645-4986-3.

36. **MacVittie, L. A.** *XAML in a Nutshell.* first edition. Sebastopol : O'Reilly Media, 2006. str. 304. ISBN 978-0-596-52673-3.
37. **Lacko, L.** *Silverlight - Výukový průvodce tvorbou interaktivních aplikací.* Brno : Computer Press, 2010. str. 464. ISBN 978-80-251-2716-2.
38. **Eisenberg, J.** *SVG Essentials.* Sebastopol : O'Reilly Media, 2002. str. 368. ISBN 978-0-596-00223-7.
39. **Staníček, P.** *CSS kaskádové styly.* 2. vydání. Brno : Computer Press, 2003. ISBN 80-7226-872-4.
40. **Schindler, R.** *Rukověť autora testových úloh.* Praha : Centrum pro zjišťování výsledků vzdělávání, 2006. str. 86. ISBN 80-239-7111-5.
41. **Koong, Ch., a další.** Interactive Item Template and Editing System. *Proceedings of the Second International Conference on Innovative Computing, Informatio and Control.* Washington : IEEE Computer Society, 2007. ISBN 0-7695-2882-1.
42. **MacDonald, M. a Szpuszta, M.** *ASP.NET 3.5 a C# 2008 - tvorba dynamických stránek profesionálně.* [překl.] J. Pokorný a J. Gregor. Brno : Zoner Press, 2008. str. 1584. ISBN 978-80-7413-008-3.
43. **Holzner, S.** *XSLT - příručka interneového vývojáře.* [překl.] B. Kiszka. Brno : Computer Press, 2002. str. 515. ISBN 80-7226-600-4.
44. **Chráska, M.** *Didaktické testy.* Brno : Paido, 1999. str. 91. ISBN 80-85931-68-0.
45. **Skonnard, A. a Gudgin, M.** *Essential XML Quick Reference: A Programmer's Reference to XML, XPath, XSLT, XML Schema, SOAP, and More.* Boston : Addison-Wesley Professional, 2001. str. 432. ISBN 978-0201740950.
46. **Robinson, S., a další.** *C# Programujeme profesionálně.* [překl.] B. Kiszka. Brno : Computer Press, 2003. ISBN 80-251-0085-5.
47. **Byčkovský, P.** *Základy měření výsledků výuky. Tvorba didaktického testu.* Praha : ČVUT, 1982.
48. **Stubblebine, T.** *Regular Expression Pocket Reference.* Sebastopol : O'Reilly Media, 2003. str. 112. ISBN 978-0596004156.
49. **Vošický, Z., Lank, V. a Vondra, M.** *Matematika a fyzika.* Havlíčkův Brod : Fragment, 2007. ISBN 978-80-253-0523-2.
50. **Sancristobal, E., a další.** Integration View of Web Labs and Learning Management Systems. *In: Education Engineering (EDUCON) 2010.* Madrid : IEEE Computer Society, 2010. stránky 1409-1417. ISBN 978-1-4244-6570-5.
51. **Balog, K.** On the Investigation of Similarity Measures for Product Resolution. *Proceedings of the IJCAI'11 Workshop on Discovering Meaning On the Go in Large Heterogeneous Data (LHD-11).* Barcelona : CSIC, červenec 2011. stránky 49-54.
52. **Garcia, E.** Cosine Similarity and Term Weight Tutorial. *Information Retrieval Intelligence.* [Online] 10. říjen 2006. [Citace: 4. únor 2012.] http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html.
53. **Jaro, M. A.** Probabilistic linkage of large public health data files. *Statistics in Medicine.* Malden : Wiley, 1995. Vol. 14, stránky 491–498. ISSN 1097-0258.
54. **Winkler, W. E.** String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods.* Washington DC : American Statistical Association, 1990. stránky 354-359.

55. **Yujian, L. a Bo, L.** A Normalized Levenshtein Distance Metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Washington : IEEE Computer Society, 2007. Sv. 29, č. 6, stránky 1091-1095.
56. **Monge, A. E. a Elkan., Ch. P.** The field matching problem: Algorithms and applications. *Proceedings of the Second international conference on knowledge discovery and data mining.* Palo Alto : AAAI press, 1996. ISBN 978-1-57735-004-0.
57. **Ukkonen, E.** Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science.* Essex : Elsevier, 1992. Vol. 92, issue 1, stránky 191–211. ISSN 0304-3975.
58. **Winkler, W. E.** Overview of Record Linkage and Current Research Directions. *U.S. Census Bureau.* [Online] 8. únor 2006. [Citace: 16. listopad 2010.] http://www.census.gov/srd/papers/pdf/rrs2006-02.pdf.
59. **Palionis, P.** *Evaluation of Similarity Metrics: Supporting Matchmaking of Patients with cardiovascular disease.* Enschede, Netherlands : University of Twente, 2011. Diplomová práce.
60. **ByungWon, O., a další.** Comparative Study of Name Disambiguation Problem using a Scalable Blockingbased. *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries.* New York : ACM, 2005. ISBN1-58113-876-8.
61. **Stein, B. a Eissen, S. M.** Near Similarity Search and Plagiarism Analysis. *From data and information analysis to knowledge engineering: proceedings of the 29th Annual Conference of the German Classification Society (GfKl).* Magdeburg : Springer, 2006. stránky 430-437. ISBN 978-3-540-31313-7.
62. **Atkinson, R. L.** *Psychologie.* [překl.] E. Herman, M. Petržela a D. Brejlová. Praha : Portál, 2003. ISBN 80-7178-640-3.
63. **Scandura, J. M. a Brainerd, Ch. J.** *Structural/process models of complex human behavior.* Banff, Alberta, Kanada : Springer, 1978. str. 624 s. ISBN 90-286-0578-9.
64. **Stára, L.** *Software pro výuku cizího jazyka s využitím matematických modelů.* Praha : ČVUT, 2010. Bakalářská práce.
65. **Mohamad, J. a Maurice, B.** A comparative study of learning curves with forgetting. *Applied Mathematical Modelling.* New York : Elsevier Science, August 1997. Volume 21, Issue 8, stránky 523-531. ISSN 0307-904X.
66. **Ross, S. M.** *Introduction to Probability Models.* 10. vydání. Los Angeles : Academic Press, 2009. ISBN 978-0-12-375686-2.
67. **Maulik, U., Bandyopadhyay, S. a Mukhopadhyay, A.** *Multiobjective Genetic Algorithms for Clustering (Applications in Data Mining and Bioinformatics).* Heidelberg : Springer, 2011. ISBN 978-3-642-16614-3.
68. **Hynek, J.** *Genetické algoritmy a genetické programování.* Praha : Grada Publishing, 2008. ISBN 978-80-247-2695-3.
69. **Jedlička, M.** *Využití genetických algoritmů pro optimalizaci výběru sanačních technologií.* Brno : Masarykova univerzita, 2006. Diplomová práce.
70. **Krpata, P.** *Separace signálů vibrací točivých strojů s využitím genetických algoritmů.* Praha : ČVUT, 2010. Diplomová práce.
71. **Naranbaatar, N.** *Vybrané problémy výpočtů na PC s pohyblivou řádovou čárkou.* Pardubice : Univerzita Pardubice, 2010. Bakalářská práce.

72. **Meloun, M. a Militký, J.** Přednosti analýzy shluků ve vícerozměrné statistické analýze. *Sborník přednášek z konference Zajištění kvality analytických výsledků.* Medlov : 2 THETA, 22. - 24. 3. 2004. stránky 29-46. ISBN 80-86380-22-X.
73. **Pavelka, F. a Klímek, P.** *Aplikovaná statistika.* 1. vydání. Zlín : FaME, 2000. ISBN 80-214-1545-2.
74. **Ashley, D.** Obfuscation used by an HTTP Bot. *Information Security Office.* [Online] 20. září 2010. [Citace: 19. únor 2012.] http://security.utexas.edu/consensus/Obfuscation.pdf.
75. **Silva, F. O., Pacheco, J. A. a Rosa, P.** A Web Service Authentication Control System Based on SRP and SAML. *12th International Conference Information Visualisation.* Washington DC, USA : IEEE Computer Society, 2005. ISBN0-7695-2409-5.
76. **Pappa, J.** *Silverlight - datové služby.* Brno : Zoner Press, 2009. ISBN 978-80-7413-041-0.
77. **Hogg, J., a další.** *Web Service Security: Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements (WSE) 3.0.* USA : Microsoft Press, 2006. ISBN 978-0735623149.
78. **Li, J., a další.** Robust Remote Authentication for ScalableWeb-Based Services. *International Conference on Intelligent Information Hiding and Multimedia Signal Processing.* Washington DC, USA : IEEE Computer Society, 2008. ISBN 978-0-7695-3278-3.
79. **Sumter, L.** Cloud Computing: Security Risk. *Proceedings of the 48th Annual Southeast Regional Conference.* Oxford, Mississippi : ACM, 2010. ISBN 978-1-4503-0064-3.
80. **Menezes, A., J., Oorschot van, P., C. a Vanstone, S., A.** *Handbook of Applied Cryptography.* Boca Raton : CRC Press, 1996. ISBN 978-0-8493-8523-0.
81. **Pinkava, Jaroslav.** Hashovací funkce v roce 2004. *Crypto-World.* roč. VI, č. 9, Praha, říjen 2004, stránky 15-18, ISSN 1801-2140.
82. **Burnett, M.** *Perfect Passwords: Selection, Protection, Authentication.* Rockland, MA : Syngress Publishing, Inc., 2005. ISBN 978-1597490412.
83. **Hubálovský, Š. a Musílek, M.** Počítačová bezpečnost ve výuce informatiky (Tvorba hesel a steganografie). *Matematika-fyzika-informatika.* Praha : Prometheus 2010, roč. 20, listopad 2010. ISSN 1210-1761.
84. **Jícha, R.** Salted hash - další krok ke zvýšení bezpečnosti. *Interval.cz.* [Online] 10. únor 2005. [Citace: 10. červen 2010.] http://interval.cz/clanky/salted-hash-dalsi-krok-ke-zvyseni-bezpecnosti/.
85. **Leach, P., Mealling, M. a Salz, R.** A Universally Unique IDentifier (UUID) URN Namespace. *RFC 4122.* [Online] červenec 2005. [Citace: 10. červen 2010.] http://www.ietf.org/rfc/rfc4122.txt.
86. **Císař, P.** *InterBase / Firebird - Tvorba, administrace a programování databází.* Brno : Computer Press, 2003. str. 453. ISBN 80-7226-956-1.
87. **Singh, S.** *Kniha kódů a šifer.* Praha : Dokořán a Argo, 2009. ISBN 978-80-7363-268-7.
88. **Piper, F. a Murphy, S.** *Kryptografie: Průvodce pro každého.* [překl.] P. Mondschein. Praha : Dokořán, 2006. ISBN 80-7363-074-5.
89. **Janeček, J.** *Gentlemani nečtou cizí dopisy.* Brno : Books, 1998. ISBN 80-85914-90-5.
90. **Shannon, C. E.** Communication Theory of Secrecy Systems. *Bell System Technical Journal.* 1949, 28, stránky 656–715.
91. **Hála, V.** Kvantová kryptografie. *Aldebaran Bulletin.* [Online] 14/2005, roč. 4. [Citace: 12. září 2010.] http://aldebaran.cz/bulletin/2005_14_kry.php. ISSN 1214-1674.
92. **Vernam, G. S.** *Secret signaling system. 1310719* U.S. Patent, 22. červenec 1919.

93. **Dušek, M.** Kvantová kryptografie. *Koncepční otázky kvantové teorie.* [Online] [Citace: 23. srpen 2010.] http://muj.optol.cz/dusek/predn/kokt/krypt.htm.
94. **Andrew, R., a další.** A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. *NIST Special Publications (800 Series).* [Online] duben 2010. [Citace: 21. srpen 2010.] http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf. SP 800-22 Rev 1a.
95. **Pierre, L. a Richard, S.** *TestU01: A C Library for Empirical Testing of Random Number Generators.* Université de Montréal : ACM Trans. Math. Softw. 33, 4, Article 22, 2007. DOI=10.1145/1268776.
96. **Klíma, V.** Blue Midnight Wish, kandidát na SHA-3 aneb poněkud privátně o tom, jak jsem k BMW přišel. [editor] P. Vondruška. *Crypto-World.* 2/2009, roč. 11, stránky 2-12.
97. **Boon, C., Philippaerts, P. a Piessens, F.** Practical experience with the .NET cryptographic API. *Katholieke Universiteit Leuven.* [Online] listopad 2008. [Citace: 24. 1 2012.] https://lirias.kuleuven.be/bitstream/123456789/208274/1/CW531.pdf. CW Reports vol: CW531.
98. **Esposito, D.** *ASP.NET a ADO.NET - tvorba dynamických webových stránek.* [překl.] J. Černý. Praha : Grada Publishing, 2003. str. 352. ISBN 80-247-0474-9.
99. **Richardson, L. a Ruby, S.** *RESTful Web Services.* Sebastopol : O'Reilly Media, 2007. str. 448. ISBN 978-0-596-52926-0.
100. **Liberty, J. a Xie, D.** *Programming C# 3.0.* 5. vydání. Sebastopol : O'Reilly Media, 2008. ISBN 978-0-596-52743-3.
101. **Hilyard, J. a Teilhet, S.** *C# 3.0 Cookbook.* 3. vydání. Sebastopol : O'Reilly Media, 2007. ISBN 978-0-596-51610-9.
102. **Pialorsi, P. a Russo, M.** *Microsoft LINQ : kompletní průvodce programátora.* [překl.] J. Fadrný. 1. vydání. Brno : Computer Press, 2009. ISBN 978-80-251-2735-3.
103. **Lowy, J.** Introducing System.Transactions. *Microsoft Developer Network.* [Online] 2005. [Citace: 12. prosinec 2010.] http://msdn.microsoft.com/en-us/library/ms973865.aspx.
104. **Jarzabek, S., a další.** XVCL: XML-based Variant Configuration Language. *Proceedings of the 25th International Conference on Software Engineering (ICSE '03).* Washington DC : IEEE Computer Society, 2003. stránky 810-811. ISBN 0-7695-1877-X.
105. **Smištík, Z.** *Moderní technologie pro vývoj webových aplikací a jejich výkon.* Brno : Vysoké učení technické v Brně, Fakulta informačních technologií, 2008. Diplomová práce.
106. **Bishopová, J.** *C# - návrhové vzory.* Brno : Zoner Press, 2010. ISBN 978-80-7413-076-2.
107. **Púdelka, I.** *Aspektovo orientovane programovanie a jeho podpora.* Brno : Masarykova univerzita, 2010. Diplomová práce.
108. **Schult, W. a Polze, A.** Aspect-oriented programming with C# and .NET. *In: Proceedings of the Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing.* Washington, DC : IEEE Computer Society, 29.4. - 1.5. 2002. stránky 241-248. ISBN 0-7695-1558-4.
109. **Ghosh, J. a Cameron, R.** *Silverlight Recipes: A Problem Solution Approach.* New York : Apress, 2009. ISBN 978-1-4302-2435-8.
110. **Cardenosa, J., Gallardo, C. a Martin, A.** Internationalization and localization after system development: A practical case. *Proceedings of the Fourth International Conference "Information Research and Applications" i.TECH 2006, Varna, Bulgaria.* Sofia : FOI-COMMERCE, 20.-25. 6. 2006. stránky 207-214. ISBN 978-954-16-0036-8.

111. **Šimůnek, M.** *SQL - kompletní kapesní průvodce.* 1. vyd. Havlíčkův Brod : Grada Publishing, 1999. ISBN 80-7169-692-7.
112. **Kuhn, M.** A summary of the international standard date and time notation. *The Computer Laboratory.* [Online] 1. prosinec 2004. [Citace: 17. leden 2012.] http://www.cl.cam.ac.uk/~mgk25/iso-time.html.
113. **Reichel, D.** Jak na elektronickou výměnu dat (EDI)? *CIO Business World.* [Online] září 2009. [Citace: 18. leden 2012.] http://data.businessworld.cz/file/elektronicka-vymena-dat.pdf.
114. **Torresen, J.** A Scalable Approach to Evolvable Hardware. *Genetic Programming and Evolvable Machines.* Dordrecht, Netherlands : Springer, 2002. Vol. 3, no. 3, stránky 259-282. ISSN 1573-7632.
115. **Moore, J. a Churchward, M.** *Moodle 1.9 Extension Development.* Birmingham : Packt Publishing, 2010. ISBN 978-1-847194-24-4.
116. **Bri, D., a další.** A Study of Virtual Learning Environments. *Wseas transactions on advances in engineering education.* Wisconsin : WSEAS, 2009. Issue 1, Volume 6, January 2009, stránky 33-43. ISSN 1790-1979.
117. **Cardinaels, K., Meire, M. a Duval, E.** Automating Metadata Generation: the Simple Indexing Interface. *Proceedings of the 14th international conference on World Wide Web.* New York : ACM, 2005. stránky 548-556. ISBN1-59593-046-9.
118. **Gonzalez-Barbone, V. a Anido-Rifon, L.** Creating the first SCORM object. *Computers & Education.* Oxford : Elsevie Science, 2008. vol. 51, no. 4, stránky 1634–164. ISSN 0360-1315.
119. **Guler, C., Altun, A. a Askar, P.** Developing Reusable Learning Objects: Hacettepe University Case. *Proceedings of the Third International Conference on Internet Technologies and Applications (ITA 09).* Wrexham, North Wales : Glyndwr University, 8. - 11. 8. 2009. ISBN 978-0-946881-65-9.
120. **Voborník, P.** *Programovací jazyk pro podporu výuky algoritmů.* Hradec Králové : Univerzita Hradec Králové, Fakulta informatiky a managementu, 2006. Diplomová práce.
121. **Kelly, M.** *English for Secondary Schools.* Dobruška : Střední škola - Podorlické vzdělávací centrum, 2011. ISBN 978-80-260-0917-7.
122. **Bayer, J.** *C# 2005 - Velká kniha řešení.* Brno : Computer Press, 2007. ISBN 978-80-251-1620-3.
123. **Kubrický, J.** Interaktivita webových aplikací prostřednictvím technologie AJAX. *Modernizace vysokoškolské výuky technických předmětů.* Hradec králové : Gaudeamus, 2009. ISBN 978-80-7041-611-2.
124. **Rahman, S., Nguyen, T. A. a Yang, A. T.** Developing certificate-based projects for web security classes. *Journal of Computing Sciences in Colleges.* Houston : Consortium for Computing Sciences in Colleges, 2006. Sv. 21, č. 5. ISSN1937-4771.
125. **Moldaschl, M.** *Rich internet application development.* Vídeň : Vienna University of Economics and Business, 2011. Bakalářská práce.
126. **Ikeda, D., Yamada, Y. a Hirokawa, S.** Expressive Power of Tree and String Based Wrappers. *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03).* Acapulco, Mexico : International Joint Conference on Artificial Intelligence, 9. -10.8. 2003. stránky 21-26.
127. **Phan, T.** *Enhanced SOAP Performance for low bandwidth environments.* Melbourne, Austrálie : RMIT University, 2007. Diplomová práce.
128. **Procházka, D.** *Outlook 2010.* Praha : Grada Publishing, 2010. ISBN 978-80-247-3499-6.

129. **Cifuentes, C.** *Reverse Compilation Techniques.* Faculty of Information Technology : Queensland University of Technology, 1994. Disertační práce.
130. **Richter, J.** *CLR via C#.* 3. vydání. Redmond : Microsoft Press, 2010. ISBN-13: 978-0-7356-2704-8.
131. **Manoharan, S.** Application state, serialization and versioning. *Proceedings of IADIS International Conference Applied Computing 2004.* Lisabon : IADIS, 2004. stránky 57-60. ISBN 972-98947-3-6.

8.1 Webové stránky

[w1] ADO.Net DataSet for Silverlight Applications, [8.12.2010], <www.silverlightdataset.net>
[w2] AICC, [20.2.2011], <http://www.aicc.org>
[w3] Alf – program na tvorbu interaktivních testů, [25.3.2011], <http://www.interaktivnitesty.cz>
[w4] Amazon web services, [9.12.2010], <aws.amazon.com>
[w5] Anki – FAQ, [19.3.2011], <http://www.ankisrs.net/docs/FrequentlyAskedQuestions.html>
[w6] Anki – friendly, intelligent flashcards, [19.3.2011], <http://www.ankisrs.net>
[w7] Anki, [19.3.2011], <http://anki.ichi2.net/account/login>
[w8] Articulate Quizmaker, [4.4.2011], <http://www.articulate.com/products/quizmaker.php>
[w9] Articulate, Moodle, and SCORM, [28.3.2011], <http://www.joedeegan.blogspot.com/2010/06/articulate-moodle-and-scorm.html>
[w10] ClassMarker, [7.1.2011], <http://www.classmarker.com>
[w11] Cryptographic hash algorithm competition, [5.6.2010], <http://csrc.nist.gov/groups/ST/hash/sha-3/>
[w12] Designer testových otázek, [15.1.2012], <www.alltest.eu/Design.aspx>
[w13] DevExpress Grid, [10.1.2012], <http://www.devexpress.com/Products/NET/Controls/Silverlight/Grid_Pro/grid.xml>
[w14] DevExpress Tree View, [10.1.2012], <http://www.devexpress.com/Products/NET/Controls/Silverlight/Grid_Pro/tree.xml>
[w15] Dokeos – Open Source E-Learning, [7.4.2011], <http://www.dokeos.com>
[w16] Dokeos – Worldwide map, [7.4.2011], <http://www.dokeos.com/en/community.php>
[w17] Dokumenty Google, [20.3.2011], <http://www.google.com/google-d-s/intl/cs/documents/>
[w18] DotNetNuke, [15.1.2012], <http://www.dotnetnuke.cz>
[w19] Drupal, [15.1.2012], <http://www.drupal.org>
[w20] e-Learning Authoring Tool, [25.3.2011], <http://www.e-learningconsulting.com/products/authoring/authoring.html>
[w21] Editor nastavení testu, [15.1.2012], <http://www.alltest.eu/demo/SettingsDemo.aspx>
[w22] English for Secondary Schools, [18.2.2012], <http://english.cjlc.eu>
[w23] eXe, The eLearning XHTML editor, [23.3.2011], <http://www.wikieducator.org/Online_manual>
[w24] eXeLearning, [23.3.2011], <http://www.exelearning.org>
[w25] Facebook Authentication, [11.2.2012], <http://developers.facebook.com/docs/authentication>
[w26] Firebird RDBMS, [20.9.2010], <http://www.firebirdsql.org>

[w27] Google – Using OAuth 2.0 to Access Google APIs, [11.2.2012], <http://code.google.com/apis/accounts/docs/OAuth2.html>
[w28] Grant GR/N15764/01, [3.11.2011], <http://gow.epsrc.ac.uk/ViewGrant.aspx?GrantRef=GR/N15764/01>
[w29] Hot Potatoes Home Page, [27.3.2011], <http://hotpot.uvic.ca>
[w30] Imaging – Stretching an Image, [21.1.2012], <http://msdn.microsoft.com/en-us/library/cc189005(v=vs.95).aspx#stretching_an_image>
[w31] IS MU – Dril - učení s prodlevami, [16.3.2011], <http://www.is.muni.cz/elportal/mmp2008/cs_dril.pl>
[w32] IS MU – Dril – Učení zpaměti snadno, efektivně a s trvalým výsledkem, [16.3.2011], <http://www.is.muni.cz/elportal/nastroje/dril>
[w33] IS MU – GEM – Generátor kódu multimediálních prvků, [15.3.2011], <http://www.is.muni.cz/do/rect/el/nastroje/gem/gem.html>
[w34] IS MU – Multiquest – generátor otázek s náhodnými parametry, [15.3.2011], <http://www.is.muni.cz/do/rect/el/nastroje/multiquest/multiquest.html>
[w35] IS MU – Oživený test, [15.3.2011], <http://www.is.muni.cz/do/1499/el/nastroje/oziveny_text/oziveny_text.html>
[w36] IS MU – Questomat - obrázkové otázky v Odpovědnících, [15.3.2011], <http://www.is.muni.cz/elportal/tipy/questomat.pl>
[w37] JSON, [9.12.2010], <www.json.org>
[w38] iŠkola.cz, [13.3.2011], <http://www.iSkola.cz>
[w39] Joomla, [15.1.2012], <http://www.joomla.org>
[w40] Localizing Silverlight-based Applications, [10.1.2012], <http://msdn.microsoft.com/en-us/library/cc838238(VS.96)>
[w41] Memostation – Učební metody, [17.3.2011], <http://www.memostation.net/cs/ucebni-metody>
[w42] Memostation, [17.3.2011], <http://www.memostation.net>
[w43] Metoda nejmenších čtverců, [19.1.2012], <http://nb.vse.cz/~tichy/MNC.htm>
[w44] MojeID, [11.2.2012], <http://www.mojeid.cz>
[w45] Moodle Demonstration Site, [28.3.2011], <http://demo.moodle.net>
[w46] Moodle Mobile Open Auth, [11.2.2012], <http://docs.moodle.org/dev/Mobile_Open_Auth>
[w47] Moodle Statistics, [28.3.2011], <http://www.moodle.org/stats>
[w48] Moodle, [28.3.2011], <http://www.moodle.cz>
[w49] MSDN Blogs – ADO.NET team blog – DataSet a Silverlight, [8.12.2010], <http://blogs.msdn.com/b/adonet/archive/2009/05/26/dataset-and-silverlight.aspx>
[w50] OAuth, [2.2.2011], <http://www.oauth.net>
[w51] OData, [10.12.2012], <www.odata.org>
[w52] OpenID, [11.2.2012], <http://www.openid.net>
[w53] ORM - eXpress Persistent Objects (XPO), [18.1.2012], <http://www.devexpress.com/products/NET/ORM/>
[w54] Pseudo-Czech Dummy Text Generator, [3.2.2012], <http://www.wellstyled.com/tools/dummy-cz>
[w55] Program Algoritmy, [1.2.2012], <http://algds.cronos.cz>

[w56] Program Testy, [2.2.2011], <http://www.mikmik.cz/produkty/testy>
[w57] Překladač Google, [3.2.2012], <http://translate.google.cz>
[w58] Python Programming Language, [17.3.2011], <http://www.python.org>
[w59] Regulární výrazy, [6.10.2010], <http://www.regularnivyrazy.info>
[w60] RFC 4632 (CIDR), [29.3.2011], <https://datatracker.ietf.org/doc/rfc4632/?include_text=1>
[w61] SCORM, [31.1.2012], <http://www.adlnet.gov/capabilities/scorm>
[w62] Seznam českých slov, [2.11.2011], <http://www.oficialni.cz/slova>
[w63] Silverlight Toolkit, [5.5.2010], <http://silverlight.codeplex.com>
[w64] SimMetrics, [3.11.2011], <http://sourceforge.net/projects/simmetrics>
[w65] StatSoft – Statistica, [6.4.2011], <http://www.statsoft.cz>
[w66] SuperMemo – Application of a computer to improve the results obtained in working with the SuperMemo method, [18.3.2011], <http://www.supermemo.com/english/ol/sm2.htm>
[w67] SuperMemo – Development of SuperMemo (1985-2006), [18.3.2011], <http://www.supermemo.com/english/sms.htm>
[w68] SuperMemo – Early implementations of SuperMemo, [18.3.2011], <http://www.supermemo.com/articles/soft/sm2.htm>
[w69] SuperMemo – SuperMemo 2 Plug-In for SuperMemo for Window: Delphi source code, [18.3.2011], <http://www.supermemo.com/english/ol/sm2source.htm>
[w70] SuperMemo: how aids learning, [18.3.2011], <http://www.supermemo.eu/how_supermemo_aids_learning>
[w71] SuperMemo UX, [18.3.2011], <http://www.supermemo.eu/supermemo_ux>
[w72] SuperMemo, [18.3.2011], <http://www.supermemo.eu>
[w73] Systémy testování – dotazník (1. verze), [20.3.2011], <http://www.goo.gl/kEbOb>
[w74] Terasoft – Výukové programy, [7.4.2011], <http://www.terasoft.cz>
[w75] Testovací systémy – dotazník (finální verze), [28.3.2011], <http://www.goo.gl/D6NCK>
[w76] Testpark, [20.3.2011], <http://www.testpark.cz>
[w77] The Mnemosyne Project - Principles, [17.3.2011], <http://www.mnemosyne-proj.org/principles.php>
[w78] The Mnemosyne Project, [17.3.2011], <http://www.mnemosyne-proj.org>
[w79] The Story of the Ribbon, [20.3.2011], <http://blogs.msdn.com/b/jensenh/archive/2008/03/12/the-story-of-the-ribbon.aspx>
[w80] Univerzální testovací prostředí – demonstrační verze, [22.2.2012], <http://www.alltest.eu/demo>
[w81] W3C Math Home, [20.3.2011], <http://www.w3.org/Math>
[w82] XML to JSON conversion, [17.12.2011], <http://jsontoxml.utilities-online.info>
[w83] XSL Transformations (XSLT) Version 2.0, [25.9.2010], <http://www.w3.org/TR/xslt20>
[w84] YouTube, [24.1.2012], <http://www.youtube.com>

9 AUTORSKÉ PUBLIKACE

9.1 Související s tématem disertační práce

[ap-1] **Voborník, P.**: *Vytváření, nasazování a provoz LMS na střední škole „Podorlické vzdělávací centrum*". In: DIVAI 2008 – Dištančné vzdelávanie v aplikovanej informatike - zborník príspevkov, 28.–29. 5. 2008, FPV UKF, Nitra, 2008, str. 328-333. ISBN 978-80-8094-317-2.

[ap-2] **Voborník, P.**: *LMS pro střední školy*. In: Sborník příspěvků z konference a soutěže eLearning 2008, 11.–13. 11. 2008, Gaudeamus, Hradec Králové, 2008, str. 355-359. ISBN 978-80-7041-143-8.

[ap-3] **Voborník, P.**: *Bezpečná autentizace aplikace klient-server v internetu pomocí povinně unikátních saltů*. In: Internet, bezpečnost a konkurenceschopnost organizací 2011, 16.–17. 3. 2011, UTB, Zlín, str. 347-354. ISBN 978-80-7454-012-7.

[ap-4] **Voborník, P.**: *Počítačové testovací systémy*, In: Sborník příspěvků z konference Alternativní metody výuky 2011, 28. 4. 2011, UK, Praha. ISBN 978-80-7435-104-4.

[ap-5] **Milková, E., Voborník, P.**: *Multimediální pomůcky pro výuku algoritmizace*, Technológia vzdelávania, vol. 18, no. 2, 2011. ISSN 1335-003X.

[ap-6] **Voborník, P.**: *Teaching algorithms using multimedia tools*, In: The proceedings of the conference ERIE 2011, 9.–10. 6. 2011, FEM CULS, Praha, str. 312-321. ISSN 1803-1617.

[ap-7] **Voborník, P.**: *Efektivní předávání objektově-relačně mapovaných dat v cloud computingu*. Informační technologie pro praxi 2011, 6.-7. 10. 2011, VŠB-TUO, Ostrava, str. 189-197. ISBN 978-80-248-2487-1.

[ap-8] **Voborník, P.**: *Univerzální testovací prostředí*. In: Sborník příspěvků z konference eLearning 2011, Hradec Králové, Gaudeamus UHK, 8.-10. 11. 2011, str. 80-85. ISBN 978-80-7435-153-2.

[ap-9] **Voborník, P.**: *Výpočetní model inteligentního výběru testových otázek*. In: The proceedings of the 14th international conference MEKON 2012, 1.-2. 2. 2012, VŠB-TUO, Ostrava. ISBN 978-80-248-2552-6.

[ap-10] **Voborník, P.**: *Rychlá multiplatformní autentizace v internetu*. In: IP Networking 1 – Theory and practice (zborník vedeckých prác), Žilinská univerzita v Žiline, Žilina, 2011, str. 41-45. ISBN 978-80-554-0494-3.

[ap-11] **Voborník, P.**: *Téměř dokonalá šifra*. Matematika-fyzika-informatika. ISSN 1210-1761. (v tisku)

9.2 Ostatní

[ap-12] **Milková, E., Voborník, P.**: *Program Algoritmy – důležitý doplněk výuky algoritmizace*. In: Sborník příspěvků ze semináře a soutěže eLearning 2006, 7.–19. 11. 2006, Gaudeamus, Hradec Králové, 2006, str. 284-288. ISBN 80-7041-416-2.

[ap-13] **Voborník, P.**: *Výuka algoritmizace pomocí programu Algoritmy*. In: VIII. Vedecká konferencia doktorandov a mladých vedeckých pracovníkov - zborník príspevkov, 18.–19. 4. 2007, FPV UKF, Nitra, 2007, str. 769-774. ISBN 978-80-8094-106-2.

[ap-14] CD k **Milková, E.**: *Algoritmy – Objasnění, procvičení a vizualizace základních algoritmických konstrukcí*. Alfa nakladatelství, Praha, 2008, s. 114. ISBN 978-80-87197-10-3.

[ap-15] **Milková, E., Beránek, L., Voborník, P.**: *Algorithms – Explanation, Exercises and Visualization of the Basic Algorithmic Constructions*. Gaudeamus, Hradec Králové, 2009, s. 114. ISBN 978-80-7041-635-8.

[ap-16] **Milková E., Haviger J., Rubáček, F., Voborník, P.**: *Algoritmy - základní konstrukce v příkladech a jejich vizualizace*. Gaudeamus, Hradec Králové, 2010, s. 100. ISBN 978-80-7435-064-1.

[ap-17] **Voborník, P.**: *Výuka programování s pomocí video-tutoriálů*. In: Sborník příspěvků z konference a soutěže eLearning 2010, 9.–10. 11. 2010, Gaudeamus, Hradec Králové, 2010. ISBN 978-80-7435-067-2.

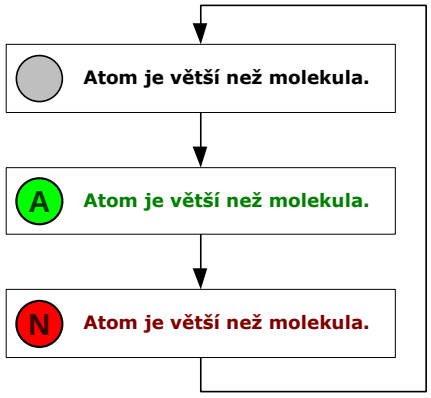
[ap-18] **Strnadová, V., Voborník, P.**: *Electronic communication with the students of the first year FIM UHK*. In: 10th WSEAS International Conference on Education and Educational Technology (EDU '11), Penang, Malaysia, 2011, str. 15-21. ISBN 987-1-61804-040-4.

Přílohy

Příloha 1 – Porovnání zápisu některých funkcí animací v XAML, SVG a QML

	XAML	SVG	QML
Opakování	`RepeatBehavior` určuje počet nebo `Forever` znamená navždy.	`RepeatCount` určuje počet nebo `indefinite` znamená navždy.	`Repeat` určuje počet nebo `f` znamená navždy.
Start	Animace (`Storyboard`) se spouští z kódu na pozadí stránky nebo v její události (`EventTrigger`).	Animace je zahájena po načtení stránky, nebo navazuje na jinou událost.	Animace je zahájena po načtení stránky. Animace položek může být zastavena při tažení a po umístění do cíle.
Časování	`Begin` může posunout začátek.	`Begin` může posunout začátek, nebo i vyčkat nějaké události (konce jiné animace).	`Begin` může posunout začátek.
Reverze	`AutoReverse` může po skončení animace obrátit její chod.	Není podporována, musí ji vyřešit jiná animace s opačnými hodnotami.	`Reverse` může po skončení animace obrátit její chod.
Centrum	Souřadnice středu animace jsou definovány u transformace (`CenterX`, `CenterY`). Defaultně je to levý roh objektu.	Defaultně je to levý roh objektu. Střed lze posunout zápornými souřadnicemi `x` a `y`, zpětný posun zajistí kontejner, transformací `translate`.	Střed lze definovat přes `centerX`, `centerY`, defaultně je jím střed objektu.
Určení cíle	Umožňuje animovat cokoli na stránce definováním objektu (`TargetName`) a jeho vlastnosti (`TargetProperty`).	Animován je prvek určený nadřazeným elementem, `attributeName`, `attributeType`, type určují, jaká jeho vlastnost se animuje.	Animován je prvek nadřazeného elementu, měněnou vlastnost určují příslušné atributy specifické pro daný animační element.
Animace transformací	Nerozlišuje animaci transformací a jiných objektů, odkazuje se na ni názvem.	Speciální element `animateTransform`.	Název elementu určuje animovanou transformaci, která je v případě potřeby přidána.
Nelineární pohyby	Jejich druh a typ určuje element `EasingFunction`, název jeho podelementu a jeho atribut `EasingMode`.	Durh křivky je určen v `calcMode`, `keySplines` body definuje její zakřivení.	Atributy `func` a `mode` jsou ekvivalentem ve verzi XAML.

Příloha 2 – Příklad stavů přepínače u dichotomické podotázky typu ANO/NE (určení, zda výrok platí nebo neplatí)



Příloha 3 – Porovnání zápisu animace obdélníku v XAML, SVG a QML

XAML 952 znaků

```
<Rectangle Stroke="Black" StrokeThickness="1" Width="200" Height="100" Fill="Lime"
    Canvas.Left="200" Canvas.Top="150">
  <Rectangle.RenderTransform>
    <TransformGroup>
      <ScaleTransform x:Name="traAllScale" CenterX="200" CenterY="65" />
    </TransformGroup>
  </Rectangle.RenderTransform>
  <Rectangle.Resources>
    <Storyboard x:Name="stbAll">
      <DoubleAnimation From="0.95" To="1.05" Duration="00:00:01"
          AutoReverse="True" RepeatBehavior="Forever"
          Storyboard.TargetName="traAllScale" Storyboard.TargetProperty="ScaleX">
        <DoubleAnimation.EasingFunction>
          <SineEase EasingMode="EaseInOut" />
        </DoubleAnimation.EasingFunction>
      </DoubleAnimation>
      <DoubleAnimation From="1.05" To="0.95" Duration="00:00:01"
          AutoReverse="True" RepeatBehavior="Forever"
          Storyboard.TargetName="traAllScale" Storyboard.TargetProperty="ScaleY">
        <DoubleAnimation.EasingFunction>
          <SineEase EasingMode="EaseInOut" />
        </DoubleAnimation.EasingFunction>
      </DoubleAnimation>
    </Storyboard>
  </Rectangle.Resources>
</Rectangle>
```

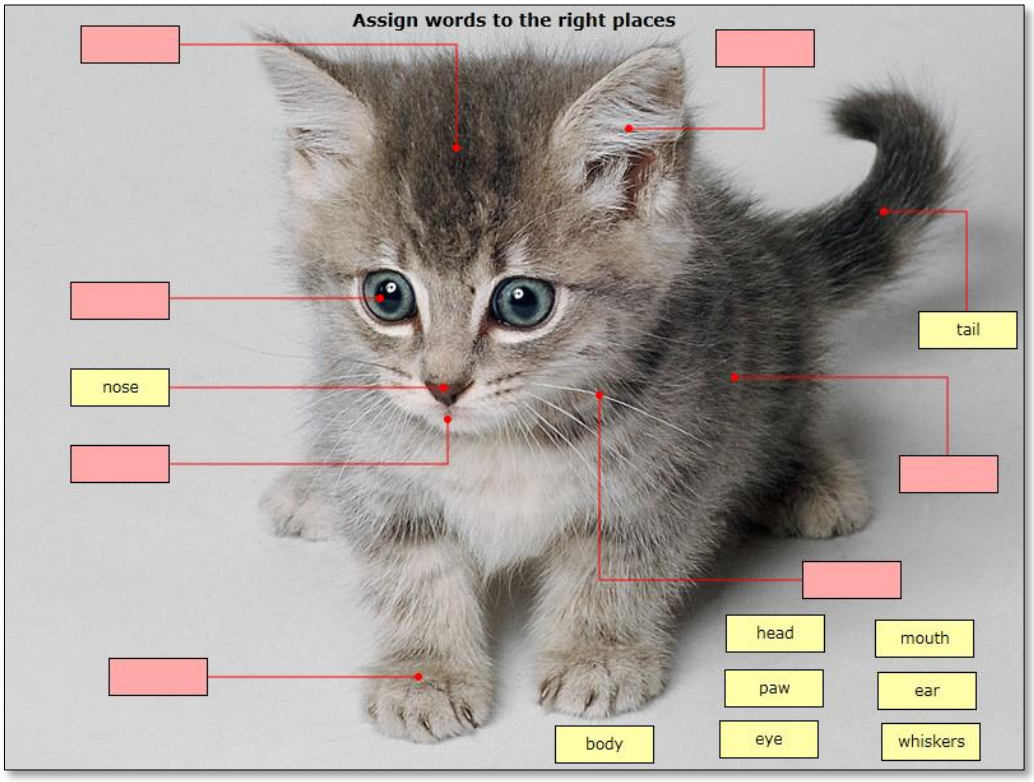
SVG 574 znaků

```
<g transform="translate(300,200)">
  <rect x="-100" y="-50" width="200" height="100"
      style="stroke: #000000; fill: #00FF00;">
    <animateTransform id="t1" attributeName="transform" attributeType="XML"
        type="scale" from="0.95 1.05" to="1.05 0.95" additive="replace"
        begin="0;t2.end" dur="1s" fill="freeze" calcMode="spline"
        keySplines="0.5 0 0.5 1" />
    <animateTransform id="t2" attributeName="transform" attributeType="XML"
        type="scale" from="1.05 0.95" to="0.95 1.05" additive="replace"
        begin="t1.end" dur="1s" fill="freeze" calcMode="spline"
        keySplines="0.5 0 0.5 1"  />
  </rect>
</g>
```

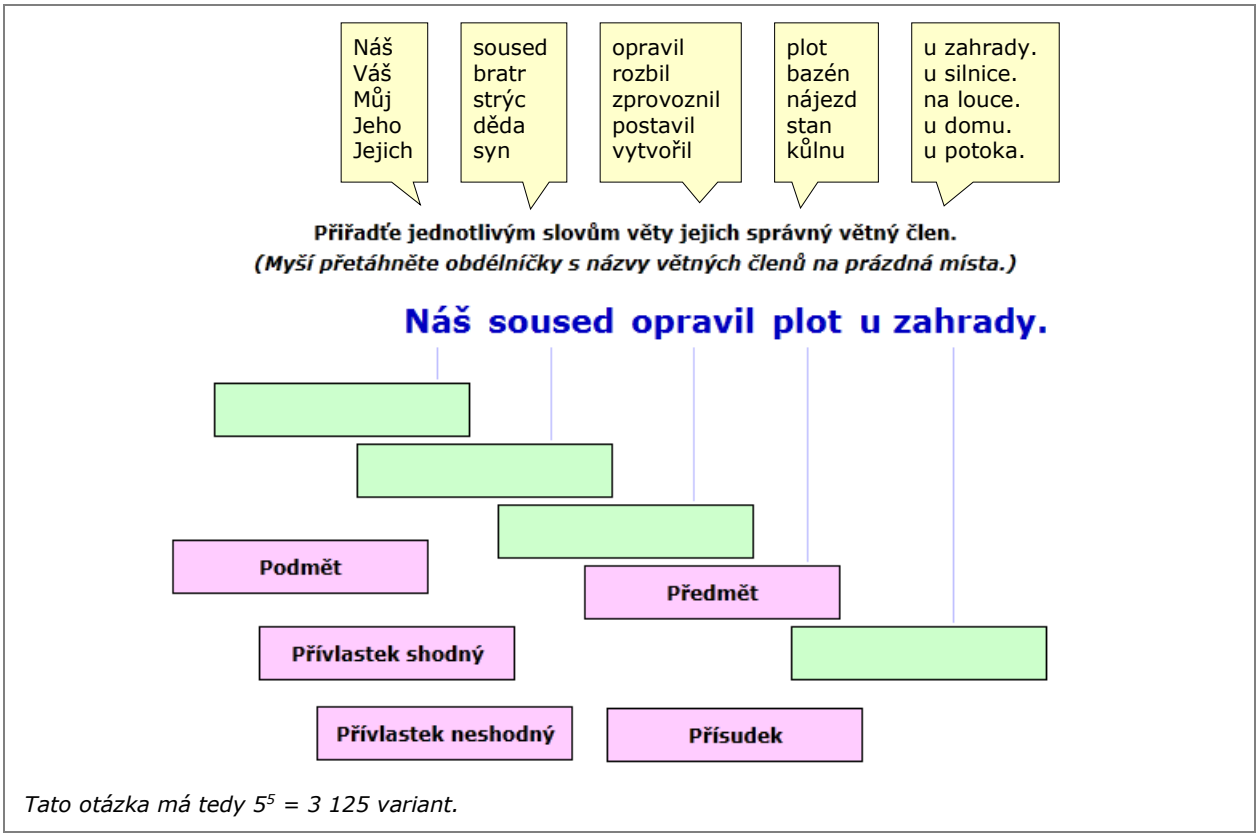
QML 259 znaků

```
<rectangle x="200" y="150" width="200" height="100"
    background="00FF00" color="000000" thickness="1">
  <anims reverse="1" repeat="forever">
    <scale duration="1" fromScaleX="0.95" toScaleX="1.05"
        fromScaleY="1.05" toScaleY="0.95" func="sine" />
  </anims>
</rectangle>
```

Příloha 4 – Ukázka otázky přiřazování anglických slovíček na správná místa v obrázku

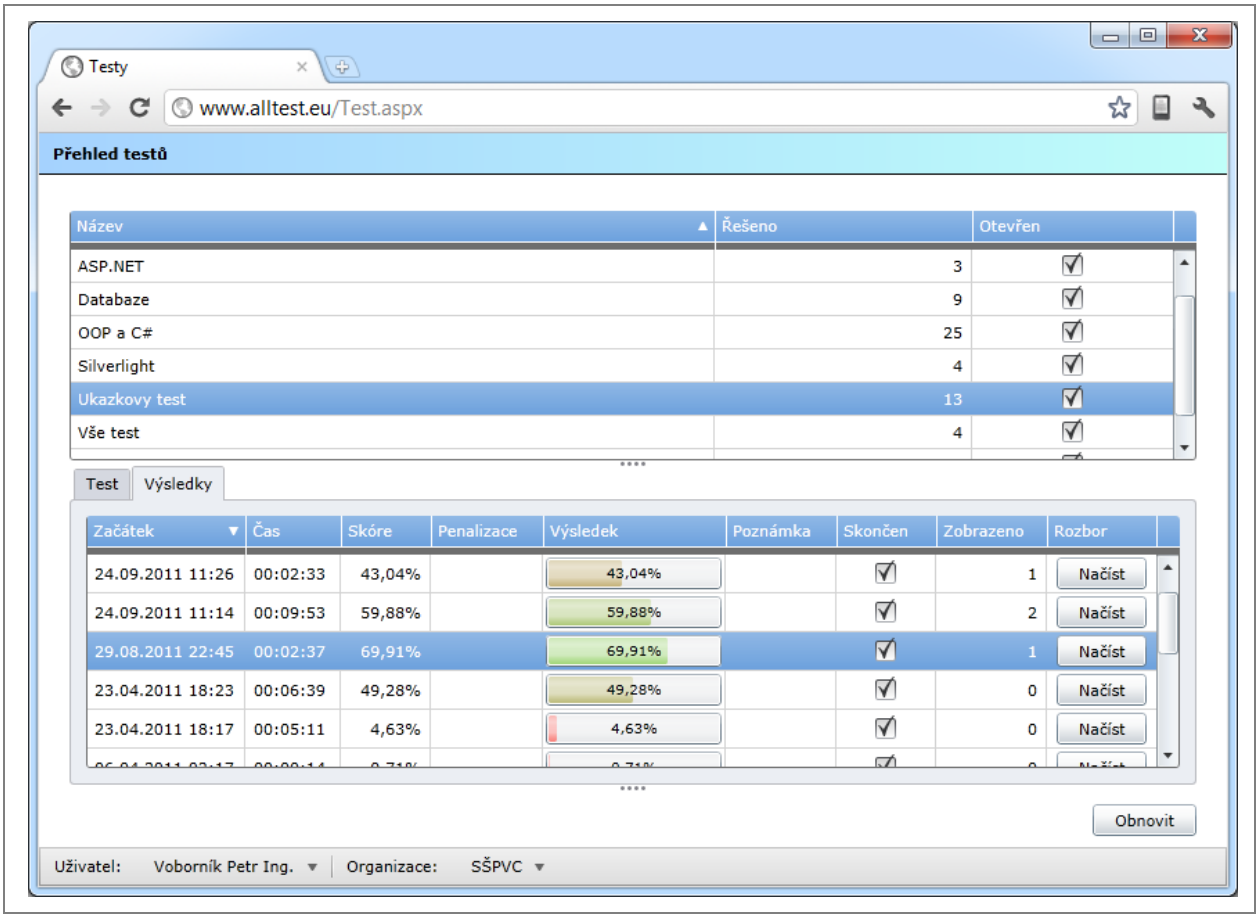


Příloha 5 – Ukázka otázky „Větný rozbor" s naznačením změn jednotlivých slov věty



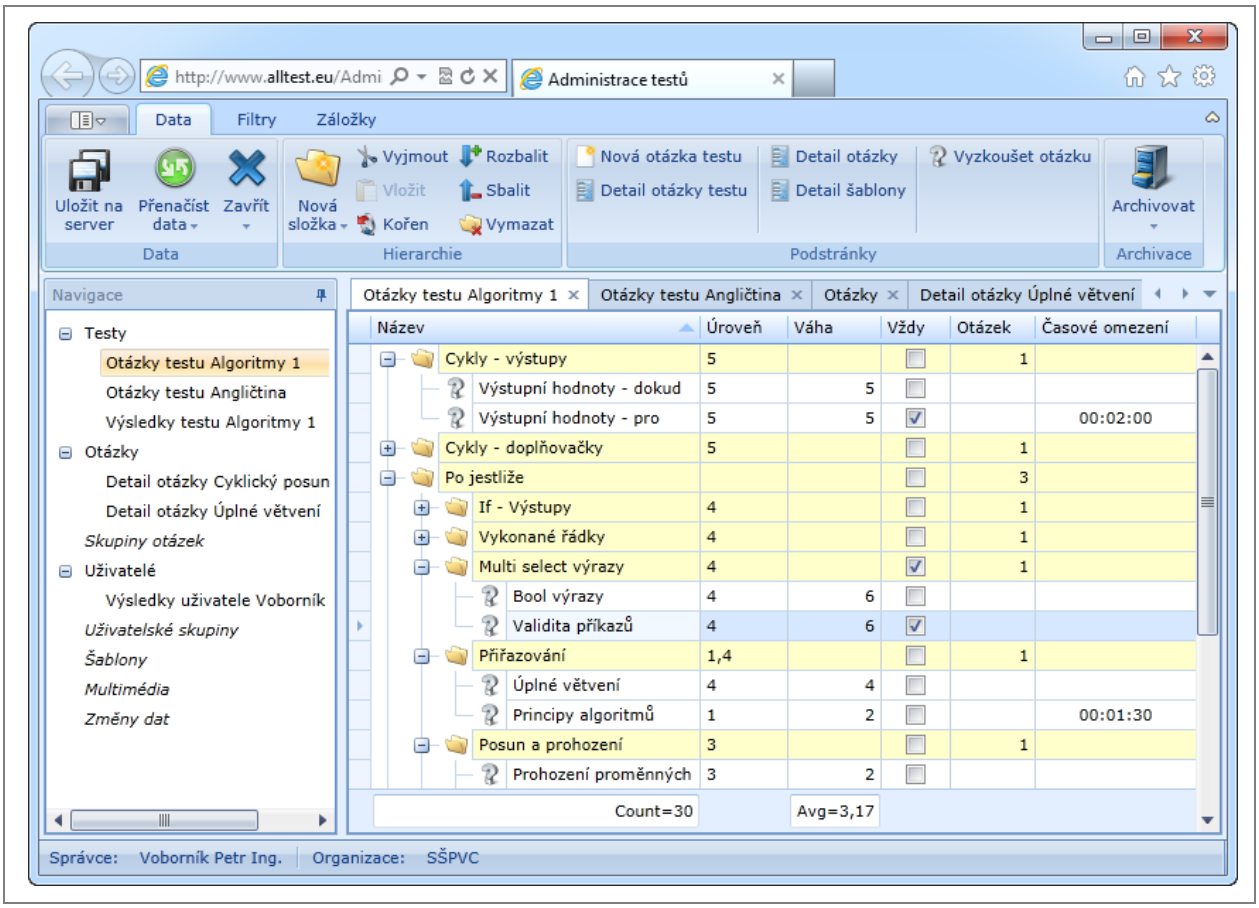
Tato otázka má tedy 5^5 = 3 125 variant.

Příloha 6 – Ukázka přehledu testů v testovacím rozhraní



Příloha 7 – Ukázka GUI testovacího rozhraní během testu

Příloha 8 – Ukázka GUI administračního rozhraní



Příloha 9 – Podobnost slova "logaritmus" s jeho různými modifikacemi dle různých srovnávacích algoritmů

Různé modifikace slova "logaritmus"			Procentní shoda dle jednotlivých algoritmů																	
Typ	Slovo	Popis	BlockDistance	ChapmanLengthDeviation	ChapmanMeanLength	CosineSimilarity	DiceSimilarity	EuclideanDistance	JaccardSimilarity	Jaro	JaroWinkler	Levenstein	MatchingCoefficient	MongeElkan	NeedlemanWunch	OverlapCoefficient	QGramsDistance	SmithWaterman	SmithWatermanGotoh	S.W.G.WindowedAffine
	logaritmus	srovnání téhož	100	100	15	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
jiné písmeno	wogaritmus	první	0	100	15	0	0	0	0	93	93	90	0	90	95	0	75	90	90	90
	logapitmus	prostřední	0	100	15	0	0	0	0	93	96	90	0	84	95	0	75	80	84	84
	logaritmub	poslední	0	100	15	0	0	0	0	93	96	90	0	90	95	0	75	90	90	90
	lobaritpus	dvě	0	100	15	0	0	0	0	87	89	80	0	68	90	0	50	60	68	68
	bcdefhjknp	vše	0	100	15	0	0	0	0	0	0	0	0	6	50	0	0	0	6	6
podobné znaky	1ogaritmus	1 místo l	0	100	15	0	0	0	0	93	93	90	0	90	95	0	75	90	90	90
	l0garitmus	0 místo o	0	100	15	0	0	0	0	93	94	90	0	84	95	0	75	80	84	84
	logaritnnus	dvě n místo m	0	91	16	0	0	0	0	91	94	82	0	86	86	0	72	75	86	86
	Iogaritmus	velké I místo l (L)	0	100	15	0	0	0	0	93	93	90	0	90	95	0	75	90	90	90
	logarifmus	f místo t	0	100	15	0	0	0	0	93	96	90	0	84	95	0	75	80	84	84
velikost písmene	Logaritmus	první	0	100	15	0	0	0	0	93	93	90	0	96	95	0	75	90	96	96
	logaRitmus	prostřední	0	100	15	0	0	0	0	93	96	90	0	96	95	0	75	80	96	96
	logaritmuS	poslední	0	100	15	0	0	0	0	93	96	90	0	90	95	0	75	90	90	90
	loGaritMus	dvě	0	100	15	0	0	0	0	87	89	80	0	92	90	0	50	60	92	92
	LOGARITMUS	vše	0	100	15	0	0	0	0	0	0	0	0	54	50	0	0	0	54	54
diakritika	logarítmus	í	0	100	15	0	0	0	0	93	96	90	0	84	95	0	75	80	84	84
	lógaritmus	ó	0	100	15	0	0	0	0	93	94	90	0	84	95	0	75	80	84	84
	logaritmuš	š	0	100	15	0	0	0	0	93	96	90	0	90	95	0	75	90	90	90
	logaritmůs	ů	0	100	15	0	0	0	0	93	96	90	0	84	95	0	75	80	84	84
	lógarítmus	ói	0	100	15	0	0	0	0	87	88	80	0	68	90	0	50	60	68	68
	logáritmuš	áš	0	100	15	0	0	0	0	87	91	80	0	74	90	0	50	70	74	74
	lógáritmůš	óůáš	0	100	15	0	0	0	0	73	76	60	0	48	80	0	25	40	48	48
stejně znějící znaky	lokaritmus	g na k	0	100	15	0	0	0	0	93	95	90	0	84	95	0	75	80	84	84
	logarytmus	i na y	0	100	15	0	0	0	0	93	96	90	0	84	95	0	75	80	84	84
	logaritmuz	s na z	0	100	15	0	0	0	0	93	96	90	0	90	95	0	75	90	90	90
prohození písmen	logairtmus	o 1: ri - ir	0	100	15	0	0	0	0	97	98	80	0	70	90	0	67	80	70	70
	logiratmus	o 2: ari - ira	0	100	15	0	0	0	0	97	98	80	0	92	90	0	58	60	92	92
	loiargtmus	o 3: gari - iarg	0	100	15	0	0	0	0	97	97	80	0	68	90	0	50	60	68	68
	lotarigmus	o 4: garit - tarig	0	100	15	0	0	0	0	97	97	80	0	68	90	0	50	60	68	68
	sogaritmul	první a poslední	0	100	15	0	0	0	0	87	87	80	0	80	90	0	50	80	80	80
	sumtiragol	pozpátku	0	100	15	0	0	0	0	57	57	0	0	34	50	0	0	10	34	34
	olgaritmsu	dvě prohození o 1	0	100	15	0	0	0	0	93	93	60	0	60	80	0	33	70	60	60
	slogaritmu	cyklický posun	0	100	15	0	0	0	0	93	93	80	0	90	85	0	58	90	90	90
vynechání písmene	ogaritmus	první	0	90	14	0	0	0	0	97	97	90	0	100	95	0	78	100	100	100
	logaritmu	poslední	0	90	14	0	0	0	0	97	98	90	0	100	90	0	78	100	100	100
	logaitmus	prostřední	0	90	14	0	0	0	0	97	98	90	0	89	90	0	78	94	89	89
	ogaritmu	první a poslední	0	80	14	0	0	0	0	93	93	80	0	100	85	0	55	100	100	100
	logatmus	dvě uprostřed	0	80	14	0	0	0	0	93	96	80	0	85	80	0	73	88	85	85
	garitmus	dvě první	0	80	14	0	0	0	0	93	93	80	0	100	90	0	73	100	100	100
	ogaitmu	tři napřeskáčku	0	70	13	0	0	0	0	90	90	70	0	86	75	0	29	93	86	86
	log	7 od konce	0	30	10	0	0	0	0	77	84	30	0	100	50	0	35	100	100	100
	l	první zbylo	0	10	9	0	0	0	0	70	73	10	0	100	50	0	13	100	100	100
	r	prostřední zbylo	0	10	9	0	0	0	0	0	0	10	0	100	50	0	0	100	100	100
přidání písmene	llogaritmus	l (L) na začátek	0	91	16	0	0	0	0	97	97	91	0	100	95	0	88	100	100	100
	logaritmuse	e nakonec	0	91	16	0	0	0	0	97	98	91	0	100	91	0	80	100	100	100
	logaritamus	a za t	0	91	16	0	0	0	0	97	98	91	0	90	91	0	80	95	90	90
	xlogaritmusx	x z obou stran	0	83	16	0	0	0	0	94	94	83	0	100	88	0	62	100	100	100
	logaritmus	mezera na začátku	100	91	16	100	100	100	100	97	97	91	100	100	95	100	80	100	100	100
	logaritmus	mezera na konci	67	91	16	71	67	42	50	97	98	91	50	100	91	100	80	100	100	100
	logar itmus	mezera uprostřed	0	91	16	0	0	0	0	97	98	91	0	100	91	0	80	95	90	90
	ljojgjajrjijtjmjujs	j mezi písmena	0	53	21	0	0	0	0	0	10	53	0	30	53	0	12	55	30	30
	l o g a r i t m u s	mezery mezi písmena	0	53	21	0	0	0	0	0	10	53	0	16	53	0	12	55	16	16
		průměr	**5**	**90**	**15**	**5**	**5**	**5**	**5**	**83**	**85**	**75**	**5**	**82**	**85**	**6**	**59**	**79**	**82**	**82**

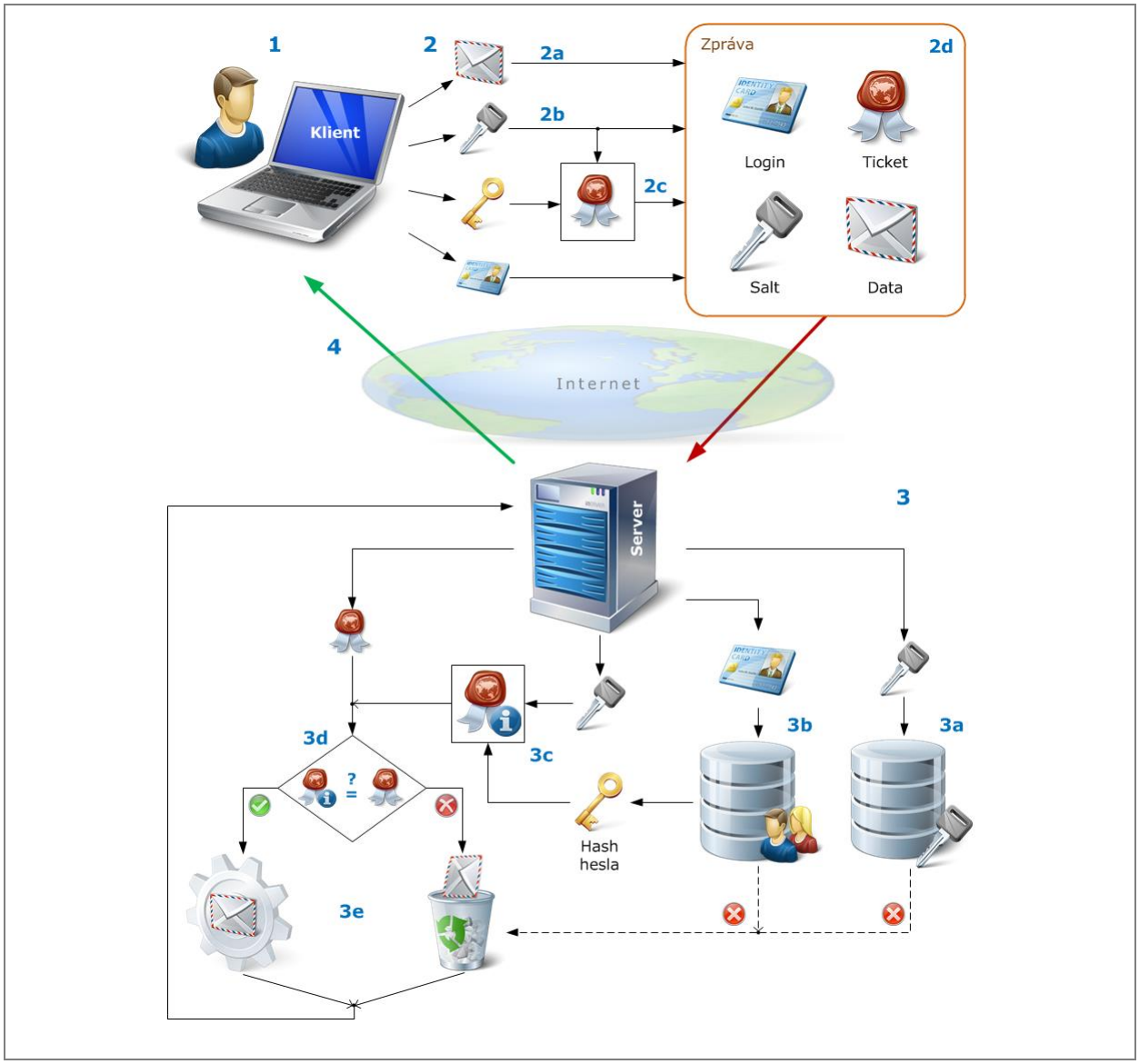
Příloha 10 – Střední doby (v μs) výpočtu podobnosti slov různých srovnávacích algoritmů

Modifikace slova		Střední doba výpočtu shody jednotlivých algoritmů [μS]																	
Typ	Slovo	BlockDistance	ChapmanLengthDeviation	ChapmanMeanLength	CosineSimilarity	DiceSimilarity	EuclideanDistance	JaccardSimilarity	Jaro	JaroWinkler	Levenstein	MatchingCoefficient	MongeElkan	NeedlemanWunch	OverlapCoefficient	QGramsDistance	SmithWaterman	SmithWatermanGotoh	S.W.G.WindowedAffine
stejné	**logaritmus**	11	7	7	12	11	11	12	12	12	17	10	163	16	12	40	14	118	122
jiné písmen	wogaritmus	9	5	5	10	9	9	9	9	10	14	9	123	12	10	34	14	120	122
	logapitmus	9	6	6	10	8	8	11	10	10	14	10	123	13	10	32	14	123	124
	logaritmub	9	7	7	10	9	8	10	10	10	13	9	123	13	8	34	15	122	121
	lobaritpus	9	7	5	9	9	10	10	10	10	14	9	124	13	10	35	14	122	122
	bcdefhjknp	11	8	8	8	10	9	8	9	7	13	7	128	12	10	40	13	126	129
podobné znaky	1ogaritmus	10	6	6	10	8	10	9	9	9	13	9	122	12	11	33	14	124	122
	l0garitmus	8	6	6	10	9	9	8	10	10	13	9	123	14	9	32	14	122	123
	logaritnnus	9	7	5	8	9	9	10	9	10	14	9	131	13	10	34	15	131	130
	Iogaritmus	9	5	5	10	9	7	9	9	10	14	9	120	13	10	32	14	130	129
	logarifmus	10	6	8	11	10	8	10	10	10	14	10	129	14	10	34	15	128	129
velikost písmene	Logaritmus	12	8	7	10	10	10	9	10	10	14	10	129	14	10	35	15	129	129
	logaRitmus	10	7	6	11	10	8	10	10	10	14	10	130	13	10	35	15	128	126
	logaritmuS	10	7	8	10	10	10	10	10	11	14	10	132	14	8	34	15	129	130
	loGaritMus	10	7	8	10	10	10	11	10	10	13	10	129	13	10	37	15	128	127
	LOGARITMUS	10	6	6	10	9	10	10	9	8	14	10	138	13	10	43	14	138	137
diakritika	logarítmus	10	6	8	9	10	10	10	10	11	15	10	131	14	10	34	16	129	129
	lógaritmus	10	7	8	9	11	8	10	10	9	14	10	130	14	10	35	17	129	132
	logaritmuš	10	8	8	10	10	10	9	10	10	14	10	129	14	10	35	16	127	127
	logaritmůs	12	7	7	11	10	8	9	10	10	14	10	130	14	10	34	15	128	128
	lógarítmus	10	6	8	11	8	10	10	11	11	15	8	129	13	10	37	15	128	128
	logáritmuš	10	6	6	10	10	10	10	10	10	14	10	131	13	10	37	15	131	130
	lógáritmůš	10	6	6	9	10	10	11	11	11	14	8	132	14	10	40	15	130	131
stejně znějící znaky	lokaritmus	10	7	6	10	9	10	10	10	10	14	10	129	13	10	35	15	126	127
	logarytmus	10	8	7	10	10	9	10	10	11	14	8	130	14	10	35	16	129	129
	logaritmuz	10	8	6	10	9	10	10	9	10	14	10	131	14	10	34	16	129	127
prohození písmen	logairtmus	10	8	5	9	10	10	10	10	10	14	9	128	14	10	36	16	127	127
	logiratmus	9	7	8	11	10	8	9	10	10	14	10	128	14	10	37	15	127	126
	loiargtmus	10	6	7	10	10	10	10	10	10	14	8	129	14	10	37	14	128	128
	lotarigmus	10	7	6	10	10	10	10	10	10	14	9	127	13	10	39	17	126	126
	sogaritmul	9	6	6	10	10	10	11	10	11	14	10	129	12	9	39	15	127	126
	sumtiragol	10	6	8	8	10	10	9	10	10	14	10	129	13	10	43	15	126	126
	olgaritmsu	10	6	6	10	10	10	9	8	10	14	8	129	11	10	39	15	126	126
	slogaritmu	10	6	6	9	10	10	10	10	10	14	10	129	13	8	37	15	128	129
vynechání písmene	ogaritmus	8	6	8	10	9	8	8	9	10	13	10	117	11	10	33	14	117	115
	logaritmu	8	6	6	10	8	8	10	10	10	14	9	116	13	10	33	16	115	113
	logaitmus	10	8	7	12	10	10	9	10	10	14	10	116	11	10	33	15	116	116
	ogaritmu	8	6	6	8	9	10	10	10	10	13	9	103	12	10	34	14	101	102
	logatmus	9	6	6	10	9	10	10	10	10	13	9	103	12	10	32	14	102	102
	garitmus	10	7	6	10	10	10	10	10	10	13	8	103	13	11	32	14	102	101
	ogaitmu	10	6	6	10	9	8	8	8	8	11	10	90	11	10	34	14	90	90
	log	10	6	6	10	10	8	8	7	9	10	10	42	10	10	27	11	41	42
	l	10	6	6	10	8	8	8	7	7	7	9	21	9	10	25	9	20	20
	r	9	6	6	10	10	10	10	6	6	7	8	21	9	10	26	9	21	20
přidání písmene	llogaritmus	10	6	8	10	10	10	10	11	11	15	8	144	14	10	35	16	143	140
	logaritmuse	10	6	6	10	10	10	10	10	10	14	10	142	14	10	36	16	142	140
	logaritamus	8	6	6	9	10	10	10	10	11	15	10	141	14	9	36	14	141	139
	xlogaritmusx	8	6	6	10	10	10	10	10	10	15	9	164	14	10	40	16	158	158
	logaritmus	10	6	6	9	10	9	10	10	11	15	10	129	14	10	36	16	144	144
	logaritmus	10	8	6	11	11	11	9	10	10	14	10	130	14	11	36	16	143	142
	logar itmus	10	6	6	11	9	11	11	10	11	14	10	124	14	11	36	16	143	142
	ljojgjajrjijtjmjujs	10	6	6	11	9	10	8	11	11	19	10	269	17	9	64	21	267	255
	l o g a r i t m u s	16	6	7	17	17	16	18	11	11	18	14	136	17	17	64	21	254	252
průměr		**10**	**6**	**6**	**10**	**10**	**9**	**10**	**10**	**10**	**14**	**9**	**125**	**13**	**10**	**36**	**15**	**126**	**126**

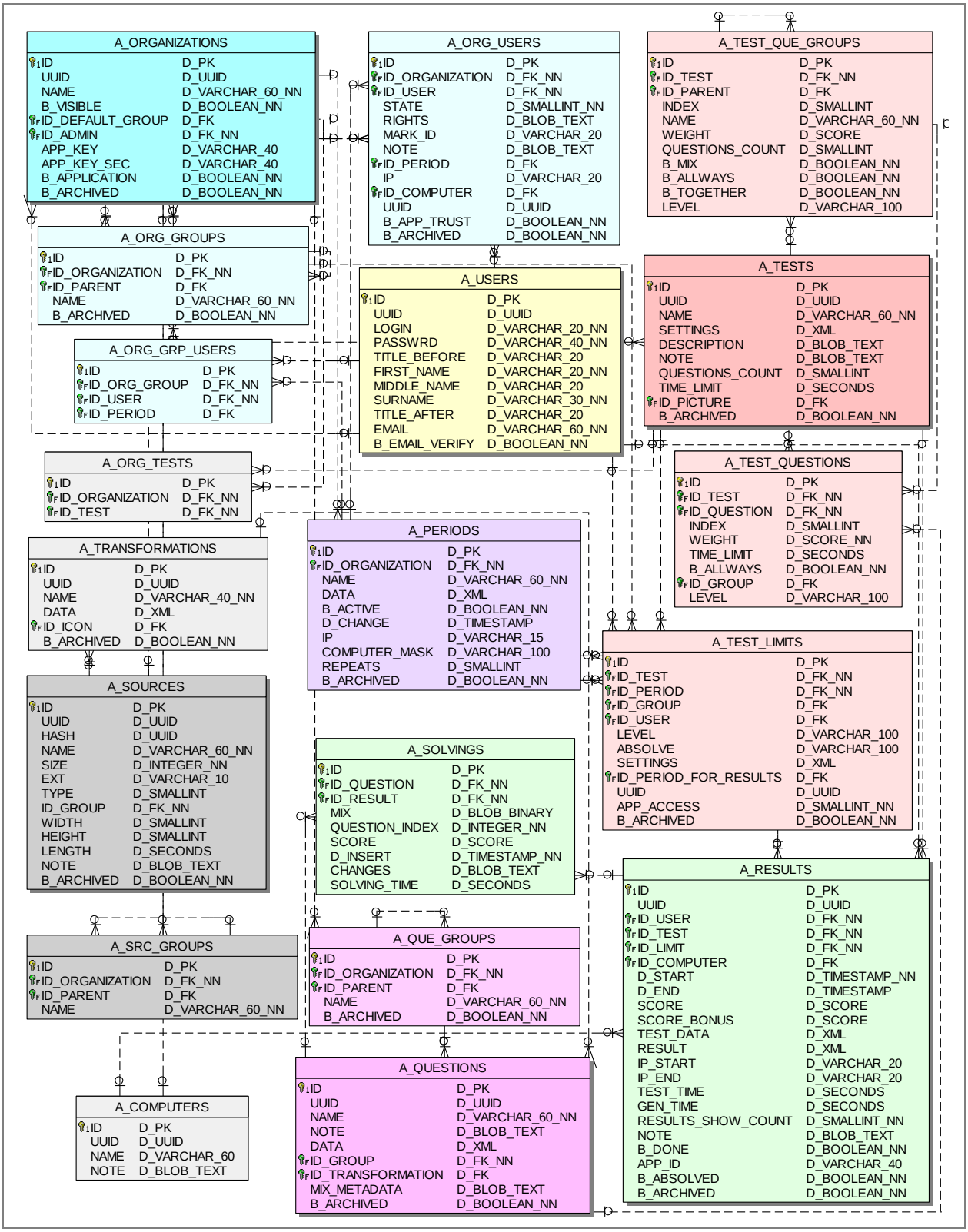
Příloha 11 – Hodnoty P (pravděpodobnost výběru otázek) pro stupnici kombinací d (t) a s (v) při c = 1, p = 4 a n = 0,2

		s	1	0,95	0,9	0,85	0,8	0,75	0,7	0,65	0,6	0,55	0,5	0,45	0,4	0,35	0,3	0,25	0,2	0,15	0,1	0,05	0
d	**t**	**v**	1	0,96	0,92	0,88	0,84	0,8	0,76	0,72	0,68	0,64	0,6	0,56	0,52	0,48	0,44	0,4	0,36	0,32	0,28	0,24	0,2
0	1,000	0,000	0,040	0,080	0,120	0,160	0,200	0,240	0,280	0,320	0,360	0,400	0,440	0,480	0,520	0,560	0,600	0,640	0,680	0,720	0,760	0,800
0,2	0,972	0,028	0,067	0,106	0,144	0,183	0,222	0,261	0,300	0,339	0,378	0,417	0,456	0,494	0,533	0,572	0,611	0,650	0,689	0,728	0,767	0,806
0,4	0,946	0,054	0,092	0,130	0,168	0,205	0,243	0,281	0,319	0,357	0,395	0,432	0,470	0,508	0,546	0,584	0,622	0,659	0,697	0,735	0,773	0,811
0,6	0,921	0,079	0,116	0,153	0,189	0,226	0,263	0,300	0,337	0,374	0,411	0,447	0,484	0,521	0,558	0,595	0,632	0,668	0,705	0,742	0,779	0,816
0,8	0,897	0,103	0,138	0,174	0,210	0,246	0,282	0,318	0,354	0,390	0,426	0,462	0,497	0,533	0,569	0,605	0,641	0,677	0,713	0,749	0,785	0,821
1	0,875	0,125	0,160	0,195	0,230	0,265	0,300	0,335	0,370	0,405	0,440	0,475	0,510	0,545	0,580	0,615	0,650	0,685	0,720	0,755	0,790	0,825
1,2	0,854	0,146	0,180	0,215	0,249	0,283	0,317	0,351	0,385	0,420	0,454	0,488	0,522	0,556	0,590	0,624	0,659	0,693	0,727	0,761	0,795	0,829
1,4	0,833	0,167	0,200	0,233	0,267	0,300	0,333	0,367	0,400	0,433	0,467	0,500	0,533	0,567	0,600	0,633	0,667	0,700	0,733	0,767	0,800	0,833
1,6	0,814	0,186	0,219	0,251	0,284	0,316	0,349	0,381	0,414	0,447	0,479	0,512	0,544	0,577	0,609	0,642	0,674	0,707	0,740	0,772	0,805	0,837
1,8	0,795	0,205	0,236	0,268	0,300	0,332	0,364	0,395	0,427	0,459	0,491	0,523	0,555	0,586	0,618	0,650	0,682	0,714	0,745	0,777	0,809	0,841
2	0,778	0,222	0,253	0,284	0,316	0,347	0,378	0,409	0,440	0,471	0,502	0,533	0,564	0,596	0,627	0,658	0,689	0,720	0,751	0,782	0,813	0,844
2,2	0,761	0,239	0,270	0,300	0,330	0,361	0,391	0,422	0,452	0,483	0,513	0,543	0,574	0,604	0,635	0,665	0,696	0,726	0,757	0,787	0,817	0,848
2,4	0,745	0,255	0,285	0,315	0,345	0,374	0,404	0,434	0,464	0,494	0,523	0,553	0,583	0,613	0,643	0,672	0,702	0,732	0,762	0,791	0,821	0,851
2,6	0,729	0,271	0,300	0,329	0,358	0,388	0,417	0,446	0,475	0,504	0,533	0,563	0,592	0,621	0,650	0,679	0,708	0,738	0,767	0,796	0,825	0,854
2,8	0,714	0,286	0,314	0,343	0,371	0,400	0,429	0,457	0,486	0,514	0,543	0,571	0,600	0,629	0,657	0,686	0,714	0,743	0,771	0,800	0,829	0,857
3	0,700	0,300	0,328	0,356	0,384	0,412	0,440	0,468	0,496	0,524	0,552	0,580	0,608	0,636	0,664	0,692	0,720	0,748	0,776	0,804	0,832	0,860
3,2	0,686	0,314	0,341	0,369	0,396	0,424	0,451	0,478	0,506	0,533	0,561	0,588	0,616	0,643	0,671	0,698	0,725	0,753	0,780	0,808	0,835	0,863
3,4	0,673	0,327	0,354	0,381	0,408	0,435	0,462	0,488	0,515	0,542	0,569	0,596	0,623	0,650	0,677	0,704	0,731	0,758	0,785	0,812	0,838	0,865
3,6	0,660	0,340	0,366	0,392	0,419	0,445	0,472	0,498	0,525	0,551	0,577	0,604	0,630	0,657	0,683	0,709	0,736	0,762	0,789	0,815	0,842	0,868
3,8	0,648	0,352	0,378	0,404	0,430	0,456	0,481	0,507	0,533	0,559	0,585	0,611	0,637	0,663	0,689	0,715	0,741	0,767	0,793	0,819	0,844	0,870
4	0,636	0,364	0,389	0,415	0,440	0,465	0,491	0,516	0,542	0,567	0,593	0,618	0,644	0,669	0,695	0,720	0,745	0,771	0,796	0,822	0,847	0,873
4,2	0,625	0,375	0,400	0,425	0,450	0,475	0,500	0,525	0,550	0,575	0,600	0,625	0,650	0,675	0,700	0,725	0,750	0,775	0,800	0,825	0,850	0,875
4,4	0,614	0,386	0,411	0,435	0,460	0,484	0,509	0,533	0,558	0,582	0,607	0,632	0,656	0,681	0,705	0,730	0,754	0,779	0,804	0,828	0,853	0,877
4,6	0,603	0,397	0,421	0,445	0,469	0,493	0,517	0,541	0,566	0,590	0,614	0,638	0,662	0,686	0,710	0,734	0,759	0,783	0,807	0,831	0,855	0,879
4,8	0,593	0,407	0,431	0,454	0,478	0,502	0,525	0,549	0,573	0,597	0,620	0,644	0,668	0,692	0,715	0,739	0,763	0,786	0,810	0,834	0,858	0,881
5	0,583	0,417	0,440	0,463	0,487	0,510	0,533	0,557	0,580	0,603	0,627	0,650	0,673	0,697	0,720	0,743	0,767	0,790	0,813	0,837	0,860	0,883
5,2	0,574	0,426	0,449	0,472	0,495	0,518	0,541	0,564	0,587	0,610	0,633	0,656	0,679	0,702	0,725	0,748	0,770	0,793	0,816	0,839	0,862	0,885
5,4	0,565	0,435	0,458	0,481	0,503	0,526	0,548	0,571	0,594	0,616	0,639	0,661	0,684	0,706	0,729	0,752	0,774	0,797	0,819	0,842	0,865	0,887
5,6	0,556	0,444	0,467	0,489	0,511	0,533	0,556	0,578	0,600	0,622	0,644	0,667	0,689	0,711	0,733	0,756	0,778	0,800	0,822	0,844	0,867	0,889
5,8	0,547	0,453	0,475	0,497	0,519	0,541	0,563	0,584	0,606	0,628	0,650	0,672	0,694	0,716	0,738	0,759	0,781	0,803	0,825	0,847	0,869	0,891
6	0,538	0,462	0,483	0,505	0,526	0,548	0,569	0,591	0,612	0,634	0,655	0,677	0,698	0,720	0,742	0,763	0,785	0,806	0,828	0,849	0,871	0,892
6,2	0,530	0,470	0,491	0,512	0,533	0,555	0,576	0,597	0,618	0,639	0,661	0,682	0,703	0,724	0,745	0,767	0,788	0,809	0,830	0,852	0,873	0,894
6,4	0,522	0,478	0,499	0,519	0,540	0,561	0,582	0,603	0,624	0,645	0,666	0,687	0,707	0,728	0,749	0,770	0,791	0,812	0,833	0,854	0,875	0,896
6,6	0,515	0,485	0,506	0,526	0,547	0,568	0,588	0,609	0,629	0,650	0,671	0,691	0,712	0,732	0,753	0,774	0,794	0,815	0,835	0,856	0,876	0,897
6,8	0,507	0,493	0,513	0,533	0,554	0,574	0,594	0,614	0,635	0,655	0,675	0,696	0,716	0,736	0,757	0,777	0,797	0,817	0,838	0,858	0,878	0,899
7	0,500	0,500	0,520	0,540	0,560	0,580	0,600	0,620	0,640	0,660	0,680	0,700	0,720	0,740	0,760	0,780	0,800	0,820	0,840	0,860	0,880	0,900
7,2	0,493	0,507	0,527	0,546	0,566	0,586	0,606	0,625	0,645	0,665	0,685	0,704	0,724	0,744	0,763	0,783	0,803	0,823	0,842	0,862	0,882	0,901
7,4	0,486	0,514	0,533	0,553	0,572	0,592	0,611	0,631	0,650	0,669	0,689	0,708	0,728	0,747	0,767	0,786	0,806	0,825	0,844	0,864	0,883	0,903
7,6	0,479	0,521	0,540	0,559	0,578	0,597	0,616	0,636	0,655	0,674	0,693	0,712	0,732	0,751	0,770	0,789	0,808	0,827	0,847	0,866	0,885	0,904
7,8	0,473	0,527	0,546	0,565	0,584	0,603	0,622	0,641	0,659	0,678	0,697	0,716	0,735	0,754	0,773	0,792	0,811	0,830	0,849	0,868	0,886	0,905
8	0,467	0,533	0,552	0,571	0,589	0,608	0,627	0,645	0,664	0,683	0,701	0,720	0,739	0,757	0,776	0,795	0,813	0,832	0,851	0,869	0,888	0,907
8,2	0,461	0,539	0,558	0,576	0,595	0,613	0,632	0,650	0,668	0,687	0,705	0,724	0,742	0,761	0,779	0,797	0,816	0,834	0,853	0,871	0,889	0,908
8,4	0,455	0,545	0,564	0,582	0,600	0,618	0,636	0,655	0,673	0,691	0,709	0,727	0,745	0,764	0,782	0,800	0,818	0,836	0,855	0,873	0,891	0,909
8,6	0,449	0,551	0,569	0,587	0,605	0,623	0,641	0,659	0,677	0,695	0,713	0,731	0,749	0,767	0,785	0,803	0,821	0,838	0,856	0,874	0,892	0,910
8,8	0,443	0,557	0,575	0,592	0,610	0,628	0,646	0,663	0,681	0,699	0,716	0,734	0,752	0,770	0,787	0,805	0,823	0,841	0,858	0,876	0,894	0,911
9	0,438	0,563	0,580	0,598	0,615	0,633	0,650	0,668	0,685	0,703	0,720	0,738	0,755	0,773	0,790	0,808	0,825	0,843	0,860	0,878	0,895	0,913
9,2	0,432	0,568	0,585	0,602	0,620	0,637	0,654	0,672	0,689	0,706	0,723	0,741	0,758	0,775	0,793	0,810	0,827	0,844	0,862	0,879	0,896	0,914
9,4	0,427	0,573	0,590	0,607	0,624	0,641	0,659	0,676	0,693	0,710	0,727	0,744	0,761	0,778	0,795	0,812	0,829	0,846	0,863	0,880	0,898	0,915
9,6	0,422	0,578	0,595	0,612	0,629	0,646	0,663	0,680	0,696	0,713	0,730	0,747	0,764	0,781	0,798	0,814	0,831	0,848	0,865	0,882	0,899	0,916
9,8	0,417	0,583	0,600	0,617	0,633	0,650	0,667	0,683	0,700	0,717	0,733	0,750	0,767	0,783	0,800	0,817	0,833	0,850	0,867	0,883	0,900	0,917
10	0,412	0,588	0,605	0,621	0,638	0,654	0,671	0,687	0,704	0,720	0,736	0,753	0,769	0,786	0,802	0,819	0,835	0,852	0,868	0,885	0,901	0,918
100	0,065	0,935	0,937	0,940	0,942	0,945	0,948	0,950	0,953	0,956	0,958	0,961	0,963	0,966	0,969	0,971	0,974	0,976	0,979	0,982	0,984	0,987
1000	0,007	0,993	0,993	0,994	0,994	0,994	0,994	0,995	0,995	0,995	0,996	0,996	0,996	0,996	0,997	0,997	0,997	0,997	0,998	0,998	0,998	0,999

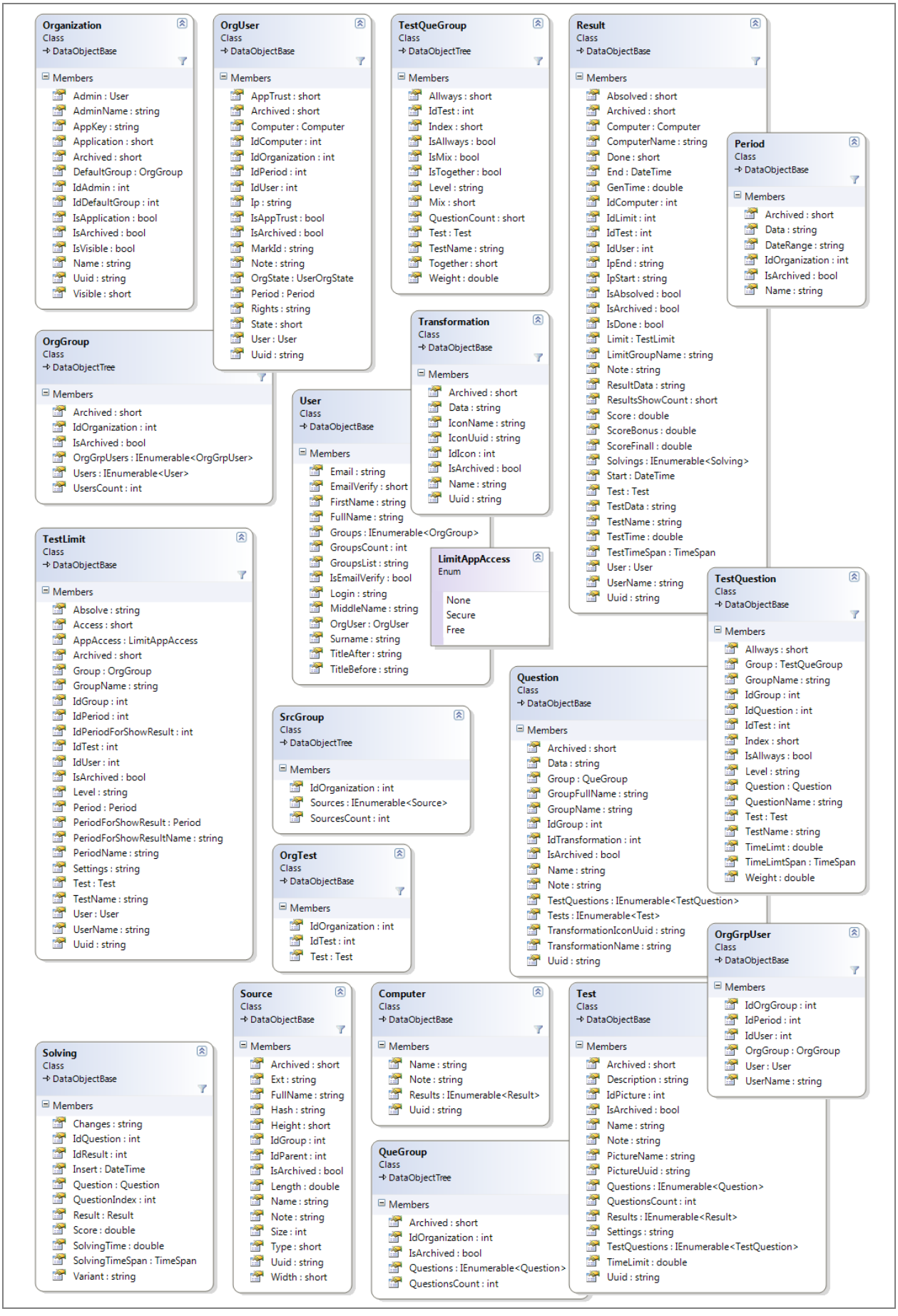
Příloha 12 – Schéma principu bezpečné autentizace aplikace klient-server v internetu pomocí povinně unikátních saltů



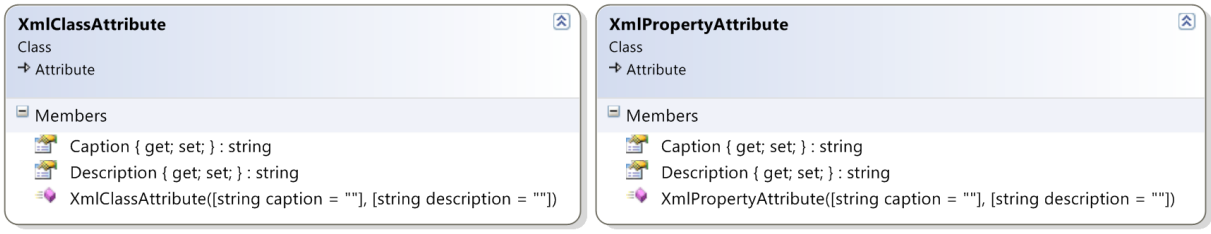
Příloha 13 – Datový model databáze (DBS: Firebird, zpracováno v IBExpert Database designer)



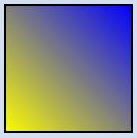
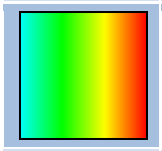
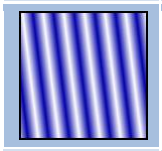
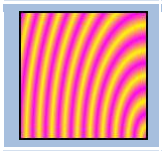
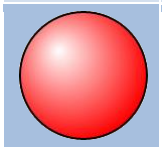
Příloha 14 – Diagram tříd pro O-R mapování v administračním rozhraní



Příloha 15 – Diagram konfiguračních tříd `XmlClassAttribute` a `XmlPropertyAttribute`



Příloha 16 – Porovnání délek zápisu týchž barevných výplní v různých jazycích (představeném mini-jazyku, XAML a SVG)

Náhled	Zápis v mini-jazyku (M-J)	Délka [znaky]			Délka v M-J oproti...	
		M-J	XAML	SVG	XAML	SVG
	`Ld:0000FF~FFFF00`	16	161	181	9,9%	8,8%
	`LH:00FFFF~00FF00~FFFF00~FF0000`	30	258	241	11,6%	12,4%
	`L:1,0.6~0,0.1:00FFFF~0.4|FFFF00~FF00FF~0.6|FFFF00~00FFFF`	56	300	322	18,7%	17,4%
	`LP:0,0.5~0.08,0.49:Sreflect:0000AA~FFFFFF`	41	192	196	21,4%	20,9%
	`R:FFFFFF~000000`	15	129	139	11,6%	10,8%
	`R:Rx0.05y0.1:C1,1:G1,1:Sreflect:FF00FF~FFFF00`	45	215	183 *(nelze nastavit různé poloměry)*	20,9%	24,6%
	`R:R0.05:Sreflect:000000~0.5|000000~0.5|FFFFFF~FFFFFF`	52	272	267	19,1%	19,5%
	`R:R1:G0.3,0.3:FFFFF~0.6|FF0000`	31	154	162	20,1%	19,1%
				průměr	**16,7%**	**16,7%**

Příloha 17 – Cena odborné poroty za inovativní produkt ze soutěže a konference eLearning 2011 (viz [ap-8])

Univerzita Hradec Králové a sdružení EUNIS-CZ

v 11. ročníku soutěže eLearning
odborná porota uděluje

Cenu

odborné poroty za inovativní produkt

Univerzitě Hradec Králové,
Fakultě informatiky a managementu

UNIVERZÁLNÍ TESTOVACÍ PROSTŘEDÍ

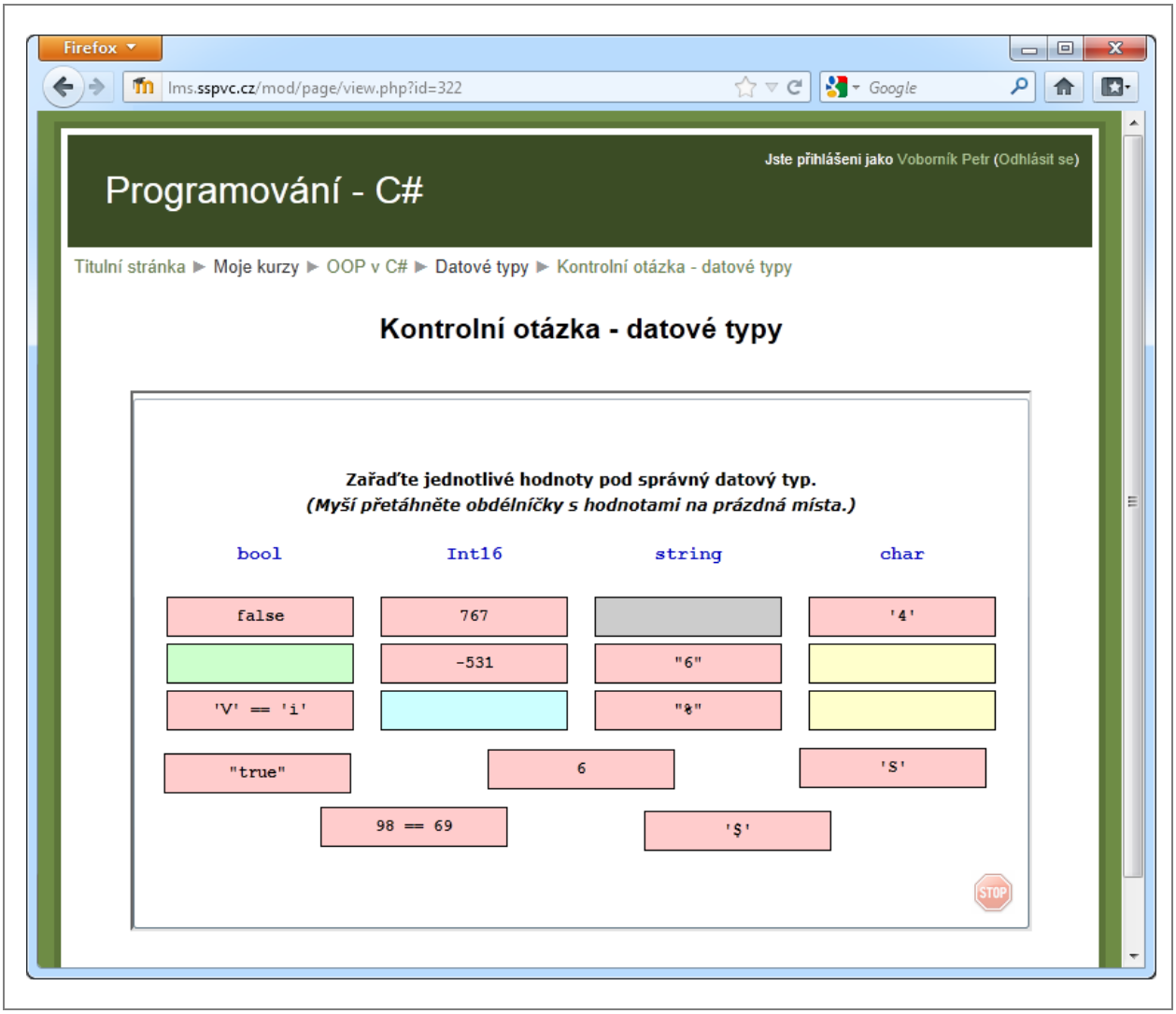
Autor:
Ing. Petr Voborník

V Hradci Králové 10. 11. 2011

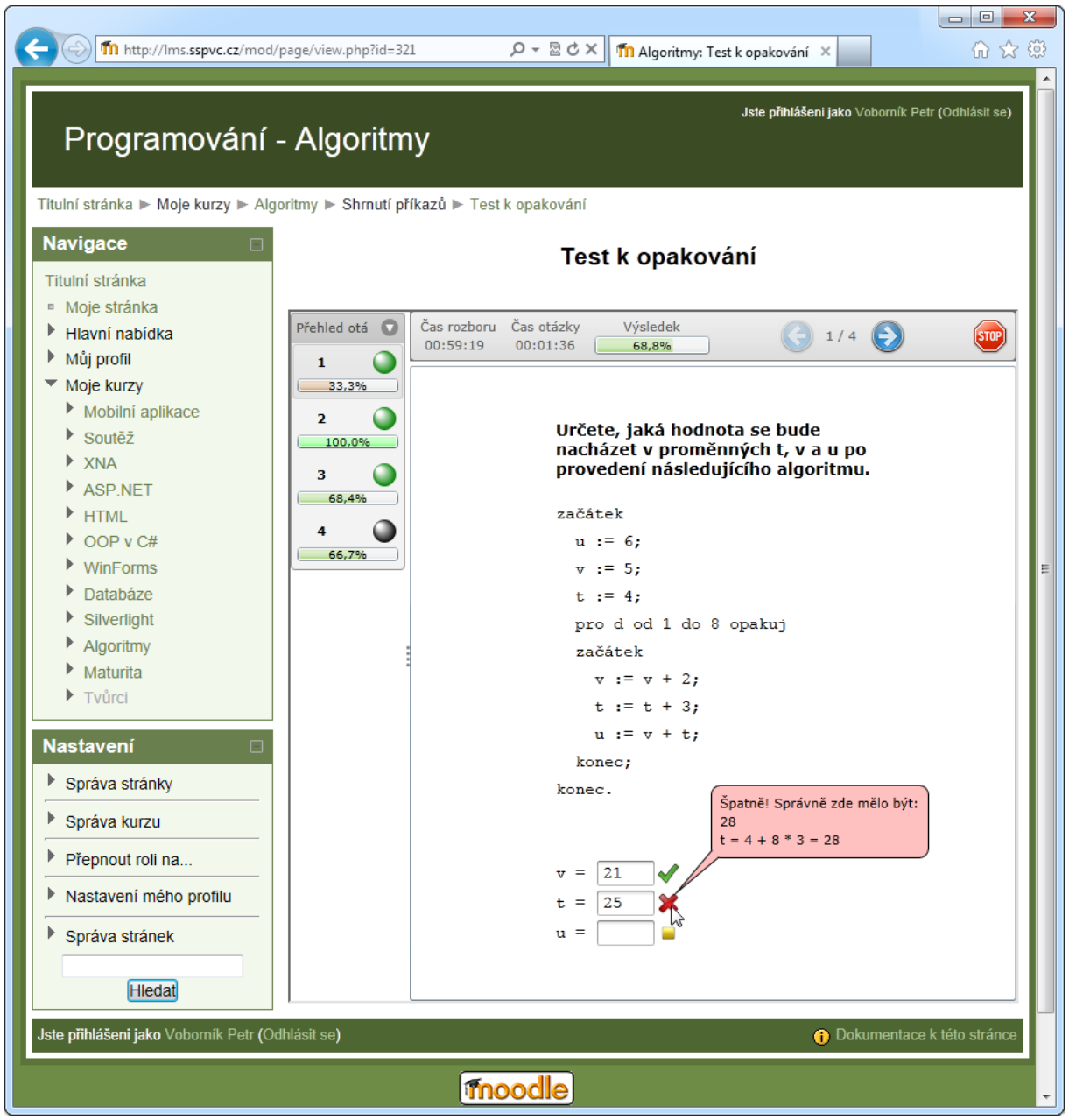
Petra Poulová

doc. RNDr. Petra Poulová, Ph.D.
Předsedkyně odborné poroty

Příloha 18 – Ukázka integrace kontrolní otázky do LMS Moodle

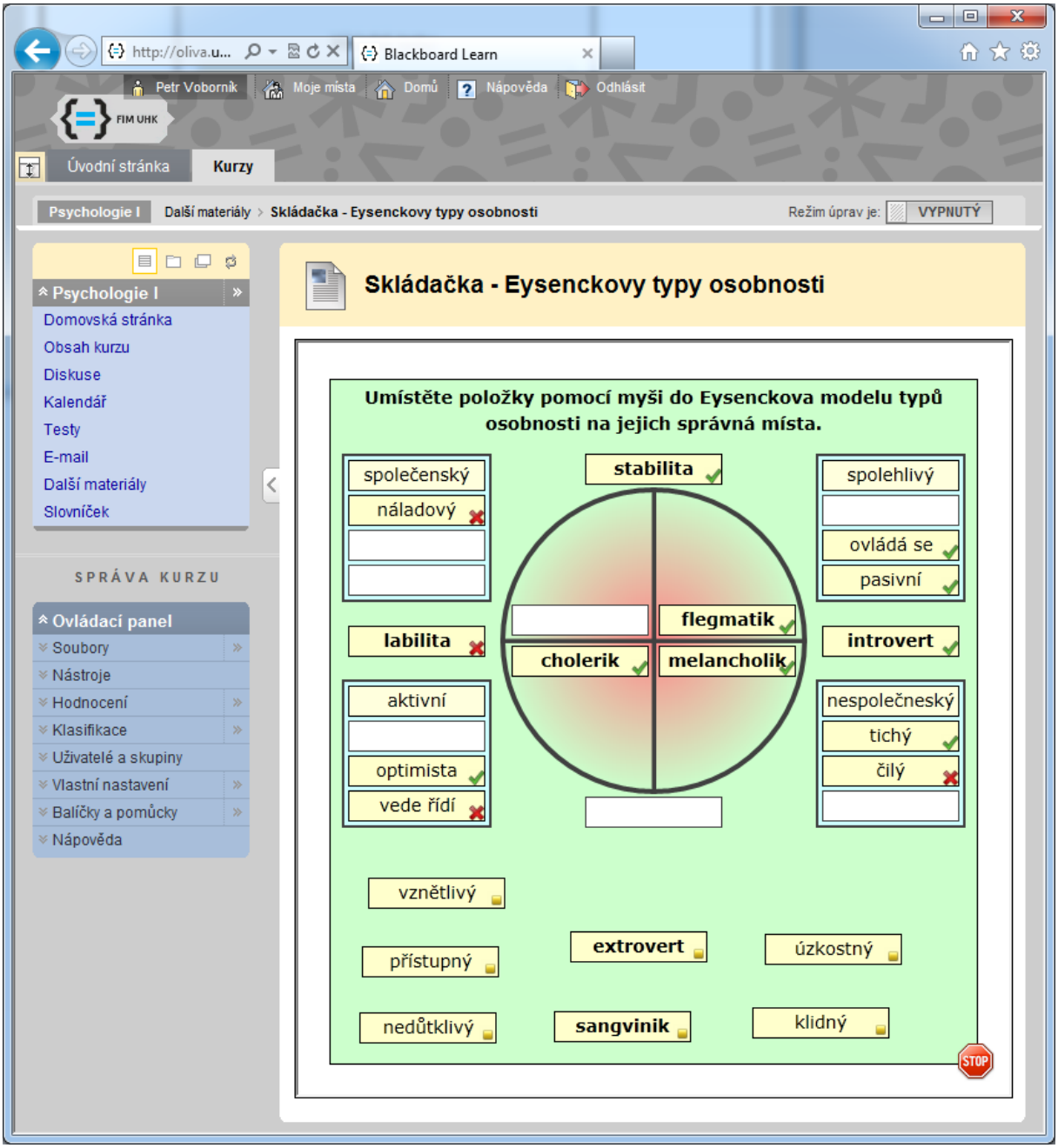


Příloha 19 – Ukázka integrace opakovacího testu do LMS Moodle



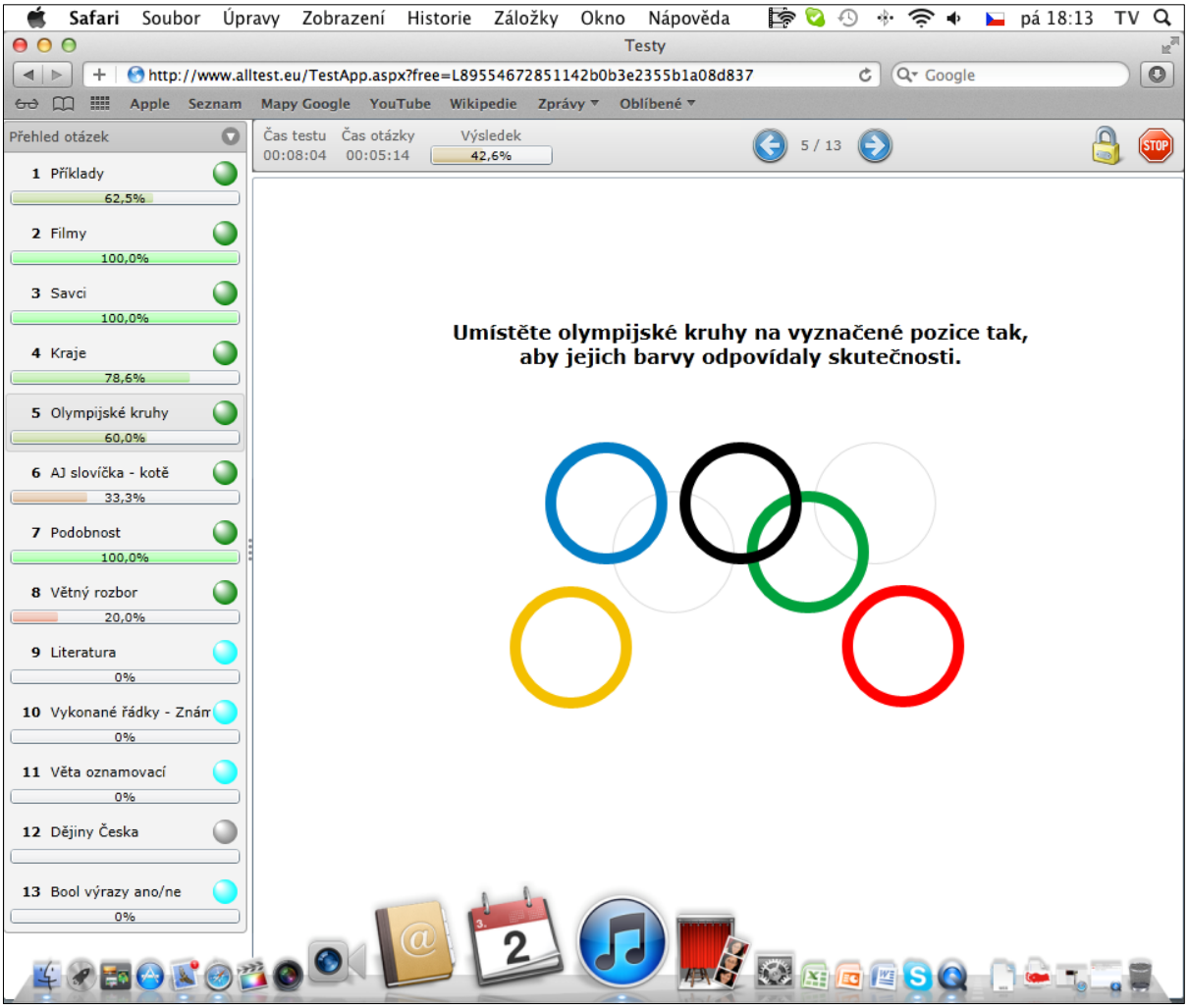
Pozn.: V otázce na tomto obrázku se náhodně vybírají názvy všech čtyř proměnných jako písmena z celé abecedy (bez duplicit), jejich výchozí hodnoty od 1 do 6, horní limit cyklu od 3 do 10 a v cyklu přičítané hodnoty od 1 do 3. Zároveň se náhodně mění pořadí řádků výchozího nastavení proměnných, pořadí inkrementace proměnných v cyklu a pořadí řádků pro zdávání výsledků.

Příloha 20 – Ukázka integrace jednoduchého výukového testu do LMS Blackboard

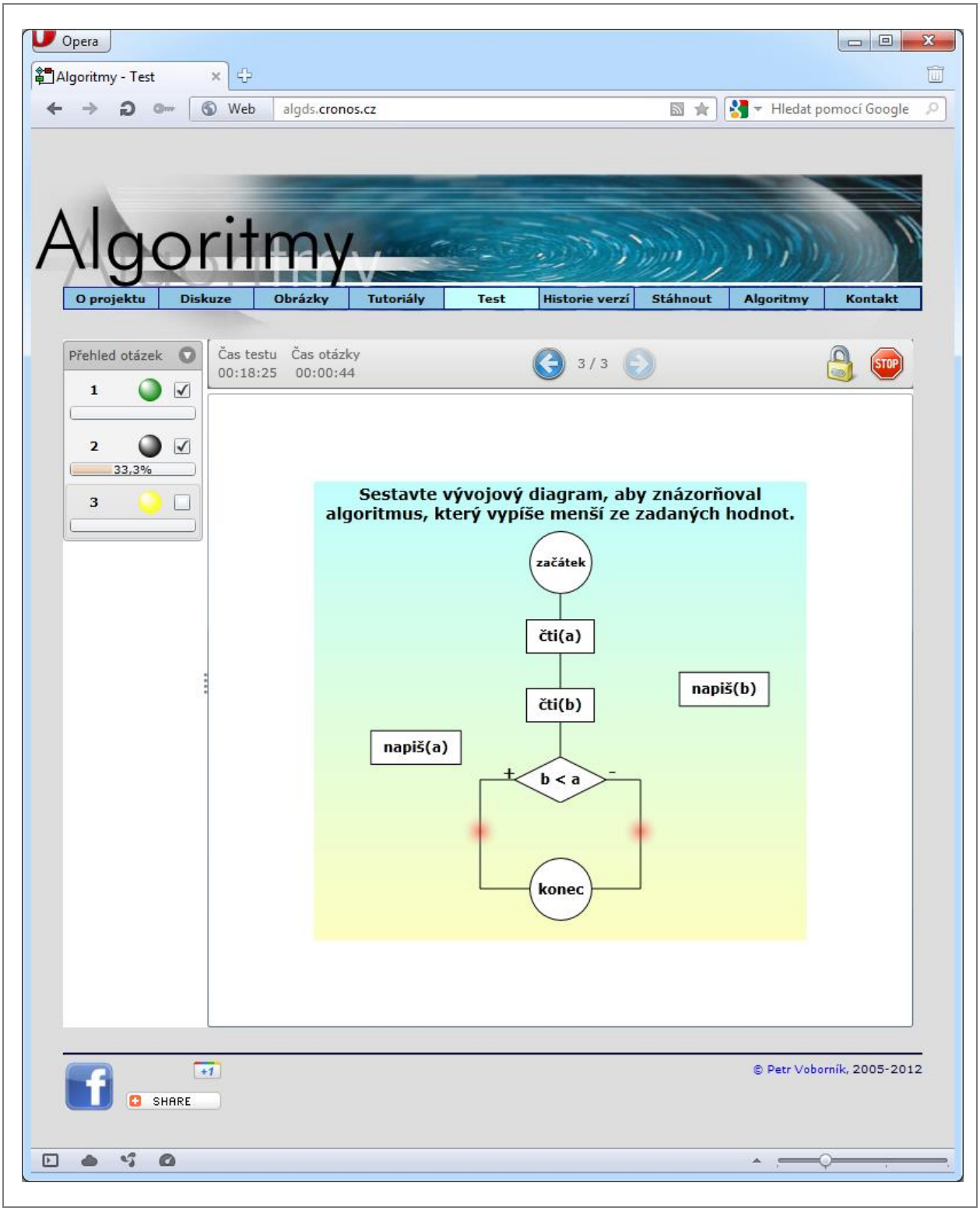


Pozn.: V této otázce se kruh s typy osobnosti otáčí, čili existují 4 jeho polohy, přičemž v rámci zadání je v každém z obdélníčků s vlastnostmi osobnosti vždy přiřazena první z nich, náhodně vybraná ze čtyř možných, tzn., existuje $4 \cdot 4^4 = 32$ možných zadání, z nichž každé má 20! možných řešení, přičemž z nich je vždy pouze $3!^4 = 1\,296$ správných (vlastnosti mohou mít libovolné pořadí).

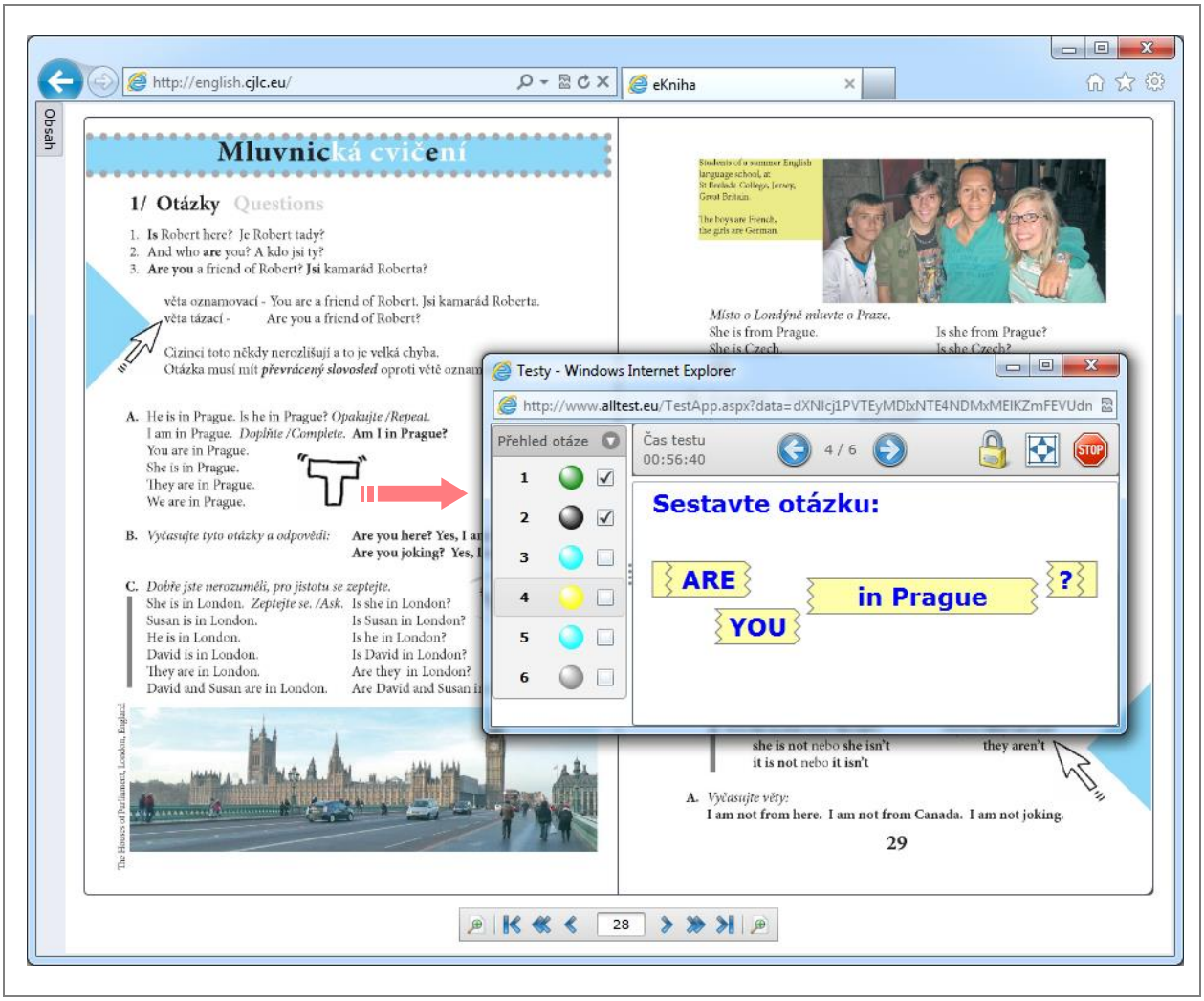
Příloha 21 – Ukázka testu pod operačním systémem Mac OS X v prohlížeči Safari



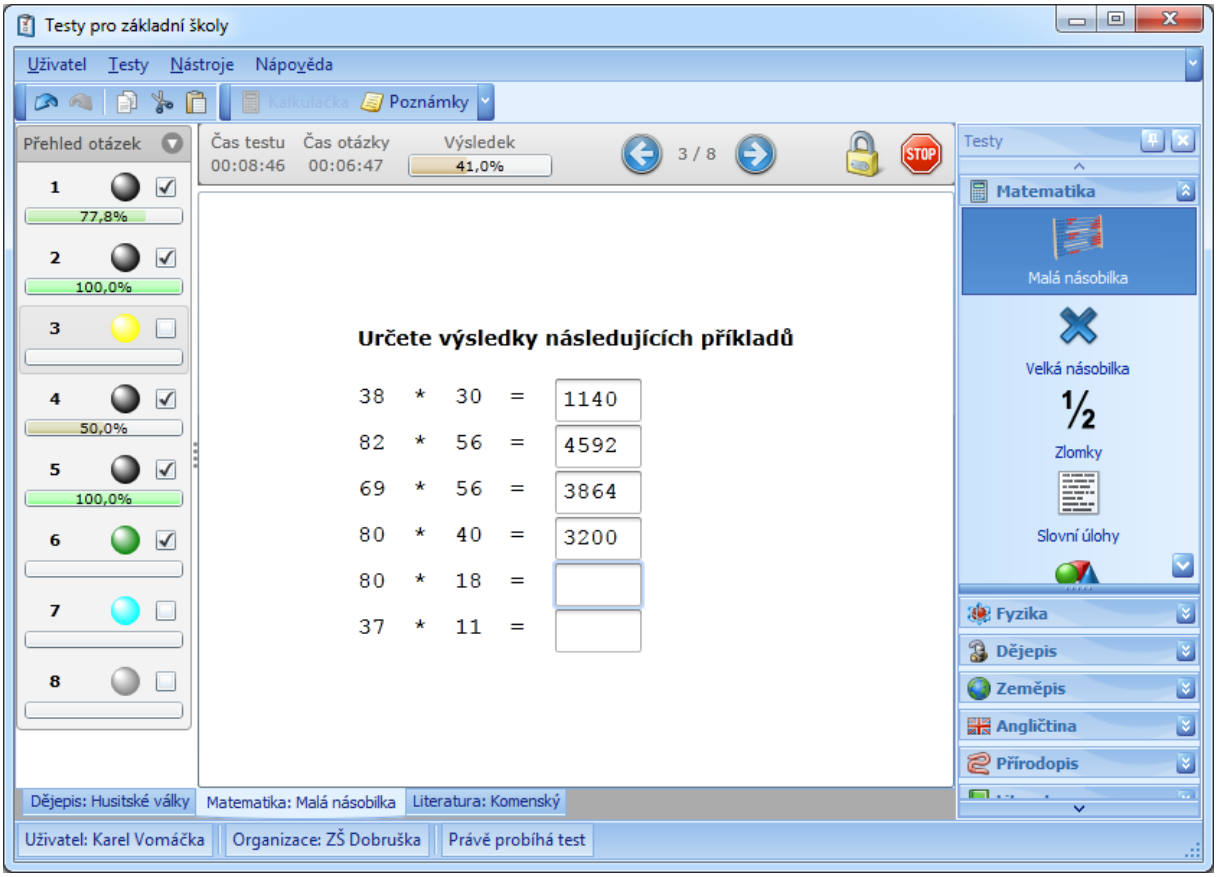
Příloha 22 – Ukázka integrace testu do webových stránek k projektu Algoritmy



Příloha 23 – Ukázka použití testu v elektronické on-line učebnici



Příloha 24 – Ilustrační ukázka integrace testů do desktopové aplikace



Příloha 25 – Parametry testovacího počítače použitého při měření

CPU	Intel Core2 Duo (T9300), 2,5 GHz
RAM	4 GB
HDD	7 200 RPM
OS	Windows 7 64bit
.NET	4.0

Příloha 26 – Metriky zdrojového kódu Univerzálního testovacího prostředí k 23.2.2012

Druh zdrojového kódu		Projekty	Soubory	Znaky
Web	Design	1	10	20 445
	Kód	-	17	28 786
Knihovny	Vlastní	6	67	257 171
	Převzatý	1	3	76 137
Silvelright	Design	5	35	127 319
	Kód vlastní	2	170	945 225
	Kód převzatý	1	45	253 153
Celkem		16	347	1 708 236