Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný odbor: INFORMATIKA

Bc. Andrej Krištofič

Objavovanie znalostí o správaní sa študenta pri učení sa programovať

Diplomová práca

Vedúca diplomovej práce: doc. Ing. Mária Bieliková, PhD.

máj 2005

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný odbor: INFORMATIKA

Autor: Bc. Andrej Krištofič

Diplomová práca: Objavovanie znalostí o správaní sa študenta pri učení sa programovať

Vedenie diplomovej práce: doc. Ing. Mária Bieliková, PhD.

máj, 2005

Cieľom diplomovej práce je aplikovať postupy objavovania znalostí a objaviť typické vzory správania sa študentov v záznamoch o používaní výučbových adaptívnych hypermediálnych (AH) systémov. Objavené vzory majú slúžiť ako základ pre skvalitnenie prispôsobovania AH systémov.

V práci analyzujeme údaje zaznamenávané v AH systémoch, pričom cieľom je vyhodnotiť vhodnosť ich použitia v procese objavovania znalostí. Navrhujeme niekoľko možných spôsobov využitia objavených znalostí – vzorov správania sa, zameriavame sa predovšetkým na odporúčanie konceptov študentom počas ich práce s AH systémom. Výsledkom je návrh procesu pre objavovanie znalostí, ktorý sme prispôsobili pre AH systémy.

Prezentujeme navrhnutú architektúru systému na odporúčanie navigácie, ktorý realizuje proces objavovania znalostí opísaný v práci. Architektúra je nezávislá od použitého AH systému. Taktiež pamätá na prezentáciu objavených znalostí autorovi obsahu AH systému. Návrh sme overili implementáciou prototypu, ktorý realizuje veľkú časť navrhnutého systému. Prototyp vytvára odporúčania (vo forme postupností relevantných konceptov) pre používateľov AH systémov. Na overenie a vyhodnotenie prototypu sme použili reálne údaje zo systému ALEA.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree course: INFORMATICS

Author: Bc. Andrej Krištofič

Thesis: Discovery of knowledge about student's behavior during the process of learning programming

Supervisor: doc. Ing. Mária Bieliková, PhD.

2005, May

The objective of this thesis is to discover potential patterns in usage data collected by educational adaptive hypermedia (AH) systems by means of knowledge discovery. Discovered usage patterns are used to improve adaptive behavior of AH systems.

In this work, we analyze data collected by AH systems and evaluate their suitability for use in the process of knowledge discovery. We propose several possible applications of discovered knowledge – usage patterns, while focusing on recommendation of relevant concepts to the students during their learning session. This resulted in a proposal of knowledge discovery process adapted for AH systems.

We devised an architecture of the software system, which follows proposed knowledge discovery process. The architecture is independent of underlying AH system. It also deals with presentation of discovered knowledge to the author of the adaptive course. To verify our design we developed a prototype, which covers the most of functionality of proposed system. The prototype is carries out recommendations (in the form of sequences of relevant concepts) for users of AH systems. Real usage data from AH system ALEA were employed in order to test and evaluate the prototype.

Čestne prehlasujem, že túto prácu som vypracoval samostatne a použil som len uvedenú literatúru.

Andrej Krištofič

Obsah

1 Úvod

V posledných rokoch Internet zaznamenal obrovský rozmach. S rozvojom Internetu sa zmenila aj povaha aplikácií – rozhranie, ktoré sprístupňuje funkcionalitu aplikácie cez sieť je dnes už súčasťou mnohých systémov. Svoje miesto na poli webových aplikácií si pomaly začínajú nachádzať aj adaptívne hypermediálne (AH) systémy. Nie je prekvapením, že jednou z rozsiahlych oblastí využitia takýchto systémov sú práve systémy na podpory výučby a vzdelávania. Je to prirodzené, veď každý študent má odlišný spôsob vnímania a učenia predkladaných materiálov a preto je výhodné, ak mu výučbový systém dokáže uľahčiť jeho štúdium tým, že mu prispôsobuje prezentáciu domény ako aj navigáciu v nej.

Výučbové AH systémy sa obyčajne prispôsobujú používateľovi na základe tzv. špecifikácie prispôsobovania, ktorá má často podobu pravidiel. Je úlohou autora obsahu AH systému, aby vytvoril vhodnú sadu pravidiel, ktorá zabezpečí čo najlepšie prispôsobovanie sa študentovi. Zostavenie takýchto pravidiel nemusí byť jednoduchá úloha, nakoľko si to vyžaduje dobrú znalosť o doméne, ktorú AH systém predkladá, ako aj o vhodných postupoch študentov pri štúdiu konkrétnej problémovej oblasti (napr. pri učení sa programovať). Pozorovaním správania sa používateľov pri používaní systému môžeme objaviť postupy štúdia, ktoré sú spoločné pre väčšie skupiny študentov. Využitím prostriedkov na objavovanie znalostí môžeme tieto postupy štúdia nájsť a využiť ich na skvalitnenie prispôsobovania (výučbových) AH systémov používateľovi.

V práci sa zaoberáme možnosťami objavovania znalostí v záznamoch o používaní AH systémov a ich využitia na skvalitnenie prispôsobovania AH systému. Prvým cieľom je navrhnúť postup na objavovanie a využitie znalostí v AH systémoch a definovať, aké údaje sú pre tento proces potrebné. Ďalšou úlohou je navrhnúť prostriedok, ktorý navrhnutý proces bude realizovať.

Nakoľko záznamy o používaní AH systémov obsahujú predovšetkým údaje o tom, ktoré koncepty používateľ navštívil pri štúdiu problémovej oblasti, objavené znalosti majú povahu charakteristických postupností alebo skupín konceptov. Takýto druh znalostí možno využiť napríklad na odporúčanie relevantných konceptov študentovi a pomôcť mu tak lepšie zvládnuť navigáciu v systéme.

Prvá časť práce obsahuje všeobecný úvod do problematiky AH systémov. Dôraz pri tom kladieme na výučbové AH systémy a ich špecifiká. Súčasným stavom v objavovaní znalostí vo výučbových AH systémoch sa zaoberá druhá časť. Okrem analýzy údajov vhodných na objavovanie znalostí sa venuje aj rôznym technikám objavovania znalostí a možnostiam využitia týchto znalostí. V nasledujúcej časti opisujeme návrh procesu na objavovanie znalostí v AH systémoch. Piata časť predstavuje systém na odporúčanie navigácie, jeho

scenáre použitia, architektúru a použité algoritmy. Takisto je v nej opísaný prototyp uvedeného systému. Experimentom s prototypom a ich výsledkom je venovaná posledná, šiesta časť.

Výsledky tejto práce sme prezentovali aj formou článkov, ktoré sme zaslali na dve konferencie. Príspevky [17,18] nájdete v prílohách A a B. Ďalšie prílohy obsahujú formálne zápisy navrhnutých algoritmov, príručku pre správcu systému, opis rozhraní tried systému, schému databázy a ukážku reprezentácie bázy znalostí systému, ako aj ukážku zdrojového kódu. Poslednou prílohou je štruktúra adresárov priloženého CD spolu s ich opisom.

2 Výučbové adaptívne hypermediálne systémy

V nasledujúcej časti bližšie predstavíme adaptívne hypermediálne (AH) systémy, základné princípy ich fungovania a naznačíme oblasti ich použitia. Podrobnejšie sa budeme venovať výučbovým AH systémom, ktoré sú primárnym predmetom záujmu tejto práce. Na záver uvedieme charakteristiku a porovnanie dvoch výučbových AH systémov, pričom sa zameriame na tie aspekty, ktoré považujeme pre túto prácu za dôležité.

2.1 Adaptívne hypermediálne systémy

Adaptívne hypermediálne (AH) systémy sú špeciálnou rodinou hypermediálnych[1] systémov, ktoré sa snažia prispôsobovať používateľovi, ktorý s nimi pracuje. Pri prispôsobovaní sa pritom berú do úvahy jednak preferencie používateľa, ale aj znalosti o jeho predchádzajúcich aktivitách v systéme (ak sú nejaké k dispozícii). Na reprezentáciu znalostí o používateľovi slúži model používateľa. Typickými atribútmi modelu používateľa sú napríklad jeho preferencie, počet návštev jednotlivých konceptov alebo úroveň jeho znalostí o konceptoch (či už zadaná samotným používateľom alebo vypočítaná systémom).

Hlavným cieľom AH systémov je uľahčiť navigáciu v častokrát rozsiahlom hypermediálnom priestore a poskytnúť používateľovi tie informácie, ktoré sú pre neho vhodné (vzhľadom na jeho preferencie, záujmy, či znalosti o oblasti). Na dosiahnutie svojho cieľa sa AH systémy prispôsobujú používateľovi v dvoch základných oblastiach [8]:

1. *Adaptívna prezentácia obsahu*

   Táto oblasť sa niekedy zvykne deliť na dve samostatné časti: prispôsobovanie vzhľadu prezentácie a prispôsobovanie obsahu prezentácie.

   Pod prispôsobovaním vzhľadu prezentácie rozumieme techniky, ktoré používateľom zobrazujú ten istý obsah, ale v rôznej forme. Príkladom takýchto techník môže byť napríklad zmena poradia a rozmiestnenia jednotlivých fragmentov na stránke, rôzne štýly zobrazenia, farby alebo fonty.

   Pri prispôsobovaní obsahu sa používateľovi prezentujú rôzne verzie daného konceptu. Ako príklad môžu poslúžiť rôzne verzie pre začínajúcich a pokročilých používateľov. Výber vhodnej verzie potom závisí od úrovne znalostí používateľa o téme.

[1] V tejto práci sa však nezameriavame na multimediálnu povahu informačných zdrojov, informačný priestor chápeme ako množinu uzlov (konceptov), ktoré sú prepojené vzťahmi (napr. odkazmi).

2. *Adaptívna navigácia*

   Neoddeliteľnou súčasťou hypermediálneho priestoru sú odkazy medzi jeho prvkami. Definujú vzťahy medzi jednotlivými prvkami priestoru a usmerňujú používateľa pri jeho prezeraní. Princíp adaptívnej navigácie spočíva v prispôsobovaní týchto odkazov pre jednotlivých používateľov. Podľa [8], nástup tzv. systémov na odporúčanie (angl. recommender systems) nás núti rozlišovať dve formy adaptívnej navigácie: prispôsobovanie statických odkazov (teda tých, ktoré boli definované v čase vytvárania obsahu AH systému) a generovanie nových odkazov.

   Prispôsobovanie existujúcich odkazov môže mať mnoho foriem. Najčastejšie používanými technikami sú: skrývanie odkazov, ich usporadúvanie a anotácia odkazov (vizuálne označovanie vzťahu odkazov k danej téme).

   Príkladom generovania nových odkazov môže byť odporúčanie relevantných dokumentov k danej téme, či už na základe príbuznosti medzi jednotlivými konceptmi, alebo založené na informáciách o používaní systému ostatnými používateľmi. Samozrejme, techniky ako usporadúvanie alebo anotácia odkazov sú aktuálne aj pre generované odkazy.

Adaptívne hypermediálne systémy nachádzajú široké možnosti uplatnenia v rôznych oblastiach. Príkladom môžu byť on-line informačné systémy, on-line help systémy alebo hypermédiá pre získavanie informácií [8]. Pravdepodobne najväčšou oblasťou, kde sa AH systémy udomácnili, je oblasť výučbových systémov. Špecifikami takýchto AH systémov sa podrobnejšie zaoberáme v nasledujúcej kapitole.

2.2 Špecifiká výučbových adaptívnych hypermediálnych systémov

Výučbové AH systémy majú niekoľko typických čŕt, ktorými sa odlišujú od AH systémov iného zamerania. Okrem tradičných metód prispôsobovania (adaptívna prezentácia obsahu a adaptívna navigácia) nachádzame vo výučbových AH systémoch ešte ďalšie špecifické metódy prispôsobovania, ktoré majú svoj pôvod v inteligentných výučbových systémoch (angl. Intelligent Tutoring Systems) (podľa [6,7]):

1. *Určenie poradia štúdia tém* (angl. curriculum sequencing)

   Cieľom je určiť poradie, v akom by mal študent témy problémovej oblasti preberať. Môžeme rozlíšiť usporiadanie na vysokej úrovni (poradie konceptov) a nízkej úrovni (poradie úloh – fragmentov v rámci konceptu).

2. *Inteligentná analýza riešení* (angl. intelligent analysis of solutions)

   Na základe inteligentného vyhodnotenia riešení problémov je možné poskytnúť študentovi vhodnú spätnú väzbu a aktualizovať jeho model používateľa. Spätná väzba pre používateľa samozrejme nezávisí iba od samotného riešenia, ale aj od aktuálneho stavu modelu používateľa. Príkladom môže byť napríklad chyba v odovzdanom

riešení. Študentovi, ktorý má vysokú úroveň znalostí o téme, systém odporučí, aby si svoje riešenie ešte raz prekontroloval (pravdepodobne išlo o chybu z nepozornosti, preklep a pod.). Na druhej strane, študentovi s menšími vedomosťami systém navrhne dodatočné materiály, ktoré by si ešte mal preštudovať.

3. *Interaktívna pomoc pri riešení problémov* (angl. interactive problem solving support)

   Výučbový AH systém môže pri riešení problémov viesť študenta krok za krokom. Rozdelenie riešenia na kroky umožňuje systému sledovať správanie sa používateľa počas riešenia problémov, vhodne aktualizovať jeho model a na jeho základe mu poskytnúť pomoc pri riešení.

4. *Riešenie problémov na základe príkladov* (angl. example based problem solving)

   Pri riešení problému môže systém študentovi odporučiť odkazy, či už na vzorové riešenia podobných problémov, alebo na riešenia problémov, ktoré študent vyriešil v minulosti.

Hypermediálny priestor, ktorý výučbové AH systémy pokrývajú nie je zvyčajne veľmi rozsiahly a spravidla je zameraný na jednu oblasť (napr. výučba programovania). Cieľom používateľa (študenta) je zvládnuť celú problémovú oblasť. Primárnou úlohou AH systému je v tomto prípade odporučiť študentovi vhodné poradie konceptov. Treba si uvedomiť, že väčšina používateľov spočiatku nemá žiadne znalosti z danej oblasti, a práve pre týchto študentov je „to správne“ poradie konceptov veľmi dôležité.

Študenti pri využívaní výučbového systému zvyčajne postupujú podľa určitých vzorov (niekedy nazývaných aj štýly učenia). Úlohou systému teda je, aby tieto vzory správania sa odhalil (napr. z historických záznamov o používaní systému) a aby na ich základe viedol používateľa pri štúdiu. To si vyžaduje analyzovať vzdelávacie aktivity používateľov v systéme. Technikami pre takúto analýzu sa podrobnejšie zaoberáme v časti 3.2.

Významnú úlohu v modeli používateľa vo výučbových AH systémoch zohráva úroveň znalostí používateľa o jednotlivých konceptoch. Používateľom s vysokou úrovňou znalostí o téme sa môžu zobrazovať doplňujúce informácie, a naopak, používateľom s horším poznaním konceptu sa môžu niektoré jeho pokročilé časti skryť.

Zaujímavým problémom je, ako zistiť úroveň znalostí študenta o danej téme. Len ťažko možno zo správania sa študenta v systéme spoľahlivo vypozorovať, aké má znalosti o konceptoch. Zvyčajne je na to potrebná spätná väzba od používateľa. Takouto spätnou väzbou môže byť napríklad vyplnenie elektronického dotazníka s testom alebo jednoducho stlačenie gombíka „Pochopil som danú tému“. Takisto tu existuje možnosť vyhodnotiť znalosti používateľa na základe externých informácií, ako sú napríklad hodnotenia priebežných testov alebo skúšok. Informáciu o úrovni vedomostí (a teda o úspešnosti, resp. neúspešnosti) študentov môžeme s výhodou využiť pri ohodnocovaní nájdených vzorov správania sa, resp. už pri výbere údajov pre vyhľadávanie týchto vzorov.

Samotná výučba programovania pomocou príkladov obsahuje ešte niekoľko špecifických aspektov. Ukazuje sa, že je vhodné rozlišovať rôzne druhy (textových) informácií v systéme [14], ako na vyššej úrovni konceptov (vysvetľujúce texty, programové schémy, príklady), tak aj na nižšej úrovni fragmentov (text, príklad, riešenie príkladu, pomôcka pri riešení). Existencia rôznych druhov konceptov, resp. fragmentov nám poskytuje užitočnú formu abstrakcie, ktorá môže prispieť k skvalitneniu výsledkov objavovania vzorov správania sa študentov. Namiesto hľadania vzorov na úrovni konceptov môžeme hľadať vzory aj na úrovni ich druhov.

2.3 Príklady výučbových adaptívnych hypermediálnych systémov

V tejto kapitole uvedieme dva príklady výučbových AH systémov. Podrobnejšie sa vyjadríme k údajom, ktoré nám systémy poskytujú o správaní sa používateľa v nich a k spôsobu uloženia týchto údajov.

AHA!

Systém AHA! (verzia 2.0) [3,10] je univerzálny[2] AH systém vyvíjaný na Technickej univerzite v Eindhovene. V súčasnosti sa najviac využíva vo výskumných projektoch alebo na podporu výučby.

Základnou jednotkou obsahu systému je koncept. Každý koncept môže mať priradených viac atribútov (napr. úroveň znalostí používateľa o koncepte) a pravidiel prispôsobovania, ktoré sa vyhodnocujú pri každej návšteve konceptu. AHA! používa tzv. prekryvný model používateľa – obsahuje teda atribúty všetkých konceptov pre jednotlivých používateľov. Systém využíva obe spomínané základné techniky adaptácie – adaptívnu prezentáciu obsahu (podmienené zobrazovanie fragmentov jednotlivých stránok, neexistuje spoločná báza fragmentov) a adaptívnu navigáciu (ukrývanie a označovanie odkazov). Podporuje vytváranie testov, ktorých výsledky môžu byť užitočným zdrojom na zistenie úrovne znalostí používateľa o tom-ktorom koncepte.

Systém používa na uloženie pravidiel prispôsobovania a modelu používateľa súborový systém (súbory vo formáte XML) alebo databázu (MySQL). Jediné informácie, ktoré sú k dispozícii, sú hodnoty atribútov v modeli používateľa. Súčasná verzia systému (2.0) nevytvára záznamy o návšteve jednotlivých konceptov ani o čase strávenom na týchto konceptoch. Nedávno bola sprístupnená verzia (3.0) tohto systému, ktorá ukladá informácie o prístupe používateľov do systému ako aj zmeny modelu používateľa do XML súborov.

[2] Pôvodne však vznikol ako systém na podporu výučby, preto sme ho zaradili do skupiny výučbových AH systémov.

ALEA

Systém ALEA je výsledkom diplomovej práce Radovana Kostelníka [14]. Používa sa na podporu výučby predmetu Funkcionálne a logické programovanie na Fakulte informatiky a informačných technológií STU v Bratislave.

Podobne ako v systéme AHA!, tak aj v systéme ALEA je základným prvkom domény koncept. Doména je modelovaná ako orientovaný graf, ktorého uzly sú koncepty a hrany vzťahy medzi nimi. Systém umožňuje ku každému vzťahu zadefinovať aj jeho druh (napr. vzťah rodič-potomok, príklad, text, schéma) a vytvoriť tak komplexný model domény. Každý koncept v systéme je možné spojiť viacerými fragmentmi, ktoré sú uložené v spoločnej báze. Je pritom zadefinovaných viac druhov vzťahov medzi konceptom a fragmentom, napríklad text, príklad, návod na riešenie alebo riešenie príkladu. Systém umožňuje vytvárať aj tzv. stratégie pre prispôsobovanie a zároveň zabezpečuje výber vhodnej stratégie pre používateľa na základe vopred stanovených pravidiel. Na prispôsobovanie používateľovi sa používa technika usporadúvania fragmentov a technika anotácie odkazov. Okrem statických odkazov priamo vo fragmentoch systém generuje a anotuje aj tzv. kontextové odkazy na základe modelu domény. Dôležitým prvkom je aj podpora vedenia používateľa pomocou špeciálneho tlačidla „Ďalej“, ktoré študentovi na základe jeho znalostí vyberie ďalšiu tému na štúdium.

Na uloženie informácií sa používa databáza Microsoft Access. Zaznamenáva sa podrobná informácia o akciách používateľa v systéme: ktoré koncepty navštívil, kedy ich navštívil a koľkokrát ich navštívil.

Porovnanie systémov

Z hľadiska poskytovaných údajov sa pre overenie výsledkov tejto práce ako vhodnejší javí systém ALEA, ktorý si na rozdiel od systému AHA! ukladá do databázy údaje a aktivitách študenta v systéme. V systéme AHA! by bolo potrebné doplniť podporu pre zaznamenávanie potrebných informácií alebo by sa tieto údaje museli získať zo záznamov webového servera, na ktorom by bol systém nasadený. Ďalšou alternatívou je použiť vývojovú verziu tohto systému, ktorá už podporuje zaznamenávanie údajov o svojom používaní.

Informácia o vzťahoch medzi konceptmi, resp. fragmentmi pritom môže prispieť k skvalitneniu výsledkov objavovania vzorov správania sa študentov. Systém ALEA umožňuje vybudovať komplexný model domény – poskytuje možnosť zadefinovania rôznych vzťahov medzi konceptmi. V systéme AHA! je táto možnosť obmedzená. Takisto nie je možné zadefinovať rôzne druhy vzťahov medzi konceptmi a fragmentmi.

Pre každý koncept je systéme AHA! možné určiť vlastnú množinu atribútov. Autor obsahu má potom viac možností, ako vytvoriť pravidlá pre zistenie úrovne znalostí o koncepte. V systéme ALEA sú atribúty pevne dané (počet návštev, čas poslednej návštevy, či používateľ označil koncept ako pochopený a úroveň znalostí). Pre účely tejto práce je najvýznamnejším atribútom úroveň znalostí. Tento atribút poskytujú oba systémy, aj keď na rôznej úrovni podrobnosti (ALEA: 4 úrovne <0,3>, AHA!: celé číslo z intervalu <0,100>).

Systém AHA! navyše umožňuje vytvárať testy, výsledky ktorých môžu slúžiť na presnejšie zistenie úrovne znalostí o konceptoch.

3 Objavovanie znalostí v AH systémoch – súčasný stav

3.1 Údaje pre objavovanie znalostí

Adaptívne hypermediálne systémy nám poskytujú niekoľko rôznych druhov údajov, ktoré môžu slúžiť ako základ pre objavovanie znalostí. V nasledujúcich kapitolách charakterizujeme tieto údaje a naznačíme možnosti ich využitia v procese objavovania znalostí.

Záznamy o navštívených konceptoch (resp. zobrazených fragmentoch)

Je zrejmé, že práve údaje o tom, ktoré koncepty používateľ navštívil pri svojej predošlej práci sú najcennejším zdrojom znalostí o správaní sa používateľa. Hovoria nám o tom, ktorým častiam používateľ venoval zvýšenú pozornosť, kam sa často vracal, či v akom poradí sa rozhodol študovať jednotlivé koncepty domény. Ideálne je, ak vieme zo záznamov zistiť jednotlivé prístupy[3] používateľa do systému, a tak ohraničiť jeho jednotlivé vzdelávacie aktivity.

Údajom, ktorý môže prispieť ku spresneniu výsledkov objavovania znalostí, je aj čas zobrazenia konceptov, resp. čas, ktorý používateľ na konceptoch strávil. Tento údaj sa dá využiť napríklad pri vyhodnocovaní podobnosti vzdelávacích aktivít používateľov [12]. Množstvo času, ktorý študent strávil na koncepte, sa dá ľahko odhadnúť na základe rozdielu časových značiek návštev jednotlivých konceptov. Tu však narážame na problém získavania spoľahlivých a dôveryhodných hodnôt. Nevieme totiž s istotou povedať, či sa študent skutočne zaoberal štúdiom predkladanej látky. Na získanie presnejšieho odhadu množstva času stráveného štúdiom zobrazovaných stránok sa často používajú programy na strane klienta [23].

O aktivitách používateľov si väčšinou systémy vedú záznamy vo vlastnej réžii (napr. v databáze alebo samostatnom súbore). V opačnom prípade je možné na získanie týchto údajov použiť záznamy z webového servera, nakoľko drvivá väčšina výučbových AH systémov je v súčasnosti realizovaná ako webové aplikácie.

Druhy konceptov/fragmentov a vzťahy medzi nimi

Znalosť druhov konceptov a fragmentov je pri výučbe programovania pomocou príkladov významná. Pomôže nám odlíšiť, ktoré koncepty predstavujú všeobecnú informáciu (napr. programovú schému), a ktoré naopak konkrétnu informáciu (napr. konkrétny príklad –

[3] Pod pojmom prístup rozumieme navštívené stránky od prihlásenia používateľa do systému až po jeho odhlásenie (angl. access session).

aplikáciu programovej schémy). Na úrovni fragmentov potom dokážeme rozlíšiť, či si študent pri riešení príkladu pozrel aj ponúkanú pomôcku, alebo či prešiel priamo na riešenie príkladu. Toto všetko nám umožní objavovať vzory správania sa aj na ďalšej úrovni – na úrovni typov konceptov alebo fragmentov.

Koncepty v doméne sú väčšinou usporiadané vo forme stromu, podobne ako kapitoly v knihe. Keď máme k dispozíciu informáciu o vzťahoch medzi konceptmi, dokážeme odhaľovať vzory správania na rôznych úrovniach ich hierarchie. Údaje o štruktúre domény ako aj o druhoch konceptov a fragmentov sú zvyčajne uložené v databáze systému alebo v štruktúrovanom textovom súbore (často vo formáte XML).

Úroveň vedomostí používateľa o konceptoch

Údaj o tom, na akej úrovni študent ovláda tému prezentovanú v koncepte, počíta AH systém na základe pravidiel zadaných autorom obsahu. Primárne sa využíva pri prispôsobovaní prezentácie alebo navigácie používateľovi, ale môžeme ho použiť aj ako kritérium na ohodnotenie jednotlivých používateľov. Pred samotným procesom hľadania charakteristických vzorov správania sa si potom môžeme vybrať cieľovú skupinu študentov (väčšinou tých úspešných, teda s vysokou úrovňou vedomostí). Nakoľko je však tento údaj vypočítaný systémom, treba pri jeho využití zohľadniť jeho presnosť a objektívnosť.

Úroveň vedomostí študenta je najčastejšie súčasťou modelu používateľa v systéme. Vzhľadom k tomu, že sa zaoberáme výučbovými AH systémami, asi ťažko by sme našli systém, ktorý by úroveň vedomostí v nejakej forme nepočítal. Väčšina systémov model používateľa ukladá do databázy alebo samostatného súboru, a preto nie je problém tento údaj zo systému získať.

Výsledky testov

Testovanie študenta je presnejšou formou zistenia jeho znalostí o problematike. V kombinácii s úrovňou vedomostí vypočítanou systémom môže pomôcť spresniť ohodnotenie používateľov. Výsledky testov sú síce určitým ohodnotením úrovne vedomostí študenta, ale považovali sme za potrebné uviesť ich samostatne, nakoľko ich považujeme za spoľahlivejšie. Tento fakt je treba zohľadniť aj pri ich využití v procese hľadania vzorov správania sa.

Pri testoch môžeme brať do úvahy jednak ich celkové výsledky (napr. na celkové ohodnotenie študenta), ale zaujímavé sú aj odpovede na jednotlivé otázky testu. Tieto odpovede nám umožnia vytvoriť si presnejší obraz o vedomostiach študenta. V takomto prípade je ale potrebné namapovať otázky, resp. odpovede testu, na jednotlivé koncepty a zadefinovať, akej úrovni vedomostí študenta zodpovedá jeho odpoveď.

Niektoré výučbové AH systémy majú v sebe priamo zabudovanú podporu testovania študentov. Ich výsledky systémy obvykle používajú na spresnenie výpočtu úrovne vedomostí študenta. Aj napriek tomu je užitočné, ak sú výsledky testov uložené priamo systéme, resp. v jeho záznamoch. V prípade, že systém nepodporuje testovanie študentov, je možné

informáciu dodať z iného zdroja. Môžu to byť výsledky zo systému určeného výhradne na testovanie vedomostí študentov, ale aj výsledky obyčajných „papierových“ testov.

3.2 Metódy a techniky objavovania znalostí

V súčasnosti neexistujú metódy a techniky určené špeciálne na objavovanie znalostí v AH systémoch. Nakoľko je však dnes väčšina AH systémov realizovaná ako webové aplikácie, budeme sa zaoberať technikami odhaľovania znalostí všeobecne vo webových systémoch. Zameriame sa pritom na tie techniky, ktoré slúžia na objavovanie znalostí za účelom personalizácie – teda určitej formy adaptácie webových aplikácií.

Údaje o používaní webových aplikácií majú podobnú povahu ako záznamy o používaní AH systémov. Kým pri webových systémoch obyčajne ide o záznamy o návštevách jednotlivých stránok, AH systémy zaznamenávajú už návštevy vzdelávacích objektov – konceptov a informačných fragmentov. Po vhodnom prispôsobení nasledujúcich techník a zohľadnení osobitostí AH systémov ich aplikujeme na objavovanie znalostí v adaptívnych hypermédiách.

Asociačné pravidlá

Asociačné pravidlá definujú vzťahy a závislosti medzi množinami objektov. Táto technika sa udomácnila predovšetkým v systémoch elektronického obchodu, kde sa často používa na analýzu obsahu nákupných košov. Výstupom sú asociačné pravidlá, ako napríklad: „12% ľudí, ktorí si kúpili výrobok A a výrobok B, si tiež kúpili výrobok C.“ Jedným z prvých publikovaných algoritmov na hľadanie asociačných pravidiel je algoritmus Apriori [1].

V kontexte webových systémov je možné túto techniku použiť na odhaľovanie vzťahov a závislostí medzi stránkami systému. Nájdené asociačné pravidlá sa dajú využiť pri odporúčaní relevantných stránok [16], praktickejší význam však majú pre autora obsahu systému, ktorý môže na ich základe upraviť prepojenia medzi stránkami a uľahčiť tak používateľom navigáciu v systéme.

Sekvenčné vzory

Myšlienka hľadania sekvenčných vzorov je veľmi podobná hľadaniu asociačných pravidiel. Hľadá závislosti medzi transakciami (napr. nákupmi zákazníkov), pričom však berie ohľad na poradie, v akom zákazník jednotlivé transakcie vykonal. Na rozdiel od hľadania asociačných pravidiel teda pracujeme s postupnosťami.

Na vyhľadávanie sekvenčných vzorov existuje pomerne veľké množstvo algoritmov. Najznámejšími sú pravdepodobne modifikácie algoritmu Apriori [2] (Apriori-All a Apriori-Some). Autori ďalších algoritmov sa zamerali predovšetkým na zefektívnenie hľadania sekvenčných vzorov vo veľkých databázach (napr. algoritmus SPAM [4]).

Pri aplikácii tejto techniky na záznamy webových systémov za transakciu môžeme považovať návštevu stránky. Nájdené sekvenčné vzory už môžeme považovať za určité charakteristické vzory správania sa, ale treba zobrať do úvahy fakt, že technika hľadania

sekvenčných vzorov síce rešpektuje poradie (v našom prípade konceptov), ale nevyžaduje, aby jednotlivé transakcie nasledovali bezprostredne za sebou. Interpretácia položiek sekvenčného vzoru teda môže byť problematická, pretože hovorí iba o tom, že používateľ navštívil dané stránky, a že ich navštívil v danom poradí. Nehovorí však nič o tom, ktoré stránky navštívil medzi stránkami v sekvenčnom vzore.

Súvislé sekvenčné vzory[4]

Súvislý sekvenčný vzor je zvláštnym prípadom sekvenčného vzoru, kedy jednotlivé transakcie, ktoré vzor obsahuje, musia za sebou bezprostredne nasledovať. Ide teda o hľadanie často sa vyskytujúcich podpostupností v postupnostiach transakcií. Technika sa často využíva pri analýze záznamov webového servera za účelom skvalitnenia štruktúry internetového sídla.

Rovnako ako pri sekvenčných vzoroch aj v tomto prípade budeme za transakciu považovať návštevu stránky v systéme. Nájdené vzory predstavujú „cesty“ v systéme, ktoré používatelia najčastejšie využívali. Takisto poslúžia autorovi obsahu AH systému, pretože predstavujú preferovaný spôsob navigácie používateľov v systéme. Môžu pomôcť odhaliť napríklad chýbajúce odkazy medzi konceptmi.

Existuje niekoľko prístupov hľadania súvislých sekvenčných vzorov. Podľa [9] sa najprv v záznamoch vyhľadajú postupnosti nazývané „maximal forward references“, ktoré v podstate vzniknú zanedbaním návratov používateľa k predošlým stránkam (autori predpokladajú, že takéto „návraty“ nie sú súčasťou sekvenčných vzorov). V takýchto špeciálnych postupnostiach sa potom vyhľadávajú sekvenčné vzory algoritmami, ktorých myšlienka je podobná ako pri algoritme Apriori (algoritmy Full Scan alebo Selective Scan [9]).

Myšlienke hľadania spojitých sekvenčných vzorov je blízke hľadanie tzv. zovšeobecnených postupností. Pri tomto prístupe sa v postupnostiach môžu nachádzať „žolíky“ (angl. wildcards), za ktoré je možné dosadiť ľubovoľnú postupnosť transakcií používateľa. Algoritmus na hľadanie zovšeobecnených postupností publikovaný v [11] je veľmi podobný algoritmu Full Scan [9]. Hoci sa na prvý pohľad môže zdať, že výstupy tohto algoritmu sa prakticky nelíšia od výstupov algoritmov na hľadanie sekvenčných vzorov, nie je to tak. V prípade zovšeobecnených postupností vieme povedať, kde stránky v nájdenej postupnosti nenasledujú bezprostredne za sebou (sú oddelené „žolíkom“). Pri obyčajných sekvenčných vzoroch táto informácia chýba.

Click-stream stromy

Myšlienka vytvárania tzv. click-stream stromov, prezentovaná v [12], sa zameriava na organizáciu prístupov používateľov do takých dátových štruktúr, aby bolo možné efektívne predpovedať (a teda napríklad aj odporúčať), aké stránky používateľ navštívi v budúcnosti. Na rozdiel od predchádzajúcich prístupov, ktoré pracovali iba s postupnosťou stránok, pri

[4] Niekedy sa táto technika označuje aj ako hľadanie „prechodových vzorov“ (angl. traversal patterns).

tejto metóde sa berie do úvahy aj čas, ktorý používateľ strávil prezeraním jednotlivých stránok.

Pri vytváraní click-stream stromov sa najprv prístupy používateľov rozdelia do skupín na základe vzájomnej podobnosti. Pri vyhodnocovaní podobnosti prístupov sa pritom berie do úvahy miera zhody jednotlivých stránok v prístupoch ako aj čas, ktorý používateľ na nich strávil. Prístupy v každej skupine sa potom zlúčia do špeciálnej štruktúry – click-stream stromu.

Zhlukovanie

Cieľom zhlukovania (angl. clustering) je nájdenie zhlukov (tried) objektov na základe ich podobnosti v niektorých atribútoch. Pri objavovaní znalostí vo webových systémoch sa metóda zhlukovania často používa na vyhľadávanie charakteristických skupín prístupov používateľov [12] alebo na zhlukovanie už nájdených vzorov [16]. Problém zhlukovania je intenzívne skúmaná oblasť, o čom svedčí aj veľké množstvo prístupov a algoritmov [5]. Vzhľadom na povahu dát, s ktorými sa pri webových systémoch pracuje (ide v podstate o vektory s veľkou dimenziou, ktorých prvkami sú navštívené stránky) sa v tejto oblasti často používajú algoritmy na rozdeľovanie hypergrafov.

3.3 Využitie objavených znalostí

Existujú dve hlavné oblasti aplikácie znalostí o správaní sa používateľov objavených v záznamoch o používaní AH systému, resp. webovej aplikácie:

1. využitie autorom obsahu systému
2. využitie na personalizáciu systému – odporúčanie

Pri prvom prístupe sa odhalené znalosti (zvyčajne asociačné pravidlá alebo sekvenčné vzory) prezentujú autorovi obsahu systému. Po ich preštudovaní sa autor môže rozhodnúť, aké zásahy má vykonať do obsahu systému, aby uľahčil používateľom prácu so systémom. Typickým príkladom takéhoto zásahu môže byť napríklad pridanie chýbajúceho spojenia medzi konceptmi v prípade, že používatelia považovali tieto koncepty za príbuzné a často sa medzi nimi pohybovali. Ďalšou ukážkou môže byť revízia obsahu informačného fragmentu, ak sa k nemu používatelia opakovane vracali alebo pri jeho štúdiu strávili viac času ako pri ostatných fragmentoch.

Príkladom systému, ktorý využíva prezentáciu objavených znalostí je systém EPRules (Educational Prediction Rules) [22]. Jeho autori použili ako zdroj údajov upravený systém AHA! [3, 10]. Na objavovanie predikčných pravidiel bol použitý evolučný algoritmus. V porovnaní s algoritmom Apriori [1] bol evolučný algoritmus síce pomalší a objavil menej pravidiel, táto vlastnosť je však pri prezentácii pravidiel skôr výhodou, nakoľko autor nie je zahltený veľkým množstvom pravidiel. Autori systému taktiež navrhli metódu, ktorá umožňuje postupné skvalitňovanie AH systému, resp. jeho obsahu. Metóda definuje štyri kroky:

1. *Vytvorenie kurzu* v AH systéme (často za pomoci autorského nástroja)
2. *Nasadenie kurzu*, študenti používajú systém a ten si robí záznamy o ich činnosti
3. *Objavovanie predikčných pravidiel*
4. *Zlepšovanie kurzu* na základe prezentovaných pravidiel (opäť pomocou autorského nástroja)

Druhým spôsobom využitia objavených znalostí je personalizácia – odporúčanie relevantných stránok (resp. konceptov) používateľovi počas toho, ako pracuje so systémom. Na odporúčanie, ktoré stránky má používateľ navštíviť v ďalších krokoch, sa používa aktuálny prístup používateľa do systému, teda postupnosť navštívených stránok, ktoré navštívil od posledného prihlásenia do systému. Súčasťou odporúčania je aj ohodnotenie vhodnosti jednotlivých stránok. Takéto odporúčanie môžeme považovať za určitú formu tzv. kolaboratívneho filtrovania (angl. collaborative filtering). Pri kolaboratívnom filtrovaní sa používateľovi odporúčajú produkty (ale aj služby alebo webové stránky) na základe preferencií ostatných používateľov, ktorí majú podobné záujmy.

Ako ukážku systému na odporúčanie stránok používateľom uvedieme systém WebPersonalizer [16]. Na odporúčanie stránok používa znalosti objavené v záznamoch webového servera. Znalosti majú podobu rôznych profilov používania systému, pričom autori použili dva prístupy na ich získanie. V prvom sa používateľské prístupy zoskupia na základe príbuznosti a pre každú skupinu sa vypočíta samostatný profil, ktorý ju reprezentuje. V druhom prístupe sa ako profily používania použijú asociačné pravidlá. Na zistenie relevantných stránok sa používa iba najaktuálnejšia časť prístupu používateľa – tzv. okno. Na základe obsahu okna sa vyberú najvhodnejšie profily, ktoré sa použijú na odporúčanie stránok.

4 Návrh procesu objavovania a využitia znalostí vo výučbových AH systémoch

Pri návrhu procesu pre objavovanie znalostí v AH systémoch a ich využitie sme vychádzali zo všeobecného procesu pre objavovanie znalostí [21]. Nami navrhnutý proces má nasledujúce kroky:

1. Zber dát
2. Predspracovanie dát
3. Dolovanie v dátach
4. Interpretácia znalostí, resp. ich využitie

Hoci podľa [21] sa pri vykonávaní jednotlivých krokoch objavovania znalostí odporúča asistencia človeka, našim cieľom bolo tento proces v čo najväčšej miere zautomatizovať. Dôvodom tohto rozhodnutia bol fakt, že pri nami navrhovanom procese nebudú väčšinou prítomní odborníci na objavovanie znalostí, ktorí poznajú vlastnosti a použitých postupov a algoritmov a vedia vhodnými zásahmi ovplyvniť kvalitu výstupov. Práve naopak, predpokladáme, že často jedinými osobami, ktoré budú tento proces využívať budú autori obsahu AH systému (pri výučbových AH systémoch najčastejšie učitelia) bez hlbších znalostí o objavovaní znalostí. Preto jediným krokom procesu, kde sa (čiastočne) vyžaduje zásah človeka, je až interpretácia objavených znalostí.

4.1 Zber dát

Zaznamenávanie údajov o práci používateľov so systémom vykonáva samotný AH systém, prípadne webový server, na ktorom je systém nasadený. Dáta sú najčastejšie ukladané do databázy systému alebo do externých súborov. V nasledujúcich kapitolách uvedieme údaje pre proces objavovania znalostí v AH systémoch, ktoré sme vybrali na základe štúdia výučbových AH systémov, ako aj techník dolovania v dátach.

Údaje o akciách používateľov v systéme

Ide o najdôležitejšiu časť vstupných údajov, nakoľko poskytujú základnú predstavu o správaní sa používateľov v systéme. Pre každú akciu používateľa v systéme navrhujeme zaznamenávať tieto údaje:

- čas, kedy akcia nastala

  Čas výskytu akcie je dôležitý pre výpočet času, ktorý používateľ strávil na určitom koncepte. Takisto definuje poradie akcií.

- identifikácia používateľa

  Ide o jednoznačnú identifikáciu používateľa v systéme, slúži na zoskupenie akcií jedného používateľa.

- druh akcie

  Na základe akcií v analyzovaných AH systémoch sme identifikovali štyri druhy elementárnych akcií v systéme. V prvom rade sú to akcie *Login* a *Logout*. Tieto akcie slúžia na ohraničenie jednotlivých vzdelávacích aktivít používateľov systéme. Ďalej sú to akcie *ConceptVisit* a *FragmentDisplay*, ktoré nám hovoria o tom, ktoré koncepty študent navštívil, a ktoré fragmenty mu pri tom boli zobrazené.

- objekt

  Objekt systému, ktorého sa akcia týkala. V prípade akcií *ConceptVisit*, resp. *FragmentDisplay* je to identifikácia konceptu, resp. fragmentu.

Údaje o typoch konceptov a fragmentov

Tieto údaje nám umožňujú pracovať so záznamami o používaní systému na vyššej úrovni – na úrovni typov konceptov a fragmentov. Takto môžeme odhaliť všeobecnejšie vzory správania sa, ktoré sa neviažu na konkrétne koncepty ani fragmenty. Rôzne typy výučbových objektov definuje štandard LOM IEEE [13] pre dátový element 5.2 Learning Resource Type. Vzhľadom k tomu, že sa v oblasti AH systémov zameriavame na výučbu programovania (ktorá je špecifická), rozhodli sme sa typy konceptov a fragmentov prevziať zo systému ALEA [14, 15], ktorý bol vytvorený špeciálne na tento účel. V zátvorke uvádzame možný ekvivalent podľa štandardu LOM IEEE (ak existuje):

Typy konceptov:
- Text (lecture)
- Programová schéma
- Príklad (exercise)

Typy fragmentov:
- Text (narrative text)
- Zadanie príkladu (problem statement)
- Pomôcka pri riešení
- Riešenie príkladu

Údaje o štruktúre domény

Podobne ako pri typoch konceptov a fragmentov, aj táto informácia nám umožní zmeniť pohľad na zaznamenané údaje. Predpokladáme pri tom, že koncepty sú v doméne usporiadané hierarchicky v stromovej štruktúre. Vďaka stromovej štruktúre sme schopní namapovať návštevy konceptov na nižšej úrovni na im nadradené koncepty (v zmysle pozície v hierarchii domény) a odhaľovať tak vzory správania sa na zvolenej úrovni konceptov.

Údaje o úspešnosti študentov

Pod úspešnosťou študentov rozumieme úroveň ich znalostí o doméne. Tento údaj je vo väčšine výučbových AH systémov súčasťou modelu používateľa, preto ho nie je potrebné dopĺňať. Na zistenie úrovne znalostí študentov môžeme použiť aj výsledky testov. Výsledky môžu pochádzať buď z testov vykonaných priamo v systéme (pokiaľ ich systém zaznamenáva) alebo môže ísť o výsledky „papierových" testov, ktoré dodá vyučujúci predmetu. Výsledky testov vykonaných priamo v AH systéme sú zvyčajne zachytené v modeli používateľa, či už priamo ako samostatné atribúty, alebo nepriamo ako úroveň znalostí o koncepte, ktorá sa na základe výsledkov vypočítala.

Výsledky sa využijú pri ohodnocovaní a filtrovaní vzdelávacích aktivít študentov. Zaujímavé sú pritom ako celkové výsledky testov (predstavujú celkové ohodnotenie študenta, resp. jeho znalostí o doméne), tak aj výsledky jednotlivých otázok testov.

4.2 Predspracovanie dát

Úlohou predspracovania je úprava údajov do podoby vhodnej pre dolovanie v dátach. Pre spracovanie údajov zo záznamov výučbového webového systému Zaïane [24] navrhuje vykonať tieto kroky:

1. Odstrániť irelevantné záznamy (šum)
2. Identifikovať prístupy do systému
3. Namapovať prístupy na vzdelávacie aktivity
4. Doplniť chýbajúce záznamy
5. Zoskupiť vzdelávacie aktivity podľa používateľov
6. Integrovať do získaných údajov externé dáta, napr. hodnotenie študentov

Tento postup je vhodný na predspracovanie záznamov z webového servera. Kroky 2. a 3. sa však dajú vynechať vtedy, keď máme k dispozícii údaje priamo z výučbového AH systému. V takom prípade totiž nie je potrebné mapovať stránky systému na jeho koncepty, toto mapovanie je už definované v modeli domény.

Výstupom predspracovania sú údaje o každom prístupe do systému: postupnosť konceptov navštívených pri prístupe, čas strávený na jednotlivých konceptoch, informácie o používateľovi, ktorý prístup vykonal. Po identifikovaní prístupov vykonáme aj ich filtrovanie, napríklad na základe ich dĺžky (krátke prístupy považujeme za šum) alebo na základe výsledkov študenta, ktorý prístup vykonal. Samozrejme, aby bolo možné efektívne aplikovať techniky dolovania v dátach, je potrebné niektoré hodnoty (napr. čas strávený na konceptoch) diskretizovať alebo rozdeliť do kategórií.

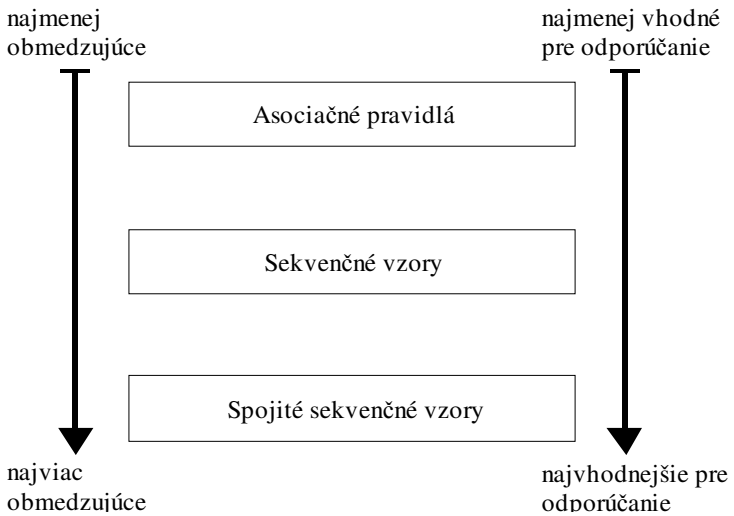
4.3 Dolovanie v dátach

V tomto kroku sa v predspracovaných údajoch objavujú znalosti o správaní sa používateľov aplikáciou postupov na dolovanie v dátach. Vzhľadom na povahu údajov, ktoré nám AH systémy poskytujú (sú to v podstate postupnosti navštívených konceptov), použijeme techniky na objavovanie asociačných pravidiel, sekvenčných vzorov a súvislých sekvenčných vzorov. Tieto techniky boli už úspešne aplikované v podobnej oblasti – v oblasti analýzy záznamov o používaní webových sídiel [16].

Hľadajú sa teda asociačné pravidlá, sekvenčné a súvislé sekvenčné vzory – štýly učenia sa. Pri objavovaní znalostí sa využijú aj údaje o typoch konceptov a štruktúre domény na interpretáciu akcií používateľov na rôznej úrovni. Výsledkom sú potom všeobecnejšie vzory správania sa. Nájdené znalosti sa uložia a budú slúžiť na skvalitnenie adaptácie AH systému používateľovi, napr. odporúčaním relevantných odkazov pre študentov.

Asociačné pravidlá, sekvenčné vzory aj súvislé sekvenčné vzory zdieľajú niekoľko spoločných znakov. Všetky sú reprezentované postupnosťami alebo množinami konceptov. Takisto algoritmy na ich objavovanie sú založené na rovnakej myšlienke generovania a orezávania množiny kandidátov na vzory. Tieto spoločné črty nám umožňujú uložiť všetky vzory v spoločnej báze znalostí a pracovať s nimi takmer rovnako. Ďalej môžeme zovšeobecniť algoritmy na objavovanie týchto vzorov a vytvoriť tak rámec, ktorý bude obsahovať spoločné časti algoritmov.

Aj napriek podobným črtám reprezentácie a spôsobu objavovania jednotlivých druhov vzorov je treba poznamenať, že samotné druhy vzorov majú rôzne vlastnosti. Práve tieto vlastnosti (napr. či a ako berú do úvahy poradie prvkov) ovplyvňujú vhodnosť jednotlivých druhov vzorov na odporúčanie konceptov (pozri obrázok 1).



Obrázok 1: Vhodnosť jednotlivých druhov vzorov na odporúčanie.

4.4 Interpretácia znalostí

Vo fáze interpretácie dochádza praktickému využitiu objavených znalostí s cieľom skvalitniť adaptáciu AH systému používateľovi. Navrhujeme použiť dva spôsoby využitia znalostí:

1. Využitie autorom obsahu AH systému
2. Využitie na automatickú úpravu špecifikácie prispôsobovania

Využitie autorom obsahu

Pri prvom spôsobe interpretácie sú znalosti prezentované autorovi obsahu AH systému, ktorý ich vyhodnotí a rozhodne sa, aké zásahy do obsahu systému vykoná. Môže upraviť štruktúru domény (napr. pridaním alebo odobratím odkazov medzi konceptmi), ale aj samotnú špecifikáciu prispôsobovania. V tomto prípade teda objavené znalosti slúžia ako určitá spätná väzba pre autora obsahu, ktorá mu hovorí o tom, ako používatelia systém využívali, v akom poradí študovali jednotlivé koncepty.

Automatická úprava špecifikácie prispôsobovania

Špecifikácia prispôsobovania určuje spôsob, akým sa má AH systém prispôsobovať používateľovi. Veľmi často je špecifikácia zadefinovaná pomocou pravidiel. Vyhodnotením pravidiel sa potom uskutočňuje samotné prispôsobovanie. Pravidlá môžu napríklad kontrolovať, či má používateľ dostatočné vedomosti na to, aby mu mohol byť prezentovaný určitý koncept.

Do špecifikácie prispôsobovania však patria aj pravidlá na vedenie používateľa v systéme. Ukazuje sa, že práve odporúčaním vhodného poradia konceptov môžeme študentovi pomôcť pri štúdiu domény. Znalosti získané z údajov o používaní systému nám hovoria, aké poradie konceptov (ale aj informačných fragmentov) uprednostňujú jednotlivé skupiny študentov. Je zrejmé, že takéto znalosti sú vhodným základom pre odporúčanie relevantných konceptov. Navrhujeme dva prístupy k úprave špecifikácie prispôsobovania sa na základe objavených asociačných pravidiel a sekvenčných vzorov.

Off–line prístup. Pri tomto prístupe sa používatelia rozdelia do skupín podľa objavených štýlov učenia (teda vzorov, ktoré preferujú) a táto informácia sa uloží do ich modelov. Pre každý zo štýlov učenia sa potom vygeneruje samostatná stratégia odporúčania konceptov – sada pravidiel prispôsobovania sa pre AH systém. Nevýhodou takéhoto prístupu je, že pre nových používateľov systém nebude mať dostatok informácií na to, aby z nich mohol zistiť štýl učenia sa. Samotné štýly učenia je však možné odhaliť aj v historických údajoch (v údajoch bývalých používateľov systému). Ako ďalšia nevýhoda sa javí statická podstata tohto prístupu. Pravidlá pre prispôsobovanie sú vygenerované a počas používania systému sa nemenia. K úprave pravidiel dôjde až v ďalšom cykle odhaľovania znalostí.

On–line prístup. Druhým prístupom je generovanie postupnosti odporúčaných konceptov na požiadanie. Vždy, keď sa používateľ presunie na ďalší koncept, vygeneruje sa nová postupnosť, ktorá sa prezentuje používateľovi. Súčasťou odporúčania je aj ohodnotenie

relevantnosti konceptov, na základe ktorého sú koncepty v odporúčaní usporiadané. Na odporúčanie sa použijú všetky tri druhy objavených vzorov – asociačné pravidlá, sekvenčné vzory a spojité sekvenčné vzory, pričom použijeme postup uvedený v [19].

Keďže je však vhodnosť vzorov na odporúčanie rôzna, treba tento fakt pri vytváraní výsledného odporúčania zohľadniť. Navrhli sme preto prístup, pri ktorom sa hodnota relevantnosti odporúčaní vytvorených na základe jedného druhu vzorov násobí koeficientom, ktorý odráža vhodnosť daných vzorov na odporúčanie. Týmto docielime, že pri výslednom odporúčaní, ktoré vznikne spojením čiastkových odporúčaní, budú mať väčšiu váhu (a teda budú uprednostnené) odporúčania vytvorené na základe vhodnejších vzorov.

5 Systém na odporúčanie navigácie

V tejto časti predstavíme systém na odporúčanie navigácie, navrhnutý v rámci projektu prezentovaného v tejto práci. Systém umožňuje obohatiť AH systémy o možnosť odporúčania konceptov, pričom využíva a automatizuje proces objavovania a využitia znalostí navrhnutý v predchádzajúcej časti práce. Postupne predstavíme scenáre použitia systému, algoritmy, ktoré využíva, ako aj jeho architektúru. V závere časti opíšeme aj implementáciu prototypu systému.

5.1 Scenáre použitia systému

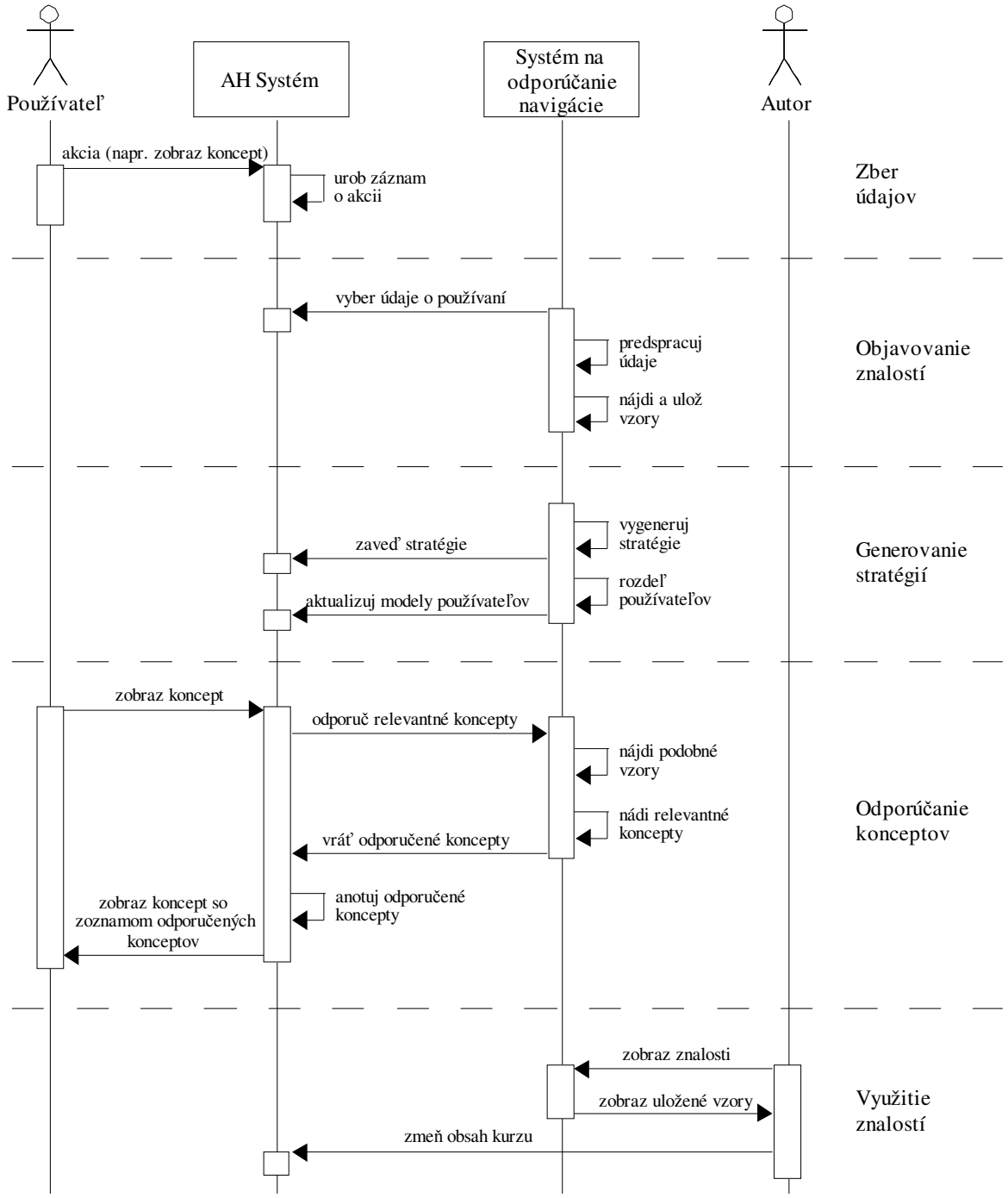
S ohľadom na navrhnutý proces objavovania a využitia znalostí sme identifikovali štyri základné činnosti, ktoré systém zabezpečuje:

1. Objavovanie znalostí v údajoch z AH systému
2. Generovanie stratégií prispôsobovania
3. Odporúčanie relevantných konceptov
4. Prezentácia znalostí

Typické scenáre použitia systému sú znázornené pomocou sekvenčného diagramu na obrázku 2. Používatelia pracujú s AH systémom, študujú jeho koncepty. Popri tom si AH systém vytvára záznamy o každej ich akcii. Zaznamenáva, aké koncepty používatelia navštívili a aké fragmenty sa im pri tom zobrazili. V určitý naplánovaný čas[5] sa vykoná prenos údajov zozbieraných za posledné obdobie do systému na odporúčanie navigácie. Po ich úspešnom predspracovaní sa dáta o správaní sa používateľov uložia do databázy systému. Systém vykoná nad údajmi v databáze algoritmy na odhaľovanie vzorov správania sa študentov a nájdené vzory uloží do bázy znalostí.

V ďalšej fáze sa používatelia rozdelia do skupín podľa ich vzorov správania sa. Nájdené vzory sa transformujú do rôznych stratégií prispôsobovania (ak to použitý AH systém podporuje) a zapíšu sa späť do AH systému. Pri ďalšom používaní AH systému sa študentom budú zobrazovať odporúčané koncepty podľa vygenerovaných stratégií prispôsobovania. Výber stratégie pre konkrétneho študenta bude závisieť od toho, do ktorej skupiny bol zaradený.

[5] Frekvencia prenosu údajov samozrejme závisí od intenzity používania AH systému, pre často používané systémy to môže byť aj niekokrát denne.



Obrázok 2: Typické scenáre použitia systému.

Ak má systém vytvorenú bázu znalostí, výučbový AH systém ho môže požiadať aj o dynamické vytvorenie množiny odporúčaných konceptov. Poskytne systému identifikáciu používateľa spolu s jeho aktuálnym prístupom – postupnosťou akcií, ktoré používateľ vykonal od posledného prihlásenia do systému. Systém na základe aktuálneho prístupu študenta nájde príbuzné vzory správania sa v báze znalostí a vytvorí z nich postupnosť odporúčaných konceptov, ktorú spolu s ohodnotením vhodnosti jednotlivých konceptov vráti výučbovému AH systému. Ten zobrazí používateľovi odkazy na odporúčané koncepty, pričom ich môže usporiadať a anotovať podľa ohodnotenia vhodnosti, resp. relevantnosti odporúčaných

konceptov. Takisto môže odporúčanie využiť na vedenie používateľa, napríklad pomocou tlačidla „Ďalej“.

Poslednou možnosťou použitia systému je preskúmanie objavených znalostí autorom obsahu systému. Ten si nechá zobraziť objavené znalosti získané z údajov o používaní AH systému. Podľa nich potom naplánuje a pomocou autorského nástroja vykoná úpravy obsahu AH systému. Pokiaľ sa výučbový AH systém používal už viac období, autor môže porovnať znalosti získané z rôznych vyučovacích období a vyhodnotiť tak úspešnosť svojich predchádzajúcich zásahov do obsahu systému.

5.2 Návrh algoritmov

Spracovanie a transformácia údajov aj znalostí v systéme vyžaduje použiť niekoľko algoritmov. V prvom rade ide o spôsob predspracovania údajov z AH systému. Ďalej sú to algoritmy na dolovanie v zozbieraných údajoch, zamerané na objavenie asociačných pravidiel, sekvenčných vzorov a spojitých sekvenčných vzorov. Nakoniec sú to algoritmy na odporúčanie konceptov na základe rôznych druhov objavených vzorov, ako aj postup na zostavenie finálneho odporúčania.

Predspracovanie údajov o používaní systému

Algoritmus na predspracovanie údajov má za úlohu identifikovať prístupy používateľov do systému a odhaliť prípadné chyby a nekonzistencie v údajoch. Jeho vstupom je postupnosť akcií všetkých používateľov usporiadaná podľa času ich výskytu. Výstupom je množina prístupov študentov – postupností navštívených konceptov, resp. zobrazených fragmentov pre jednotlivých používateľov.

Na ohraničenie prístupov používateľov do systému algoritmus používa akcie *Login* a *Logout*. Všetky akcie používateľa, ktoré sa vyskytujú v čase medzi týmito dvomi akciami, budú považované za jeden prístup. Myšlienka algoritmu spočíva v „simulácii“ prihlásenia používateľov do systému. Akcie na vstupe sa spracovávajú v tom poradí, v akom nastali v systéme, pričom sa vytvárajú záznamy o aktuálne prihlásených používateľoch, do ktorých sa postupne ukladajú akcie prihláseného používateľa. Každá akcia je spracovaná nasledujúcim spôsobom:

1. Ak je to akcia *Login*, vytvorí sa nový záznam o používateľovi.
2. Ak je to akcia *Logout*, zoznam akcií zo záznamu používateľa sa uloží do výstupnej množiny prístupov používateľov.
3. Akákoľvek iná akcia sa pridá do zoznamu akcií príslušného používateľa a zároveň sa vypočíta čas trvania akcie (napr. návštevy konceptu) na základe času jej výskytu a času výskytu nasledujúcej akcie.

Formálnejší opis algoritmu, ktorý ošetruje aj výnimočné situácie, ktoré môžu nastať, uvádzame v prílohe C.

Dolovanie znalostí – vzorov správania sa

Existujúce algoritmy na hľadanie vzorov (asociačných pravidiel [1], sekvenčných vzorov [2] a spojitých sekvenčných vzorov [9]) sú navzájom veľmi podobné a v podstate sa líšia iba rôznou interpretáciou vstupných údajov. Vstupom algoritmu je postupnosť transakcií používateľov. Za transakciu v našom prípade považujeme postupnosť konceptov (resp. fragmentov), ktoré používateľ navštívil v jednom prístupe do systému. Výstupom sú často sa vyskytujúce postupnosti (resp. množiny) konceptov, fragmentov.

Pred objavovaním spojitých sekvenčných vzorov sa podľa [9] najprv v záznamoch majú vyhľadať postupnosti nazývané *„maximal forward references"*, ktoré vzniknú zanedbaním návratov používateľa k predošlým stránkam a až v týchto postupnostiach sa vyhľadávajú. Pri prvých experimentoch s algoritmom sme však zistili, že takýmto predspracovaním strácame príliš veľké množstvo údajov, a preto sme sa rozhodli tento krok algoritmu vynechať. Myslíme si, že v kontexte výučbových AH systémov návrat k predošlému konceptu je významný (napr. mal študent potrebu oživiť si vedomosti z daného konceptu), nehovoriac o tom, že pri návrate už môže byť daný koncept prezentovaný odlišným spôsobom ako dôsledok prispôsobovania sa používateľovi.

Algoritmy na hľadanie všetkých uvedených druhov vzorov majú rovnaký základ. V každej iterácii algoritmu sa vykonajú dva kroky:

1. *vygenerovanie kandidátov* na vzory
2. *redukcia kandidátov* na základe ich podpory v databáze

Výsledkom jednej iterácie sú vzory dĺžky k [6], pričom prvá iterácia algoritmu vygeneruje vzory dĺžky 2. Kandidáti na vzory dĺžky k sa generujú na základe už nájdených vzorov z predchádzajúcej iterácie (majú teda dĺžku $k-1$). Pri redukcii množiny kandidátov sa prehľadáva databáza transakcií a zisťuje sa, či sa vzor nachádza (má podporu) v dostatočnom množstve transakcií. Minimálna podpora potrebná na akceptovanie vzoru je parametrom algoritmu. Algoritmus končí, keď je množina vygenerovaných kandidátov na vzory prázdna. Formálny zápis kostry algoritmu je uvedený v prílohe C.

V uvedenej kostre algoritmu sú dve miesta, ktoré sú špecifické pre hľadanie rôznych druhov vzorov. Je to spôsob generovania kandidátov na vzory a spôsob zisťovania ich podpory. Pre jednotlivé druhy vzorov sa kandidáti vytvárajú nasledujúcimi spôsobmi:

- *Asociačné pravidlá:* Kandidáti sa generujú operáciou množinového zjednotenia, pričom spojenie dvoch množín s počtom prvkov k je možné iba vtedy, ak majú spoločných práve $(k-1)$ prvkov.

[6] Za dĺžku vzoru budeme považovať počet konceptov v ňom obsiahnutých.

- *Sekvenčné vzory:* Kandidáti vzniknú spojením dvoch postupností, ktoré majú spoločných prvých *(k-1)* prvkov, takým spôsobom, že za prvých *(k-1)* prvkov sa pripoja posledné, *k*-te, prvky spájaných postupností.
- *Spojité sekvenčné vzory:* Noví kandidáti sa generujú spájaním postupností, pričom postupnosti dĺžky *k* sa musia prekrývať (zhodovať) v *(k-1)* po sebe nasledujúcich prvkoch.

Na to, aby mal vzor podporu v jednej transakcii (t.j. prístupe používateľa), musia platiť tieto podmienky:

- *Asociačné pravidlá:* Vzor má podporu v transakcii vtedy, ak je jej podmnožinou.
- *Sekvenčné vzory:* Vzor je podporovaný transakciou vtedy, keď obsahuje koncepty v tom istom poradí, v akom sa vyskytujú v transakcii.
- *Spojité sekvenčné vzory:* Spojitý sekvenčný vzor má podporu v transakcii, ak je jej podpostupnosťou.

Aby sme umožnili efektívnejšie prehľadávanie nájdených vzorov pri odporúčaní a zároveň ušetrili pamäťový priestor, nájdené vzory zlúčime do stromovej štruktúry tak, že každá cesta v strome (od koreňa po ľubovoľný uzol stromu) predstavuje jeden vzor. Uzly stromu majú meno, ktoré je zhodné s menom konceptu a ohodnotenie, ktoré je zhodné s podporou vzoru, ktorý končí v danom uzle. Vzory jednotlivých druhov sú zlúčené do samostatných stromov. Nakoniec teda vzniknú tri stromy, pričom každý z nich predstavuje objavené vzory jedného druhu.

Odporúčanie konceptov na základe vzorov

Úlohou algoritmu na odporúčanie je na základe objavených vzorov správania sa a aktuálneho prístupu používateľa do systému vytvoriť postupnosť konceptov, ktoré by mal používateľ navštíviť. Výstupom je teda postupnosť odporúčaných konceptov spolu s ohodnotením ich relevantnosti. Na odporúčanie konceptov z jednotlivých druhov vzorov použijeme prístup uvedený v [19] a [20].

Na odporúčanie sa využíva iba posledná, najnovšia časť aktuálneho prístupu používateľa. Táto časť sa niekedy zvykne označovať aj ako okno. Veľkosť okna (označíme ju *w*) sa zvyčajne volí ako priemerná dĺžka prístupov používateľov do systému. Znamená to, že na odporúčanie bude mať vplyv iba *w* posledných akcií používateľa.

Algoritmus postupne prehľadáva všetky vzory dĺžky *(w+1)* a snaží sa nájsť tie vzory, ktoré sú podobné obsahu okna. Kritériá, kedy je vzor podobný oknu sú pre jednotlivé druhy vzorov nasledovné:

- *Asociačné pravidlá:* Vzor je podobný oknu, ak je okno jeho podmnožinou.
- *(Spojité) sekvenčné vzory:* Vzor je podobný oknu, ak je okno jeho prefixom.

Zo všetkých vzorov, ktoré sú podobné oknu, sa odstránia koncepty, ktoré sa nachádzajú aj v okne. Zvyšný koncept, ktorý ostane vo vzore, je odporúčaný. Ako hodnota relevantnosti odporúčaného konceptu sa použije hodnota spoľahlivosti (angl. confidence) pravidla: <*koncepty okna*> => <*odporučený koncept*>.

V prípade, že sa nepodarí vytvoriť odporúčanie pre veľkosť okna *w*, hľadanie pokračuje s veľkosťou okna *(w-1)*. Algoritmus končí, keď sa podarí vytvoriť odporúčanie, alebo je veľkosť okna nulová.

Uvedeným algoritmom získame odporúčania vytvorené z rôznych druhov vzorov. Ako sme uviedli v kapitole 4.3, jednotlivé vzory majú rôzne vlastnosti a preto aj rôznu vhodnosť na odporúčanie. Vhodnosť vzorov, z ktorých odporúčania vznikli, zohľadníme v hodnote relevantnosti odporúčaných konceptov. Relevantnosť konceptov odporučených z jedného druhu vzorov prenásobíme konštantou, ktoré vyjadruje vhodnosť druhu vzorov (čím je vhodnosť väčšia, tým je väčšia aj konštanta). Týmto spôsobom uprednostníme odporúčania zo vzorov, ktoré sú na odporúčanie vhodnejšie.

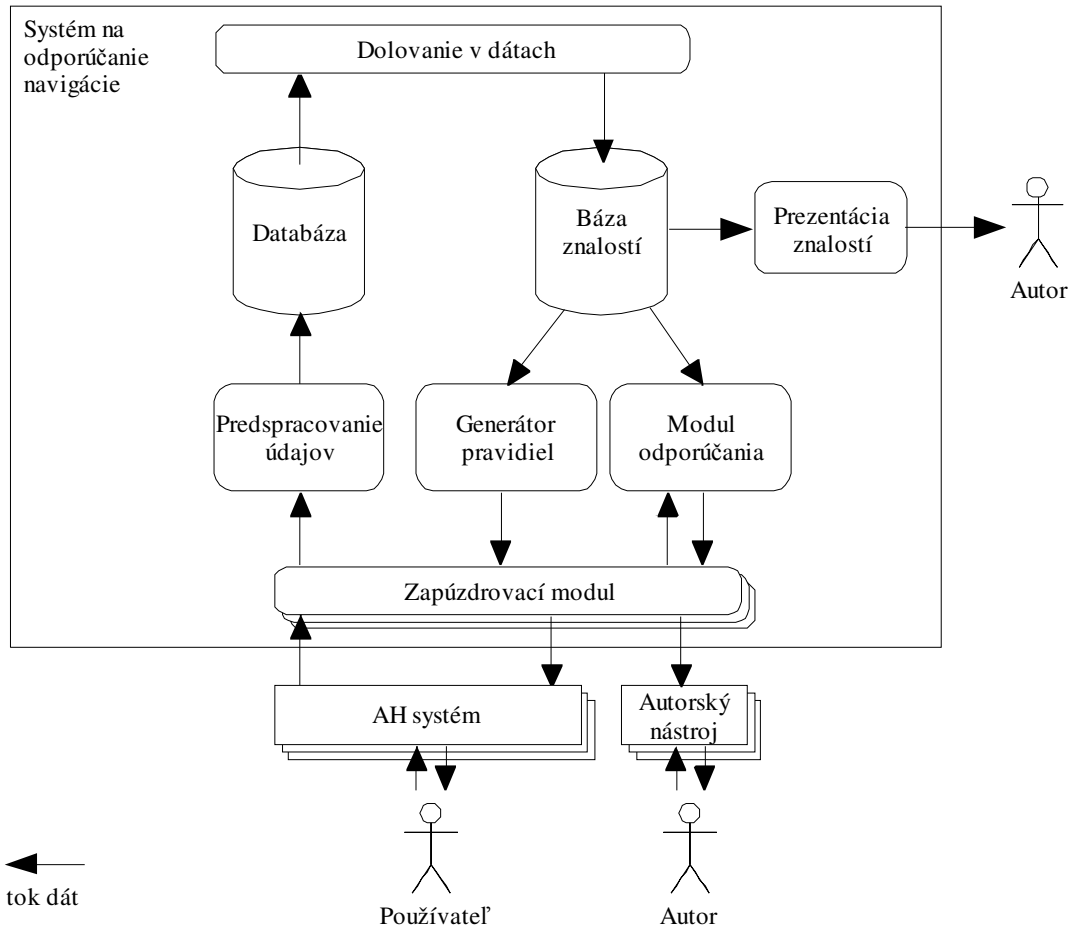
Na záver je potrebné zlúčiť odporúčania na základe jednotlivých vzorov a vytvoriť tak finálne odporúčanie. Na zlúčenie použijeme operáciu množinového zjednotenia. V prípade, že sa jeden koncept nachádza v oboch zlučovaných množinách, sa hodnota jeho relevantnosti z oboch odporúčaní sčíta. Po zlúčení všetkých čiastkových odporúčaní normalizujeme hodnotu relevantnosti konceptov, pričom hodnotu 1.0 budú mať tie koncepty, ktoré boli odporučené na základe všetkých druhov vzorov. Usporiadaním prvkov výslednej množiny vznikne postupnosť konceptov, ktorá predstavuje výsledok odporúčania.

5.3 Architektúra systému

Pri návrhu architektúry systému sme vychádzali z navrhnutého procesu pre objavovanie znalostí a procesu pre využitie týchto znalostí – odporúčanie navigácie. Moduly systému a ich vzájomné prepojenie dátovými tokmi sú znázornené na obrázku 3. V ďalších kapitolách vysvetlíme význam jednotlivých častí architektúry.

Zapúzdrovací modul (wrapper)

Ako je vidieť z obrázku architektúry, predpokladáme využitie systému viacerými a rôznymi AH systémami. Z tohto dôvodu sme v architektúre navrhli zapúzdrovací modul, ktorý oddeľuje konkrétny AH systém od častí systému, ktoré sú od použitého AH systému nezávislé. Úlohou modulu je vytvoriť jednotné rozhranie pre prístup k dátovým štruktúram výučbového AH systému. Funkcionalita a spracovanie údajov špecifické pre konkrétny AH systém sú teda sústredené do jedného modulu. Toto nám v budúcnosti umožní jednoduchšie rozšíriť systém o podporu ďalších AH systémov. Modul môžeme rozdeliť na dve časti – vstupnú a výstupnú:



Obrázok 3: Architektúra systému.

- *Vstupná časť* zabezpečuje načítanie všetkých potrebných údajov z AH systému. Sem patria predovšetkým akcie používateľov v systéme, ale aj údaje o štruktúre domény, či typoch konceptov a fragmentov.
- *Výstupná časť* modulu poskytuje prostriedky na úpravu modelu používateľa, zavedenie nových pravidiel pre prispôsobovanie, ako aj spracovanie množiny odporúčaných konceptov do formátu vhodného pre AH systém.

Predspracovanie údajov

Modul má za úlohu naplniť databázu systému. Jeho vstupom je postupnosť akcií používateľov. Po predspracovaní sa akcie skontrolujú, aby sa odhalili prípadné chyby alebo nekonzistentnosti v údajoch. Identifikujú sa prístupy jednotlivých používateľov do systému, ktoré sa filtrujú na základe zadaných obmedzení na minimálnu dĺžku a/alebo úroveň úspešnosti príslušného študenta a uložia do databázy systému. Do databázy sa tiež uložia údaje o štruktúre domény AH systému a potrebné údaje o druhoch konceptov a fragmentov.

Databáza

V databáze systému sú uložené všetky údaje potrebné pre dolovanie znalostí. Nachádzajú sa tu jednak informácie o prístupoch používateľov do AH systému, ale aj údaje o štruktúre domény AH systému a druhoch konceptov a fragmentov. Modul poskytuje prostriedky na vytvorenie rôznych pohľadov na uložené údaje (napr. pohľad na úrovni druhov konceptov a fragmentov).

Dolovanie v dátach

Modul pre dolovanie v dátach zapúzdruje algoritmy na objavovanie vzorov správania sa. Aplikáciou algoritmov na údaje v databáze sa objavia vzory správania sa. Nájdené vzory ako aj rozdelenie používateľov podľa týchto vzorov modul uloží do bázy znalostí.

Báza znalostí

V báze znalostí sú uložené znalosti objavené v údajoch databázy. Obsahuje charakteristické vzory správania sa v AH systéme aj priradenie vzorov k jednotlivým používateľom systému. Bázu znalostí môžeme rozdeliť na dve časti – špecifickú a všeobecnú:

1. *Báza špecifických znalostí* obsahuje znalosti, ktoré sú špecifické pre jednotlivé AH systémy. Môže ísť napríklad o vzory správania sa, ktoré sa viažu na koncepty konkrétneho AH systému.

2. *Báza všeobecných znalostí* obsahuje znalosti, ktoré nie sú svojou povahou viazané na konkrétny AH systém, ako napríklad vzory správania sa na úrovni typov konceptov, resp. fragmentov.

Prezentácia znalostí

Modul slúži na prezentáciu znalostí z bázy znalostí vo forme zrozumiteľnej pre autora obsahu AH systému, napríklad vo forme tabuľky alebo grafu (resp. stromu), ktorý obsahuje objavené pravidlá správania sa. Vzhľadom na potenciálne veľké množstvo znalostí, je potrebné poskytnúť používateľovi prostriedky na ich vhodné filtrovanie.

Generátor pravidiel

Generátor transformuje objavené vzory správania sa do formy statických pravidiel pre prispôsobovanie (teda takých, pri ktorých sa nevyužíva aktuálny prístup používateľa do systému). Pre každý vzor správania v báze znalostí vytvorí vlastnú sadu pravidiel. Vytvorené pravidlá potom prostredníctvom zapúzdrovacieho modulu zapíše do AH systému. Zabezpečí tiež aktualizáciu modelov používateľov na základe identifikovaných vzorov správania sa jednotlivých študentov.

Modul odporúčania

Úlohou modulu je zostaviť postupnosť odporúčaných konceptov pre konkrétneho študenta na základe vzorov v báze znalostí. Vstupom pre vytvorenie postupnosti odporúčaných konceptov je informácia o aktuálnom prístupe používateľa do systému, ktorú poskytne zapúzdrovací

modul pre konkrétny AH systém. Odporúčanie sa vytvára na žiadosť AH systému, ktorý modul osloví vždy, keď potrebuje používateľovi odporučiť koncepty (napr. pri každom presune používateľa na nový koncept alebo po uplynutí určitého časového intervalu).

5.4 Prototyp systému

Aby sme overili navrhnutý proces objavovania a využitia znalostí, algoritmy a architektúru, vytvorili sme prototyp systému. Pri prototypovaní sme sa zamerali na odporúčanie konceptov pre používateľov, pričom sme ako zdroj dát použili systém ALEA, ktorý sme niekoľkými úpravami prispôsobili pre náš projekt. V tejto kapitole opíšeme jednotlivé časti prototypu a implementačné prostredie. Takisto zhrnieme úpravy systému ALEA.

Zapúzdrovacie moduly

Hlavným zdrojom testovacích údajov pre tento projekt bol systém ALEA, ktorý obsahuje dáta o používaní systému z predchádzajúcich troch rokov. Práve preto sme sa sústredili na implementáciu zapúzdrovacieho modulu práve pre tento systém. Modul implementuje rozhranie uvedené v prílohe E.

Údaje o používaní a časť modelu domény sú v systéme ALEA uložené v databázach MS Access. Ďalšie časť údajov o doméne sa nachádza v XML súbore. Na prístup k databázam systému ALEA modul využíva rozhranie JDBC, ktoré umožňuje vykonávať dotazy nad databázou v jazyku SQL. Na spracovanie údajov o doméne sme použili knižnicu Xerces.

Pri podrobnejšej analýze záznamov o používaní systému ALEA, ako aj postupov pri vytváraní týchto záznamov, sme prišli k záveru, že údaje o používaní systému nemusia byť kompletné a niektoré časti prístupov používateľov nebude možné spoľahlivo rekonštruovať. Modul preto implementuje algoritmus, ktorý sa snaží vo veľkej miere zrekonštruovať prístupy používateľov a upozorniť na možnosť prípadných chýbajúcich údajov. Počas projektu sme upravili spôsob zaznamenávania akcií v systéme ALEA a upravenú verziu systému sme nasadili do prevádzky. Pre upravenú verziu systému ALEA sme vytvorili aj novú verziu zapúzdrovacieho modulu, nakoľko po vykonaných úpravách už nie je potrebná rekonštrukcia údajov.

Pre prístup k zapúzdrovacím modulom je v systéme vytvorená trieda, ktorá slúži ako ich register. Pri spustení systému sa vytvoria zapúzdrovacie moduly pre všetky nakonfigurované AH systémy a uložia sa do tohto registra. Keď potom niektorá z tried potrebuje prístup k AH systému, jednoducho požiada register, aby vyhľadal zapúzdrovací modul pre daný systém. Každý AH systém je jednoznačne identifikovaný svojim menom, uvedeným v konfigurácii systému.

Predspracovanie údajov

Na spracovanie informácií poskytnutých zapúzdrovacím modulom a ich uloženie do databázy systému sme vytvorili triedu na predspracovanie údajov. V triede je implementovaný

navrhnutý algoritmus na identifikáciu prístupov používateľov do systému (pozri príloha C). Pre každý koncept alebo fragment, ktorý sa nachádza v prístupe používateľa, sa od zapúzdrovacieho modulu zisťuje jeho druh a tento údaj sa uloží do informačnej bázy. Pred uložením samotného prístupu do informačnej bázy sa ešte testuje jeho dĺžka. V prípade, že je dĺžka prístupu menšia ako nastavená hodnota (pri experimentoch so systémom sme použili nieľko rôznych hodnôt), prístup sa považuje nespoľahlivý a do databázy sa nezapíše.

Aby sme zefektívnili proces spracovania a zabránili prípadným duplicitným záznamom o prístupoch používateľov v databáze, modul na predspracovanie si od zapúzdrovacieho modulu vyžiada iba tie akcie, ktoré ešte doposiaľ nespracoval. Ide teda o tie akcie, ktorých časová známka je novšia ako časová známka poslednej akcie spracovanej systémom.

Databáza

V databáze systému sú uložené údaje o prístupoch používateľov do AH systémov ako aj údaje o štruktúre domény a druhoch konceptov a fragmentov. Je realizovaná ako relačná databáza (použili sme databázový systém MySQL), ktorej schéma je uvedená v prílohe F.

Prístup k údajom v databáze zabezpečuje samostatná trieda, ktorá poskytuje základné metódy pre prácu s uloženými údajmi. Sem patria funkcie pre zápis údajov do databázy a funkcie pre získanie údajov z databázy, v ktorých sa dajú pomocou parametrov špecifikovať kritériá, na základe ktorých sa majú prístupy používateľov z databázy vybrať.

Dolovanie v dátach

Implementovali sme základnú kostru pre algoritmy na objavovanie vzorov, z ktorej sú potom odvodené triedy, ktoré implementujú špecifické časti algoritmov pre objavovanie asociačných pravidiel, sekvenčných vzorov a spojitých sekvenčných vzorov.

Implementácia algoritmov umožňuje špecifikovať, aká časť prístupov používateľov v databáze sa má použiť na objavovanie vzorov. Toto nám potom umožňuje použiť zvyšnú časť prístupov na otestovanie presnosti odporúčaní konceptov na základe objavených vzorov a sledovať tak vplyv veľkosti pomeru trénovacej a testovacej množiny prístupov na výsledky a presnosť odporúčania (bližšie pozri kapitolu 6.4). Ďalším parametrom algoritmov je tzv. minimálna podpora (počet, resp. percento prístupov používateľov, v ktorých sa vzor vyskytuje), ktorú musia objavené vzory mať, aby boli považované za signifikantné a boli uložené do bázy znalostí.

Báza znalostí

Vzhľadom k tomu, že vzory sú po ich objavení zlúčené do stromovej štruktúry, rozhodli sme sa ukladať znalosti do súborového systému vo forme XML súborov (príklad takéhoto XML súbore je uvedený v prílohe G). Samotný formát XML má formu stromu a preto považujeme tento prístup za vhodnejší, ako ukladať znalosti do relačnej databázy. Navyše, XML súbor je dostatočne čitateľný pre človeka a je ho možné ľahko naformátovať a zobraziť pomocou dostupných nástrojov, takže aj bez existencie modulu pre vizualizáciu si môže autor obsahu AH systému nechať zobraziť objavené znalosti. Rôzne druhy znalostí (vzorov) sú uložené

v samostatných súboroch. V prototype sme sa nezaoberali štruktúrovaním bázy znalostí na špecifické a všeobecné znalosti. V báze sú v samostatných adresároch uložené iba znalosti špecifické pre jednotlivé AH systémy.

Prístup k znalostiam v báze zabezpečuje samostatná trieda, ktorá poskytuje metódy na načítanie aj zápis stromu vzorov určitého druhu do bázy. Na spracovanie a kontrolu XML súborov so znalosťami sme použili knižnicu Xerces.

Modul odporúčania

Podobne ako pri algoritmoch na objavovanie znalostí, aj pri odporúčaní konceptov sme vytvorili kostru, ktorá je spoločná pre odporúčanie na základe všetkých druhov znalostí. Z tejto kostry sú potom odvodené triedy, ktoré dopĺňajú časti špecifické pre odporúčanie na základe asociačných pravidiel, sekvenčných vzorov a spojitých sekvenčných vzorov.

Implementované algoritmy využíva trieda na odporúčanie, ktorá z ich výsledkov zostaví výsledné odporúčanie. V triede sú tiež implementované metódy na zisťovanie presnosti odporúčaní. Vplyv rôznych faktorov na presnosť odporúčaní bližšie skúmame v kapitole 6.4.

Rozhranie pre odporúčanie

Aby mohli AH systémy využívať služby nášho systému na odporúčanie navigácie, bolo treba vytvoriť rozhranie, ktoré bude slúžiť na získavanie odporúčaní. Nakoľko AH systémy môžu byť implementované v rôznych prostrediach a nasadené na rôznych serveroch, rozhodli sme sa na komunikáciu medzi systémami použiť protokol XML-RPC, ktorý bol vytvorený na vzdialené spúšťanie procedúr. Samozrejme, existujú aj ďalšie alternatívy na prepojenie systémov (napr. protokol SOAP), ale pri výbere XML-RPC prevážila jeho jednoduchosť a dostupnosť implementácií pre rôzne platformy.

Nosným protokol pre XML-RPC je protokol HTTP. Pri spustení systému na odporúčanie navigácie sa preto vytvorí HTTP server, ktorý obsluhuje požiadavky na odporúčanie na porte 1605. Ide teda o webovú službu, ktorá je prístupná prostredníctvom protokolu XML-RPC. Rozhranie sme implementovali s využitím knižnice Apache XML-RPC, ktorá poskytuje aj vlastnú implementáciu jednoduchého HTTP servera.

Prezentácia znalostí

Modul načíta z bázy znalostí zvolený strom vzorov a zobrazí ho vo forme grafu. Používateľ má možnosť filtrovať uzly stromu (a teda aj vzory) na základe ich podpory. Na vykreslenie stromu sme použili knižnicu Jung.

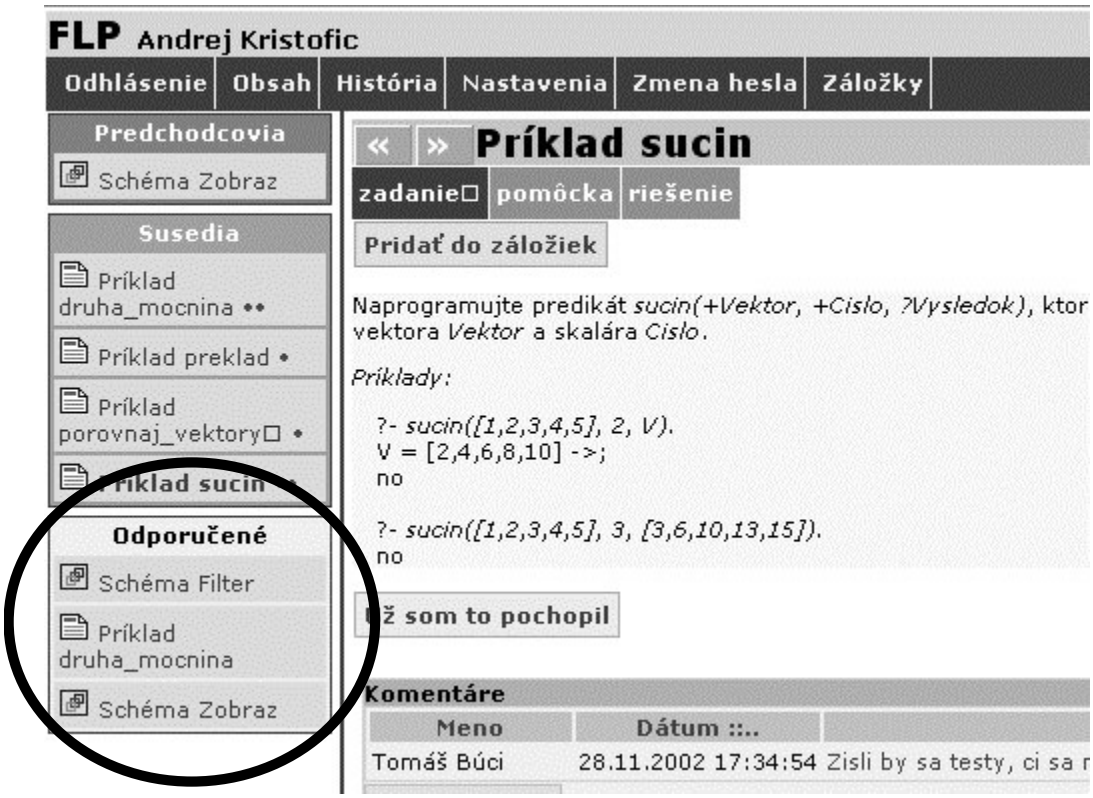
Úpravy systému ALEA

Výučbový AH systém ALEA je základným zdrojom údajov pre tento projekt. Aby bolo možné tento systém použiť, bolo potrebné vykonať v ňom dve úpravy.

Prvá úprava sa týkala zaznamenávania údajov a akciách používateľov. Systém ALEA pomáha študentom pri navigácií v doméne tým, že im poskytuje tlačidlo „Ďalší“ a „Predošlý“.

Stlačeniu týchto tlačidiel zodpovedajú akcie *Next* a *Previous*. Parametrom týchto akcií však nebol cieľový koncept, na ktorý sa používateľ stlačením tlačidla dostal, ale zdrojový koncept, teda ten, na ktorom študent tlačidlo použil. Dôsledkom takéhoto spôsobu vytvárania záznamov bolo, že pri niektorých špecifických postupnostiach akcií používateľov nebolo možné zistiť všetky koncepty, ktoré používateľ navštívil. Vytváranie záznamov sme teda upravili tak, aby parametrom akcií *Next* a *Previous* boli cieľové koncepty. V rámci tejto úpravy sme pridali navyše zaznamenávanie akcie *DisplayFragment*, na základe ktorej vieme zistiť, ktoré fragmenty sa používateľovi zobrazili.

Cieľom druhej úpravy bolo prispôsobenie systému tak, aby mohol využívať odporúčania zostavované nami navrhnutým systémom. Pridali sme preto možnosť komunikácie so systémom na odporúčanie navigácie prostredníctvom protokolu XML-RPC, pričom sme použili knižnicu PocketXML-RPC. Odporúčané koncepty zobrazujeme v samostatnej sekcii v ľavej časti obrazovky (pozri obrázok 4).



Obrázok 4: Odporúčanie konceptov integrované do systému ALEA .

Generovanie testovacích údajov

Aby sme získali ďalšie testovacie údaje, rozhodli sme sa umelo vygenerovať testovacie záznamy o používaní systému. Preto sme vytvorili softvérový nástroj, ktorý umožňuje vytvoriť niekoľko „robotov“, ktoré navštevujú stránky systému ALEA. Každý „robot“ analyzuje stránky, ktoré mu systém zobrazuje a identifikuje v nich odkazy. Odkazy roztriedi do rôznych skupín podľa toho, o aký druh odkazov ide (napr. odkazy v kontextovej ponuke, odkazy v hlavnej ponuke, odkazy „Ďalší“ a „Predošlý“ na podporu navigácie a pod.).

Program sa náhodne rozhoduje, ktorú skupinu odkazov si vyberie a ktorý koncept navštívi ako ďalší. Jeho preferencie je však možné ovplyvniť pomocou parametrov. Experimentmi s takto získanými údajmi sa budeme zaoberať v časti 6.

Implementačné prostredie a použité prostriedky

Všetky časti prototypu sme implementovali v jazyku Java. Dôvodom tohto rozhodnutia boli predovšetkým skúsenosti s týmto programovacím jazykom, existencia veľkého množstva implementovaných dátových štruktúr a v neposlednom rade aj nezávislosť tohto jazyka od použitého operačného systému. Pri vývoji sme použili run-time prostredie pre programy v jazyku Java vo verzii 1.4.2.

Pri implementácii sme využili služby niekoľko knižníc, ktorých názvy a použité verzie sú zhrnuté v tabuľke 1.

Názov knižnice	Verzia	Opis
Jung http://jung.sourceforge.net/	1.5	Knižnica pre prácu s grafmi a ich vizualizáciu.
Log4j http://logging.apache.org/log4j/docs/	1.2.8	Knižnica pre vytváranie testovacích a ladiacich záznamov – logov.
MySQL Connector Java http://www.mysql.com/products/connector/j/	3.0.15	JDBC ovládač pre databázový systém MySQL.
PocketXML-RPC http://www.pocketsoap.com/pocketXMLRPC/	1.2.1	Implementácia protokolu XML-RPC vo forme COM objektu.
Xerces http://xml.apache.org/xerces2-j/	2.6.0	Knižnica pre spracovanie XML súborov.
Apache XML-RPC http://ws.apache.org/xmlrpc/	1.2-b1	Implementácia protokolu XML-RPC v jazyku Java.

Tabuľka 1: Použité softvérové knižnice.

Zhodnotenie prototypu

Implementovaním prototypu sa nám podarilo overiť návrh systému. Prototyp implementuje moduly systému pre predspracovanie údajov, hľadanie vzorov a odporúčanie konceptov na základe objavených vzorov. Jeho funkčnosť sme overili pri experimentoch s reálnymi údajmi z AH systému ALEA.

Vďaka iteratívnemu a inkrementálnemu prístupu k vývoju sme boli schopní overiť kľúčové časti systému, ako aj jednotlivé algoritmy už počas ich návrhu. To umožnilo skorú identifikáciu problémov a ich riešenie. Týmto spôsobom sme napríklad prišli na to, že časť algoritmu na objavovanie spojitých sekvenčných vzorov nie je vhodné vykonávať v kontexte AH systémov (bližšie pozri kapitolu 5.2). Ďalším príkladom môže byť odhalenie nekonzistencie údajov zo systému ALEA, ktorú sme vyriešili jednak úpravou samotného systému, ako aj vývojom algoritmu na čiastočnú rekonštrukciu týchto údajov. Takisto nás to viedlo k vytvoreniu nástroja na generovanie vlastných testovacích údajov.

6 Experimenty a výsledky

S prototypom systému na odporúčanie navigácie sme vykonali niekoľko experimentov. Experimenty sledovali dva základné ciele:

1. Overiť vplyv parametrov algoritmov na ich výsledky a nájsť vhodné hodnoty týchto parametrov.
2. Zistiť, nakoľko ovplyvňujú samotné údaje výsledky algoritmov, predovšetkým výsledky algoritmu na odporúčanie konceptov.

V nasledujúcich kapitolách opíšeme priebeh jednotlivých experimentov, uvedieme ich výsledky, ktoré sa pokúsime zdôvodniť a vysvetliť.

6.1 Testovacie údaje

Pri experimentoch sme pracovali s troma rôznymi sadami testovacích údajov. Všetky tri sady pochádzali zo systému ALEA. Základné vlastnosti jednotlivých sád testovacích údajov sú uvedené v tabuľke 2. V ďalších tabuľkách sa budeme na jednotlivé sady odkazovať už iba ich číslami.

Prvá sada údajov predstavuje záznamy o používaní systému ALEA v období od novembra roku 2002 až do októbra roku 2004. Koncom októbra roku 2004 sme nasadili upravenú verziu systému. Nakoľko sa v tejto verzii zmenil (resp. opravil) spôsob vytvárania záznamov o používaní, rozhodli sme sa údaje získané novou verziou zaradiť do samostatnej skupiny, s cieľom zistiť, aký vplyv mala možná strata informácií pri pôvodnom spôsobe vytvárania záznamov na výsledky. Údaje z upravenej verzie systému ALEA pokrývajú obdobie od konca októbra roku 2004 do januára nasledujúceho roku.

Číslo	Testovacia sada	Obdobie	Počet používateľov[1]	Počet akcií
1.	ALEA – pôvodná	11/2002 – 10/2004	287	37 947
2.	ALEA – upravená	10/2004 – 01/2005	93	28 016 (14 250) [7]
3.	ALEA – generovaná	–	25	4 325 (2 211)[7]

Tabuľka 2: Vlastnosti sád testovacích údajov.

[7] Pri úprave systému ALEA sme pridali zaznamenávanie akcie FragmentDisplay. Číslo v zátvorke uvádza počet akcií bez uvedeného typu akcií.

Údaje o používaní systému ALEA majú niekoľko obmedzení, ktoré majú vplyv na výsledky experimentov. V prvom rade je to ich možná neúplnosť v dôsledku chyby pri ich zaznamenávaní. Ďalej je nutné spomenúť, že samotný systém ALEA už má v sebe zabudované vedenie používateľov (prostredníctvom kontextových odkazov a pomocou tlačidla „Ďalší“). Toto vedenie používatelia pomerne často využívali a preto sú údaje o používaní skreslené. Ďalšie aspekty, ktoré je potrebné vziať do úvahy sú malý rozsah domény, ktorú systém prezentuje a pomerne krátky čas, počas ktorého študenti systém používali.

Kvôli uvedeným obmedzeniam údajov získaných používaním systému ALEA sme sa rozhodli pomocou vlastného generátora vytvoriť tretiu, umelú sadu testovacích údajov. Aj tieto údaje však budú čiastočne ovplyvnené vedením v systéme ALEA, nakoľko generátor využíval pri návštevách stránok aj tie, ktoré systém odporúča pomocou tlačidla „Ďalší“. Frekvencia využívania vedenia je však menšia ako pri reálnych údajoch.

6.2 Predspracovanie údajov

Pri predspracovaní akcií používateľov je hlavnou úlohou identifikácia prístupov používateľa. Do databázy sa ukladajú iba tie prístupy, ktorých dĺžka prekročí minimálnu hranicu. Pri tomto experimente sme skúmali, aké množstvo prístupov sa odfiltruje pri rôznom nastavení tejto hranice. Takisto sme sa zaujímali o priemernú dĺžku prístupov, resp. o rozdelenie prístupov podľa ich dĺžky. Výsledky predspracovania a štatistické údaje zozbierané počas neho uvádzame v tabuľkách 3 a 4.

Z výsledkov je vidieť, že prístupy používateľov zo sady 1 sú kratšie ako prístupy identifikované v sade 2, pričom rozdiel v priemernej dĺžke je približne 3 akcie. Toto môže naznačovať neúplnosť údajov v prvej sade, spôsobenú nesprávnym spôsobom vytvárania záznamov. Takisto je možné pozorovať, že v dátach je pomerne veľké množstvo kratších prístupov. Napríklad až 50%, resp. 40% všetkých prístupov je kratších ako 5. Možným vysvetlením je, že študenti systém používali väčšinou počas cvičení a teda mali obmedzený čas na prácu so systémom. Na sadu umelo vytvorených údajov zmena parametra nemala až taký silný vplyv, nakoľko sme spôsobom jej generovania zabezpečili, aby v nej prevládali dlhšie prístupy používateľov. Na vhodnú voľbu parametra hranice minimálnej dĺžky prístupu bude treba zvážiť aj jeho vplyv na výsledky objavovania znalostí, resp. samotného odporúčania konceptov.

Sada	Počet identifikovaných prístupov	Priemerná dĺžka prístupu	Maximálna dĺžka prístupu
1.	2 109	10,06	214
2.	602	14,64	140
3.	1213	19,32	49

Tabuľka 3: Štatistické údaje o identifikovaných prístupoch.

Minimálna dĺžka na akceptovanie prístupu	Sada	Množstvo akceptovaných prístupov (v %)	Priemerná dĺžka akceptovaného prístupu
5	1.	48,55	19,53
	2.	61,79	22,56
	3.	96,26	20,01
10	1.	30,82	26,89
	2.	41,03	30,52
	3.	89,72	20,86
15	1.	21,48	33,48
	2.	31,23	36,48
	3.	57,94	25,71
20	1.	16,45	38,55
	2.	23,29	42,33
	3.	41,12	29,48

Tabuľka 4: Vplyv nastavenia hranice minimálnej dĺžky na priemernú dĺžku akceptovaného prístupu.

6.3 Objavovanie vzorov

Na predspracované prístupy používateľov sme aplikovali algoritmy na hľadanie asociačných pravidiel (AP), sekvenčných vzorov (SV) a spojitých sekvenčných vzorov (SSV). Zisťovali sme, ako vplýva množstvo dát odfiltrované pri predspracovaní na množstvo objavených vzorov a ich dĺžku. Ďalej sme sledovali vplyv parametra algoritmov (minimálnej podpory vzorov) na výsledky.

Výsledky v tabuľke 5 ukazujú ako vplýva filtrovanie prístupov pri predspracovaní na celkový počet objavených vzorov. Hodnota parametra minimálnej podpory bola pri tomto experimente nastavená na 0,15. Vo výsledkoch si môžeme všimnúť zaujímavé závislosti:

1. Čím dlhšie vzory sa nachádzali v databáze (pretože kratšie boli odfiltrované pri predspracovaní), tým väčší bol počet nájdených vzorov. Takisto rástla ich dĺžka. Z toho vyplýva, že základ pre vzory tvoria práve dlhšie prístupy používateľov. Prítomnosť kratších prístupov im bránila, aby získali väčšiu podporu. Odstránením kratších prístupov sme umožnili, aby vzory objavené v dlhších prístupoch mohli získať dostatočnú podporu na to, aby mohli byť zaradené do výsledkov.

2. Je vidieť výrazný rozdiel v počte objavených vzorov pri rôznych testovacích sadách dát, pričom oveľa väčší počet vzorov sa našiel v údajoch, ktoré pochádzajú z upravenej verzie systému ALEA. Je možné, že problémy so zaznamenávaním akcií v pôvodnej verzii spôsobili, že údaje o používaní nie sú kompletné. Menší počet

vzorov v umelo vytvorenej sade údajov je pravdepodobne spôsobený tým, že generovanie tejto sady je čiastočne založené na náhodnom výbere.

3. Rozdielny počet vzorov objavených rôznymi technikami súvisí s reštriktívnosťou jednotlivých druhov vzorov. Najmenej obmedzení určujú asociačné pravidlá, najviac obmedzujúce sú spojité sekvenčné vzory.

V tabuľke 6 uvádzame výsledky objavovania vzorov pre rôzne hodnoty minimálnej podpory vzorov. Boli použité prístupy používateľov, ktorých minimálna dĺžka bola 10. Opäť je badať výrazný rozdiel medzi jednotlivými sadami dát. Zdá sa, že hodnota parametra minimálnej podpory silne ovplyvňuje počet objavených vzorov. Ukazuje sa, že pri hodnotách okolo 0,15 algoritmy nájdu dostatočné množstvo vzorov. Pomerne nízka hodnota podpory vzorov môže byť spôsobená rozmanitosťou správania sa používateľov v systéme. Pravdepodobne len malé skupiny používateľov sa správali podobným spôsobom.

6.4 Odporúčanie konceptov

Pri experimentoch s odporúčaním konceptov sme sa zamerali predovšetkým na jeho presnosť. Snažili sme sa posúdiť vplyv parametrov predspracovania a objavovania vzorov na úspešnosť presnosť.

Počas experimentov bola sada údajov náhodne rozdelená na dve časti – trénovaciu a testovaciu množinu údajov. V trénovacej množine sa hľadali vzory správania sa, prístupy používateľov v testovacej množine sme použili ako vstup algoritmu pre odporúčanie. V procese testovania sme postupne vytvárali odporúčania na základe každého prefixu daného prístupu používateľa, pričom sa výsledok odporúčania porovnával s konceptom, ktorý nasledoval v prístupe používateľa bezprostredne za použitým prefixom.

Minimálna dĺžka na akceptovanie prístupu	Sada	Počet objavených vzorov			Max. dĺžka vzorov		
		AP	SV	SSV	AP	SV	SSV
5	1.	37	33	30	2	2	1
	2.	229	153	69	6	4	3
	3.	165	142	48	5	5	3
10	1.	113	93	61	4	3	2
	2.	2 072	944	111	8	7	5
	3.	219	158	55	6	5	4
15	1.	282	209	91	6	5	3
	2.	11 774	5 423	155	11	9	5
	3.	1 438	785	89	9	8	6

Tabuľka 5: Vplyv minimálnej dĺžky prístupov na počet objavených vzorov.

Minimálna podpora	Sada	Počet objavených vzorov			Max. dĺžka vzorov		
		AP	SV	SSV	AP	SV	SSV
0,15	1.	113	93	61	4	3	2
	2.	2 072	944	111	8	7	5
	3.	219	158	55	6	5	4
0,20	1.	37	35	32	2	2	1
	2.	322	201	68	6	5	3
	3.	43	42	25	3	4	2
0,25	1.	24	24	23	2	2	1
	2.	126	79	49	4	3	2
	3.	16	16	16	1	1	1

Tabuľka 6: Vplyv minimálnej podpory vzorov na počet objavených vzorov.

Za presné odporúčanie sme považovali takú odporučenú postupnosť konceptov, ktorá obsahovala očakávaný koncept (bez ohľadu na to, na ktorej pozícii sa nachádzal). Pri vykonávaní testov sme zaznamenávali údaje o priemernom umiestnení presne odporučených konceptov v postupnosti. Odporúčanie sme testovali samostatne na základe všetkých troch druhov vzorov (asociačné pravidlá – AP, sekvenčné vzory – SV, spojité sekvenčné vzory – SSV). Takisto sme vyhodnocovali presnosť odporúčania, ktoré vzniklo ich zlúčením (zlúčené odporúčanie – ZO).

Treba poznamenať, že nami vyhodnocovaná presnosť odporúčaní nemá priamy súvis s úspešnosťou odporúčania. Za úspešné odporúčania totiž považujeme tie, ktoré pomôžu študentovi lepšie zvládnuť predkladanú doménu. Z toho vyplýva, že úspešnosť odporúčaní by sme museli vyhodnocovať na základe spätnej väzby od študenta, resp. na základe jeho dosiahnutých výsledkov. To, že odporúčanie nie je presné ešte neznamená, že by nebolo úspešné (samozrejme, platí to aj naopak – presné odporúčanie nemusí byť úspešné). Ničmenej, presnosť odporúčaní nám umožní aspoň čiastočne vyhodnotiť vplyv rôznych parametrov na výsledky.

Tabuľka 7 poskytuje prehľad o výsledkoch odporúčania získaných pri rôznych veľkostiach trénovacej sady dát (minimálna dĺžka prístupu používateľa bola 5 a minimálna podpora vzorov 0,15). Celkovo môžeme sledovať rastúci trend presnosti odporúčaní s rastúcou veľkosťou sady testovacích údajov. Tento trend je výraznejší pri údajoch získaných z upravenej verzie systému ALEA. Možným vysvetlením je, že pri väčšej trénovacej sade dát sa podarí nájsť viac vzorov správania sa, na základe ktorých je možné robiť odporúčania. Takisto je možné pozorovať veľmi nízku presnosť odporúčaní na základe spojitých

sekvenčných vzorov. Nastavenie ostatných parametrov pri tomto experimente malo za následok, že sa týchto vzorov podarilo nájsť iba veľmi málo, čo sa prirodzene odrazilo aj na presnosti odporúčania.

Minimálna dĺžka prístupu výrazne ovplyvňuje počet objavených vzorov. Tabuľka 8 ukazuje vplyv tohto parametra na presnosť odporúčaní (minimálna podpora vzorov 0,15, veľkosť trénovacej množiny 90%). S rastúcim počtom nájdených vzorov stúpa aj presnosť odporúčania, samozrejme na úkor času, ktorý je potrebný na nájdenie vzorov a odporúčanie konceptov na ich základe. Najmenší vplyv mal parameter na umelo vytvorenú sadu údajov (pri hodnote 10 sme dokonca zaznamenali pokles presnosti).

Pri experimentoch s hodnotou parametra minimálnej podpory vzorov (tabuľka 9, minimálna dĺžka prístupov používateľov 5, veľkosť trénovacej množiny 90%) sa opäť potvrdilo, že presnosť rastie s množstvom objavených vzorov. Nakoľko s rastúcou hodnotou parametra minimálnej podpory klesá počet objavených vzorov, klesá aj presnosť odporúčania.

Veľkosť trénovacej množiny	Sada	Počet odporúčaní	Presnosť odporúčania (v %)				Priemerné umiestnenie			
			AP	SV	SSV	ZO	AP	SV	SSV	ZO
75%	1.	4 211	4,35	10,57	0,85	13,42	1,49	12,21	1,00	10,09
	2.	1 755	14,64	26,15	17,66	33,90	1,51	5,77	1,19	4,90
	3.	523	6,12	19,50	5,53	23,52	1,13	8,89	1,00	7,78
80%	1.	3 373	3,82	10,79	0,98	13,73	1,22	12,02	1,00	13,73
	2.	1 367	15,52	27,00	18,58	34,16	1,55	5,20	1,22	4,51
	3.	419	5,49	20,05	4,30	23,15	1,17	6,77	1,00	6,24
85%	1.	2 272	4,40	11,00	0,00	14,26	1,20	11,10	–	8,91
	2.	941	15,52	27,10	19,02	34,11	1,50	3,81	1,21	3,43
	3.	273	7,69	23,44	6,23	27,47	1,19	6,50	1,00	5,97
90%	1.	1 442	4,37	12,48	0,00	15,81	1,21	11,59	–	9,53
	2.	353	18,70	35,69	25,21	43,06	1,41	3,82	1,18	3,49
	3.	179	7,82	24,58	5,03	29,05	1,29	7,14	1,00	6,44

Tabuľka 7: Vplyv veľkosti trénovacej množiny na presnosť odporúčania.

Min. dĺžka prístupov	Sada	Počet odporúčaní	Presnosť odporúčania (v %)				Priemerné umiestnenie			
			AP	SV	SSV	ZO	AP	SV	SSV	ZO
5	1.	1 442	4,37	12,48	0,00	15,81	1,20	11,59	–	9,53
	2.	353	18,70	35,69	25,21	43,06	1,41	3,82	1,18	3,49
	3.	179	7,82	24,58	5,03	29,05	1,29	7,14	1,00	6,44
10	1.	1 604	9,16	15,09	6,80	19,14	1,68	13,16	1,05	10,73
	2.	601	16,31	37,60	27,79	44,09	1,72	7,44	1,17	6,26
	3.	203	6,40	21,67	11,82	26,11	1,69	8,18	1,00	7,15
15	1.	1 379	12,04	17,11	10,22	22,04	2,11	7,59	1,10	6,36
	2.	472	22,67	50,64	34,32	57,63	2,22	7,09	1,27	5,97
	3.	153	9,15	29,41	28,95	33,33	2,07	4,36	1,10	4,02

Tabuľka 8: Vplyv minimálnej dĺžky prístupov na presnosť odporúčania.

Min. podpora vzorov	Sada	Počet odporúčaní	Presnosť odporúčania (v %)				Priemerné umiestnenie			
			AP	SV	SSV	ZO	AP	SV	SSV	ZO
0,13	1.	1 442	8,39	13,87	2,57	18,31	1,45	10,42	1,00	8,33
	2.	353	24,93	44,19	30,59	52,12	1,43	2,99	1,17	2,71
	3.	179	10,61	29,61	12,85	33,52	2,21	2,7	1,00	5,28
0,14	1.	1 442	5,69	12,41	2,57	16,37	1,50	11,96	1,00	9,58
	2.	353	22,95	38,81	26,91	48,16	1,44	3,75	1,19	3,48
	3.	179	6,70	26,26	8,94	29,05	1,33	6,68	1,00	6,31
0,15	1.	1 442	4,37	12,48	0,00	15,81	1,20	11,59	–	9,53
	2.	353	18,70	35,69	25,21	43,06	1,41	3,82	1,18	3,49
	3.	179	7,82	24,58	5,03	29,05	1,29	7,14	1,00	6,44

Tabuľka 9: Vplyv minimálnej podpory vzorov na presnosť odporúčania.

Pri všetkých experimentoch boli najpresnejšie odporúčania, ktoré vznikli na základe sekvenčných vzorov, aj keď umiestnenie odporúčaného konceptu bolo podstatne nižšie ako pri odporúčaniach vytvorených z ostatných druhov vzorov. Pri spájaní odporúčaní z rôznych druhov vzorov sme použili nasledujúce váhy:

- asociačné pravidlá – 2
- sekvenčné vzory – 4
- spojité sekvenčné vzory – 90

Výsledné odporúčanie bolo vždy presnejšie ako čiastkové odporúčania. To znamená, že jednotlivé druhy vzorov sa navzájom čiastočne dopĺňajú. Hoci sme pomocou váh silne preferovali spojité sekvenčné vzory ukázalo sa, že na výsledku výsledného odporúčania sa najviac podieľali práve obyčajné sekvenčné vzory. Umiestnenie konceptu vo výslednom odporúčaní sa totiž výrazne posunulo smerom nadol, k hodnotám, ktoré dosahovalo pri odporúčaní na základe sekvenčných vzorov. Je to spôsobené pravdepodobne tým, že odporúčanie na základe sekvenčných vzorov sa často podarí urobiť aj vtedy, keď nie je možné urobiť odporúčanie na základe ostatných druhov vzorov.

Ukázalo sa, že zmena spôsobu zaznamenávania akcií v systéme ALEA zmenila povahu údajov. Presnosť odporúčaní pre pôvodnú verziu systému bola podstatne menšia ako presnosť odporúčaní pri upravenej verzii. Výsledky však môžu byť skreslené, pretože sady dát z oboch verzií systému ALEA obsahovali rozdielne množstvo akcií aj používateľov.

Parametre algoritmov predspracovania aj dolovania v dátach ovplyvňujú presnosť odporúčania konceptov, ktorá sa pri sade dát z upravenej verzie systému ALEA pohybovala od 30% do 60%. Pozorovali sme, že čím väčšie množstvo vzorov sa podarí objaviť, tým je presnosť odporúčania väčšia. Na druhej strane je časovo náročné objaviť veľké množstvo vzorov, takisto ako vytvárať na ich základe odporúčania. Príliš veľké množstvo vzorov môže spôsobiť, že systém bude na dynamické (on-line) odporúčanie nepoužiteľný. Aj s prihliadnutím na tento fakt sa nám ako najvhodnejšie javia hodnoty nasledujúce parametrov:

- minimálna dĺžka prístupov používateľa – 10
- minimálna podpora vzorov – 0,14

7 Zhodnotenie

Cieľom práce bolo navrhnúť, akým spôsobom je možné aplikovať postupy objavovania znalostí v údajoch z adaptívnych hypermediálnych (AH) systémov, so zámerom využiť objavené znalosti na skvalitnenie prispôsobovania AH systému študentovi. Na základe analýzy údajov, ktoré nám AH systémy môžu poskytnúť sme sa orientovali na techniky a algoritmy na hľadanie charakteristických vzorov (asociačných pravidiel, sekvenčných vzorov a spojitých sekvenčných vzorov). Navrhli sme niekoľko spôsobov využitia týchto vzorov, pričom sme sa zamerali predovšetkým na odporúčanie konceptov študentovi, teda na vytváranie výučbových postupností vhodných pre študenta. Vychádzali sme pri tom z existujúcich postupov na personalizáciu navigácie v prostredí Internetu. Výsledkom je návrh procesu na objavovanie znalostí v AH systémoch, ktorý pokrýva zber vhodných údajov, ich predspracovanie, dolovanie znalostí z údajov a využitie (interpretáciu) objavených znalostí.

Navrhli sme systém, ktorý realizuje proces na objavovanie znalostí v AH systémoch. Vďaka tomu, že sme sústredili všetku funkcionalitu špecifickú pre konkrétny výučbový AH systém do jedného modulu, je systém dostatočne flexibilný a nezávislý od AH systému, použitého ako zdroj údajov. Použili sme tri príbuzné algoritmy na hľadanie vzorov správania sa, aby mal systém dostatok znalostí na odporúčanie konceptov. Pri návrhu systému sme brali do úvahy aj fakt, že objavené znalosti môžu poskytnúť užitočnú informáciu aj autorovi obsahu AH systému. Preto sme do architektúry systému zaradili aj modul na vizualizáciu objavených znalostí.

Na základe návrhu sme implementovali prototyp systému, v ktorom sme sa sústredili na predspracovanie údajov, dolovanie znalostí a odporúčanie konceptov študentovi na základe objavených znalostí. Rozhranie pre odporúčanie konceptov je v prototype systému implementované ako webová služba, ktorá spracuje požiadavky na odporúčanie a vytvorí odporúčanú postupnosť konceptov. Keďže sme ako zdroj údajov pre prototyp využili systém ALEA, implementovali sme zapúzdrovací modul pre tento systém a upravili sme ho tak, aby mohol využiť odporúčania poskytované prototypom.

S prototypom sme vykonali niekoľko experimentov, pri ktorých sme skúmali vplyv rôznych parametrov algoritmov predspracovania údajov a objavovania vzorov na výsledky a presnosť odporúčaní. Na základe výsledkov experimentov sme stanovili odporučené hodnoty jednotlivých parametrov. Pri experimentoch sme pracovali s viacerými sadami údajov zo systému ALEA. Výsledky naznačili, že spôsob zaznamenávania údajov v systéme ALEA ako aj ďalšie faktory (napr. zabudované vedenie používateľov v systéme) čiastočne ovplyvnili výsledky odporúčaní.

Je veľa smerov, ktorými by sa ďalšia práca v tejto oblasti mohla uberať. Jedným z nich môže byť zohľadnenie viacerých atribútov používateľa (napr. jeho znalosť o koncepte alebo čas, ktorý strávil štúdiom konceptu) pri objavovaní znalostí, resp. pri odporúčaní. V práci sme sa zaoberali predovšetkým AH systémami zameranými na výučbu, konkrétne na výučbu programovania. Zaujímavým krokom by mohlo byť zovšeobecnenie myšlienok práce na AH systémy všeobecne alebo ich overenie v inej doméne výučby.

Samotný prototyp poskytuje niekoľko ďalších oblastí skúmania. Bolo by vhodné vyhodnotiť úspešnosť jeho odporúčaní na základe spätnej väzby od študentov a na základe ich dosiahnutých výsledkov. Prospešným by bolo aj získanie väčšieho množstva údajov pre experimenty z rôznych AH systémov a z rôznych predmetov, kde sa vyučuje programovanie (napr. výučba jazykov C, C++ alebo Java).

Literatúra

1. AGRAWAL, R. – SRIKANT, R.: *Fast Algorithms for Mining Association Rules*. Proc. of the 20th Int. Conference on Very Large Databases (Bocca, J. B. – Jarke, M. – Zaniolo, C., eds.), s. 487-499. Santiago. 1994. ISBN 1-55860-153-8. http://www.almaden.ibm.com/cs/people/srikant/papers/vldb94.pdf (20.5.2004)

2. AGRAWAL, R. – SRIKANT, R.: *Mining Sequential Patterns*. Proc. of the 11th Int. Conference on Data Engineering (ICDE) (Yu, P. S. – Chen, A. S. P., eds.), s. 3-14. Taipei, Taiwan. IEEE CS Press 1995. ISBN 0-8186-6910-1. http://www.dsi.unive.it/~dm/icde95.pdf (21.4.2004)

3. AHA!: *Adaptive Hypermedia For All*. Domovská stránka projektu. http://aha.win.tue.nl (20.5.2004)

4. AYRES, J. – FLANNICK, J. – GEHRKE , J. – YIU, T.: *Sequential PAttern Mining Using Bitmaps*. Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, s. 429-435. Edmonton, Alberta, Canada. ACM Press 2002. ISBN 1-58113-567-X. http://www.cs.cornell.edu/johannes/papers/2002/kdd2002-spam.pdf (25.11.2004)

5. BERKHIN, P.: *Survey of Clustering Data Mining Techniques*. Accrue Software, Inc. San Jose, CA. 2002. http://www.accrue.com/products/rp_cluster_review.pdf (20.5.2004)

6. BRUSILOVSKY, P.: *Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies*. Proc. of Workshop on WWW-Based Tutoring. ITS'98. San Antonio, TX. 1998. http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/brusilovsky.html (20.5.2004)

7. BRUSILOVSKY, P.: *Adaptive and Intelligent Technologies for Web-based Education*. Special Issue on Intelligent Systems and Teleteaching (Rollinger, C. – Peylo, C., eds.), Künstliche Intelligenz, 4, s. 19-25. 1999. http://www2.sis.pitt.edu/~peterb/papers/KI-review.pdf (18.9.2004)

8. BRUSILOVSKY, P.: *Adaptive Hypermedia.* User Modeling and User Adapted Interaction. Ten Year Anniversary Issue (Kobsa, A., ed.), s. 87-110. 2001. http://www2.sis.pitt.edu/~peterb/papers/brusilovsky-umuai-2001.pdf (22.11.2004)

9. CHEN, M. S. – PARK, J. S. – YU, P. S.: *Data mining for path traversal patterns in distributed systems*. 16th Int. Conference on Distributed Computing Systems (ICDCS '96), s. 385-393. Hong Kong. IEEE CS Press 1996. ISBN 0-8186-7398-2. http://www.ee.ntu.edu.tw/~mschen/paperps/icdcs16.ps (29.11.2004)

10. DE BRA, P. ET AL.: *AHA! The Adaptive Hypermedia Architecture*. Proc. of the 14th ACM Conference on Hypertext and Hypermedia, s. 81-84. Nottingham. ACM Press 2003. ISBN 1-58113-704-4. http://wwwis.win.tue.nl/~debra/ht03/pp401-debra.pdf (20.5.2004)

11. GAUL, W. – SCHMIDT-THIEME, L.: *Mining web navigation path fragments.* Proc. of the Workshop on Web Mining for E-Commerce – Challenges and Opportunities. Boston, MA. 2000. http://robotics.stanford.edu/~ronnyk/WEBKDD2000/papers/thieme.pdf (9.11.2004)

12. GÜNDÜZ, S. – TAMER ÖZSU, M.: *A Web Page Prediction Model Based on Click-stream Tree Representation of User Behavior*. Proc. of the 9th ACM SIGKDD Int.Conference on Knowledge discovery and data mining (Getoor, L. – Senator, T. E. – Domingos, P. – Faloutsos C., eds.), s. 535-540. Washington, D.C. ACM Press 2003. ISBN 1-58113-737-0. http://db.uwaterloo.ca/~ddbms/publications/web/kdd03.pdf (25.11.2004)

13. IEEE WG12: *IEEE 1484.12.1-2002.* Standard for Information Technology – Education and Training Systems – Learning Objects and Metadata. 2002. http://ltsc.ieee.org/wg12/index.html (28.4.2005)

14. KOSTELNÍK, R.: *Podpora výučby programovania pomocou príkladov v pavučine*. Diplomová práca. FEI STU. Bratislava. 2002.

15. KOSTELNÍK, R. – BIELIKOVÁ, M.: *Web-Based Environment using Adapted Sequences of Programming Exercises*. Proc. of Int. Conference on Information Systems Implementation and Modelling (Beneš, M., ed.), s. 33-40. Brno. 2003. ISBN 80-85988-84-4. http://www.fiit.stuba.sk/~bielik/publ/abstracts/2003/isim2003.pdf (19.4.2005)

16. MOBASHER, B. – COOLEY, R. – SRIVASTAVA, J.: *Automatic Personalization Based On Web Usage Mining*. Communication of ACM. Volume 43. Issue 8. 2000. http://maya.cs.depaul.edu/~mobasher/papers/MCS00.pdf (21.11.2004)

17. KRIŠTOFIČ, A. – BIELIKOVÁ, M.: *Improving adaptation in web–based educational hypermedia by means of knowledge discovery.* Submitted to: Hypertext 2005 – 16th ACM Conference on Hypertext and Hypermedia. 2005.

18. KRIŠTOFIČ, A.: *Recommender System for Adaptive Hypermedia Applications*. Proc. of the IIT.SRC – Student Research Conference in Informatics and Information Technologies (Bieliková, M., ed.), s. 229-234. Vydavateľstvo STU, Bratislava. 2005. ISBN 80-227-2217-0. http://www.fiit.stuba.sk/iit-src/35-kristofic.pdf (3.5.2005)

19. MOBASHER,B. – DAI, H. – LUO, T. – NAKAGAWA, M.: *Using Sequential and Non-Sequential Patterns for Predictive Web Usage Mining Tasks.* Proc. of the IEEE International Conference on Data Mining (ICDM'2002), s. 669-672. Maebashi. 2002. ISBN 0-7695-1754-4. http://maya.cs.depaul.edu/~mobasher/papers/MDLN02b.pdf (21.11.2004)

20. NAKAGAWA, M. – MOBASHER, B.: *Impact of Site Characteristics on Recommendation Models Based On Association Rules and Sequential Patterns*. Proc. of the IJCAI'03 Workshop on Intelligent Techniques for Web Personalization. Acapulco, Mexico. 2003. http://maya.cs.depaul.edu/~mobasher/papers/NM03a.pdf (21.2.2005)

21. PARALIČ, J.: *Objavovanie znalostí v databázach*. Elfa, Košice 2003. 80 s. ISBN 80-89066-60-7. http://neuron.tuke.sk/~paralic/prezentacie/OZ/ObjavovanieZnalostivDB.pdf (19.4.2005)

22. ROMERO, C. – VENTURA, S. – DE BRA, P. – DE CASTRO, C.: *Discovering Prediction Rules in AHA! Courses*. Proc. of the 9th Int. User Modeling Conference (Brusilovsky, P. – Corbett, A. T. – de Rosis, F., eds.), s. 25-34. Johnstown, PA. Springer 2003. ISBN 3-540-40381-7. http://wwwis.win.tue.nl/~debra/um2003/um03.pdf (20.5.2004)

23. SHAHABI, C. – ZARKESH, A. M. – ADIBI, J. – SHAH, V.: *Knowledge discovery from users Web-page navigation*. Proc. of the 7th Int. Workshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications, s. 20. IEEE CS Press 1997. ISBN 0-8186-7849-6.
http://www.isi.edu/~adibi/papers/adibi_ride97.pdf (17.2.2005)

24. ZAÏANE, O. R.: *Web Usage Mining for a Better Web-Based Learning Environment*. Proc. of Conference on Advanced Technology for Education, s. 450-455. Banff, Alberta. 2001.
http://www.cs.ualberta.ca/~zaiane/postscript/CATE2001.pdf (20.5.2004)

Príloha A Príspevok na konferencii IIT.SRC 2005

V prílohe sa nachádza článok, ktorý sme prezentovali na študentskej konferencii IIT.SRC, ktorá sa konala 27. apríla 2005 na Fakulte informatiky a informačných technológií STU v Bratislave.

KRIŠTOFIČ, A.: *Recommender System for Adaptive Hypermedia Applications*. Proc. of the IIT.SRC – Student Research Conference in Informatics and Information Technologies (Bieliková, M., ed.), s. 229-234. Vydavateľstvo STU, Bratislava. 2005. ISBN 80-227-2217-0.
http://www.fiit.stuba.sk/iit-src/35-kristofic.pdf (3.5.2005)

Recommender System for Adaptive Hypermedia Applications

Andrej KRIŠTOFIČ*

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
kristofic@zoznam.sk

Abstract. Most adaptive web-based hypermedia systems adapt presentation of the content and/or navigation using predefined set of rules. Considering different behavior and preferences of each user it may be hard to generalize and construct all appropriate rules in advance. In this paper we explore this problem in context of educational adaptive hypermedia systems. We focus on recommending lessons (learning objects or concepts) that students should study next while using adaptive hypermedia system. We present architecture of a web-based recommender system, which uses results of knowledge discovery to improve adaptation.

1 Introduction

Adaptive hypermedia systems have become popular in last few years in many application areas such as educational hypermedia, on-line information systems or information retrieval hypermedia [1]. Adaptive hypermedia application in the field of educational systems is natural, as each student generally has different characteristics related to learning, so an adaptation to the student's characteristics can help the student to master topics more effectively.

However, it is quite difficult for an author of the course to develop appropriate adaptation model. This requires respectable knowledge of students' behavior to create such adaptation rules, which will result into an effective adaptation of presented knowledge together with appropriate navigation in the course information space. We propose to use knowledge extracted from educational adaptive hypermedia system by means of knowledge discovery for the adaptation model improvement.

---

* Supervisor: doc. Ing. Mária Bieliková, PhD., Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

M. Bieliková (Ed.), IIT.SRC 2005, April 27, 2005, pp. 229-234.

In this paper we present a recommender system, which employs knowledge discovery for purpose of finding patterns of a characteristic behavior of students from data collected by the adaptive hypermedia educational system. Discovered patterns are used in the process of recommendation of relevant concepts to students, mainly for curriculum sequencing.

2 Navigation recommendation approaches

Following discovered patterns various approaches have been proposed for navigation recommendation (global or local, direct or indirect guidance). In [5] authors have proposed to conduct the recommendation directly with discovered patterns. Active user session is used to search the patterns and find the matching ones. Suffixes of these patterns become candidates for the recommendation. Only the most recent part of the active user session is used for matching.

Another approach [4] is aimed to group stored user sessions or discovered frequent item sets into session clusters using techniques of clustering. Representative usage profile is computed for each cluster. Active user session is then matched against computed usage profiles.

Mentioned approaches deal with one kind of patterns and provide support for this specific kind. We propose using several kinds of patterns (e.g., sequential and traversal patterns) distinguished during the process of recommendation and describe software framework for a recommender system, which extends architecture of adaptive web-based hypermedia applications. Properties of used patterns and methods for discovering them are discussed in greater detail in [3].

3 Recommender system for adaptive hypermedia

3.1 Recommender system architecture

Proposed recommender system as a back-end for an adaptive hypermedia system is based on modular software architecture. The software architecture (see Figure 1) follows the flow of two main processes carried out in the system: (1) knowledge discovery and (2) concept recommendation.

The system is designed to be independent of underlying adaptive hypermedia application as much as possible. Therefore the architecture incorporates a wrapper, which acts as a facade between an adaptive hypermedia system and the recommender system itself. This approach allows using various adaptive hypermedia systems as sources of data for knowledge discovery without modification of the recommender system itself.

Wrapper. All the adaptive hypermedia system dependent tasks are performed in the wrapper module. This includes usage data retrieval as well as tasks related to the processing and deploying generated adaptation rules, and the user model update. Using the recommender system with several adaptive hypermedia systems requires a wrapper

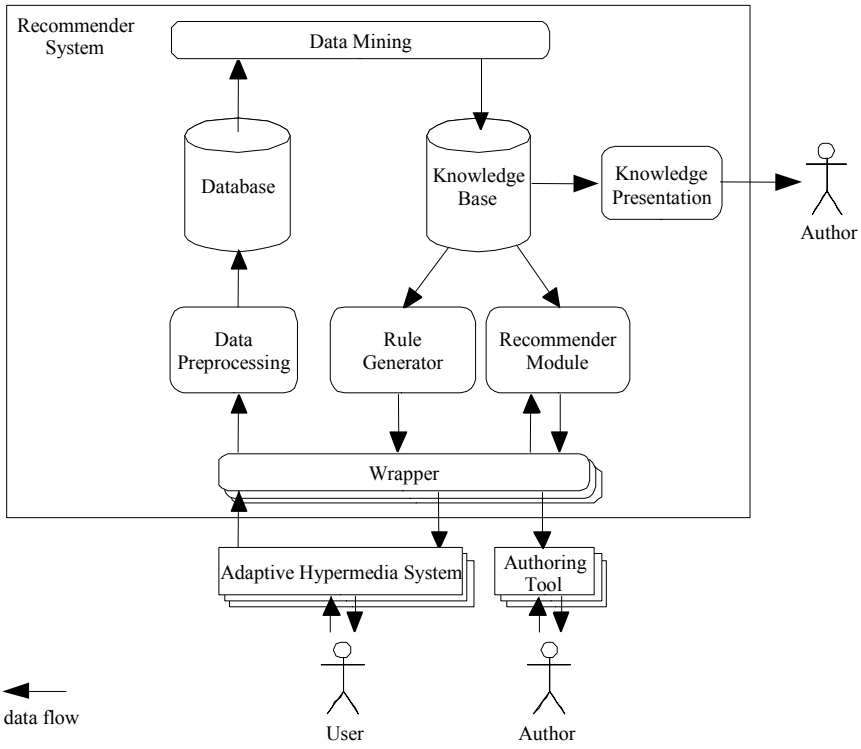


Fig. 1. Architecture of recommender system.

module for every adaptive hypermedia system. This module forms also the interface to an authoring tool, where adaptation knowledge is defined by the author.

Data preprocessing. The data preprocessing module takes the stream of user actions as its input. These actions are preprocessed and searched for potential errors and inconsistencies. The module also takes care of identification of user's sessions. Sessions are filtered according to their length (short sessions are dropped) and/or overall knowledge level of respective user. Finally, the module stores sessions into database of the recommender system.

Data mining. In the data mining module various algorithms for pattern mining (see [3]) are applied on data stored in database. Traversal patterns, sequential patterns and association rules are discovered using different views on data (e.g., different levels of concepts in domain hierarchy, concept types). Discovered patterns are stored into the knowledge base.

Knowledge presentation. The main purpose of the knowledge presentation module is to present selected discovered knowledge (e.g., association rules) to the author of a course. The author can then make modifications to the domain structure and the contents of the information fragments.

Rule generator. The task of the rule generator module is to construct static adaptation rules (i.e., rules where current learning session of a student is not considered) based on patterns in the knowledge base. These rules are deployed to the adaptive hypermedia system through the wrapper.

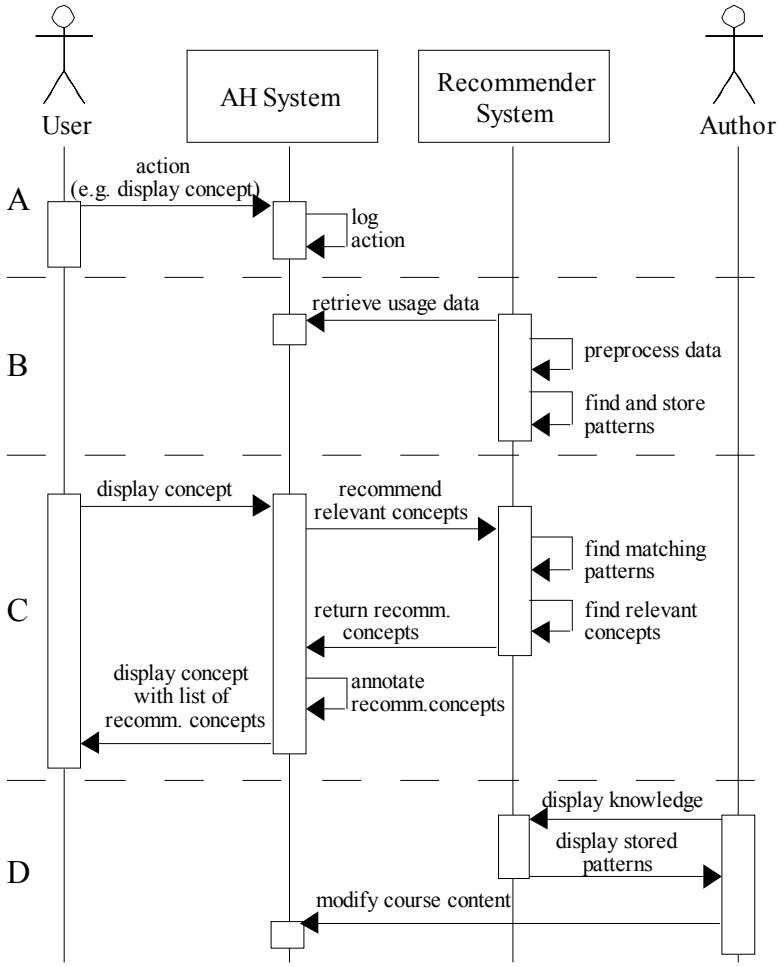


Fig. 2. Typical scenario of the recommender system usage.

Recommender module. The recommender module creates a sequence of recommended concepts based on the current session of a user and the knowledge stored in the knowledge base. The concepts in the current user session are matched against the ones in patterns of the knowledge base. When appropriate (similar) patterns are found, the concept recommendation is carried out. The result is a sequence of recommended concepts and the evaluation of their suitability and relevance for the user. This sequence is generated on demand of the adaptive hypermedia system (each time the user moves to another concept or fragment, or after some time period or at the beginning of each session).

3.2 Use case scenario

Sequential diagram in Figure 2 describes typical usage scenario of proposed recommender system. While users work with the adaptive hypermedia system, it logs and

stores all their actions (e.g., logins, logouts, concept visits) (see part A in figure). When scheduled, the recommender system retrieves usage data, preprocesses it and stores it into database. Data mining algorithms are executed on stored usage data and usage patterns are discovered and stored in the knowledge base (see part B in figure).

With knowledge base filled up, the recommender system can recommend a sequence of relevant concepts for particular user, when asked to do so. An adaptive hypermedia system provides the recommender system with the user identification and his current session and the recommender system returns recommended sequence of concepts, which the user should visit next. The adaptive hypermedia system may annotate concepts in the sequence according to suitability evaluation provided by the recommender system (see part C in figure).

When author of the content decides to improve it, visualization of discovered knowledge can help to decide which modifications should be made. The author can also compare historical records and evaluate the effect of his previous changes to the adaptive hypermedia system content (see part D in figure).

4 Experiments

To verify our approach to knowledge discovery in adaptive web-based hypermedia we developed a prototype of the recommender system. We implemented three data mining algorithms for pattern discovery (traversal patterns, sequential patterns and association rules) suitable for educational hypermedia applications.

Results of the knowledge discovery are markedly dependent on source data. We chose the ALEA educational adaptive hypermedia system [2] for our experiments since we have had available usage logs gathered during 3 years of its usage in the Functional and logic programming course. At time of experiments the database contained 1 170 sessions.

Experimenting with the ALEA data, we found out that even though algorithms find some patterns in the data, these patterns have quite low support (around 15%). To outline this fact, we analyzed and visualized visits of individual concepts as well as the navigation between them. We discovered that all visits are almost equally distributed between all concepts. There were only slight trails of increased navigation between certain concepts. This issue may be caused by the restricted way the ALEA system was used. The primary goal of students was not to master whole domain but to prepare themselves for a programming test. The second reason may be the existence of a guidance that the ALEA system already offers. We suppose that direct guidance implemented in the ALEA system (by means of the "Next" button) influenced such students' behavior and was proved in this way as appropriate.

5 Conclusions

The adaptation in most of current educational adaptive hypermedia systems is driven by a fixed set of rules. The construction of such rules is a complex task and the author may not be able to completely assess all intricacies found in different learning styles of

students. Techniques for data mining can provide knowledge needed either to recommend students concepts according to their characteristics or to assist the author of the course to improve the structure of the domain.

Main contribution of this paper is proposal of the recommender system, which is based on an idea of adaptation model development using data mining techniques. It connects the process of knowledge discovery with on-line recommendation of concepts based on discovered knowledge. The architecture of system allows using different underlying adaptive hypermedia systems, enabling us to share common knowledge (not specific to certain adaptive hypermedia system and its content). The architecture also incorporates the module for visualization of discovered knowledge helping the author of the course to prepare modification to the content.

In our approach, association rules, sequential patterns and traversal patterns represent automatically generated knowledge about students' behavior. Based on a student's current learning session and discovered patterns, the recommender system is able to recommend a sequence of concepts, which the student should study next.

Future development of the recommender system will focus on enhancing the set of data mining algorithms, e.g., to use clustering techniques in order to discover user clusters according their learning style. We plan to evaluate quality of recommendations carried out by our recommender system using data produced by other educational hypermedia systems. The evaluation will be based on a feedback from students as well as on results of the recommendation performed on a testing set of data.

Acknowledgement: This work was supported by Science and Technology Assistance Agency under the contract No. APVT-20-007104.

References

1. Brusilovsky, P.: Adaptive hypermedia. *User Modeling and User-Adapted Interaction Ten Year Anniversary Issue, 11*, 1/2 (2001), 87-110.
2. Kostelník, R. - Bieliková, M.: Web-based environment using adapted sequences of programming exercises. In: *Proc. of Int. Conf. on Information Systems Implementation and Modelling*, M. Beneš, Ed. (2003), 33-40.
3. Krištofič, A. - Bieliková, M.: Improving adaptation in web–based educational hypermedia by means of knowledge discovery. Submitted to: *HT 2005 – Sixteenth ACM Conference on Hypertext and Hypermedia*.
4. Mobasher, B. - Cooley, R. - Srivastava, J.: Automatic personalization based on web usage mining. *Communications of the ACM 43*, 8 (2000), 142-151.
5. Mobasher, B. - Dai, H. - Luo, T. - Nakagawa, M.: Using sequential and non-sequential patterns in predictive web usage mining tasks. In: *Proc. of the IEEE Int. Conf. on Data Mining* (2002), 669-672.

Príloha B Článok odoslaný na konferenciu Hypertext 2005

V prílohe je uvedený článok, v ktorom prezentujeme výsledky tejto diplomovej práce. Článok sme odoslali na medzinárodnú ACM konferenciu Hypertext 2005.

KRIŠTOFIČ, A. – BIELIKOVÁ, M.: *Improving adaptation in web–based educational hypermedia by means of knowledge discovery.* Submitted to: Hypertext 2005 – 16th ACM Conference on Hypertext and Hypermedia. 2005.

Improving Adaptation in Web-Based Educational Hypermedia by means of Knowledge Discovery

Andrej Krištofič
Faculty of Informatics and Information Technologies
Slovak University of Technology
Ilkovičova 3
Bratislava, Slovakia
kristofic@zoznam.sk

Mária Bieliková
Faculty of Informatics and Information Technologies
Slovak University of Technology
Ilkovičova 3
Bratislava, Slovakia
bielik@fiit.stuba.sk

ABSTRACT

Most adaptive web-based hypermedia systems adapt presentation of the content and/or navigation using predefined set of rules. Considering different behavior and preferences of each user it may be hard to generalize and construct all appropriate rules in advance. This problem is more noticeable in educational adaptive hypermedia systems, where adaptation to individual learning style of a student is important for the student to effectively assess particular domain. In this paper we present techniques for data mining, which can be used to discover knowledge about students' behavior during learning, as well as techniques, which take advantage of such knowledge to recommend students lessons they should study next. We also describe a process of recommendation based on knowledge discovery and present an architecture of a web-based system, which uses proposed approach to improve adaptation. Proposed architecture is independent of actual adaptive hypermedia system used.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia

Keywords

adaptive web-based educational hypermedia, knowledge discovery, adaptive navigation, concept recommendation, usage patterns

1. INTRODUCTION

Adaptive hypermedia systems have become popular in last few years in many application areas such as educational hypermedia, on-line information systems or information retrieval hypermedia [4]. Their growth is supported also by widespread use of the web as a basis for making a 'new education' possible, because it is available *anyplace* and *anytime* [8]. Adaptive hypermedia application in the field of educational systems is natural, as each student generally has different characteristics related to learning (goals, learning styles, preferences, etc.), so an adaptation to the student's characteristics can help the student to master topics more effectively.

However, it is quite difficult for an author of the course to develop appropriate adaptation model (usually represented using if-then rules). This requires respectable knowledge of students' behavior to create such adaptation rules, which will result into an effective adaptation of presented knowledge together with appropriate navigation in the course information space. We propose to use data collected by the educational adaptive hypermedia system during its usage to extract implicit, unknown and potentially useful information related to the adaptation from data. Methods and techniques of knowledge discovery provide us with discovering interesting knowledge, which can be employed to improve the adaptation model together with simplification of the adaptation model development.

The aim of this paper is to present a proposal to employ knowledge discovery for purpose of finding trends and patterns of a usage from data collected by the adaptive hypermedia educational system. These patterns reflect characteristic behavior of students (individuals or groups) during learning with the aid of the adaptive hypermedia system. Discovered patterns are used in the process of recommendation of relevant concepts to students, mainly for curriculum sequencing. We explore this proposal in the context of learning programming using programming exercises.

The rest of the paper is structured as follows. First, we give short overview of knowledge discovery in adaptive hypermedia in the context of related works. Next, in Section 3 we describe interesting data selected as a source for knowledge discovery in educational adaptive hypermedia. Section 4 discusses techniques appropriate for data mining with the aim of discovering interesting patterns in learners' behavior. In Section 5 we present the usage of discovered knowledge for recommendation in adaptive hypermedia. We describe design of a recommender system, which is based on discovering behavioral patterns and the usage of such discovered knowledge. The paper concludes with a summary, and a description of future directions of our research.

2. RELATED WORKS

Knowledge discovery in educational adaptive hypermedia usage data is a complete process, rather than a particular algorithm. It is aimed to reveal interesting patterns hidden in data collected during learning. The process consists of the basic steps identified for knowledge discovery (e.g., [9]) and later for web usage mining (e.g., [18]). The process begins with understanding application domain, stating the problem and collecting data. Next, acquired data is cleaned and preprocessed. This step may also include integration of data from various sources, selection of relevant attributes and data transformation for the next step – data mining. During the data mining step, various methods for pattern extraction or classification are applied. In the end of knowledge discovery process extracted knowledge is analyzed, interpreted and introduced into practical use (e.g., using for adaptation in educational hypermedia system).

Knowledge discovery for navigation recommendation is often based on usage data mining. Current approaches mainly fall on web usage mining area. Their primary goal is usually evaluating web site designs through an analysis of the browsing behavior of the web site users. Recently some authors focus on personalization issues rather than producing analytical knowledge (e.g., [17]).

There exists a variety of techniques for data mining, each with specific characteristics and possible usage. Commonly used techniques are cluster mining, association rules mining and sequential pattern mining. These techniques have drawn a lot of attention lately and a variety of algorithms have been developed (see survey in [14]). Other artificial intelligence approaches for discovering interesting patterns can be used. For example, in [19] the Education Prediction Rules tool for prediction rules discovery in adaptive hypermedia system using genetic algorithm is presented. Authors used adapted AHA! system as the source of usage data.

Following discovered patterns various approaches have been proposed for navigation recommendation (global or local, direct or indirect guidance). In [16] authors have proposed to conduct the recommendation directly with discovered patterns. The active user session is used to search the patterns and find the matching ones. Suffixes of these patterns become candidates for the recommendation. Only the most recent part (called window – w) of the active user session is used for matching, and it is matched against the patterns with length $|w|$[1]$+1$. When no recommendation can be made with current window, its size is decreased until it is possible to generate a recommendation.

Another approach [15] is aimed to group stored user sessions or discovered frequent item sets into session clusters using techniques of clustering. Representative usage profile is computed for each cluster. Active user session is then matched against computed usage profiles.

Mentioned approaches deal with one kind of patterns and provide support for this specific kind. We propose using several kind of patterns (e.g., sequential and traversal patterns) distinguished during the process of recommendation and describe software framework for a recommender system, which extends architecture of educational adaptive web-based hypermedia applications.

[1]where $|w|$ denotes the size of window

3. DATA FOR KNOWLEDGE DISCOVERY

Basic goal of knowledge discovery in an educational adaptive hypermedia system is to discover characteristic patterns of behavior of students, or groups of students during learning. Source data are substantial for sound results of knowledge discovery. We have identified several types of data, which represent useful source for knowledge discovery in the educational adaptive hypermedia system:

- user activity logs,
- user knowledge level (performance value),
- domain model structure.

In the following paragraphs we explain importance of each type of data for the process of knowledge discovery. As we experimented with two adaptive educational systems: AHA! [7] and ALEA [13], we provide examples and discussion related to data that can be found in these two adaptive hypermedia systems. However, our observations are valid also for other systems because data related to the user behavior in adaptive hypermedia systems do not differ significantly (common ontologies such as [12] and standards such as [11] can be used).

3.1 User activity logs

Almost every adaptive hypermedia system maintains its usage logs, which contain data related to users' activities. These logs are the main source of data for mining. The data show us how the educational system was being used by students – which concepts (knowledge elements) students visited and which information they were presented with.

Log entries usually contain *timestamps*, which are helpful in estimating the time a student spent on certain concept. Time can be used also to evaluate similarity between different user learning activities [10]. Time usage this way may originate inaccuracies though, because timestamps denote only the time of concept accessing, but not the time of concept studying. We can not say, whether the student really worked on displayed concept, or he was just idling. However, after preprocessing time access values (e.g., by filtering extreme values) it can be used as an auxiliary characteristic in the process of knowledge discovery. There exist approaches, which employ a client-side program to capture the time spent on information fragments more accurately [20].

Most of today's adaptive hypermedia systems are realized as web-based applications. Consequently, when there are no hypermedia system logs available, logs from the web server can be used instead. Naturally, additional data preprocessing and cleaning has to be carried out.

Our proposal considers following user action attributes usable in the process of knowledge discovery:

- *timestamp*
  defines time when action occurred. It is used to order

actions in time and to compute the time a user spent on certain concept;

- *user identification*
  having each user uniquely identified enables us to perform knowledge discovery under actions of individual user and allows personalization;

- *type*
  allows reasoning on different levels. We defined following elementary action types (with possibility of an extension):

  - *Login* and *Logout* which are used to delimit user's learning session;
  - *ConceptVisit* and *FragmentDisplay*, which carry information about visited concepts by particular user and information fragments he was presented with;

- *object*
  specifies identification of an object on which the action is carried out. For example for the *ConceptVisit* action it holds an identification of the concept visited by the user.

3.2 Domain model structure

Domain model structure is important source for discovering interesting patterns in usage data. Domain model of an educational hypermedia system is composed of a set of domain knowledge elements [5], which represent elementary fragments of knowledge for the given domain. They can be named differently in various educational hypermedia systems, e.g. concepts, knowledge items, topics, knowledge elements, learning outcomes. We denote elementary fragments of domain knowledge in this paper as concepts. Moreover, we distinguish concepts and fragments. Fragments represent actual information content, i.e. knowledge in case of educational hypermedia system. Differentiation of concepts and fragments enables more accurate data mining as various concept-fragment relations can exist. However, our approach is applicable also in situations where only concepts are considered.

One possible structuring of domain concepts in educational adaptive hypermedia systems is a hierarchy [5]. By mapping concepts to their parents in the concept hierarchy, we are able to work with concepts on different levels and thus discover more general usage patterns.

Several different types of concepts and fragments can exist in the educational adaptive hypermedia system. For example in the context of learning programming using program exercises, the concept type can provide information whether the concept contains general knowledge (e.g., a programming scheme) or it contains specific knowledge (e.g., an exercise together with an example of source code – a specific application of the programming scheme). Providing availability of information about the concept type or fragment type we can look for more general behavioral patterns. Instead of working with concrete concepts, we can work with their types only on different levels of abstraction and generalization.

In the context of learning programming we use the following concept and fragment types (inspired by the ALEA system [13]):

- Concept types
  - Text
  - Programming scheme
  - Exercise
  - Test
- Fragment types
  - Text
  - Exercise definition
  - Exercise hint
  - Exercise solution
  - Source code

3.3 User knowledge level

For the task of knowledge discovery it is important to know knowledge level related to particular concept for each student. Usually, we are interested in mining data of "successful" students, i.e. those with high knowledge level of visited concepts (or good performance results), because our goal is to help students to successfully learn the domain. Estimating the user knowledge level is a complex task and it is usually performed by the educational adaptive hypermedia system itself. After the knowledge level for certain concept and user is determined it is stored into the user model.

The results of on-line tests offered by many educational adaptive hypermedia systems can also be valuable (and possibly more precise) estimation of users' knowledge. Along with students' results provided externally (e.g., results of "paper" tests or overall score in the course) they can help to assess the overall knowledge level of students more precisely. Overall level of knowledge is used for determination of a subset of acquired data used in the next step of knowledge discovery process for data mining.

3.4 Examples of data for knowledge discovery

In this section we provide examples of data for knowledge discovery provided by two above mentioned adaptive hypermedia systems AHA! and ALEA.

AHA!

AHA! is an adaptive web-based hypermedia system developed at the Eindhoven University of Technology [7]. As of upcoming version 3.0, the system records information about users' activities either into the database or into a separate XML file. Log entry is created every time a resource is being accessed. One entry contains a timestamp, identification of the user's session, name of accessed resource, identification of the user and a flag, which marks an access to the fragment. Even though AHA! does not record the Login and Logout actions, learning session of the user can still be identified thanks to the user's session field in the log entry. The AHA! system also logs changes in the user's model. This allows considering the context in which the action occurred during the process of knowledge discovery.

AHA! allows building a complex hierarchy of concepts, since arbitrary concept relationship types can be defined. Moreover, it is possible to assign certain type to each concept. The system maintains a table of concepts, which is filled in from the XML file. This enables defining concept and fragment types as described in Section 3.2.

The system uses overlay user model, i.e. values of all concept attributes for particular user are stored and maintained. Concepts representing domain knowledge fragments usually define the *knowledge* attribute, which represents user's level of knowledge about the concept. The value of the knowledge attribute (and the other attributes as well) is updated when the user reads related page. AHA! also offers possibility to create an on-line test, which results are used to update the value of the knowledge attribute. This way we can assess the knowledge of the user more precisely than just considering visits of fragments.

ALEA

Educational adaptive web-based hypermedia system ALEA (Adaptive LEArning) is used in the course Functional and logic programming at the Slovak University of Technology [13] since academic year 2002/2003. We have used the data gathered by this system to verify an approach for knowledge discovery in educational adaptive hypermedia proposed in this paper. ALEA stores detailed information about users' actions during learning into the database. Each time the user selects a concept or fragment, visits the fragment or explicitly changes his preferences, a log entry is created. It contains a timestamp, identification of the user, type of the action and an optional parameter of the action. Unlike AHA!, changes in the user model are not logged in ALEA. ALEA maintains only the most recent version of the user model. It means that we are only aware of the outcome of actions – the resulting user model. We do not know in which context (i.e., state of the user model) particular action occurred. This can be considered as a disadvantage, though it does not influence knowledge discovery at the level proposed here. For performing more detailed analysis of data, it seems useful to have such information.

The domain model is represented using XML. The XML file contains definition of concepts and relations between concepts. ALEA employs several concept relation types and concept types as well, which can be easily extended. Information fragments are stored in separate XHTML files and their types along with their mappings to the concepts are stored in the database.

For each visited concept ALEA stores a level of user's knowledge related to domain knowledge represented by the concept and the time when the user last visited concept together with the overall number of visits. The ALEA system maintains also information whether the user marked the concept as "understood".

4. MINING ADAPTIVE HYPERMEDIA USAGE DATA

Main source of data for knowledge discovery in an educational adaptive hypermedia system are logs of users' learning activities. These logs contain records about every single user action in the adaptive hypermedia system. As a result of preprocessing, we get learning sessions of all users identified (one learning session is delimited by the Login and Logout actions). The goal of data mining step in educational adaptive hypermedia is to discover characteristic patterns of navigating during users' learning sessions. We propose using the following techniques for accomplishing this task: association rules mining, sequential patterns mining and traversal patterns mining. Association rules and sequential associations are two notable types of web pattern generally. Traversal patterns can be denoted as a special case of sequential patterns proposed for mining a navigation in both open or closed information spaces.

Association rules mining. This technique was originally proposed to analyze customer transactions in the market-basket domain. It is commonly used for example in analysis of contents of on-line shopping carts. In this case it helps to find groups of related products based on preferences of customers. In educational adaptive hypermedia systems we employ association rules mining to find relations between concepts. Even though the technique does not take into account the order of concepts, its results can still be used in the process of recommending relevant concepts. Nevertheless, discovered relations between concepts are more useful for the author of information space who can modify the course to better suit users' needs. There exist several algorithms for association rules mining. One of the first presented algorithms for this technique is *Apriori* algorithm [1].

Sequential patterns mining. Sequential patterns mining is technique similar in some sense to association rules mining. Unlike association rules this technique takes into account the order of concepts visited. However, the concepts do not necessarily have to be adjacent. That is why the interpretation of sequential patterns in context of navigation within the hypermedia information space could be quite difficult. We only know that users have frequently visited the concepts in certain order, but we cannot tell, which concepts they had visited in the meantime. These patterns are still useful for recommending concepts, which user should visit in the future. There exist modifications of Apriori algorithm for sequential pattern discovery, e.g., *Apriori-All* and *Apriori-Some* algorithms [2]. The other algorithms (e.g., SPAM [3]) focus on improving performance for mining on large databases.

Traversal patterns mining. Traversal patterns, sometimes also referenced as contiguous sequential patterns, are used for discovering frequent subsequences of students' learning sessions. They are often used in web server logs analysis. Traversal patterns form the base for recommending relevant concepts in our approach. When properly visualized, they can be also helpful to the author of the course as they show preferred "paths" that students tend to follow in the educational hypermedia system. The representative algorithms for traversal patterns mining are *Full Scan* and *Selective Scan* algorithms [6].

It is worth noting that patterns discovered using above mentioned techniques share some common traits. They all represent sequences or sets of items (concepts, in our case). The algorithms used for their discovery share the same idea of

generating and pruning the set of candidate patterns. Thus, we can take advantage of this similarity and store discovered patterns in common knowledge base. We can also generalize the algorithms for pattern discovery in educational adaptive hypermedia and introduce common software framework shared between different algorithms.

5. RECOMMENDER SYSTEM FOR ADAPTIVE HYPERMEDIA

Patterns revealed during knowledge discovery process are used for recommendation of relevant concepts. The result of recommendation is a sequence of concepts, which student should probably visit next, together with numeric representation of suitability of these concepts.

Because we are dealing with several kinds of patterns, it is necessary to distinguish between recommendations based on each kind. We proposed ordering of pattern kinds by their suitability for recommendation in educational adaptive hypermedia systems (see Figure 1).

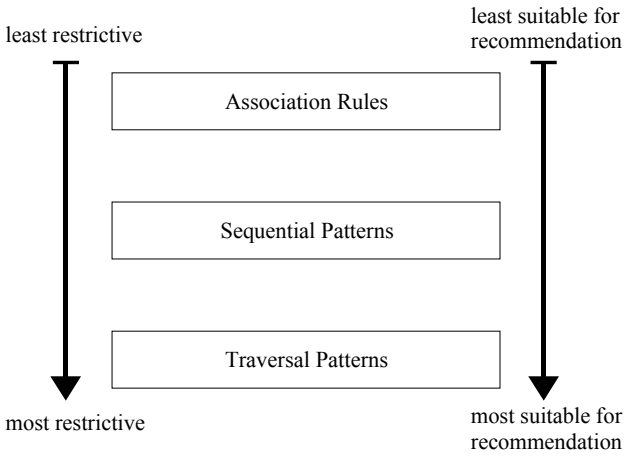


Figure 1: Suitability of various patterns for recommendation.

We consider traversal patterns the most reliable base for recommendation, because in educational adaptive hypermedia systems the proper order of concepts is crucial for the student to master a subject. Certainly, sequential patterns as well as association rules can be also used for recommendation but they provide broader guidance than traversal patterns. Therefore the suitability of recommended concepts has to be evaluated with respect to the kind of patterns on which the recommendation was based.

5.1 Recommender system architecture

Proposed recommender system as a back-end for an adaptive hypermedia system is based on modular software architecture. The software architecture (see Figure 2) follows the flow of two main processes carried out in the system:

- knowledge discovery and
- concept recommendation.

The system is designed to be independent of underlying adaptive hypermedia application as much as possible. Therefore the architecture incorporates a wrapper, which acts as a facade between an adaptive hypermedia system and the recommender system itself. This approach allows using various adaptive hypermedia systems as sources of data for knowledge discovery without modification of the recommender system itself. The architecture is not restricted by application area of the adaptive hypermedia system (i.e., its usage is broader than educational adaptive hypermedia).

Wrapper

All the adaptive hypermedia system dependent tasks are performed in the wrapper module. This includes usage data retrieval as well as tasks related to the processing and deploying generated adaptation rules, and the user model update. The wrapper module hence defines common interface for the access to the adaptive hypermedia system resources and data. Using the recommender system with several adaptive hypermedia systems requires a wrapper module for every adaptive hypermedia system. This module forms also the interface to an authoring tool, where adaptation knowledge are defined by the author.

Data preprocessing

The data preprocessing module takes the stream of user actions as its input. These actions are preprocessed and searched for potential errors and inconsistencies. The module also takes care of identification of users sessions. Sessions are filtered according to their length (short sessions are dropped) and/or overall knowledge level of respective user. Finally, the module stores sessions into database of the recommender system. Information about the domain structure and types of concepts and fragments is also stored here.

Data mining

In the data mining module various algorithms for pattern mining (see Section 4) are applied on data stored in database. Traversal patterns, sequential patterns and association rules are discovered using different views on data (e.g., different levels of concepts in domain hierarchy, concept types). Discovered patterns are stored into the knowledge base.

Knowledge presentation

The main purpose of the knowledge presentation module is to present selected discovered knowledge (e.g., association rules) to the author of a course. The author can then make modifications to the domain structure and the contents of the information fragments.

Rule generator

The task of the rule generator module is to construct static adaptation rules (i.e., rules where current learning session of a student is not considered) based on patterns in the knowledge base. These rules are deployed to the adaptive hypermedia system through the wrapper.

Recommender module

The recommender module creates a sequence of recommended concepts based on the current session of a user and the

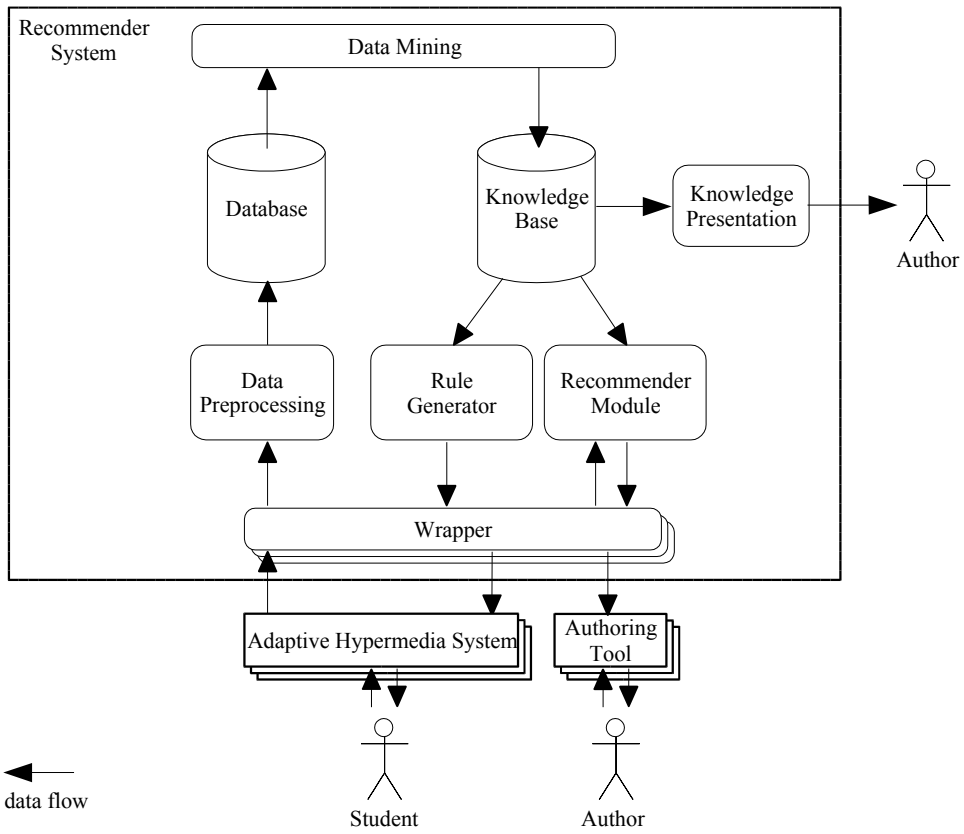


Figure 2: Architecture of recommender system.

knowledge stored in the knowledge base. The concepts in the current user session are matched against the ones in patterns of the knowledge base. When appropriate (similar) patterns are found, the concept recommendation is carried out. The result is a sequence of recommended concepts and the evaluation of their suitability and relevance for the user. This sequence is generated on demand of the adaptive hypermedia system (each time the user moves to another concept or fragment, or after some time period or at the beginning of each session).

5.2 Use case scenario

Sequential diagram in Figure 3 describes typical usage scenario of proposed recommender system. While users (students in case of educational adaptive hypermedia system) work with the adaptive hypermedia system, it logs and stores all their actions (e.g. logins, logouts, concept visits). When scheduled, the recommender system retrieves usage data, preprocesses it and stores it into database. Data mining algorithms are executed on stored usage data and usage patterns are discovered and stored in the knowledge base.

With knowledge base filled up, the recommender system can recommend a sequence of relevant concepts for particular user, when asked to do so. An adaptive hypermedia system provides the recommender system with the user identification and his current session and the recommender system returns recommended sequence of concepts, which the user should visit next. The adaptive hypermedia system may annotate concepts in the sequence according to suitability evaluation provided by the recommender system.

When author of the content decides to improve it, visualization of discovered knowledge can help to decide which modifications should be made. The author can also compare historical records and evaluate the effect of his previous changes to the adaptive hypermedia system content.

5.3 Experiments

To verify our approach to knowledge discovery in adaptive web-based hypermedia we developed a prototype of the recommender system. We implemented three data mining algorithms for pattern discovery (traversal patterns, sequential patterns and association rules) suitable for educational hypermedia applications.

In some experiments with traversal patterns mining we created also maximal forward references [6] where returns to previously visited fragments are ignored. We have found that in this way loss of information is not acceptable. This is a consequence of the fact that the navigation in educational hypermedia has different characteristics as a navigation (browsing) in the web information space. The return back action during learning session is important part of behavioral patterns in educational hypermedia systems and should not be ignored.

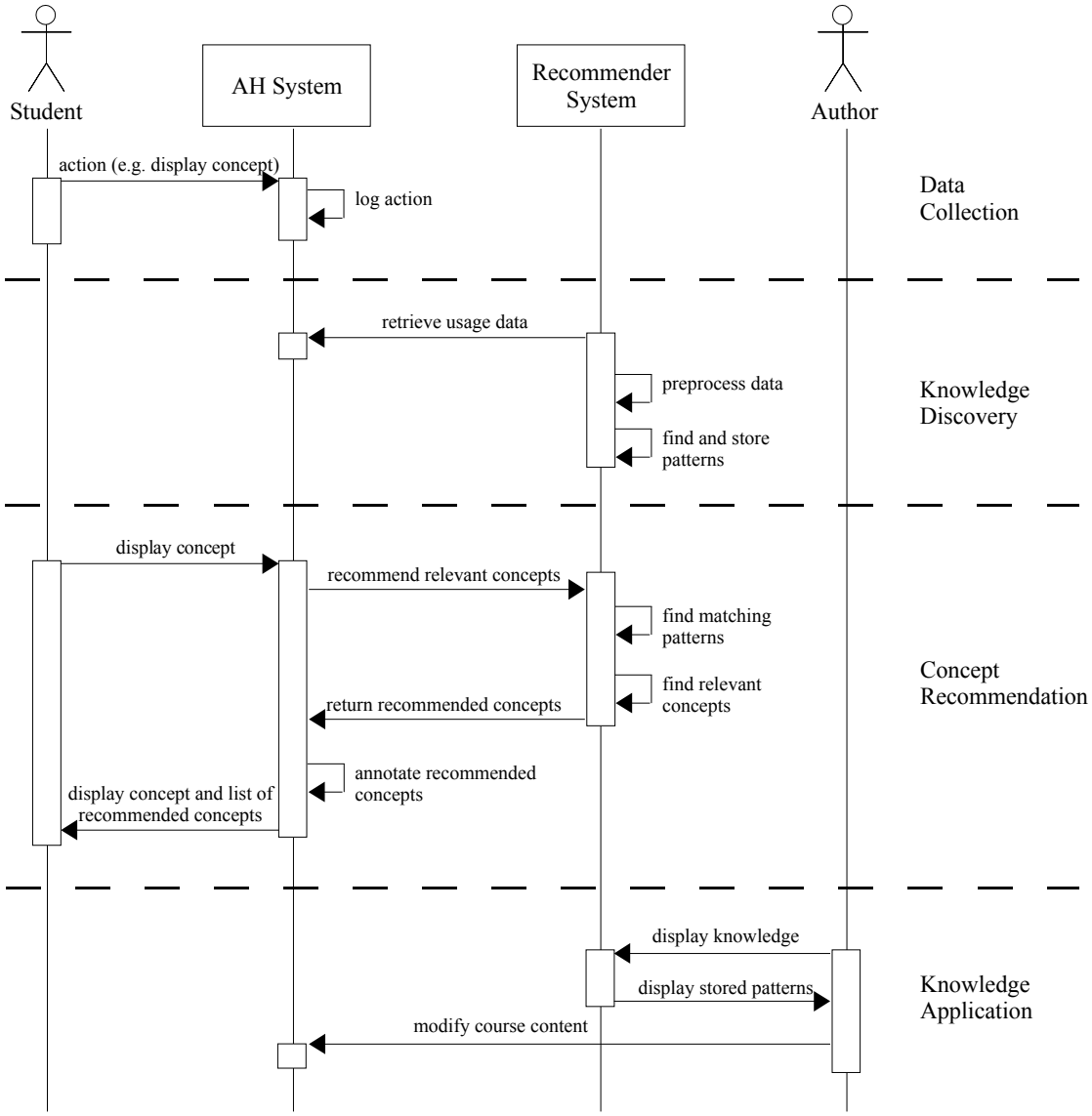


Figure 3: Typical scenario of the recommender system usage.

Results of the knowledge discovery are markedly dependent on source data. We chose the ALEA educational adaptive hypermedia system [13] for our experiments since we have had available usage logs gathered during 3 years of its usage in the Functional and logic programming course. We successfully preprocessed usage logs and were able to identify learning sessions of individual users. At time of experiments the database contained 1 170 sessions.

Experimenting with the ALEA data, we found out that even though algorithms find some patterns in the data, these patterns have quite low support (around 15%). To outline this fact, we analyzed visits of individual concepts as well as the navigation between them. We discovered that all visits are almost equally distributed between all concepts. There were only slight trails of increased navigation between certain concepts. This issue may be caused by the restricted way the ALEA system was used. The primary goal of students was not to master whole domain but to prepare themselves for a programming test. The second reason may be the existence of a guidance that the ALEA system already offers. Visualizations of concept visits by students showed that the students navigated through the information space of the ALEA more or less evenly. We suppose that direct guidance implemented in the ALEA system (by means of the "Next" button) influenced such students' behavior and was proved in this way as appropriate.

6. CONCLUSIONS AND FUTURE WORK

The adaptation in most of current educational adaptive hypermedia systems is driven by a fixed set of rules. The construction of such rules is a complex task and the author may not be able to completely assess all intricacies found in different learning styles of students. Techniques for data

mining can provide knowledge needed either to recommend students concepts according to their characteristics or to assist the author of the course to improve the structure of the domain.

Main contribution of this paper is a proposal of adaptation model development support using data mining techniques. We analyzed data provided by educational adaptive hypermedia systems, which can be useful in the process of knowledge discovery. Usage data (i.e., logs of users' actions) are valuable source for knowledge discovery. We selected three well-known techniques for mining patterns in such data. In our approach, association rules, sequential patterns and traversal patterns represent automatically generated knowledge about students' behavior. Based on a student's current learning session and discovered patterns, the recommender system is able to recommend a sequence of concepts, which the student should study next.

Important outcome presented in this paper is architecture of the recommender system, which connects the process of knowledge discovery with on-line recommendation of concepts based on discovered knowledge. The architecture of system allows to use different underlying adaptive hypermedia systems, enabling us to share common knowledge (not specific to certain adaptive hypermedia system and its content). The architecture also incorporates the module for visualization of discovered knowledge helping the author of the course to prepare modification to the content.

Future development of the recommender system will focus on enhancing the set of data mining algorithms, e.g., to use clustering techniques in order to discover user clusters according their learning style. We plan to evaluate quality of recommendations carried out by our recommender system using data produced by other educational hypermedia systems. The evaluation will be based on a feedback from students as well as on results of the recommendation performed on a testing set of data. We also intend to conduct more experiments with the other educational adaptive hypermedia systems and with their different content with primary attention to AHA! considering also data related to user model changes.

7. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. S. P. Chen, editors, *11th Int. Conf. on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[3] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 429–435, 2002.

[4] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, Ten Year Anniversary Issue, 11(1/2):87–110, 2001.

[5] P. Brusilovsky. *Authoring Tools for Advanced Technology Learning Environments*, chapter Developing Adaptive Educational Hypermedia Systems: from Design Models to Authoring Tools, pages 377–409. Kluwer Academic Publishers, 2003.

[6] M. S. Chen, J. S. Park, and P. S. Yu. Data mining for path traversal patterns in a web environment. In *16th Int. Conf. on Distributed Computing Systems*, pages 385–392, 1996.

[7] P. De Bra, A. Aerts, B. Berden, B. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. AHA! the adaptive hypermedia architecture. In *HYPERTEXT'03: Proc. of the 14th ACM Conf. on Hypertext and Hypermedia*, pages 81–84. ACM Press, 2003.

[8] P. De Bra, L. Aroyo, and A. Cristea. *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, chapter Adaptive Web-Based Educational Hypermedia, pages 387–410. Springer Berlin, 2004.

[9] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. E. Uthurusamy. *Advances in Knowledge Discovery in Data Mining*. AAAI Press/MIT Press, 1996.

[10] S. Gündüz and M. Tamer Özsu. A web page prediction model based on click-stream tree representation of user behavior. In *KDD'03: Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 535–540. ACM Press, 2003.

[11] IEEE. IEEE P1484.2/D7, 2000-11-28. Draft Standard for Learning Technology. Public and Private Information (PAPI) for Learners. Available at http://ltsc.ieee.org/, 2000.

[12] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. Butler, and L. Tran. Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies 1.0. W3C Recommendation, January 2004.

[13] R. Kostelník and M. Bieliková. Web-based environment using adapted sequences of programming exercises. In M. Beneš, editor, *Proc. of Int. Conf. on Information Systems Implementation and Modelling*, pages 33–40, 2003.

[14] M. Koutri, N. Avouris, and S. Daskalaki. *Adaptable and Adaptive Hypermedia Systems*, chapter A survey on web usage mining techniques for web-based adaptive hpermedia systems. Idea Publishing Inc., 2004.

[15] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[16] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Proc. of the IEEE Int. Conf. on Data Mining*, pages 669–672, 2002.

[17] B. Mobasher and M. Nakagawa. Impact of site characteristics on recommendation models based on association rules and sequential patterns. In *Proc. of the IJCAI'03 Workshop on Intelligent Techniques for Web Personalization*, 2003.

[18] D. Pierrakos, G. Paliouras, C. Papatheodorou, and C. Spyropoulos. Web usage mining as a tool for personalization: A survey. *User Modeling and User-Adapted Interaction*, 13(4):311–372, 2003.

[19] C. Romero, S. Ventura, P. De Bra, and C. de Castro. Discovering prediction rules in AHA! courses. In *Proceedings of the User Modeling Conference*, pages 25–34, 2003.

[20] C. Shahabi, A. M. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users web-page navigation. In *RIDE '97: Proc. of the 7th Int. Workshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications*, page 20. IEEE Computer Society, 1997.

Príloha C Algoritmy na spracovanie údajov a odporúčanie

Algoritmus na identifikáciu prístupov používateľov

Vstup: postupnosť akcií *A*

Výstup: množina identifikovaných prístupov používateľov *P*

pre každú $a \in A$

ak *a* = *Login* **potom**

ak ∃ *prístup* [*a.používateľ*] **potom**

varovanie “Používateľ sa neodhlásil“

$P = P \cup$ *prístup* [*a.používateľ*]

vytvor *prístup* [*a.používateľ*]

inak

vytvor *prístup* [*a.používateľ*]

koniec

koniec

ak *a* = *Logout* **potom**

ak ∃ *prístup* [*a.používateľ*] **potom**

$P = P \cup$ *prístup* [*a.používateľ*]

inak

varovanie “Používateľ nebol prihlásený“

koniec

koniec

ak *a* = *ConceptVisit* **alebo** *a* = *FragmentDisplay* **potom**

ak ∃ *prístup* [*a.používateľ*] **potom**

a.trvanie = vypočítaj trvanie akcie

prístup[*a.používateľ*] = *prístup* [*a.používateľ*] +*a*

koniec

koniec

koniec

výstup *P*

Kostra algoritmu na objavovanie vzorov správania sa

Ide o zovšeobecnenie algoritmu Apriori [1]. Funkcie *generuj kandidátov* a *podporuje* sú špecializované pre objavovanie asociačných pravidiel a (spojitých) sekvenčných vzorov.

Vstup: množina prístupov používateľov P, minimálna podpora *min_pod*

Výstup: množina objavených vzorov V

V_1 = vyber koncepty, ktorých podpora > *min_pod*

k = 2

opakuj pokiaľ $V_{k-1} \neq \emptyset$

C_k = generuj kandidátov (V_{k-1})

pre každý $p \in P$

pre každý $c \in C_k$

ak podporuje (p,c) **potom**

c.podpora = c.podpora + 1

koniec

koniec

koniec

V_k = vyber kandidátov, ktorých podpora > *min_pod*

koniec

$V = \cup_k V_k$

výstup V

Kostra algoritmu na odporúčanie konceptov

Predstavuje spoločný základe pre odporúčanie konceptov na základe asociačných pravidiel a (spojitých) sekvenčných vzorov. Funkcia *vyhľadaj uzly* je špecifická pre každý druh vzorov a zabezpečuje vyhodnotenie podobnosti prístupu používateľa a vzoru. Jej výstupom je množina uzlov v strome vzorov, ktoré predstavujú posledné prvky vzorov, ktoré sú podobné zadanému prístupu používateľa.

Vstup: aktuálny prístup používateľa s, veľkosť okna w, strom vzorov t

Výstup: postupnosť odporučených konceptov K

opakuj pokiaľ $w > 0$

okno = posledných w prvkov s

uzly = vyhľadaj uzly (*okno*, t)

pre každý $u \in$ *uzly*

potomkovia = nájdi priamych potomkov uzla u v strome t

pre každý $p \in$ *potomkovia*

relevantnosť = *p.podpora* / *u.podpora*

$K = K +$ (*p.koncept*, *relevantnosť*)

koniec

koniec

ak $|K| > 0$ **potom**

ukonči cyklus

inak

$w = w - 1$

koniec

koniec

usporiadaj K podľa relevantnosti

výstup K

Príloha D Príručka pre správcu systému

Požiadavky systému

Hardvér (odporučená konfigurácia):

- *procesor:* Intel – kompatibilný s taktovacou frekvenciou minimálne 1GHz
- *pamäť:* 256MB RAM
- *diskový priestor:* systém a podporné knižnice vyžadujú 6MB diskového priestoru (tento priestor nezahŕňa databázu systému ani jeho bázu znalostí)

Softvér:

- operačný systém s podporou prostredia pre vykonávanie Java programov
- prostredie pre vykonávanie Java program – Java 2 Runtime Environment, Standard Edition, verzia 1.4.2
- relačný databázový systém s podporou rozhrania JDBC a príslušné JDBC ovládače
- všetky ostatné potrebné softvérové knižnice sú súčasťou inštalácie systému

Inštalácia systému

1. Nakopírujte súbory systému do zvoleného adresára. Súbory sa nachádzajú na CD v adresári *Prototype/AHminer*.
2. Vytvorte databázu systému a vytvorte jej štruktúru. SQL príkazy na vytvorenie štruktúry prázdnej databázy nájdete na CD v adresári *Data/AHminer* v súbore *empty.sql*. Podrobnejšie inštrukcie ako vytvoriť a inicializovať databázu nájdete v manuáli pre váš databázový systém.
3. Zabezpečte, aby sa JDBC ovládač pre váš databázový systém nachádzal v niektorom z adresárov, ktoré sú uvedené v premennej prostredia *CLASSPATH*.

Konfigurácia systému

Systém sa konfiguruje pomocou konfiguračných súborov umiestnených v adresári *config* systému:

config.xml

Hlavný konfiguračný súbor aplikácie vo formáte XML. Jeho definíciu nájdete v adresári *dtd* v súbore *ahminer.dtd*. Ukážkový konfiguračný súbor nájdete už vytvorený v adresári *config*. Celá konfigurácia sa musí nachádzať v sekcii *ahminer.* Ďalej opíšeme jednotlivé časti konfiguračného súboru:

- *database* – nastavenia databázy systému, sekcia musí byť umiestnená v sekcii *ahminer*

  Atribúty sekcie:

  - driverClass – celé meno triedy JDBC ovládača pre databázový systém
  - url – adresa databázy systému
  - user – meno používateľa pre prístup k databáze
  - password – prístupové heslo k databáze

  Obsah sekcie: žiadny

- *dataImport* – nastavenia parametrov pre predspracovanie údajov, sekcia musí byť umiestnená v sekcii *ahminer*

  Atribúty sekcie:

  - minSessionLength – minimálna dĺžka prístupu používateľa, ktorá bude akceptovaná pri predspracovaní údajov, všetky kratšie prístupu budú ignorované

  Obsah sekcie: žiadny

- *knowledgeBase* – nastavenia bázy znalostí, sekcia musí byť umiestnená v sekcii *ahminer*

  Atribúty sekcie:

  - directory – celá cesta k adresáru, v ktorom budú uložené súbory bázy znalostí; **používateľ**, ktorý systém spúšťa musí **mať práva na zápis a čítanie** z tohto adresára.

  Obsah sekcie: žiadny

- *AHsystems* – sekcia obsahuje konfigurácie jednotlivých AH systémov, musí byť umiestnená v sekcii *ahminer*

  Atribúty sekcie: žiadne

  Obsah sekcie: sekcie *AHsystem* pre jednotlivé AH systémy (pozri nižšie)

- *recommender* – nastavenia modulu pre odporúčanie konceptov, sekcia musí byť umiestnená v sekcii *ahminer*

  Atribúty sekcie: žiadne

  Obsah sekcie: sekcia *xml-rpc* na konfiguráciu rozhrania pre odporúčanie (pozri nižšie)

- *control* – nastavenie rozhrania pre ovládanie systému, sekcia musí byť umiestnená v sekcii *ahminer*

  Atribúty sekcie: žiadne

  Obsah sekcie: sekcia *xml-rpc* na konfiguráciu rozhrania pre ovládanie systému (pozri nižšie)

- *AHsystem* **–** konfigurácia konkrétneho AH systému, sekcia s môže nachádzať iba v sekcii *AHsystems*

  Atribúty sekcie:

  - name –jedinečné meno AH systému, slúži ako identifikácia systému
  - wrapperClass – celé meno triedy, ktorá implementuje zapúzdrovací modul pre AH systém

  Obsah sekcie: sekcie *parameter*, na zadanie parametrov pre zapúzdrovací modul (pozri nižšie). O tom, aké parametre zapúzdrovací modul podporuje sa dočítate v jeho dokumentácii.

- *xml-rpc* – nastavenie XML-RPC servera pre ovládanie systému resp. odporúčanie, sekcia musí byť umiestnená v sekcii *recommender* aj *control*

  Atribúty sekcie:

  - address – adresa, na ktorú bude naviazané rozhranie pre ovládanie systému
  - port – číslo portu, na ktorom bude akceptovať požiadavky server na ovládanie systému

  Obsah sekcie: žiadny

- *parameter* **–** nastavenie parametra pre zapúzdrovací modul AH systému, sekcia sa môže nachádzať iba v sekcii *AHsystem*

  Atribúty sekcie:

  - name – meno parametra
  - value – hodnota parametra

  Obsah sekcie: žiadny

log4j.xml

Konfiguračný súbor pre knižnicu log4j, ktorú sme použil na vytváranie správ pre používateľa. Systém prichádza sa nastavením, pri ktorom sa správy vypisujú na štandardný výstup. O ďalších možnostiach konfigurácie tejto knižnice (napr. zápis správ do súborov) sa dočítate na adrese http://logging.apache.org/log4j/docs/ .

Pridanie nového AH systému

Na pridanie nového AH systému použite nasledujúce kroky:

1. Pridajte novú sekciu AHsystem do sekcie AHsystems konfiguračného súboru, kde nastavíte meno systému a celé meno triedy zapúzdrovacieho modulu. Pridajte tiež parametre pre zapúzdorovací modul, ak je to potrebné. Príklad:

```
<AHsystem name = "Alea-FLP" wrapperClass = "aleawrapper.AleaWrapperNew">
        <parameter name="systemPath" value="C:\School\ROK5\DP\alea-v1.1"/>
</AHsystem>
```

2. Zabezpečte, aby sa trieda zapúzdrovacieho modulu pre váš AH systém nachádzala v niektorom z adresárov, ktoré sú uvedené v premennej prostredia *CLASSPATH*. Na tento účel môžete použiť aj adresár *wrappers* v adresári systému, ktorý je automaticky zahrnutý do premennej *CLASSPATH*.

3. Ak si prajete využívať odporúčania vo vašom AH systéme, nastavte prepojenie vášho AH systému so systémom na odporúčanie navigácie (podrobnosti nájdete v dokumentácii príslušného AH systému). Adresu a port rozhranie pre odporúčanie konceptov zistíte v konfiguračnom súbore.

Používanie a ovládanie systému

Hlavnou triedou systému je trieda *AhMiner*, ktorú spustíme príkazom (aktuálny adresár, je adresár systému):

```
java –classpath ahminer.jar ahminer.AhMiner
```

Po spustení systém inicializuje svoje moduly a vytvorí XML-RPC rozhrania pre odporúčanie konceptov (štandardne na porte 1605) a pre ovládanie systému (štandardne na porte 1981). Nastavenie portov je možné zmeniť v konfiguračnom súbore. Systém môžete ovládať pomocou XML-RPC rozhrania, ktoré vytvára objekt *controller* s nasledujúcimi metódami:

- refresh(String system)

  Pomocou tejto metódy zabezpečíte načítanie nových akcií zo špecifikovaného AH systému a následné obnovenie bázy znalostí vyhľadaním vzorov. *Upozornenie: Tieto operácie môžu trvať dlhší čas v závislosti od množstva nových údajov.*

- showPatterns(String system, String type)

  Metóda zobrazí obsah bázy znalostí – strom vzorov uvedeného typu, pre zvolený AH systém. Typ vzorov môže byť *association, sequential* alebo *traversal*.

- shutdown()

  Metóda umožní bezpečné vypnutie systému.

Uvedené metódy môžete zavolať pomocou nástroja na ovládanie systému *AhMinerController*, ktorý spustíte príkazom

```
java –classpath ahminer.jar ahminer.AhMinerController
```

Jeho argumentmi sú:

- *adresa* rozhrania pre ovládanie systému
- *port, na ktorom je* spustené rozhranie pre ovládanie systému
- *meno metódy*, ktorú chcete zavolať
- *parametre metódy* (ak sú potrebné)

V prípade, že si prajete načítavať údaje z AH systémov v pravidelných intervaloch, odporúčame použiť špecializovanú aplikáciu na plánovanie spúšťania, ktorá zabezpečí spustenie nástroja na ovládanie systému s vhodnými parametrami v zvolený čas. V operačnom systéme Windows je možné použiť napríklad program *at*, v Unix-like operačných systémoch aplikáciu *cron* alebo *anacron*.

Prepojenie systému s AH systémami

V tejto časti príručky sa budeme venovať postupu prepojenia systému na odporúčanie navigácie s inými AH systémami. Prepojenie vyžaduje dva kroky:

1. Implementáciu zapúzdrovacieho modulu na strane systému na odporúčanie navigácie
2. Implementáciu pripojenia na rozhranie pre odporúčanie a zobrazenie odporúčaných konceptov na strane AH systému

Implementácia zapúzdrovacieho modulu

Modul je realizovaný ako trieda v jazyku Java, ktorá implementuje rozhranie *ahminer.dataimport.Wrapper*. Modul teda musí zabezpečiť načítanie akcií z AH systému, získanie informácii o doméne a typoch konceptov a fragmentov. Samozrejeme, musí vykonať mapovanie typov akcií, konceptov a fragmentov na typy, ktoré sa používajú v systéme na odporúčanie navigácie (pozri balík *ahminer.types*). Nasleduje zoznam metód rozhrania *Wrapper*, spolu s ich opisom:

- boolean initialize( )

  Metóda sa zavolá pred prvým použitím wrappera. Zabezpečuje jeho inicializáciu, ak je potrebná.

  Vracia: *true* ak bola inicializácia úspešná, inak *false*.

- FragmentType getFragmentType(String fragmentName)

  Metóda zistí typ zadaného fragmentu.

  Argumenty:

  - fragmentName – technické meno fragmentu v AH systéme

  Vracia: Typ fragmentu.

- int getFragmentLevel(String fragmentName)

  Metóda vráti pozíciu (úroveň vnorenia) fragmentu v modeli domény.

  Argumenty:

  - fragmentName – technické meno fragmentu v AH systéme

  Vracia: Úroveň vnorenia fragmentu v strome domény.

- ConceptType getConceptType(String conceptName)

  Metóda zistí typ zadaného konceptu.

  Argumenty:

  - conceptName – technické meno konceptu v AH systéme

  Vracia: Typ konceptu.

- String getConceptParent(String conceptName)

  Metóda vyhľadá rodičovský koncept k zadanému konceptu.

  Argumenty:

  - conceptName – technické meno konceptu v AH systéme

  Vracia: Typ konceptu.

- int getConceptLevel(String conceptName)

  Metóda vráti pozíciu (úroveň vnorenia) konceptu v modeli domény.

  Argumenty:

  - conceptName – technické meno konceptu v AH systéme

  Vracia: Úroveň vnorenia konceptu v strome domény.

- List getActions( )

  Úlohou metódy je vrátiť všetky dostupné akcie používateľov z AH systému.

  Vracia: Zoznam všetkých akcií používateľov v AH systéme.

- List getActions(Date date)

  Úlohou metódy je vrátiť akcie používateľov z AH systému, ktoré novšie ako je zadaný dátum.

  Argumenty:

  - date – dátum, od ktorého sa majú akcie z AH systému získať

  Vracia: Zoznam akcií používateľov v AH systéme novších ako zadaný dátum.

- List getActiveUserSession(int userId)

  Metóda vracia aktuálny prístup používateľa, teda postupnosť akcií od jeho posledného prihlásenia.

  Argumenty:

  - userId – identifikácia používateľa v AH systéme

  Vracia: Zoznam akcií používateľov v AH systéme novších ako zadaný dátum, *null* ak používateľ práve nie je prihlásený

- void cleanup( )

  Metóda sa zavolá pri vypínaní systému. Wrapper by mal uvoľniť všetky alokované zdroje.

Implementácia pripojenia na rozhranie

Systém sprístupňuje rozhranie na odporúčanie konceptov pomocou protokolu XML-RPC. Rozhranie vytvára objekt *recommender*, ktorý poskytuje jedinú metódu:

- Vector recommend(String system, int userId)

  Parametrami tejto metódy sú meno systému (*system*) a identifikácia používateľa v systéme (*userId*). Metóda vracia pole (<array> v XML-RPC), ktoré obsahuje štruktúry (<struct>), reprezentujúce odporúčania. Každá štruktúra má dve polia:

  - object – objekt, ktorý sa odporúča, t.j. technické meno konceptu
  - relevance – relevantnosť odporúčania, desatinné číslo od 0 do 1

  Jednotlivé odporúčania sú v poli usporiadané zostupne na základe relevantnosti, takže odporúčanie s najväčšou relevantnosťou bude na prvom mieste v poli.

Implementácia protokolu XML-RPC je dnes už dostupná na mnohých platformách, napríklad:

- Apache XML-RPC (http://ws.apache.org/xmlrpc/) pre platformu Java
- PocketXML-RPC (http://www.pocketsoap.com/pocketXMLRPC/) vo forme COM objektov pre platformy spoločnosti Microsoft

Odporúčania, ktoré AH systém získa prostredníctvom rozhrania XML-RPC môže prezentovať používateľovi vo forme odkazov na koncepty s prípadnou anotáciou na základe ich relevantnosti, alebo ich môže priamo využiť na vedenie používateľa (napr. na základe prvého odporučeného konceptu).

Prepojenie so systémom ALEA

Štandardnou súčasťou systému sú aj dva zapúzdrovacie moduly pre AH systém ALEA:

- *AleaWrapper* – zapúzdrovací modul pre pôvodnú verziu systému ALEA
- *AleaWrapperNew* – zapúzdrovací modul pre nami upravenú verziu systému ALEA

Oba moduly majú iba jediný parameter *systemPath*, ktorý definuje adresár, v ktorom sa systém (resp. jeho databázy) nachádzajú.

V upravenej verzii systému ALEA (nájdete ju na CD v adresári *AHsystems*) môžete využiť aj odporúčanie vytvárané systémom. Prepojenie nastavíte v súbore *selectnodes.asp* pomocou premenných:

- *recommenderURL* – adresa a port rozhrania pre odporúčanie, tak ako je uvedené v konfiguračnom súbore v sekcii recommender.
- *systemId* – identifikácia (meno) vášho AH systému, tak ako je uvedené v konfiguračnom súbore v sekcii AHsystem.

Upozornenie: Upravená verzia systému ALEA využíva na prepojenie COM objekty PocketXML-RPC. Inštalačný súbor pre tieto objekty nájdete na CD v adresári *xmlrpc* pod systémom ALEA.

Príloha E Vybrané triedy systému a ich rozhrania

V prílohe nájdete rozhrania tried prototypu, ktoré implementujú funkcionalitu základných modulov architektúry systému. Sú tu uvedené iba vybrané triedy a dôležité časti ich rozhraní. Kompletná dokumentácia ku všetkých triedam prototypu vo formáte *javadoc* sa nachádza na priloženom dátovom nosiči.

Wrapper

Rozhranie pre prístup k AH systému. Všetky zapúzdrovacie moduly musia toto rozhranie implementovať.

- boolean initialize( )

  Metóda sa zavolá pred prvým použitím wrappera. Zabezpečuje jeho inicializáciu, ak je potrebná.

  Vracia: *true* ak bola inicializácia úspešná, inak *false*.

- FragmentType getFragmentType(String fragmentName)

  Metóda zistí typ zadaného fragmentu.

  Argumenty:

  - fragmentName – technické meno fragmentu v AH systéme

  Vracia: Typ fragmentu.

- int getFragmentLevel(String fragmentName)

  Metóda vráti pozíciu (úroveň vnorenia) fragmentu v modeli domény.

  Argumenty:

  - fragmentName – technické meno fragmentu v AH systéme

  Vracia: Úroveň vnorenia fragmentu v strome domény.

- ConceptType getConceptType(String conceptName)

  Metóda zistí typ zadaného konceptu.

  Argumenty:

  - conceptName – technické meno konceptu v AH systéme

  Vracia: Typ konceptu.

- String getConceptParent(String conceptName)

  Metóda vyhľadá rodičovský koncept k zadanému konceptu.

  Argumenty:

  - conceptName – technické meno konceptu v AH systéme

  Vracia: Typ konceptu.

- int getConceptLevel(String conceptName)

  Metóda vráti pozíciu (úroveň vnorenia) konceptu v modeli domény.

  Argumenty:

  - conceptName – technické meno konceptu v AH systéme

  Vracia: Úroveň vnorenia konceptu v strome domény.

- List getActions( )

  Úlohou metódy je vrátiť všetky dostupné akcie používateľov z AH systému.

  Vracia: Zoznam všetkých akcií používateľov v AH systéme.

- List getActions(Date date)

  Úlohou metódy je vrátiť akcie používateľov z AH systému, ktoré novšie ako je zadaný dátum.

  Argumenty:

  - date – dátum, od ktorého sa majú akcie z AH systému získať

  Vracia: Zoznam akcií používateľov v AH systéme novších ako zadaný dátum.

- List getActiveUserSession(int userId)

  Metóda vracia aktuálny prístup používateľa, teda postupnosť akcií od jeho posledného prihlásenia.

  Argumenty:

  - userId – identifikácia používateľa v AH systéme

  Vracia: Zoznam akcií používateľov v AH systéme novších ako zadaný dátum, *null* ak používateľ práve nie je prihlásený

- void cleanup( )

  Metóda sa zavolá pri vypínaní systému. Wrapper by mal uvolniť všetky alokované zdroje.

DataImporter

Trieda implementuje funkcionalitu modulu pre predspracovanie údajov z AH systému.

- boolean importData(String system)

  Metóda prostredníctvom wrappera pre príslušný systém získa akcie používateľov a identifikuje v nich prístupy jednotlivých používateľov do systému, ktoré uloží do databázy.

  Argumenty:

  - system – identifikácia AH systému, z ktorého sa majú údaje predspracovať

  Vracia: *true* ak bolo predspracovanie úspešné, inak *false*.

KnowledgeBase

Predstavuje bázu znalostí systému. Znalosti sú ukladané vo forme XML súborov do súborového systému.

- PatternTree loadPatternTree(String system, PatternTreeType type)

  Metóda vráti strom vzorov zvoleného typu.

  Argumenty:

  - system – identifikácia AH systému, s ktorého bázou znalostí sa má pracovať
  - type – typ stromu vzorov (založený na asociačných pravidlách, sekvenčných vzorov alebo spojitých sekvenčných vzoroch)

  Vracia: Strom, v ktorom sú uložené vzory.

- void writePatternTree(String system, PatternTree tree, PatternTreeType type)

  Metóda zapíše do bázy znalostí strom vzorov zvoleného typu.

  Argumenty:

  - system – identifikácia AH systému, s ktorého bázou znalostí sa má pracovať
  - type – typ stromu vzorov (založený na asociačných pravidlách, sekvenčných vzorov alebo spojitých sekvenčných vzoroch)

DataMine

V triede je implementovaná databáza systému. Tvorí rozhranie pre relačný databázový systém.

- void writeSession(String system, UserSession session)

  Metóda zapíše prístup používateľa do databázy.

  Argumenty:

  - system – identifikácia AH systému, ku ktorému prístup prislúcha
  - session – prístup používateľa

- void storeFragmentType(String system, String name, FragmentType type, int level)

  Metóda zapíše informácie o fragmente do databázy.

  Argumenty:

  - system – identifikácia AH systému, ku ktorému fragment prislúcha
  - name – technické meno fragmentu
  - type – typ fragmentu
  - level – úroveň vnorenia fragmentu v doméne

- void storeConceptType(String system, String name, ConceptType type, int level, String parent)

  Metóda zapíše informácie o koncepte do databázy.

  Argumenty:

  - system – identifikácia AH systému, ku ktorému koncept prislúcha
  - name – technické meno konceptu
  - type – typ konceptu
  - level – úroveň vnorenia konceptu v doméne

- Date getLastActionTime(String system)

  Metóda vráti časovú známku poslednej spracovanej akcie zadaného AH systému.

  Argumenty:

  - system – identifikácia AH systému

  Vracia: Dátum a čas naposledy spracovanej akcie zo zadaného AH systému.

- void setLastActionTime(String system, Date time)

  Metóda nastaví časovú známku poslednej spracovanej akcie zadaného AH systému.

  Argumenty:

  - system – identifikácia AH systému
  - time – časová známka naposledy spracovanej akcie z AH systému

---

- ResultSet retrieveActions(String system)

  Metóda vráti všetky prístupy používateľov zadaného AH systému.

  Argumenty:

  - system – identifikácia AH systému

  Vracia: Prístupy používateľov tak, ako sú uložené v databáze.

- ResultSet retrieveActions(String system, String where)

  Metóda vráti všetky prístupy používateľov zadaného AH systému, pričom umožňuje ich prvky filtrovať na základe zadaných kritérií.

  Argumenty:

  - system – identifikácia AH systému
  - where – kritéria pre výber prístupov používateľov vo forme výrazu pre SQL klauzulu WHERE

  Vracia: Prístupy používateľov tak, ako sú uložené v databáze.

- ResultSet retrieveActions(String system, int level, String where)

  Metóda vráti všetky prístupy používateľov zadaného AH systému, pričom umožňuje ich prvky filtrovať na základe zadaných kritérií. Všetky prvky prístupov s úrovňou vnorenia väčšou ako zadanou sú namapované na svojich predchodcov (rodičov) v strome domény. Vo prístupoch sa teda nebudú nachádzať prvky s úrovňou vnorenia väčšou ako je zadaný argument.

  Argumenty:

  - system – identifikácia AH systému
  - level – maximálna úroveň vnorenia prvkov prístupov
  - where – kritéria pre výber prístupov používateľov vo forme výrazu pre SQL klauzulu WHERE

  Vracia: Prístupy používateľov tak, ako sú uložené v databáze.

- ResultSet queryDatabase(String system, String what, String groupBy, String where, String orderBy)

  Metóda umožňuje vykonať SQL SELECT dopyt na databázou systému. Dopyt sa robí nad tabuľkami databázy, ktoré sú navzájom prepojené pomocou cudzích kľúčov. Sú sprístupnené iba údaje, ktoré sa týkajú iba zvoleného AH systému.

  Argumenty:

  - system – identifikácia AH systému
  - what – určuje stĺpce, ktoré sa majú vybrať
  - groupBy – GROUP BY časť SQL SELECT dopytu
  - where – WHERE časť SQL SELECT dopytu
  - orderBy – ORDER BY časť SQL SELECT dopytu

  Vracia: Údaje z databázy na základe zvoleného dopytu.

PatternDiscoveryFramework

V triede je implementovaná kostra algoritmu pre objavovanie vzorov. Kurzívou sú uvedené abstraktné metódy, ktoré implementujú jednotlivé algoritmy pre objavovanie vzorov.

- boolean mine(String system)

  Metóda vyhľadá vzory v údajoch zadaného AH systému a uloží ich do bázy znalostí.

  Argumenty:

  - system – identifikácia AH systému

  Vracia: *true* ak bolo vyhľadanie úspešné, inak *false*.

- boolean mine(String system, double trainSize)

  Metóda vyhľadá vzory v údajoch zadaného AH systému a uloží ich do bázy znalostí. Vzory sa hľadajú iba v zadanej časti údajov.

  Argumenty:

  - system – identifikácia AH systému
  - trainSize – veľkosť množiny údajov na hľadanie vzorov (1.0 = celá množina)

  Vracia: *true* ak bolo vyhľadanie úspešné, inak *false*.

- *List generateCandidates(List patterns)*

  Abstraktná metóda, ktorú implementujú jednotlivé algoritmy na hľadanie vzorov. Jej úlohou je vytvoriť novú množinu kandidátov na vzory.

  Argumenty:

  - patterns – vzory, z ktorých sa budú generovať noví kandidáti na vzory

  Vracia: Zoznam nových kandidátov na vzory.

- *boolean supports(List session, List pattern)*

  Abstraktná metóda, ktorú implementujú jednotlivé algoritmy na hľadanie vzorov. Jej úlohou je zistiť, či má vzor podporu v prístupe používateľa.

  Argumenty:

  - session – prístup používateľa
  - pattern – vzor

  Vracia: *true* ak má vzor podporu v prístupe používateľa, inak *false*.

PatternTreeRecommender

V triede je implementovaná kostra algoritmu pre odporúčanie konceptov. Kurzívou je uvedená abstraktná metóda, ktorú implementujú jednotlivé algoritmy pre odporúčanie

- List recommend(List session, int windowSize)

  Metóda zostaví postupnosť odporúčaných konceptov na základe zadaného prístupu používateľa.

  Argumenty:

  - session – aktuálny prístup používateľa do systému
  - windowSize – veľkosť okna, t.j. aká veľká časť prístupu používateľa sa berie do úvahy

  Vracia: Zoznam odporúčaných konceptov, usporiadaný podľa relevantnosti konceptov.

- *Set searchTree(List session)*

  Abstraktná metóda, ktorú implementujú jednotlivé algoritmy na odporúčanie konceptov. Jej úlohou je nájsť vzory, ktoré sú podobné zadanému prístupu a vytvoriť množinu uzlov stromu vzorov, v ktorých nájdené vzory končia.

  Argumenty:

  - session – aktuálny prístup používateľa do systému

  Vracia: Množinu uzlov zo stromu vzorov.

Príloha F Schéma databázy

UserSession – zoznam prístupov jednotlivých používateľov

Názov atribútu	Dátový typ	Opis
sessionId	bigint(20) unsigned	primárny kľúč, identifikácia prístupu
systemId	bigint(20) unsigned	príslušnosť prístupu k AH systému
userId	bigint(20) unsigned	identifikačné číslo používateľa
random	double	náhodné číslo vygenerované pri zápise prístupu do databázy, používa sa pri rozdeľovaní prístupov na testovaciu a trénovaciu sadu

Session – akcie používateľov v jednotlivých prístupoch

Názov atribútu	Dátový typ	Opis
id	bigint(20) unsigned	identifikácia prístupu
sequence	bigint(20) unsigned	poradové číslo akcie v rámci prístupu
name	varchar(45)	meno konceptu/fragmentu
length	bigint(20) unsigned	čas v sekundách strávený na koncepte
fragment	tinyint(1) unsigned	príznak, či meno reprezentuje fragment

System – údaje o nakonfigurovaných AH systémoch

Názov atribútu	Dátový typ	Opis
id	bigint(20) unsigned	identifikácia AH systému
name	varchar(45)	meno AH systému
lastAction	datetime	časová známka naposledy spracovanej akcie z AH systému

ConceptType – údaje o druhoch konceptov a ich pozícii v doméne

Názov atribútu	Dátový typ	Opis
name	varchar(45)	primárny kľúč, meno konceptu
systemId	bigint(20) unsigned	príslušnosť konceptu k AH systému
type	int(10) unsigned	druh konceptu
level	int(10) unsigned	úroveň konceptu v rámci stromu domény
parent	varchar(45)	najbližší nadradený koncept v doméne

FragmentType – údaje o druhoch fragmentov a ich pozícii v doméne

Názov atribútu	Dátový typ	Opis
name	varchar(45)	primárny kľúč, meno fragmentu
systemId	bigint(20) unsigned	príslušnosť fragmentu k AH systému
type	int(10) unsigned	typ fragmentu
level	int(10) unsigned	úroveň fragmentu v rámci stromu domény

Príloha G Ukážka súboru bázy znalostí

V prílohe je uvedená ukážka XML súboru, ktorý reprezentuje časť bázy znalostí – strom, v ktorom sú reprezentované nájdené vzory správania sa.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE node SYSTEM "dtd/patterntree.dtd">
<node name="ROOT NODE" support="1.0">
   <node name="sLispRedukcia" support="0.17611337"></node>
   <node name="c22" support="0.5728745">
      <node name="SchemaZobraz" support="0.1902834"></node>
   </node>
   <node name="c21" support="0.5202429">
      <node name="c22" support="0.19838056">
        <node name="SchemaZobraz" support="0.13765182"></node>
      </node>
   </node>
   <node name="SpracovanieZoznamu" support="0.27732792"></node>
   <node name="SchemaZobrazL" support="0.37449393"></node>
   <node name="SchemaZobraz" support="0.6639676">
      <node name="SchemaFilter" support="0.16396761"></node>
      <node name="PrikladDruha_Mocnina" support="0.1680162"></node>
   </node>
   <node name="SchemaTest" support="0.29352227"></node>
   <node name="SchemaHladajL" support="0.31376517"></node>
   <node name="SchemaHladaj" support="0.42510122">
      <node name="SchemaPocitajPrvky" support="0.15991902"></node>
   </node>
   <node name="SchemaFilterL" support="0.35020244"></node>
   <node name="SchemaFilter" support="0.5748988">
      <node name="SchemaHladaj" support="0.14777328"></node>
      <node name="PrikladDel_Nuly" support="0.13765182"></node>
   </node>
   <node name="PrikladSuma" support="0.18016194"></node>
   <node name="PrikladSucin" support="0.25101215"></node>
   <node name="PrikladPreklad" support="0.29554656">
      <node name="PrikladPorovnaj_Vektory" support="0.15182187"></node>
   </node>
   <node name="PrikladPorovnaj_Vektory" support="0.24493927">
      <node name="PrikladSucin" support="0.14574899"></node>
   </node>
   <node name="PrikladParny" support="0.17206478"></node>
   <node name="PrikladDruha_Mocnina" support="0.41497976">
      <node name="PrikladPreklad" support="0.15789473"></node>
   </node>
   <node name="PrikladDeleteL" support="0.15182187"></node>
   <node name="PrikladDel_Nuly" support="0.25506073">
      <node name="PrikladDelete" support="0.13765182"></node>
   </node>
   <node name="PrikladCard" support="0.41700405"></node>
   <node name="Lisp" support="0.17408907"></node>
 </node>
```

Príloha H Ukážka zdrojového kódu

Ako ukážku zdrojového kódu prototypu uvedieme funkciu, ktorá implementuje hlavnú časť kostry algoritmu na odporúčanie konceptov.

```
public final List recommend(List session, int windowSize) {
  List recommendations = new LinkedList();

  if (windowSize < 0) {
    logger.warn("Recommendation requested with negative window size");

    return recommendations;
  }

  if (session.size() < windowSize) {
    logger.warn("The size of session is insufficient for selected window size");

    return recommendations;
  }

  while (windowSize > 0) {

    // create window of session
    int to = session.size();
    int from = to - windowSize;
    List window = session.subList(from, to);

    Set lastNodes = searchTree(window);
    Iterator i = lastNodes.iterator();

    while (i.hasNext()) {

      PatternTreeNode lastNode = (PatternTreeNode) i.next();

      if (lastNode != null) {
        Iterator nodeIterator = lastNode.childrenIterator();

        while (nodeIterator.hasNext()) {
          PatternTreeNode node = (PatternTreeNode) nodeIterator.next();

          if (node.isLast()) {

            float relevance = node.getSupport() / lastNode.getSupport();
            insertRecommendation(
                       recommendations,
                       new Recommendation(
                                node.getValue(),
                                relevance
                                 )
                     );
          }
        }
      }
    }

    // decrease window size until recommendation can be made
    if (!recommendations.isEmpty()) {
      break;
    } else {
      windowSize--;
    }
  }

    Collections.sort(recommendations);
  return recommendations;
}
```

Príloha I Obsah dátového nosiča

/AHsystems – AH systémy, ktorými sa zaoberáme v práci

- **/AHA!** – systém AHA!
- **/ALEA** – pôvodná verzia systému ALEA
- **/ALEA-v1.1** – verzia systému ALEA s upraveným zaznamenávaním údajov
- **/ALEA-v1.2** – verzia systému ALEA rozšírená o odporúčanie konceptov

/Data – údaje použité pri experimentoch s prototypom

- **/AHminer** – databázy systému, ktoré obsahujú predspracované údaje
- **/ALEA** – databázy systému ALEA
  - **/alea_db_2002** – databáza systému ALEA z roku 2002/2003
  - **/alea_db_2003** – databáza systému ALEA z roku 2003/2004
  - **/alea_db_2004** – databáza systému ALEA z roku 2004/2005
  - **/alea_merged_db_2002-2004** – spojené databázy systému ALEA za obdobie 2002-2004
  - **/alea_generator** – databáza vytvorená generátorom

/Prototype – prototyp systému

- **/AHminer** – systém na odporúčanie navigácie
  - **/bin** – skompilované triedy systému
  - **/config** – konfiguračné súbory systému
  - **/doc** – javadoc dokumentácia tried systému
  - **/dtd** – definície XML súborov
  - **/lib** – knižnice, potrebné pre fungovanie systému
  - **/src** – zdrojové texty systému
  - **/wrappers** – adresár na umiestnenie zapúzdrovacích modulov
- **/AleaWrapper** – zapúzdrovacie moduly pre systém ALEA
  - **/bin** – skompilované triedy modulov
  - **/lib** – knižnice, potrebné pre fungovanie modulov
  - **/src** – zdrojové texty modulov
- **/AleaVisitor** – nástroj pre generovanie databáz v systéme ALEA
  - **/bin** – skompilované triedy nástroja
  - **/src** – zdrojové texty nástroja

/Resources – literatúra použitá v práci v elektronickej podobe

/Thesis – text diplomovej práce v elektronickej podobe