Masarykova univerzita
Fakulta informatiky

Univerzální digitální repozitář

Diplomová práce

Bc. Vlastimil Krejčíř

Brno, 2005

Prohlášení

Prohlašuji, že tato diplomová práce práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: RNDr. Miroslav Bartošek, CSc.

Poděkování

Děkuji vedoucímu mé práce RNDr. Miroslavu Bartoškovi, CSc. za to, že mi umožnil vypracovat tuto práci v rámci mé pracovní výpomoci na Ústavu výpočetní techniky Masarykovy univerzity, za cenné rady a připomínky a za čas, který mi věnoval. Dále chci poděkovat Petru Kovářovi za cenné rady, které pomohly vyřešit problémy spojené s instalací a správou testovaných systémů. Poděkování patří také Mgr. Davidu Hlaváčkovi za jeho konzultace týkající se programování v jazyce Java. V neposlední řadě pak děkuji Bc. Davidu Cimbůrkovi za jeho trpělivost při korekturách této diplomové práce.

Shrnutí

Dnes, v době prudkého nárůstu šíření informací v digitální podobě, narůstá potřeba řešit uspokojivým způsobem jejich ukládání, správu, archivaci a zpřístupnění uživatelům. Proto se v současnosti rozvíjí oblast digitálních knihoven. Byly publikovány teoretické studie, které se touto problematikou zabývají. Následující práce se pokouší zmapovat stav v oblasti praktického využívání digitálních knihoven. Zjišťuje, jaký konkrétní software je v současné době pro implementaci digitální knihovny k dispozici a provádí testování a vzájemné porovnání vybraných systémů.

Na základě provedené analýzy v segmentu volně dostupných softwarových řešení se ukázal být současným nejvhodnějším systémem pro nasazení v praxi systém DSpace. DSpace je obecný systém pro archivaci jakýchkoli digitálních dat. V rámci této práce byl rozšířen o nadstavbu, která umožňuje jeho větší specializaci a významně zvyšuje jeho použitelnost v reálných aplikacích. Využití této nadstavby je popsáno a prakticky ukázáno na různých typech dat.

Klíčová slova

digitální knihovna, digitální objekt, repozitář, metadata, identifikátor, *DSpace*

Sem bude v papírové podobě diplomové práce vložena kopie oficiálního zadání.

Obsah

Kapitola 1

Úvod

Oblast digitálních knihoven zahrnuje širokou škálu poznatků, přístupů a technologií, ve kterých není snadné se orientovat. Co je digitální knihovna? Digitální knihovna je organizovaná sbírka informací udržovaných a šířených pomocí digitálních technologií. Zjednodušeně lze říci, že digitální knihovna má rysy a plní funkce klasické knihovny, ale využívá k tomu současné moderní počítačové a komunikační technologie a těží z jejich výhod. Jaké jsou základní funkce a cíle digitálních knihoven?

Stejně jako u klasických knihoven jde především o shromažďování, ochranu a archivaci vědomostí, současně se službami poskytování, hledání, třídění, řízení kvality, jednoznačné identifikace a výměny informací – a to jak ve vztahu k jiným knihovnám, tak ve vztahu k uživatelům. Tyto funkce jsou prováděny systematicky a organizovaně a nikoli jen pro klasické dokumenty, ale pro jakákoli data – textová i multimediální. Tato data společně s popisnými metadaty jsou uložena v počítači a nazývají se digitální objekty. Služby se realizují v prostředí počítačové sítě, v současnosti v prostředí Internetu.

Proč potřebujeme digitální knihovny? Odpovědí je obrovský nárůst digitálních informací na světovém webu. Samotný web není řešením, není digitální knihovnou, protože není organizován. Navíc nesplňuje profilované zaměření kladené na knihovny – sledování kvality, jednoznačnou identifikaci informací a dlouhodobou ochranu a archivaci. Tyto problémy současného webu by měly řešit právě digitální knihovny.

Digitální knihovny si nekladou za cíl zcela nahradit současné klasické „kamenné“ knihovny. Naopak s nimi musí spolupracovat a těžit z jejich znalostí a stovkami let prověřených postupů. Tyto znalosti pak implementovat a využít výhod počítačových technologií pro zlepšení funkcí a služeb, které klasické knihovny poskytují.

Jaký je současný stav v praktické realizaci digitálních knihoven? Jsou již dostupná použitelná řešení? Která z těchto řešení jsou perspektivní? Na tyto otázky se tato práce pokusí najít odpověď. Současně bude navrženo praktické řešení digitální knihovny pro potřeby Masarykovy univerzity.

1.1 Přehled práce a kapitol

Cílem práce bylo detailně prozkoumat a porovnat *volně dostupné* systémy pro tvorbu a správu digitálních knihoven. Na základě výsledků této analýzy pak zvolit vhodný systém pro tvorbu digitální knihovny na Masarykově univerzitě a na různých typech dokumentů ověřit praktickou použitelnost systému.

Řešení probíhalo v několika fázích. Nejdříve byl proveden průzkum zaměřený na vyhledání potenciálně vhodných kandidátů pro implementaci digitální knihovny. Celkem bylo vyzkoušeno třináct volně dostupných systémů. Z nich byly vybrány čtyři systémy, Fedora, RIB, EPrints a DSpace, které byly testovány podrobněji. Zvolené systémy jsou vyvíjeny v akademickém prostředí a do tohoto prostředí jsou i cíleny. Primárně jsou určeny jako repozitáře pro správu a archivaci dat a jeví se jako zajímavé z hlediska budoucího vývoje. Tyto systémy byly všechny v rámci řešení diplomové práce nainstalovány, byly podrobně vyzkoušeny možnosti jejich konfigurace (běh pod protokolem HTTPS, nastavení prostředí a dalších parametrů, ...), testovány jejich dokumentované vlastnosti a prozkoumány možnosti přizpůsobení pro různé typy vkládaných objektů. Byly taktéž částečně testovány programátorské knihovny, které se spolu se systémy distribuují. Na základě těchto testů byl vybrán systém DSpace jako vhodný kandidát pro digitální knihovnu Masarykovy univerzity. Pro něj jsme implementovali podporu pro různé typy sbírek digitálních objektů a ukázali využití této typové podpory pro ukládání fotografií a časopisů.

Kapitola 1 – Úvodní kapitola této práce, přehled.

Kapitola 2 – Seznámení s oblastí digitálních knihoven. Stručný přehled používaných pojmů, protokolů a principů. Popis jedné ze základních teoretických prací v oblasti digitálních knihoven (Kahn/Wilenského architektura).

Kapitoly 3, 4, 5 a 6 – Popis a analýza čtyř testovaných systémů (Fedora, RIB, EPrints a DSpace).

Kapitola 7 – Srovnávací analýza testovaných systémů, přehledové srovnávací tabulky.

Kapitola 8 – Stručný přehled dalších (testovaných) systémů pro správu a archivaci dat.

Kapitola 9 – Praktická část diplomové práce – popis implementace podpory typů v systému DSpace, praktická ukázka vytváření typů, implementace repozitáře.

Kapitola 10 – Závěr práce, shrnutí dosažených výsledků.

1.2 Seznam použitých zkratek

Níže je přiložen seznam zkratek, které jsou často používány v této práci.

BDef	Behaviour Definition (Object)
BMech	Behaviour Mechanism (Object)
CMS	Content Management System
DC	Dublin Core
DOI	Digital Object Identifier
DL	Digital Library
CD	Compact Disc
GPL	GNU Public Licence
LoC	Library of Congress
MIT	Massachussetts Institute of Technology
MPL	Mozilla Public Licence
OAI (PMH)	Open Archives Initiative (Protocol for Metadata Harvesting)
PID	Persistent Identifier
STI	Scientific & Technical Information

Kapitola 2

Digitální knihovny

V této části práce budou stručně vysvětleny základní pojmy, které se v oblasti digitálních knihoven objevují. Většina z nich je dále používána v samotné práci. Podrobnější rozbor problematiky je uveden například v [1] a [2].

2.1 Digitální knihovna, digitální objekt

S pojmem **digitální knihovna** (v literatuře se často používá zkratky DL, dle anglického termínu Digital Library) je čtenář intuitivně seznámen v úvodu této práce. Přesných definic lze nalézt celou řadu – liší se zejména úhlem pohledu na celou problematiku. Pohled počítačového odborníka může být například takový:

> Digitální knihovna je *spravovaná sbírka informací* spolu s odpovídajícími službami, přičemž informace jsou uloženy v digitální podobě a jsou dostupné prostřednictvím sítě.
>
> *W.Y.Arms, 2000* [3]

Naopak pohled knihovníka na digitální knihovnu je odlišný:

> Digitální knihovny jsou *organizace*, které poskytují zdroje (včetně specializovaného personálu) umožňující provádět výběr, strukturování a zpřístupnění sbírek digitálních prací, tyto práce dále distribuovat, udržovat jejich integritu a dlouhodobě uchovávat – a to vše s ohledem na snadné a ekonomické využití určitou komunitou nebo množinou komunit uživatelů.
>
> *US Digital Library Federation, 1997* [3]

Ve zbytku práce bude užíván především počítačově odborný pohled na celou problematiku, který se více přibližuje první z obou citovaných definicí.

Základním prvkem digitální knihovny je **digitální objekt**. Digitální objekt je digitální struktura, reprezentující libovolný objekt reálného i digitálního informačního prostředí. Sestává jednak z vlastních dat popisujících informační obsah a jednak z metadat, které realizují popisné, strukturální a administrativní vlastnosti digitálního objektu. Digitální objekt může být ekvivalentem knihy v klasické knihovně (která kromě samotného intelektuálního obsahu má obvykle uvedena i metadata – informace o nakladateli, roku vydání apod.). Speciálním případem je digitální objekt, který není nosičem dat, ale pouze dává informaci, kde lze tato data nalézt – v tomto případě mluvíme o digitálním meta-objektu. Obvykle se však i pro tento speciální případ používá obecný termín digitální objekt a následná specializace se určí dle kontextu.

Dalším důležitým pojmem je **repozitář**. Repozitář je místo fyzického uložení digitálních objektů. Repozitář se „stará" o digitální objekty v něm uložené a prostřednictvím vhodného protokolu je zpřístupňuje externímu prostředí. Obvykle je jednoznačně identifikován (lokálně nebo celosvětově) a obsahuje rozhraní, na které se napojují služby umožňující přístup k objektům.

2.2 Metadata

Metadata jsou (strukturované) informace o informacích. Obecně slouží k popisu libovolného předmětu, služby, systému, . . . V prostředí digitálních knihoven obvykle popisují digitální objekt (ale mohou popisovat i samotnou knihovnu, repozitář apod.). Můžeme je dělit do tří základních typů – metadata *popisná*, *strukturální* a *administrativní*. *Popisná* metadata slouží k identifikaci objektu a popisu jeho obsahu, používají se například při hledání konkrétního objektu. Příkladem popisných metadat je informace o autorovi, datu vzniku objektu nebo autorských právech vztahujících se k objektu. *Strukturální* metadata zachycují informace o formátu objektu a jeho struktuře. Například říkají, že objekt obrázek je uložen ve formátu JPG a na základě této informace lze daný obrázek správně zobrazit na monitoru počítače. Jiným příkladem je objekt kniha – strukturálními metadaty jsou identifikátory jednotlivých stran. *Administrativní* metadata uchovávají technické informace o objektu. Například přístupová práva k objektu, kde je objekt fyzicky uložen (v databázi, na konkrétním místě na pevném disku, . . .). V praxi nemusí být toto rozdělení metadat striktně dodržováno a jednotlivé typy se často překrývají.

Kvůli vzájemné spolupráci mezi (digitálními) knihovnami, přesnému vyhledávání a správné interpretaci informací je vyvíjena snaha zavést celosvětový standard pro metadatový popis objektu. Na rozdíl od klasických knihoven, které popisují pouze poměrně úzkou skupinu objektů, je bohatost dat v oblasti digitálních knihoven obrovská. Vyvstávají důležité otázky: Jaké metadatové položky jsou potřebné pro každý digitální objekt? Jsou dostačující pro vyhledávání? Co například s nestandardními objekty, které vyžadují podrobnější popis? Jak metadata interpretovat – co skutečně znamená položka *Autor* (je to skutečný autor, je to překladatel, je to ilustrátor, . . .)? Jak zapsat datum vytvoření objektu (2003-11-21, 21.11.2003, . . .)?

Tyto problémy se snaží vyřešit několik standardů. Jak bude patrné z následujících kapitol, velmi známým a používaným standardem pro popis digitálních objektů v prostředí webu se v současnosti stává **Dublin Core** (často označován zkratkou DC) [4]. Definuje základní jádro – 15 metadatových položek, které by měl mít každý informační objekt, udává jejich význam a navíc dovoluje upřesnění a rozšíření tohoto jádra. Metadata standardu Dublin Core jsou popisnými a částečně strukturálními metadaty. Z dalších popisných metadatových standardů lze zmínit **LoC – Core Metadata Elements** [5] používaný Kongresovou knihovnou (Library of Congress, zkratka LoC) ve Spojených státech amerických.

Dalšími standardy, které se v souvislosti (nejen) s metadaty používají, jsou standardy realizující jednoznačné zakódování metadat. Jak zakódovat metadata v Dublin Core, aby byl okolní svět (lidé i počítače) schopný je přečíst, jednoznačně interpretovat a být schopen si je spojit s konkrétními daty (digitálními objekty)? Pro toto zakódování lze použít současných prostředků jazyka HTML (s pomocí tagů `<meta>`), ale takto lze uchovat pouze jednoduché informace. Používanější je jazyk XML spolu se standardy definujícími sémantiku zakódovaných metadat. Jedním takovým standardem je například **RDF** (Resource Description Framework) [6], který definuje XML schéma, umožňující vyjádřit vztahy mezi metadatovými položkami a jejich sémantiku. Například dává návod na jednoznačné zakódování informace „Karel Čapek je autorem knihy Válka s mloky“, přičemž jednotlivé metadatové položky mohou být podle standardu Dublin Core. Jiným standardem, který říká jak zakódovat metadata a jejich vazbu na digitální objekty, je **METS** (Metadata Encoding and Transmission Standard) [7]. Definuje XML schéma, které sémanticky dělí metadata do pěti kategorií – popisná metadata (například DC), administrativní metadata (jak byla data vytvořena, s jakou licencí, . . .), metadata chování (pomocí jakého programu se zobrazí daný objekt), metadata udávající seznam souborů příslušejících k objektu a strukturální metadata (stromová struktura spolu s popisem napojení logických komponent digitálního objektu na jednotlivé zdrojové soubory).

2.3 Identifikátory

Identifikátor je jednou z typických metadatových položek digitálního objektu. Slouží pro jednoznačnou a trvalou identifikaci digitálního objektu, repozitáře, ... Z klasického knihovnického prostředí je znám například pojem ISBN (International Standard Book Number), jednoznačný identifikátor, který je používán pro identifikaci tištěných knih. Podobná snaha o jednoznačnou identifikaci je žádoucí i pro digitální objekty. Vzhledem k tomu, že digitálním objektem může být prakticky jakákoli elektronická informace, je úkol nalezení obecného mechanismu pro jednoznačnou identifikaci netriviálním problémem. Požadavky na identifikátory jsou široké (nezávislost na umístění, jednoznačnost, perzistence, škálovatelnost, ...) a v prostředí Internetu je nutný nějaký mechanismus, který by byl vyhovující pro všechny uživatele a poskytoval spolu s identifikátory i potřebné služby – přidělování jednoznačného identifikátoru a jeho resoluci (automatizované vyhledání objektu identifikovaného daným identifikátorem v globální síti). Standardů v oblasti digitálních identifikátorů existuje v současnosti několik, například je to URN (Uniform Resource Name) [8], PURL (Persistent URL) [9], CNRI handles (Corporation for National Research Initiatives) [10], DOI (Digital Object Identifier) [11], ARK (Archival Resource Key) [12] a další.

Technologie URN není konkrétním standardem poskytujícím mechanismy přidělování a resoluce identifikátorů, ale poskytuje obecné zastřešení pro různé identifikační mechanismy v prostředí webu. Identifikačními mechanismy jsou standardy CNRI handles a DOI – definují přidělování identifikátorů i postupy jejich resoluce. Standardy PURL a ARK nedefinují přímo přidělování identifikátorů digitálním objektům, ale zajišťují resoluční mechanismy pro tyto objekty. Obvyklá realizace resolučního mechanismu výše zmíněných standardů je řešena pomocí speciálních serverů, které k zadanému identifikátoru vracejí fyzické umístění digitálního objektu (případně i další informace o tomto objektu – to v závislosti na daném standardu).

2.4 Interoperabilita

Interoperabilita zaručuje možnost vzájemné spolupráce mezi nezávisle spravovanými knihovnami. Můžeme mluvit o různých pohledech na interoperabilitu. Uživatelský pohled může znamenat jednotné rozhraní pro přístup k různým knihovnám. Správce knihovny naopak spíše zajímá, jak si vyměňovat data s jinými knihovnami a zajistit tak alespoň částečnou koherenci obsahu. Správce (knihovník) bude tedy požadovat standardy a protokoly, jejichž použití mu zajistí spolupráci s co největším počtem jiných digitálních knihoven rozptýlených na Internetu.

Jedním z problémů týkajících se spolupráce mezi knihovnami je výměna metadat o spravovaných objektech. Další otázkou je realizace vzájemného provázání dokumentů. Je nutné definovat postupy a protokoly, které určí, jakým způsobem tyto problémy řešit.

V současnosti existuje několik protokolů spadajících do oblasti interoperability, například OAI-PMH (Open Archives Initiative – Protocol for Metadata Harvesting) [13], Z39.50 [14] (standard LoC), OpenURL [15] a další. V této práci bude často zmiňován zejména protokol OAI-PMH. Jedná se o jednoduchý protokol pro „sklízení" metadat. Digitální knihovny poskytují rozhraní kompatibilní s OAI, které umožňuje třetím stranám získávat metadata o objektech, které digitální knihovna spravuje. Těmito třetími stranami jsou instituce, které se specializují na shromažďování metadat z různých zdrojů, nad nimiž pak implementují další služby, například poskytují rozhraní, které slouží pro vyhledávání a vrací metadata objektu a informace o tom, v jaké digitální knihovně lze daný objekt nalézt, je-li veřejně dostupný apod. Technicky probíhá komunikace pomocí protokolu HTTP a vracené informace jsou kódovány pomocí XML (metadata o objektu musí být k dispozici minimálně v Dublin Core). Podrobné informace o protokolu lze nalézt na výše zmíněném odkazu na www stránky organizace OAI.

Standard OpenURL definuje způsob kódování metadat o informačních objektech do URL a slouží k realizaci otevřených, kontextových, dynamických vazeb mezi zdroji na webu. Nad OpenURL lze vytvořit různá aplikační prostředí – jedním z nich je například SFX. V praxi to vypadá tak, že každý odkaz z dokumentu na jiný dokument vede přes prostředníka, jímž je speciální server SFX, který například provozuje instituce poskytující dané dokumenty, a který na základě informace o tom, kdo zdroj z daného odkazu žádá, poskytne vhodný odkaz na objekt (nebo jiné informace o objektu). Například koncový uživatel klikne na odkaz vedoucí na konkrétní elektronickou publikaci. Server SFX tento požadavek vyhodnotí a zjistí, že požadavek vznesl uživatel, který nemá příslušnou publikaci právo prohlížet. Na základě toho přesměruje uživatele jen na stránku obsahující metadata o publikaci, případně na stránku s recenzemi této publikace apod.

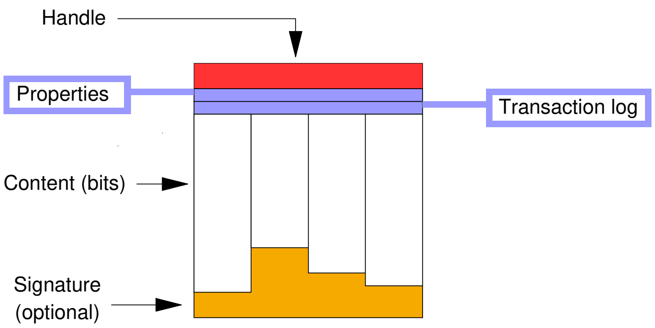
2.5 Další aspekty oblasti DL

Výčtem výše zmíněných problémů v oblasti digitálních knihoven jejich studium nekončí. Lze si klást další otázky. Jak zajistit globální vyhledávání nad digitálními knihovnami? Jak dlouhodobě archivovat digitální data (vytištěná kniha má životnost stovky let, ale obyčejné CD možná desítky...)? Jak se vypořádat se společenskými problémy, autorskými právy a zneužíváním digitálních knihoven? Kdo zaplatí veškeré náklady nutné k provozu a poskytování služeb spojených s digitálními knihovnami? Diskuse těchto aspektů by pravděpodobně vydala na několik knih, proto se zde omezíme na konstatování, že dané problémy se v současnosti stále řeší.

2.6 Kahn/Wilenského architektura

Jednou ze základních teoretických prací v oblasti digitálních knihoven je článek *Framework for Distributed Digital Object Services* [16] dvojice autorů Robert Kahn a Robert Wilensky. Ti v této práci definují základní architekturu spojenou s digitálními knihovnami.

Práce nejdříve definuje základní pojmy. Především pojem **digitální objekt** (*digital object*) jako datovou strukturu, jejímž obsahem je digitální materiál (*data*) a jedinečný identifikátor pro tento materiál nazývaný **handle**, který je součástí klíčových metadat (*key-metadata*). Takovýto objekt může být uložen v jednom nebo více **repozitářích**, což je úložný prostor, který je přístupný po síti a má přidělené jedinečné jméno od globální jmenné autority (*global naming authority*). Objekt uložený v repozitáři je nazýván **uložený digitální objekt** (*stored digital object*). Uložení objektu v repozitáři je spojeno s registrací identifikátoru (*handle*) tohoto objektu u některé jmenné autority, která zajišťuje resolvování objektu. Takový objekt se nazývá **registrovaným objektem** (*registered object*).



Obrázek 2.1: Digitální objekt (převzato z [17] a vektorově překresleno).

Repozitář udržuje o každém svém objektu informace zahrnující popisná, strukturální i administrativní metadata (*properties record*) a transakční záznam (*transaction record*). Transakční záznam obsahuje informace o veškerých operací prováděných nad repozitářem.

Všechna data v objektu mají své typy. Základními typy jsou bitové sekvence (*bit-sequence*), digitální objekty (*digital-object*) a identifikátory (handle). Složené typy jsou množinami základních typů (*set-of-...*). Kahn a Wilensky dále mluví o **meta-objektu** (*meta-object*), což je objekt, který pouze odkazuje na jiné digitální objekty a fyzicky není nosičem dat (je pouze nosičem metadat). Objekty mohou být měnitelné (*mutable*) a trvalé (*immutable*). Identifikátor (*handle*) je objektu přidělen jako „identifikátor repozitáře“ + „lokálně přidělený identifikátor objektu“. Lokálně přidělený identifikátor objektu je jedinečný v rámci repozitáře.

Přístup k repozitáři se realizuje jednoduchým protokolem s názvem RAP (*Repository Access Protocol*). Tento protokol definuje tři základní operace nad repozitářem. Operace **přístupu k objektu** (*Access to a digital object*) poskytuje základní metody pro získávání metadat i dat digitálního objektu. Na každém objektu mohou být definovány služby (*additional services*), které ho před vrácením uživateli zpracují. Uživatel pak obdrží tzv. diseminaci objektu (*disseminition*) – tj. objekt prezentovaný v určité lidskými smysly vnímatelné podobě (nikoli jako proud bitů). Druhou definovanou operací je **vložení objektu** (*Deposit of digital object*), což je pouze jednoduchá operace, která vloží objekt spolu se zadanými metadaty. Třetí základní operací je operace **získávání informací o repozitáři** (*Access to reference services*). Tato operace slouží pro klientské aplikace a vzdálené servery a vrací informace o repozitáři, o typu služeb a protokolů, které repozitář nabízí pro komunikaci, atp.

Kahn/Wilenského práce dále diskutuje službu resolvování identifikátorů (*handles*) – je postavena na stromové struktuře serverů (globální a lokální jmenné autority), které dokáží resolvovat příslušné identifikátory v rámci globální sítě. Celý systém funguje tak, že klient podá nejprve dotaz na svůj lokální server, který v případě, že zná odpověď, ji vrátí, v opačném případě se ptá svého nadřazeného serveru, který postupuje stejným způsobem (v nejhorším případě až ke kořenovému serveru). Podobný mechanismus resolvování jmen používá v současnosti na Internetu DNS (Domain Name System) [18].

Poslední část práce se věnuje návrhu zápisu jednoznačného identifikátoru a principu jeho tvorby.

Celkově práce uvádí obecný rámec, který může být (a mnohdy je) vodítkem návrhářů a programátorů vyvíjejících digitální knihovny. Mnohé principy, které Kahn a Wilensky popisují, jsou často používány v praxi, což bude vidět i na testovaných systémech v dalších kapitolách práce.

2.7 Z historie digitálních knihoven

První zmínky o digitálních knihovnách se datují do doby vzniku prvních počítačů. V práci *As We May Think* [19] z roku 1945 zmiňuje její autor Vannevar Bush základní principy fungování „digitální knihovny“ a jeho vizionářský systém Memex je v podstatě „digitální knihovnou“ postavenou na tehdy dostupných analogových technologiích (i když nebyl nikdy zkonstruován). Skutečný rozvoj v oblasti digitálních knihoven začíná v polovině 90. let v souvislosti s rozvojem počítačových technologií a Internetu. Začínají vznikat první experimentální projekty, je ustaven program Digital Library Initiative [20]. Rozvoj digitálních knihoven probíhá do současnosti.

2.8 Shrnutí

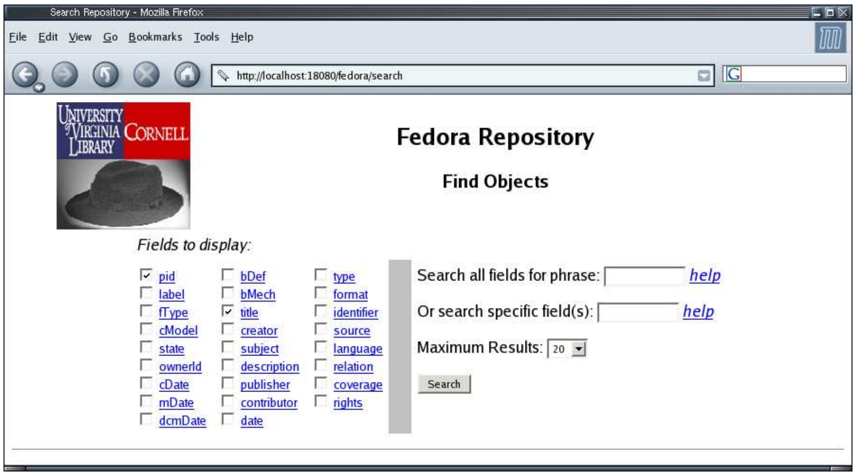
Téma digitálních knihoven je skutečně velmi široké. Pro hlubší pochopení pojmů a principů odkazovaných dále v textu je pravděpodobně nutné sáhnout po materiálech, které se na daná témata specializují. Základní principy však tento úvod dostatečně zahrnuje.

V další části práce již se budeme zabývat současnými volně dostupnými systémy, které více či méně dobře implementují jádro digitální knihovny, digitální repozitář. Postupně se seznámíme se systémy Fedora, RIB, EPrints a DSpace.

Kapitola 3

Fedora

Systém Fedora [21] je svým způsobem mezi jednotlivými digitálními knihovnami výjimečný. Vývojáři si dali za cíl vybudovat systém, který by realizoval základní teoretické poznatky a modely v oblasti digitálních knihoven. Staví mimo jiné na Kahn/Wilenského architektuře. Systém Fedora poskytuje jen repozitářové služby, jakési jádro ošetřující ukládání, správu a archivaci digitálních objektů. Zároveň obsahuje knihovnu funkcí a volání, která umožňuje programátorovi provádět činnosti nad repozitářem. Systém Fedora zatím není zcela kompletní, okamžitě v praxi nasaditelný a použitelný, protože neposkytuje potřebné uživatelské rozhraní – to si musí instituce, která by chtěla systém používat, vytvořit sama.



Obrázek 3.1: Ukázka úvodní stránky www rozhraní systému Fedora.

Systém Fedora je volně šiřitelný včetně zdrojových kódů poskytovaných programů (pro tyto typy programů budu dále používat zavedený termín *open source software*, případně *open source program/aplikace*) pod licencí Mozilla Public Licence [22]. Projekt je vyvíjený na Cornell University ve spolupráci s University of Virginia v USA. Název je složen z prvních písmen charakteristiky **Flexible Extensible Digital Object and Repository Architecture**. Cílem projektu je poskytnout *univerzální* repozitář pro University of Virginia a další, především akademické, instituce.

3.1 Historie

První základní práce [23] o systému Fedora byla publikována v roce 1998 Sandrou Payette a Carlem Lagoze, vědci z výzkumné skupiny pro digitální knihovny na Cornellově univerzitě v USA. Navrhli systém Fedora jako modulární systém a provedli jeho implementaci za pomocí technologie CORBA[24]. Na této práci začal tým z University of Virginia budovat digitální repozitář a experimentovat se sbírkami digitálních objektů (převážně uložených v jazyce SGML). Tato experimentální část projektu pomohla vývojářům určit, že navržená koncepce je vhodná. Programátorský tým tedy vyvinul základní jádro postavené na relační databázi a, pro administraci a přístup, uživatelské rozhraní naprogramované v jazyce Java.

V srpnu 2001 obdržela University of Virginia grant Mellonovy nadace ve výši 1 milión amerických dolarů, aby na základech prvních verzí systému Fedora vybudovala skutečně sofistikovaný digitální repozitář. Projekt začal být realizován za pomoci moderních technologií (webové služby, Java). Vývoj, jehož se účastní sedm řešitelských týmů z USA a Velké Británie, probíhá do současnosti.

3.2 Základní vlastnosti

Repozitář systému Fedora je univerzální a umožňuje vkládat digitální objekty libovolného typu. Metadata digitálních objektů jsou popisována v jazyce XML dle schématu standardu **METS**. Hledání je fulltextové v metadatových položkách, metadata jsou ukládána a dávána k dispozici ve formátu **Dublin Core** (jako součást METS). Pro sdílení metadat podporuje Fedora **OAI-PMH** protokol verze 2.0. Také obsahuje utilitu pro hromadné vytváření objektů, správu verzí objektů a autentizovaný přístup. Fedora zatím nemá hotové uživatelské prostředí.

Systém Fedora je napsán v jazyce Java, může být provozován na počítačích s operačními systémy Windows, Linux, MacOS a další (aktuální kompletní seznam podporovaných platforem lze nalézt na stránkách projektu). Pro běh ukázkového webového rozhraní používá www server Tomcat. Předpokládá se síťové prostředí a z toho také veškerá filozofie práce vychází. Repozitář s objekty je identifikován IP adresou (a portem). Celý systém stojí na webových službách (Web Services [25]), přes které je realizována vzdálená práce s repozitářem. Fedora má i prostředky pro vytváření nových objektů, jejich export a import.

3.3 Architektura

3.3.1 Digitální objekt

Základním prvkem, se kterým systém Fedora pracuje, je digitální objekt. DO je složen z perzistentního jednoznačného identifikátoru – *identifikátoru PID* (z anglického Persistent Identifier), systémových metadat, *diseminátorů* (Disseminators), které propojují digitální objekt s jeho chováním, a *datastreamů* (Datastreams), což jsou samotné soubory představující vlastní obsah digitálního objektu. Digitální objekty mohou být tří typů:

Data Objects – klasické *datové objekty*, reprezentují konkrétní digitální data (obrázky, videa, publikace, software, . . .). Každý datový objekt je složen z *datastreamů* (metadata a samotná data) a navázán na *diseminátory* (určují jakým způsobem bude objekt poskytován, ve skutečnosti je to ukazatel na **Behaviour Definition Object**). Příkladem datového objektu může být objekt, jehož datastreamy jsou obrázky typu JPG.

Behavior Definition Objects – objekty *popisu chování* (dále také BDef objekty). Tento typ objektů slouží k abstraktnímu popisu služby, která je složena z různých metod (dá se říci, že se jedná o deklaraci služeb). Je tím určen model chování, který může přijímat i několik datových objektů. Například služba pro zpracování obrázků definuje funkce zmenšení obrázku, převod do stupňů šedi, získání náhledu obrázku atp.

Behavior Mechanism Objects – objekty *implementace chování* (dále také BMech objekty). Slouží k reprezentaci konkrétní služby (která splňuje požadavky a definice služby tak, jak jsou dány v BDef objektu). Společně s datovým objektem a BDef objektem získáme prostředky pro popis chování digitálních objektů v repozitáři. BMech objekt obsahuje metadata služeb (zapsána pomocí jazyka WSDL [26]) a popis, jak tyto služby spouštět (kde jsou uloženy, které typy datastreamů jim mohou být předávány jako parametry, . . .).

Objekty popisu chování a implementace chování budou dále označovány souhrnně *objekty chování*.

Pro popis běžného scénáře konstruování DO je také nutné se nejdříve seznámit s typy datastreamů. Datastreamy mohou být čtyř typů:

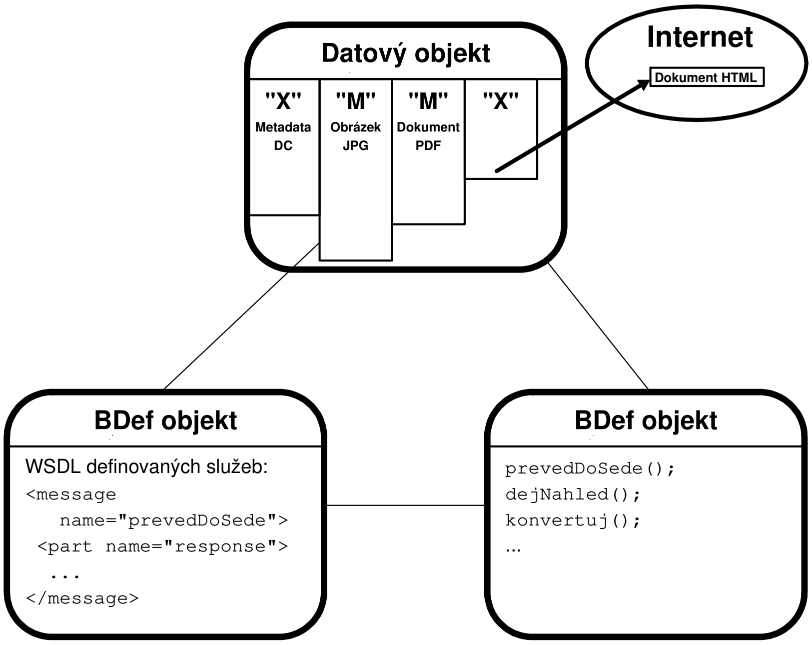
Referenced External Content – (značí se „**E**“). Jde vlastně o ukazatel na objekt, který je fyzicky uložen mimo repozitář.

Repository-Managed Content – („**M**“). Obsah, který je spravován fyzicky přímo repozitářem. Takto přijatá data jsou zkopírována přímo do repozitáře a nadále uchovávána v databázi serveru.

Implementor-Defined XML Metadata – („**X**“). Metadata v XML, která jsou uchovávána přímo v objektu, mohou být šířena stejným způsobem jako klasické datastreamy. Každý datový objekt standardně obsahuje datastream typu „X“. Tento datastream má název DC a obsahuje metadatový popis objektu dle Dublin Core. Uživatel si však může přidat vlastní datastreamy typu „X“ (vlastní metadatové popisy).

Redirected Content – („**R**“). Jedná se o speciální typ datastreamů, který je zaveden kvůli uchovávání audia a videa, které může být poskytováno vzdáleně (tzv. streamovaně – video nebo audio je na žádost po malých částech posíláno na klientský počítač, kde je průběžně přehráváno). Z tohoto důvodu je nutné zavést přímé spojení mezi klientem a poskytovatelem (poskytovatel může nabízet různé datové toky atp.). Systém Fedora proto nabízí možnost obejít klasické spojení, které zprostředkovává – tedy poskytne klientovi přímý odkaz na fyzické umístění objektu.

Typický scénář vytvoření digitálního objektu vypadá následovně. Nejdříve se vytvoří datový objekt. Jeden jeho datastream bude typu „X“ (s názvem DC) a bude obsahovat metadatový popis objektu. Dále může tento datový objekt obsahovat různé typy dalších datastreamů. Podle typu těchto datastreamů je možné k objektu přiřadit několik objektů typu BDef. Každý objekt BDef má odkazy na nějaký objekt BMech, který služby nabízené objektem BDef realizuje. Těchto BMech objektů (což jsou vlastně samotné implementace webových služeb) může být více a lze pak snadno zaměnit jednu implementaci služby za jinou. Posledním krokem je svázání konkrétní služby s příslušným datastreamem. Služba se skládá obvykle z několika funkcí (metod), které lze nad příslušným dříve asociovaným datastreamem volat. Pro lepší představu uvedeme konkrétní příklad. Vytvoříme si datový objekt s názvem *Fotografie budovy ÚVT*. Vyplníme příslušná metadata (uložená v datastreamu typu „X“) a importujeme soubory různých typů

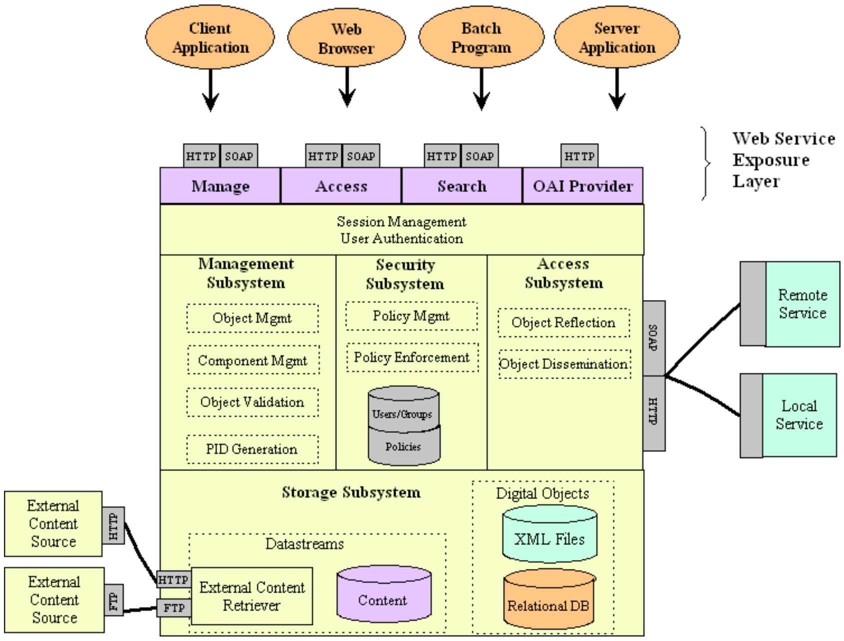


Obrázek 3.2: Digitální objekt v systému Fedora.

a velikostí s fotografiemi budovy ÚVT, to budou datastreamy typu „M". Jedna z fotografií je ve formátu JPG. Vybereme BDef objekt, který se jmenuje *Služby pro zpracování obrázků* – ten deklaruje metody `dejNahled()`, `prevedDoSede()`, `konvertuj()` a jiné. Přiřadíme ho jako možný model chování pro datový objekt *Fotografie budovy ÚVT*. K tomuto BDef objektu přiřadíme konkrétní BMech objekt, který dané služby realizuje (ví, kde jsou k dispozici). Nyní zbývá spojit naši fotografii ve formátu JPG s vybraným BMech objektem. Tímto je digitální objekt vytvořen. Pokud tedy budeme žádat po repozitáři danou fotografii, uděláme to přes příslušnou službu, se kterou je fotografie spojena. Například chceme tuto fotografii pouze ve stupních šedi, repozitář nám ji přes diseminátor `prevedDoSede()` poskytne. Systém Fedora přiřazuje každému objektu výchozí diseminátory, které realizují základní operace, jako je vrácení položky (v případě, že chceme od repozitáře naši fotografii budovy ÚVT, tak jak je v něm uložena) apod.

3.3.2 Systémová architektura a přístup

Systém Fedora je postaven na klasické vícevrstvé architektuře. Nejnižší vrstva realizuje fyzické uložení digitálních objektů (datastreamů, metadat, odkazů přes protokoly HTTP a FTP na externí obsah), další vrstva pak obsahuje systémy pro přístup (napojení na webové služby), správu obsahu a přístupovou politiku. Nejvyšší vrstva je rozhraním pro přístup do systému (přístup do repozitáře, jeho správa, dávkové programy, . . .).



Obrázek 3.3: Architektura systému Fedora (převzato z [37]).

Přístup k repozitáři je umožněn přes administrátorského klienta nebo webové služby, které jsou řazeny do tří kategorií, podle typu jejich volání. Máme tedy tři rozhraní pro přístup:

Management API (API-M) – Rozhraní pro práci s digitálními objekty, pro jejich správu (přidávání, mazání, správa verzí, . . .). Implementováno je jako webová služba (přes protokol SOAP (Simple Object Access Protocol) [27]).

Access API (API-A) – Rozhraní pro přístup k objektům v repozitáři (rozšiřování informací, hledání informací, . . .). Opět jako SOAP. (Možnost poskytování přes proxy server, viz sekce 3.3.3).

Access-lite API (API-A-Lite) – Totéž jako Access API, ale přístup je řešen přes protokol HTTP (metodou GET).

Takto definované přístupové body dávají uživateli volnou ruku v možnosti vytvoření vlastního přístupového rozhraní. Systém Fedora implementuje vlastně programátorskou knihovnu poskytující jádro pro digitální knihovnu. Všechny poskytované webové služby pro přístup přes rozhraní API-M a API-A jsou definovány v dokumentaci, kterou lze nalézt na www stránkách systému Fedora (bude-li dále zmiňována dokumentace, pak se jedná o tuto dokumentaci) a jejich popis by překročil záměr této práce. Zajímavě řešený je přístup přes API-A-Lite. Je umožněn přes protokol HTTP metodou GET. Požadavky jsou tedy zadávány přímo v URL Fedora serveru, například URL volání

```
http://domain.name/fedora/get/PID/bDefPID/
       methodName[/dateTime][?parmArray]
```

odešle na server, na kterém běží systém Fedora, požadavek na objekt (v něm uložený konkrétní datastream) určený vloženými parametry. `get` říká, že chceme data, příslušející objektu s identifikátorem `PID`. Dále je zadán identifikátor objektu chování `/bDefPID`, který je danému objektu přidělen a nakonec je určena metoda `methodName`, která se má na daném objektu (některém z jeho datastreamů) provést. Konkrétně může výše zmíněné obecné volání vypadat takto

```
http://domain.name/fedora/get/foto_UVT:5/chovani_UVT:1/
       getThumbnail?itemID=fotografie1
```

Výsledkem volání je náhled `fotografie1` budovy ÚVT. Identifikátor `foto_UVT:5` určuje objekt, příslušnou konkrétní fotografii (datastream) určuje parametr `itemID= fotografie1` metody `getThumbnail`. Na všech fotografiích týkajících se budovy je pak definována například politika chování s PID *chovani_UVT:1*.

V dokumentaci jsou podrobně a přehledně popsány metody pro vyhledávání, získávání objektů, získávání metadat (přes OAI-PMH) a další. Údaje jsou vraceny buď v HTML tabulce, která informace zobrazí v prohlížeči nebo v XML (pro další zpracování na straně klienta).

3.3.3 Poskytování dat

Klasický objekt se obvykle skládá z datastreamů. Ve skutečnosti se jedná o ukazatele na uložení konkrétních dat. Digitální objekty poskytujeme i mimo repozitář vzdáleným počítačům, což znamená, že přes jisté mechanismy může vzdálený klient přistupovat k datastreamům. Tyto mechanismy, které vzdálený přístup umožňují, ovšem dávají zároveň s objektem i informaci, kde jsou jeho datastreamy fyzicky uloženy. Nedůvěryhodný klient pak může z této informace vytěžit přístup přímo k daným datastreamům a obejít tak mechanismus diseminátorů, který systém Fedora poskytuje. Aby se zamezilo tomuto stavu, používá systém Fedora zprostředkování datastreamů (Datastreams mediation).

V podstatě jde o zavedení jednoduché proxy vrstvy, která zakrývá fyzické umístění digitálního objektu. V momentě, kdy přijde dotaz, proxy jej převezme za svůj, vyřídí a vrátí klientovi informace (případně konkrétní data jako proud bytů). Tímto je klient nucen komunikovat přímo s repozitářem a nedostane se k fyzickému umístění dat. Nevýhoda tohoto přístupu je, že zavádí další vrstvu, která komunikaci zpomaluje.

3.3.4 Správa verzí

Nejnovější verze systému Fedora přináší správu verzí (Versioning). V praxi to znamená, že při jakékoli změně datastreamu nebo diseminátoru se původní verze uloží (včetně data a času) a je kdykoli přístupná zpětně. Všechny tyto verze datastreamů a diseminátorů jsou stále udržovány v rámci jednoho digitálního objektu. Uživatel se může vrátit k jakékoli verzi (například k původní fotografii, která byla nahrazena kvalitnější verzí). Jednotlivé verze lze samozřejmě smazat.

3.4 Fedora verze 1.2

3.4.1 Instalace

Všechny potřebné programy jsou volně dostupné na Internetu. Kromě samotného distribučního balíku systému Fedora je k instalaci potřeba pouze standardní instalace vývojového prostředí J2SDK (implementace jazyka Java firmy Sun Microsystems [28]) ve verzi 1.4 a instalace databázového serveru (vynikající je podpora několika databází, doporučená je MySQL [29]). Výhodné je, že lze použít malý javový databázový stroj Mc-Koi obsažený přímo v distribučním balíku systému Fedora (je doporučený pro testování). Pro překlad ze zdrojových kódů je též nutný Apache Ant [30] (ale součástí distribuce jsou i binární verze jak serveru, tak klientských programů). To, že veškeré knihovny a další programy jsou obsaženy přímo v distribuci, je možno hodnotit velmi pozitivně, protože není nutné dělat velké zásahy před překladem. Diskutabilní je zařazení do distribučního balíku www serveru Tomcat [31].

Samotná instalace je dobře dokumentovaná a snadná. Sestává z nastavení asi dvou až tří proměnných. Vše se přehledně přeloží do jednoho adresáře, který se nainstaluje pouhým zkopírováním a nastavením systémových proměnných.

Před spuštěním je nutné provést konfiguraci sytému. Vše je opět výborně popsáno a dokumentováno. Konfigurační soubor je v XML. Konfigurace je velmi pestrá, nastavit lze prakticky cokoli, nicméně pro rychlejší orientaci před prvním spuštěním autoři v dokumentaci uvádějí nejdůležitější prvky spolu s radami, jak je nastavit. Celý konfigurační soubor je přehledně dělen na části (nastavení serveru, modulů a databází). Nadefinovat lze porty, jméno serveru nebo logovací soubory. Také lze měnit jednotlivé javovské třídy (moduly) obsluhující příslušné funkce (např. pro přístup k databázi – ukládání digitálních objektů atp.)– to znamená, že se dá standardní modul nahradit modulem vlastním. U standardního modulu můžeme nastavit jaký řetězec se bude přidávat do identifikátorů PID nových objektů (jednoznačný pro daný repozitář, například DOI[1]), jaké kódování se volí pro ukládání textu atd. Jiný modul upravuje chování vůči OAI serverům, nastavovat lze kódování metadat (výchozí nastavení je METS) nebo takové detaily jako je jméno, kterým se bude systém představovat OAI serverům.

Po spuštění lze k prvnímu seznámení se systémem Fedora využít několik tzv. demo objektů. Jedná se o předem připravené různé typy digitálních objektů včetně diseminátorů.

3.4.2 Programové vybavení – klient

Spolu s Fedora serverem se distribuuje i řada užitečných utilit pro práci s repozitáři.

Aplikace pro správu

K dispozici je interaktivní grafický program (Fedora Administrative Client) napsaný v Javě, který je uživatelskou bránou k modulu API-M a umožňuje správcům repozitářů základní operace nad objekty (přidávání, zálohování, vytváření, procházení, hledání). Po spuštění je nejdřív nutné přihlásit se k repozitáři (samozřejmostí je přístup ze vzdáleného počítače).

Přidáváním nového objektu z menu se v repozitáři vytvoří kostra nového objektu. Vytvářet lze jak datové objekty, tak objekty chování. V dokumentaci je podrobný průvodce přidáváním pro jednotlivé typy. U datového objektu je nutné ho asociovat s chováním, přidat datastreamy, popř. editovat metadata (tohle už se musí udělat ručně v hotové XML šabloně – ale postup je standardní, např. úprava titulku `<dc:title> </dc:title>`).

[1] Digital Object Identifier, viz kapitola 2

Datastreamy se dají přidávat dle typů. Pokud jsou datastreamem metadata v XML, lze zvolit ještě typ (copyright, technická, ...), dále druh metadat (DC, MARC, ...), a pak data doplnit ručně nebo importovat z XML souboru (není nutné mít je validovány podle XML schématu). U normálního datastreamu se pak ještě zvolí jeho MIME typ [32], který je důležitý pro asociování s diseminátory (každý diseminátor akceptuje pouze určité MIME typy).

Klient umí importovat objekty ze souboru, z jiného repozitáře (zde je nutné znát identifikátor PID objektu ve vzdáleném repozitáři). Objekty však lze importovat i hromadně. Objekty lze samozřejmě i exportovat, mazat, číst jejich XML záznamy, vyhledávat (včetně pokročilého vyhledávání) a prohlížet. Veškerá práce probíhá za pomoci kontextových menu, označit lze například i více objektů a některé akce nad nimi provádět hromadně. Správa verzí (viz část 3.3.4) je při prohlížení zobrazena časovou lištou na níž si lze posunem ukazatele vybírat příslušné verze objektů.

Dalšími funkcemi aplikace Fedora Administrative Client jsou hromadné vkládání objektů pomocí šablon, terminál pro ladění systému Fedora (umožňuje zkoušet si jednotlivá volání včetně parametrů a sledovat výsledky) a nástroj na vytváření objektů chování (s hezky zpracovaným průvodcem).

Skripty

Kromě výše popsaného klienta má systém Fedora v instalaci ještě několik skriptů (spouštějí se z příkazové řádky operačního systému), které usnadňují práci administrátorovi. K dispozici jsou skripty pro vyhledávání, export, vkládání objektu ze zdrojového XML souboru a mazání objektu.

Přístup k repozitáři přes www prohlížeč

Fedora server poskytuje pouze minimální možnosti přístupu přes www – nabízí pouze stránku s formulářem pro jednoduché vyhledávání a chudé informační stránky. Lze vyhledávat, nechat si vypsat informace o repozitáři, informace o objektu a službách, které jsou k němu vázány a tyto služby na něj zavolat. Vzhled, přístup a intuitivnost byly zcela potlačeny na úkor čisté funkcionality – v praxi to znamená, že je k dispozici pouze naprosté minimum pro přístup k funkcím repozitáře. Na druhou stranu lze snadno ověřovat například funkčnost diseminátorů, zkontrolovat byl-li objekt správně vložen apod. Absence jakýchkoli uživatelských práv, vzhledově a uživatelsky kvalitnějšího rozhraní odsuzují tento „webový server" pro použití pouze administrátorem k testování, případně pro ukazování síly samotného systému Fedora.

3.4.3 Programové vybavení – server

Jak bylo řečeno výše, každý datastream lze spojit s nějakou službou. V distribučním balíku systému Fedora jsou dodávány některé lokální služby (Local services), které jsou uloženy přímo v repozitáři. Služby mohou být uloženy i mimo repozitář a volány mohou být vzdáleně přes SOAP/RPC (SOAP Remote Procedure Call). Implementovány jsou většinou jako Java servlety – speciální třídy s podporou protokolu HTTP. Problematika servletů je podrobně rozebrána například v [33]. Systém Fedora poskytuje následující služby:

Saxon XSLT Processor Local Service – K zadanému XML dokumentu a XSLT stylu vyrobí výstupní dokument. Volání se provádí přímo na serveru, kde běží Fedora, volá se servlet **SaxonServlet**, důležité parametry jsou vstupní XML soubor a styl XSL.

FOP Local Service – Jedná se o FO procesor (FOP – Formatting Objects Processor [34]), transformuje dokumenty napsané v XML-FO [35] do PDF (případně i dalších typů dokumentů). Volání se provádí metodou GET s voláním servletu **FOPServlet** na serveru, kde Fedora běží. Argumentem, je soubor v XSL-FO.

The Image Manipulation Local Service – Slouží k různým transformacím obrázků (změna velikosti, převod na šedotónní obraz apod.). Jedná se o servlet **ImageManipulation** volaný na Fedora serveru. Jako parametry přijímá URL na soubor s obrázkem a operaci, která se má provést. Možné operace jsou oříznutí, konverze (GIF, JPG, TIFF, PNG, a BMP), změna velikosti, přiblížení, změna světlosti a převod do stupňů šedé. Služba je postavena na programu **ImageJ** [36].

3.5 Dokumentace

Dokumentace je zpracována velmi kvalitně, včetně popisu instalace. Podrobně jsou vysvětleny speciální funkce, popis nejdůležitějších nastavení, popis konstrukce objektů pomocí XML, kompletní seznam všech volání pro jednotlivá rozhraní (včetně příkladů) a odkaz na další informace a tutoriály. Všechny konfigurační soubory mají přehledné komentáře u jednotlivých položek, včetně doporučených hodnot, které může uživatel nastavit (i s vysvětlením proč zrovna tuto hodnotu nastavit určitým způsobem). Příkladem, jak by měl vypadat tutoriál, je průvodce pro aplikaci Fedora Administrative Client, který provádí uživatele krok za krokem jednotlivými menu spolu s náhledy obrazovek a vysvětlením akcí a jejich odezev. Pro programátory jsou k dispozici ukázky volání webových služeb v programovacích jazycích Perl, C# a PHP.

3.6 Závěr

Systém Fedora není v současné době kompletní hotový knihovní systém, který je možno okamžitě nasadit do běžného provozu. Jedná se spíše o jakési jádro, základ pro budování digitální knihovny. Nevýhodou tohoto přístupu je, že v současném stavu vývoje není systém hned připraven na „ostré" nasazení do provozu. Zejména v oblasti přidávání objektů a administrace je situace horší. Buď je nutné používat aplikaci Fedora Administrative Client (v Javě pro jakoukoli platformu, takže to není takový problém) nebo si naprogramovat vlastní prostředí. Bohužel v aplikaci Fedora Administrative Client může objekty přidávat jen administrátor (resp. je nutné znát pro přidávání objektů heslo). Navíc neexistuje něco jako API-M-Lite pro přístup přes metodu GET a je nutné tyto věci dělat čistě přes SOAP/RPC. Samotná prezentace objektů uživatelům není v přijatelné formě také vyřešena. Podrobněji jsou možnosti konkrétního praktického využití popsány v [37].

Z výše popsaného plyne, že neexistují uživatelské účty (i když je autoři slibují v budoucích verzích). Dále nelze realizovat tvorbu zanořených složek s objekty a tím provádět jejich třídění – nelze to obejít ani pomocí digitálního objektu typu „E" (External Content, autoři na stránkách projektu uvádějí, že to bude možné ve verzi 1.3). Lze pouze odkazovat na datastreamy.

Další nedostatek se týká národních prostředí. Při importu objektů z jiného Fedora repozitáře se bohužel špatně importovaly znaky s interpunkcí, podobný problém nastal u vyhledávání. Při práci na lokálním počítači (běží na něm server Fedora) při prohlížení a vkládání objektů vše funguje dobře, u vyhledávaní však nikoli – na dotazy obsahující české znaky s diakritikou nevrací žádné výsledky. Pravděpodobně to ukazuje na vnitřní problém, protože data byla odeslána serveru správně (ověřeno přímou kontrolou odesílaných paketů). Lze předpokládat, že s dalším vývojem systému Fedora a jeho přijímáním uživateli mimo anglosaské prostředí bude tento problém vyřešen.

Závěrem lze nabídnout přirovnání vystihující roli systému Fedora mezi ostatními systémy. Systém Fedora je spíš skládačka monopostu formule 1 – dostáváte do rukou výkonný motor a podvozek, a je jen na vás, jaké pneumatiky a kryty zvolíte. Každopádně základem je stále monopost F1, který může být rychlý a výkonný, má k tomu ty nejlepší předpoklady. Na druhou stranu nestačí jen nasednout a jet, je nutné umět tento monopost řídit a ovládat. Hotové systémy pro digitální knihovny, jakými jsou DSpace a EPrints, lze přirovnat k pohodlnému rodinnému vozu – snadno se řídí a ovládá, jeho výkonové schopnosti se však obtížněji mění a uživatel pořádně neví, co je vlastně „pod kapotou“.

Kapitola 4

RIB

Název RIB [38] je zkratkou ze spojení **Repository In a Box**. Jedná se o systém, který je zamýšlen jako univerzální repozitář a knihovní systém pro správu metadat se zaměřením na softwarové balíky. Uživatel dostává webové rozhraní a nástroje a okamžitě po instalaci je systém RIB připraven pro nasazení do provozu (pokud se uživatel spokojí s tím, co je v základní konfiguraci nabízeno). Využívá hojně standardů organizace IEEE [39]. Je vyvíjen na University of Tennessee za podpory a řízení z NHSE [40] (National HPCC Software Exchange) – odtud také zaměření na správu software.

Obrázek 4.1: Úvodní stránka správce systému RIB.

Systém RIB je na pomezí mezi publikačním systémem pro správu objektů a digitální knihovnou. Filozofie práce vychází spíše z publikačních systémů, proces udržování dat v repozitáři a jeho sdílení ven pak spíše z oblasti digitálních knihoven. Částečně se však systém RIB blíží i k systému Fedora. Stejně jako systém Fedora má systém RIB k dispozici balík služeb (volání), na kterých může programátor postavit vlastní rozhraní k systému.

Se systémem RIB jsou dodávána dvě výchozí uživatelská rozhraní – administrátorské a přístupové (pro uživatele).

4.1 Historie

První **verze 1.0** systému RIB vznikla v roce 1997. Cílem bylo vybudovat katalog digitálních objektů (dat) a metadat za pomoci webových technologií. Již v té době bylo rozhodnuto o použití standardu BIDM (Basic Interoperability Data Model, více v části 4.3.2). Generování www stránek pro přístup k datům v repozitáři bylo prováděno za pomocí perlových CGI skriptů a HTML formulářů. Pro podporu certifikace a verifikace bylo k modelu BIDM přijato rozšíření ACF (Asset Certification Framework), pro správu autorských práv a přístupu rozšíření IPRF (Intellectual Property Rights Framework). Tyto standardy byly použity i v následující verzi systému RIB a tou je **verze 2.0**.

Systém RIB verze 2.0 byl dán k dispozici v roce 2001. Od doby vzniku původní verze se objevilo mnoho nových technologií a standardů, na kterých byl nový RIB postaven. Z velké části byla přepsána administrátorská část, která je nyní v Javě. Je více využíváno XML spolu se standardem RDF. Přidána byla možnost přidávat nové třídy modelu BIDM, což dovolilo používat RIB jako univerzální repozitář pro jakékoli typy dat.

V roce 2003 byla uvolněna **verze 2.2**. Obsahuje opravy chyb a několik spíše drobnějších vylepšení. Na konec roku 2004 slibují autoři novou verzi, nyní již kompletně přepsanou v jazyce Java. Tato verze má být stejným přelomem jako verze 2.0 ve srovnání s verzí 1.0.

4.2 Základní vlastnosti

Systém RIB je zčásti naprogramován v jazyce Perl (s použitím několika modulů pro www programování), zčásti v jazyce Java. V jazyce Perl je naprogramována část pro prohlížení a přístup k administrátorské části, administrátorská část je pak realizována Java appletem (aplikace napsaná v jazyce Java, která se stáhne ze vzdáleného serveru na lokální počítač, kde běží obvykle v okně prohlížeče). Jako www server je použit Apache, pro správu databáze MySQL server. Není to čistě open source projekt, zdrojové kódy tříd napsaných v Javě nejsou k dispozici (s autory lze však jejich zpřístupnění domluvit). Systém je k dispozici pro většinu běžných operačních systémů (Linux, Solaris, AIX, IRIX, Windows).

Systém RIB používá speciální model **Basic Interoperability Data Mo-**

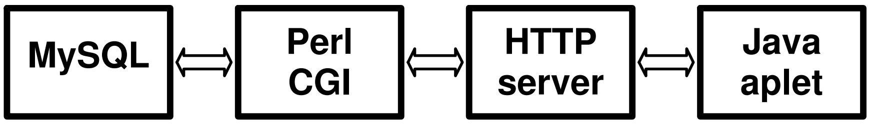
del (BIDM). Jedná se o standard organizace IEEE, konkrétně číslo 1420.1. RIB je jediným z testovaných systémů, který se hlouběji zabývá problémem mezinárodních autorských práv, certifikací a verifikací (zejména softwaru). Na digitálních objektech definuje (jako rozšíření modelu BIDM) speciální IEEE standard **Intellectual Property Rights Framework** (IPRF), který přidává metadata pro určení práv (licencování apod.) a **Asset Certification Framework** (ACF) zajišťující metadata pro certifikáty (software). Pro kódování metadat používá standard **RDF**.

V systému RIB je zabudována podpora pro současný provoz více nezávislých repozitářů. Každý repozitář může mít vlastní definovaný model a vzhled webového rozhraní (katalog). Správa jednotlivých repozitářů je prováděna přes dodávanou aplikaci (podrobnosti viz část 4.4).

4.3 Architektura

4.3.1 Systémová architektura a RIB API

Systém RIB se skládá ze čtyř hlavních modulů. Jsou to databáze MySQL, Perl CGI, HTTP server (Apache) a Java applet. Přístup do databáze se realizuje pouze přes Perl CGI modul, který samotný realizuje vnitřně jádro systému a je nejdůležitějším modulem. Administrátorskou část pak zajišťuje modul Java applet.



Obrázek 4.2: Moduly systému RIB.

Na HTTP serveru systému RIB je pomocí skriptů CGI definováno rozhraní **RIB API** – Application Programmer's Interface. Jedná se o programátorskou knihovnu funkcí, které realizují různé operace nad repozitářem. Tato knihovna je k dispozici pro programovací jazyky Perl, Java a C. Testována byla knihovna pro jazyk C. Programování s její pomocí je přehledné a na slušném stupni abstrakce (není nutné řešit nízkoúrovňové záležitosti). Prostým `#include "rib.h"` získáme přístup k funkcím nad repozitářem. Například `catalogInfo()` vrací informace o repozitáři (v XML), `changeName()` mění jméno repozitáře a další. Ke všem datům, která funkce vracejí v XML, jsou definovány příslušné DTD (Document Type Definition, informace o tomto standardu lze nalézt například na [41] nebo [42]). Programátor však nemusí využít jen připravené knihovny, ale může psát i vlastní knihovny, pouze je pak odkázán na nízkoúrovňové programování (je nutné pomocí metody POST protokolu HTTP posílat správná data CGI skriptům). Kompletní RIB API referenční příručka je k dispozici na [43].

4.3.2 BIDM

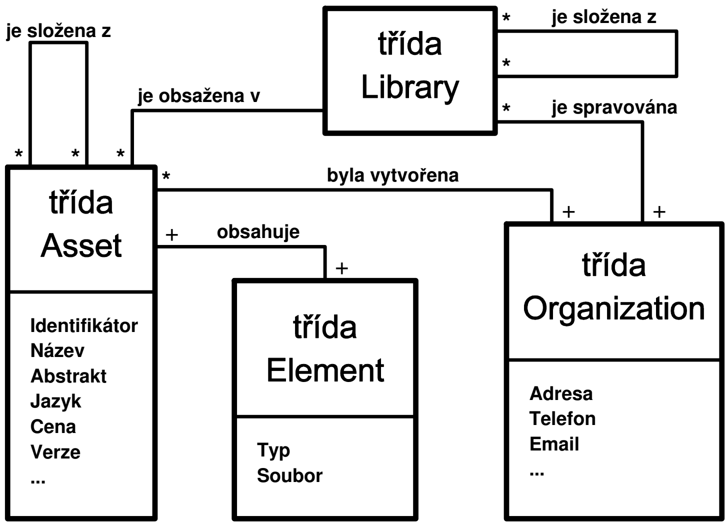
BIDM je základní model, který používá systém RIB pro ukládání digitálních objektů. Jedná se o IEEE Standard 1420.1 pro výměnu a kódování metadat (podobně jako Dublic Core s METS). Je to objektový model se základními třídami s atributy, který lze rozšiřovat o další třídy s jinými atributy. Atributy v jednotlivých třídách reprezentují metadata digitálního objektu, jeho provázanost s jinými objekty a digitálními knihovnami. Základní model BIDM má definovány 4 třídy.

Hlavní a nejzákladnější je třída **Asset**, reprezentující abstraktní objekt. Tato třída má definována metadata pro popis objektu a speciální metadatové položky pro definování vztahů. Objekty této třídy jsou typicky dokumentace k projektu, zdrojové kódy programu, testovací balíčky (jejich metadatové popisy) atd. Možné atributy jsou název, abstrakt, klíčová slova, jazyk, verze subjektu, cena, omezení a další – popisy, které lze u softwarových balíků očekávat. Kromě toho obsahuje tato třída i další atributy vyjadřující vztahy k ostatním třídám. Důležitý je atribut odkazující na jiné objekty třídy Asset – tímto je zaručena velmi jemná granularita ve smyslu složek a kolekcí, protože získáme prakticky neomezenou hloubku zanoření a možnosti vytvářet jemné struktury (viz část 4.4.2). Dalšími atributy realizujícími vztahy jsou atributy odkazující na zbylé tři základní třídy.

Další třídou modelu BIDM je třída **Element**. Objekt této třídy obsahuje konkrétní fyzický soubor (např. přeloženou hotovou aplikaci) spolu s minimálními atributy. Objekt třídy Element se váže na objekt(y) třídy Asset. Jeden objekt třídy Asset může vázat více elementů a jeden element může být vázán na více objektů třídy Asset.

Třída **Library** je důležitou třídou, definující abstraktní digitální knihovnu, která obsahuje objekty třídy Asset, tedy digitální objekty. Platí, že digitální objekt třídy Asset může být uložen ve více knihovnách (minimálně však v jedné). I objekty třídy Library mohou být navzájem spojovány a zanořovány.

Poslední základní třídou je **Organization**. V logickém členění hierarchicky nejvyšší třída. Může odkazovat na několik objektů třídy Library a Asset, které „spravuje“. Základními atributy jsou adresa, email, telefon a fax.



Obrázek 4.3: Přehledový diagram modelu BIDM.

Přesný model (diagram) lze nalézt v dokumentaci systému RIB (dostupná na www stránkách systému, dále bude označována pouze jako dokumentace). Popis jednotlivých metadatových položek (atributů) tříd je v dokumentaci velmi stručný (s odkazem na stránky organizace IEEE). Pomocí administrátorského klienta lze vytvářet své vlastní třídy a metadatové položky – pak však nelze zaručit interoperabilitu s jinými repozitáři (viz část 4.3.4).

4.3.3 Rozšíření BIDM modelu

Použití základního modelu BIDM zaručí interoperabilitu mezi repozitáři, ale pro potřeby ukládání a distribuce softwarových balíků není dostatečné. Proto umožňuje RIB použití dalších tříd definujících bohatější metadatové položky při zachování interoperability. Tato rozšíření jsou směřována opět na oblast software.

Prvním rozšířením je **Asset Certification Framework** (ACF), které definuje několik tříd. Jejich provázání je patrné z diagramu, který lze nalézt v dokumentaci. Struktura je poměrně bohatá, bohužel však špatně popsaná. Pro bližší informace autoři opět odkazují na stránky IEEE.

Druhým rozšířením je **Intellectual Property Rights Framework** (IPRF). Opět definuje několik tříd (*Licencing terms*, *Rights policy* a *Rights Category*) a váže je na objekty tříd Element, Library a Asset.

Tato rozšíření jsou standardy IEEE (1420.1a a 1420.1b). Zde vyvstává nepříjemný problém – autoři popisují všechny atributy velmi stručně nebo vůbec. U některých lze jen vytušit jejich význam. Při pátrání po bližším popisu se nakonec dostáváme vždy na stránky IEEE, kde si lze popis standardu spolu s několika třídami zakoupit (za cca 153 USD). Striktní dodržení definovaných standardů tedy bohužel není zadarmo.

Následující dvě rozšíření jsou z dílny NHSE (National HPCC Software Exchange). Týkají se uživatelů, kteří chtějí používat RIB jako repozitář pro softwarové balíky. Standard **NHSE extension** doplňuje do třídy Asset nové atributy (URL na stránku produktu apod.) a **NHSE Deployment** definuje novou třídu *Deployment*, která může sloužit pro vytváření dalších spojení mezi třídami – autoři uvádí příklad spojení mezi třídou Machine a Asset. Prakticky software reprezentovaný objektem třídy Asset je spojen s třídou Deployment, která je napojena na objekt třídy Machine, tedy nějaký počítač. Tato vazba pak říká, který software je instalován na kterém počítači – lze pak z této vazby generovat přehledné tabulky ukazující, kde je co nainstalováno.

4.3.4 Interoperabilita

Interoperabilita je zaručena jen mezi servery podporujícími BIDM standard. Změny typu metadatových záznamů tuto interoperabilitu porušují. Jak je zmiňováno výše, uživatel si může díky dodávaným nástrojům definovat libovolné modely, třídy a jejich provázanost, s tím rizikem, že ztrácí možnost kontaktu s okolním světem (pokud není domluven nějaký jiný standardní model – nejlépe takový, který by nebyl zpoplatněn žádnou organizací – zatím však takový v prostředí systému RIB neexistuje). Podpora rozšířeného protokolu pro sklízení metadat OAI-PMH v systému RIB není deklarována. Stejně nikde nejsou zmíněny další standardy (Dublin Core). Pokud bychom chtěli použít OAI-PMH a jiné volně dostupné standardy, bylo by potřeba vše realizovat vlastními silami za použití rozhraní RIB API. Znamenalo by to nutnost provádět transformace v XML a dopsat části pro OAI-PMH server. Otevřenou otázkou zůstává, jaké problémy by se vynořily v průběhu realizace – je možné, že by se rozhraní RIB API ukázalo jako nedostatečné.

Technicky je interoperabilita zajištěna tak, že metadata objektů jednoho repozitáře jsou odkazována do katalogu jiného repozitáře (je dělán pouze odkaz, přístupová práva a další zůstávají v kompetenci repozitáře, který data fyzicky spravuje). Toto odkazování lze provádět pouze mezi RIB repozitáři a příslušnými metadatovými modely. Prakticky se vzájemná výměna metadat řeší dávkově spouštěným skriptem.

4.4 RIB verze 2.2

4.4.1 Instalace

Instalace je na první pohled velmi snadná. Proběhne spuštěním jediného dávkového skriptu, který udělá vše. Až na nastavení hesla administrátora, instalačního adresáře, doménového jména a portu počítače není třeba udělat nic. Veškerý software nutný pro systém RIB je obsažen přímo v distribuci. Bohužel to platí i pro Apache, MySQL a další balíky, u kterých se dá předpokládat, že již budou na koncovém počítači nainstalovány. Nikde na webových stránkách také není možné nalézt návod, jak systém RIB nainstalovat jinak než standardní cestou – vývojáři pouze, emailem na konkrétní dotaz, sdělují, že žádný takový podrobnější návod neexistuje (s doporučením prostudovat asi 50 kB velký instalační skript . . .).

V praxi se však instalace neukázala tak snadnou. Pro správnou funkčnost systému je nutné mít nastavené prostředí tak, aby to vyhovovalo instalačnímu skriptu. Při troše štěstí proběhne instalace dobře – v opačném případě to znamená spoustu práce, počínaje procházením souboru s chybovými hlášeními a ručními opravami instalačního skriptu konče. Vzhledem k tomu, že se dá předpokládat, že systém RIB nebude zprovozňovat uživatel začátečník, je řešení instalace nevhodné. Ve skutečnosti (při absenci podrobné dokumentace) přidá víc práce než užitku.

4.4.2 Prostředí pro správu

K repozitářům se sice přistupuje přes webové rozhraní, ale samotné rozhraní v HTML tvoří jen malou část. V ní lze vytvářet a mazat repozitáře, přistupovat do prostředí pro správu repozitáře, importovat jiné repozitáře RIB a nahrávat soubory do repozitáře (tyto soubory pak mohou být asociovány s objekty v repozitáři). V HTML je také prezentován tzv. katalog – stránky určené uživatelům k prohlížení obsahu repozitáře a vyhledávání v repozitáři. Zbytek prostředí správy repozitáře je kompletně naprogramovaný jako Java applet. Což znamená, že je nutné mít webový prohlížeč s Java pluginem. Vypadá to tak, že v okně prohlížeče běží korektní Java aplikace.

V ní se odehrává i celá práce s repozitářem. Je zde několik záložek (stejně jak v běžné okenní Java aplikaci) pro různá nastavování repozitáře. První záložkou je stručná nápověda pro ostatní záložky. Přidávání, mazání a editace objektů se provádí pomocí menu, k dispozici je i stručná nápověda. Vytváření objektů všech základních typů je snadné. Opět je jediný problém v tom, že nejsou dostatečně popsány jednotlivé metadatové položky, takže není úplně jasné, co do nich vyplňovat. Ale tento problém by při vytvoření vlastního modelu odpadl.

Další záložka obsahuje nástroje pro úpravu katalogu (nastavení vzhledu katalogu – webových stránek přístupných uživatelům). Následuje záložka pro nastavení interoperability. Konečně poslední záložka se týká nastavení repozitáře. Zde je možné definovat vlastní třídy objektů a vlastní metadata – nový metadatový model. Kromě toho lze nastavit jméno repozitáře, heslo pro přístup a další údaje.

Pro vytváření vlastních datových modelů je administrátorovi k dispozici aplikace **Data model editor** (je součástí administrátorského rozhraní). S její pomocí lze vytvářet třídy, jejich podtřídy, zadávat vztahy mezi třídami a definovat metadata tříd. Vytvořený model lze pak snadno zahrnout do katalogu a okamžitě začít používat. Editor je velmi intuitivní a přehledný, opět má však poměrně stručnou dokumentaci, která začne chybět v momentě, kdy se pokusíme vytvořit nějaké speciálnější třídy.

4.4.3 Katalog

Katalog je částečně generován v závislosti na obsahu repozitáře. Je to veřejná část repozitáře, se kterou budou pracovat koncoví uživatelé. Jedná se o klasické stránky HTML, které jsou poměrně jednoduché a velmi přehledné. Obsahuje nástroje pro hledání v repozitáři a pro jeho procházení. Hledání je jak jednoduché, tak rozšířené – v něm lze prohledávat jednotlivé třídy objektů a jejich jednotlivé atributy (to vše generováno dynamicky). Speciálním typem hledání je tzv. slovníkové hledání (Vocabullary search) ve třídách, které mají nadefinovaný slovník. V případě BIDM modelu se jedná o členění softwaru dle typu. Například *Benchmark and Example Programs* nebo *Distributed Processing Tools*. Jednotlivé typy mohou mít další podtypy, míra zanoření je z praktického hlediska neomezená – toto je právě struktura, kterou lze vytvářet pomocí administrátorské aplikace. Granularita je tedy vynikající.

Další funkcí katalogu je sledování změn, které je však velmi jednoduché. V podstatě může uživatel pouze zadat dotaz na objekty, které byly přidány za posledních x dní (kde x je hodnota z roletového menu, buď 1 den, týden, 14 dní, měsíc, . . .).

Kromě informací o repozitáři (typu kontaktní email) už další funkce katalog neposkytuje.

Další podstatnou funkcí je editace vzhledu katalogu. Ke každé www stránce katalogu je k dispozici šablona v HTML, kterou lze editovat. Kromě samotného HTML, kterým lze ovlivnit především vzhled (barvy, loga, nadpisy), jsou k dispozici i direktivy systému RIB umožňující přístup k některým dynamicky generovaným informacím – především se jedná o jméno repozitáře a různé druhy odkazů na repozitář. Většina podstatných dynamicky generovaných informací se děje přes jedinou direktivu, která je však černou skříňkou, do kterého není vidět (dokumentace se o ní blíže nezmiňuje).

4.5 Dokumentace

Na webových stránkách autorů se toho moc podrobného najít nedá (s výjimkou popisu rozhraní RIB API), spíše je tam jen uživatelská příručka. Popis použitých standardů je slabší, odkazováno je na IEEE, u některých metadatových položek není díky stručnosti jejich popisu zcela zřejmý jejich význam (jaká data s jakou syntaxí zapisovat). Chybí nějaká administrátorská dokumentace k vlastní instalaci. Dokumentace k administrátorské části systému je obsažena on-line a je velmi stručná. Některé položky kontextových menu například nejsou popsány vůbec. Popis architektury systému je dostačující a referenční příručka k rozhraní RIB API je perfektní.

4.6 Závěr

Systém RIB je kompletní, okamžitě v praxi použitelný systém. Bohužel má několik nedostatků, které překrývají některé jeho skvělé vlastnosti. Především je to prakticky neexistující systém přístupových práv a uživatelů. Systém uživatelů a práv má být kompletně k dispozici až v nové verzi systému RIB. Nevýhodou je nutnost mít v prohlížeči Java plugin kvůli spouštění appletu (pro správu repozitářů). Administrační stránky jsou naprogramovány v Javě a na pomalejších počítačích může být proto práce zdlouhavá, vyřízení požadavku na nějakou dříve neprohlédnutou část aplikace trvá dlouho. Tyto problémy by se daly odstranit pouze kompletním napsáním vlastního rozhraní s použitím rozhraní RIB API (což je na jednu stranu výhoda, ale nevýhodné je, že to vyžaduje netriviální úsilí, protože je nutné prakticky vybudovat kompletní systém).

Sporné je použití standardů IEEE – má své výhody (silná a stabilní instituce v pozadí) i nevýhody (nutnost platby za podrobné informace o standardu). Další netriviální problém vyvstal při použití znaků s diakritikou. RIB si s nimi absolutně neumí poradit. Podpora mezinárodních znaků by měla být obsažena v následující verzi systému RIB, která již má být kompletně naprogramována v Javě. Nutno zdůraznit, že původní termín vydání této verze je (oproti původním deklaracím autorů) téměř o půl roku opožděn . . .

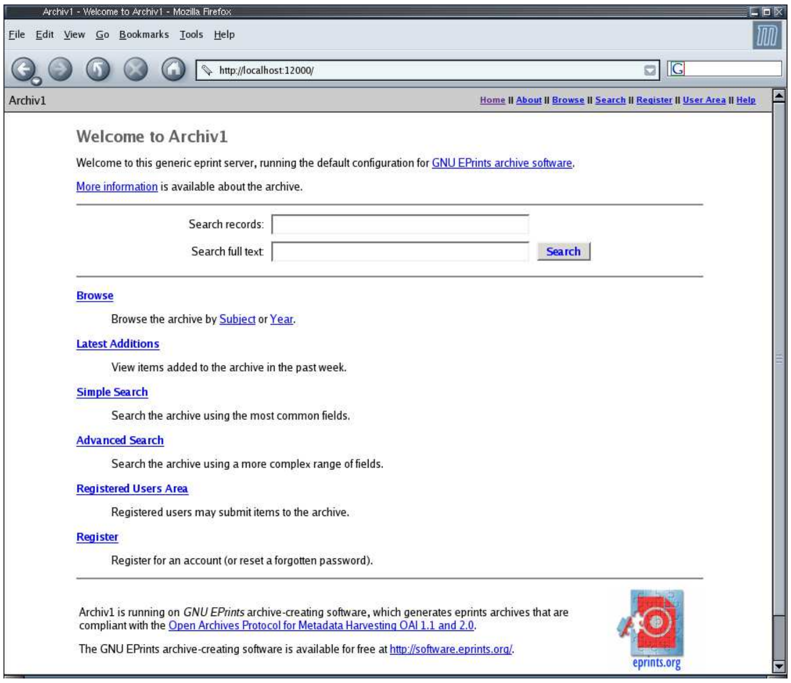
Celkově systém, až na výše uvedené nedostatky, působí vcelku kompaktně a pro účel, za jakým byl vytvořen (uchovávání a sdílení software), je bezesporu velmi kvalitní. To částečně dokládají některá praktická nasazení – systém RIB se pro sdílení a ukládání software provozuje asi na 15 serverech a používají ho i takové instituce jako NASA (neveřejně) a Army Research Laboratory armády USA (pro některé své volně dostupné softwarové produkty). Slovo „částečně" je voleno v předchozím souvětí záměrně – repozitář NASA je neveřejný a repozitáře US Army mají pouze několik objektů a z velké části jsou nepřístupné (prohlížeč vrací chybu). Čtenář sám může vyzkoušet aktuální odkazy na další repozitáře z www stránek systému RIB.

Lehká specializace systému RIB na softwarovou oblast z něj dělá nástroj, který nemůže vyhovovat naprosto všem. Autoři se pokusili umožnit uživatelům snadné přizpůsobení a jejich snaha byla částečně úspěšná – zejména pak možnost tvorby vlastních struktur datového modelu a možnost použití rozhraní RIB API. Do budoucna bude zajímavé počkat si na další verzi, která má přinést užitečná vylepšení.

Kapitola 5

EPrints

Systém EPrints [44] je určen zejména pro správu vědeckých a technických informací a pro sdílení výsledků a novinek vědeckého výzkumu (tyto informace se často označují zkratkou STI podle anglického termínu Scientific & Technical Information). Příkladem takových informací jsou diplomové práce, vědecké články, technické zprávy apod. EPrints je kompletní systém připravený k okamžitému nasazení. Ve srovnání se systémy RIB a Fedora je již vyzrálejší a uživateli poskytuje hotové řešení, včetně kompletního rozhraní, uživatelských účtů a systému přístupových práv. Systém je vyvíjen na University of Southampton ve Velké Británii. Šířen je pod hlavičkou organizace GNU [45].



Obrázek 5.1: Úvodní stránka systému EPrints.

5.1 Historie

Vývoj systému EPrints začal v roce 2000. První **verze 1.0** byla vydána v listopadu téhož roku. Již od této první verze využívá systém standardů OAI, dnes již historické verze protokolu PMH 0.2. V lednu 2001 byla vyvinuta **verze 1.1** s podporou OAI 1.0. Verze systému EPrints 1.x měly za úkol sloužit spíše jako testovací, aby se včas odhalily požadavky a problémy – vše s cílem uplatnit nabyté znalosti ve verzi 2. Vývoj systému **EPrints 2** začíná v září 2001 a pokračuje až do současnosti (číslo 2 za názvem systému je používáno pro snadné odlišení obou řad jednotlivých verzí – označení EPrints 2 přísluší všem verzím 2.x.x). Od verze 2.0 má každá oficiální 2.x verze vlastní jméno – jména jsou odvozena od názvů jídel. První oficiální verze se objevila v únoru 2002 a nesla název *Olive* (předcházeli jí dvě testovací verze *Anchovy* a *Pepperoni*). Mezi touto verzí a následujícími verzemi v červnu 2002 se systém EPrints zařadil do rodiny programů distribuovaných pod hlavičkou organizace GNU. Postupně následovaly verze *Tuna*, *Pineapple* (implementovala OAI 2.0) a *Pumpkin*. Verze Pumpkin vyšla na konci září 2002. V lednu 2004 je zveřejněna první verze řady 2.3.x (tato řada již nenese žádné speciální označení) a vývoj probíhá do současnosti.

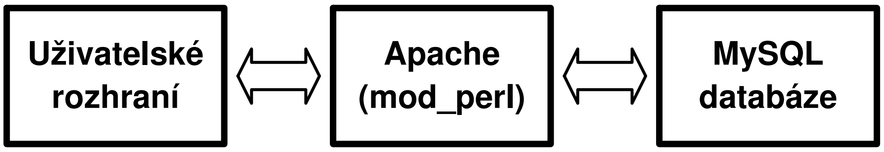
5.2 Základní vlastnosti

Je požadován server Apache s dynamicky přilinkovanou knihovnou *mod_perl*. Uživatelské rozhraní je tedy webové. Dále je požadován server SMTP (kvůli registraci nových uživatelů). Celý systém je postaven na programovacím jazyce Perl. Generování stránek je částečně prováděno staticky (kvůli zvýšení rychlosti). Je proto nutné jednou za čas spouštět skripty pro přegenerování stránek (podrobněji bude diskutováno v části 5.6). Jako databázový server je použit MySQL. EPrints mohou být provozovány na jakémkoli GNU operačním systému – je však doporučována některá distribuce operačního systému deklarující se jako GNU/Linux (ale je například možné volit i MacOS).

EPrints jsou repozitářem zaměřeným na vědecké publikace v akademické sféře, tomu odpovídají metadata i procesy. Vše je samozřejmě konfigurovatelné a nic nebrání s jistým úsilím vytvořit obecný repozitář. Je zde podpora archivů – jsou to samostatné systémy s vlastní konfigurací, samostatné digitální knihovny běžící na jádře EPrints. Interoperabilitu zajišťuje podpora protokolu **OAI-PMH**. Pro metadata používá systém EPrints vlastní vnitřní formát.

5.3 Architektura

Systém EPrints používá (ve srovnání s ostatními testovanými systémy) jednoduchou architekturu klasického www serveru. Nejvyšší vrstvou je uživatelské rozhraní, jehož požadavky obsluhuje server Apache spouštějící příslušné perlové skripty, které vytvářejí procesy, přistupující k databázi MySQL. Níže je uveden zjednodušený diagram této architektury. Kompletní diagram (včetně relačních vztahů) je k dispozici na www stránkách projektu [46].



Obrázek 5.2: Základní části architektury systému EPrints.

5.3.1 Archivy

EPrints server umožňuje provozovat a obsluhovat několik samostatných archivů (obvykle jako virtuální servery Apache – tedy na zvláštních portech nebo www adresách). Každý z archivů má vlastní uživatele, vlastní konfiguraci a vlastní data (jsou nezávislé - proto se musí i zvlášť spravovat). Výhodou je, že lze každý archiv upravit pro různé typy digitálních objektů.

5.3.2 Struktura

Strom zanoření dokumentů, v EPrints označovaný jako Subject, je ve výchozím nastavení velmi bohatý. Systém EPrints předpokládá vkládání objektů textového typu a podle toho nabízí členění podle LoC. Kořen stromu je označován jako oblast (Area), tedy Library of Congress Subject Areas – jedná se o strom, který identifikuje položku dle obsahu (například technická práce z oblasti medicíny). EPrints nabízí administrátorům editační nástroj (Subject editing tool), který je přímo v uživatelském rozhraní a ve kterém lze vytvářet vlastní oblasti s libovolným zanořením. Tento strom pak používají koncoví uživatelé pro snadné a přehledné procházení archivu nebo při vyhledávání (mohou zadat, ve které větvi se bude vyhledávat).

5.3.3 Statické generování stránek

EPrints je zaměřen na oblast STI, takže se předpokládá, že vkládání článků nebude příliš časté (myšleno ve stovkách denně). Proto byl z důvodů rychlosti přístupu zvolen přístup přes generování statických stránek. Očekává se, že většina operací na serveru bude spojena se čtením, hledáním, listováním atp. a zápis a editace budou tvořit jen malou část procesorového času. Neustálé dynamické generování by pak znamenalo zbytečnou zátěž. Proto pouze část systému běží dynamicky (práce pod uživatelským profilem, administrace) a zbytek je generován staticky. Tj. uživatel vloží položku do fronty pro schvalování, tato položka je následně schválena a vložena do archivu. Nicméně je vložena pouze do databáze a uživatelé, kteří bezprostředně v archivu hledají ji nevidí a nemají k ní přístup. Jednou za čas je potřeba spustit sadu skriptů, které statické stránky přegenerují podle aktuálního stavu databáze (záleží na posouzení administrátora, jak často a kdy se budou tyto operace provádět – jak čerstvé mají být údaje prezentované uživatelům).

5.4 E-prints verze 2.2

5.4.1 Instalace

Instalace je poměrně dobře popsána. Nutností je doinstalovat poměrně velké množství balíků jazyka Perl (automatický skript dodávaný v distribuci s EPrints nefunguje zcela korektně, záleží velmi na aktuální konfiguraci operačního systému a příslušných programátorských knihoven). Po nainstalování všech potřebných balíčků lze přikročit k instalaci samotného systému EPrints. Instalace je snadná, systém má solidního průvodce. Bohužel je instalace silně nedotažená, takže je potřeba dělat netriviální zásahy do konfigurace webového serveru. Kvůli registraci uživatelů je nutné mít také v provozu SMTP server.

5.4.2 Rozhraní

Rozhraní je poměrně jednoduché (evidentně čistě HTML), nicméně velmi přehledné a intuitivní. Úvodní obrazovka nabízí přístup do různých částí systému a prakticky ukazuje, co vše systém umí. Zejména jsou to pole pro vyhledávání, dále odkazy na jednotlivé části systému, pro anonymní přístup je to procházení položek, složitější hledání, výpis nejnovějších přidaných položek, dále odkazy na vstup do uživatelské oblasti a registrace nového uživatele.

Po přihlášení může uživatel pracovat se svými položkami (DO v EPrints je označován také jako *Item*, budu proto dále používat i odpovídajícího českého výrazu *položka*). EPrints je dělán zejména pro ukládání textových informací, a proto obsahuje šablony pro vložení knihy, kapitoly z knihy, článku, monografie atp. Podle toho jsou nabízena příslušná metadata k vyplnění (byl publikován, v jakém časopisu, jaký nakladatel, ISSN, DOI, . . .). Vkládat lze potom data typu PDF, HTML, PS, ASCII, obrázek (zde spíše titulní strana časopisu, kde byl článek publikován), popř. data nespecifikovaného typu. Vkládání probíhá v několika krocích, kterýkoli krok může být přerušen a je možné se k němu později vrátit a pokračovat. Pomocí menu lze omezit skupinu uživatelů, kteří budou smět článek stahovat (všichni uživatelé, jen registrovaní, ...). Nahrávat data lze přímo z lokálního disku nebo vložením URL s článkem. Jedna položka pak může obsahovat více souborů (ty mohou být nahrávány i komprimované ve formátech .zip a .tar.gz). Bohužel je NUTNÉ (ve výchozí konfiguraci), aby položka obsahovala minimálně jeden soubor ve formátu PDF, PS, HTML nebo ASCII. Což znamená, že nelze mít položku skládající se např. pouze z obrázků.

Workflow proces

Pojmem workflow proces rozumíme proces vkládání digitálního objektu v případě, že digitální objekt není vložen v jednom kroku, ale musí projít několika fázemi. Obvykle má ke každé fázi přístup jiný uživatel (nebo skupina uživatelů), kteří po prohlédnutí a kontrole objektu buď objekt schválí a pošlou do další fáze nebo zamítnou (návrat do předchozí fáze či jiná akce). V systému EPrints je workflow proces kvalitně implementován. Spočívá ve vložení objektu v několika krocích. Vkládání lze v jakémkoli kroku přerušit a později se do něj vrátit. Obyčejný uživatel, který se zaregistroval běžným způsobem, pak provede uložení (Depozit) položky, tj. uloží ji do složky dokumentů, čekajících na schválení. Až do schválení položky uživatelem s příslušnými právy není položka definitivně uložena v archivu (ve stavu Accepted). Všechny položky v jakémkoli stavu může uživatel samozřejmě editovat (nicméně u již akceptovaných položek musí po změně projít položka opět celou schvalovací procedurou).

Zasílání novinek emailem

Zaregistrovaný uživatel má možnost nechat si zasílat emailem informace o nově přidaných objektech, změnit svůj záznam, prohledávat archiv atp. Samozřejmě může sledovat, v jakém stavu schvalování ve workflow procesu jsou jím vložené objekty (Pending items).

Privilegovaní uživatelé

Administrátor a editor mají k dispozici další menu. Zejména mají možnosti prohlížet a schvalovat položky (Pending items), editovat uživatele, editovat a mazat položky (mohou je hledat už v archivu a mezi položkami, které uživatelé editují, ale ještě je ani nezařadili do procesu schvalování).

Uživatele lze přidávat v administrátorském menu, speciálními skripty a uživatel se sám může přidat do systému tím, že se zaregistruje. Následně je mu poslán email, ve kterém je adresa, na kterou se uživatel musí přihlásit, aby byl skutečně přidán do systému. Uživatelovu registraci do systému může potvrdit i administrátor. Uživatelé systému jsou tří typů. *Obyčejný uživatel* – přidá ho administrátor nebo se sám zaregistruje. *Editor* – má práva schvalovat nové položky a editovat (ne přidávat) uživatele. A *administrátor* – může provádět veškeré akce. Privilegovaný uživatel má také možnost zjišťovat statistické informace o archivu.

Indexace a hledání

Vyhledávání v systému EPrints je dobře propracováno. Uživatelé mají možnost klást poměrně bohaté dotazy. Kromě klasického vyhledávání v metadatech autoři umožňují indexovat i obsah dokumentů. Nutností je nainstalovat si externí programy, které umí různé formáty dokumentů převádět na textové soubory, v nichž pak systém EPrints vyhledává. Podporovány jsou dokumenty PDF (pomocí nástrojů z balíku programů xpdf), Microsoft DOC (wvware) a HTML (lynx). Podrobně je nastavení popsáno v dokumentaci k EPrints.

5.4.3 Konfigurace systému

Vše je velmi dobře, kvalitně a téměř úplně popsáno v dokumentaci na webu systému EPrints. Výborný je podrobný popis konfiguračních souborů, včetně slušně zpracovaných návodů „jak udělat . . . “ (How-to . . .) – jak přidat nový typ metadat, jak změnit vzhled, . . . Tentýž archiv lze provozovat v několika jazykových mutacích (staticky se generují). Tedy stránky jsou klientovi nabídnuty v jazyce, na který má nakonfigurovaný prohlížeč (vyjednávání o obsahu umožňuje protokol HTTP/1.1).

Možnosti konfigurace jsou velmi rozsáhlé a není problémem postavit si systém ke svému obrazu. Například lze odstranit výše zmiňované omezení na to, že v každé položce (digitálním objektu) musí být některý typ souboru – lze povolit všechny typy nebo definovat vlastní množinu povolených typů (obrázky, audio). Dále lze třeba vypnout schvalování položek, ty pak jsou vkládány přímo do systému bez souhlasu editora, lze upravit vzhled i systém přidávání položek (nastavit výchozí hodnoty metadatových záznamů), změnit typy metadat pro vyhledávání, nastavit různá práva pro třídy uživatelů (editor, administrátor) a přidat nové třídy, změnit vzhled stránky s vyhledaným objektem (například aby se zobrazil náhled obrázku).

Možnosti konfigurace jsou bohaté, systém lze upravit podle vlastních požadavků.

5.5 Dokumentace

Dokumentace je velmi bohatá. Velmi podrobné návody, tutoriály, odstraňování problémů, odkazy na jiné stránky s podobnou problematikou – to vše hodnotím velmi kladně. Online nápověda je obsažena také přímo v uživatelském rozhraní. Bohužel v některých částech webu systému EPrints jsou bílá místa a celá nápověda tak působí nedodělaně – například k některým důležitým skriptům dokumentace zcela chybí. Jedná se však pouze o malé části dokumentace, zbytek je výborný.

5.6 Diskuse ke statickému generování stránek

Statické generování obsahu www stránek je často používaným postupem. Je však potřeba zvážit, zda-li je vhodné i pro systém EPrints. Definované generování statického obsahu se spouští na základě toho, jak určí správce. Samotní autoři varují, že pokud jsou v archivu řádově tisícovky záznamů, pak může celý proces aktualizace statických stránek trvat i více než hodinu (v závislosti na použitém hardware). Po tuto dobu je systém velmi zpomalen. Řešením může být použití snížení priority běhu příslušného skriptu – server je však stále zatížen. Spouští-li se toto generování ve vhodnou dobu (v nočních hodinách) a používají-li server převážně uživatelé z oblasti, ve které je fyzicky umístěn, pak může být tento přístup výhodný.

Pokud má přibývat jen několik dokumentů denně, je statické generování stránek velmi výhodné (samotné procházení archivu je velmi rychlé, prakticky už rychlejší být nemůže, protože webový server pouze poskytuje statické stránky). Uvažujeme-li však o stovkách dokumentů (digitálních objektů) denně, pak se mi jeví lepší nasazení některého jiného systému. Stačí si představit vkládání a následné editace i několika stovek fotografií denně – může být žádoucí zkontrolovat si ihned konečný výsledek po vložení a na jeho základě fotografie editovat, pak následně opět vložit . . . V takovém případě se jeví jako vhodnější dynamické zobrazování obsahu (i vzhledem k tomu, že současná technika je dostatečně výkonná, aby podobnou zátěž zvládala).

Přístup se dá shrnout do dvou základních otázek. Jestli předpokládáme řádově více operací realizujících čtení a procházení obsahu, nebo více operací zápisu. A jakou rychlostí budou nové dokumenty přibývat a jak rychle je chceme dát k dispozici uživatelům. Budeme-li spouštět generování příliš často (navíc se nesmí spustit nové generování dříve, než skončí původní, aby nedocházelo ke konfliktům), systém bude touto činností neúměrně vytížen, v opačném případě nebude okamžitě aktuální. Jsem toho názoru, že v případě účelu, k jakému je systém EPrints vytvořen, je architektura vhodná. Dokumenty procházejí vždy schvalovacím procesem (někdo si je musí přečíst, provést hlubší studii – to trvá nezanedbatelnou dobu), který vylučuje příliš masivní přidávání. Pro tyto účely je výkonově optimalizován. Měl-li by však sloužit i pro jiné typy objektů, u kterých lze předpokládat vyšší frekvenci vkládání a s tím související větší zatížení, pak není statické generování vhodné.

5.7 Závěr

Na systému EPrints jsem v aktuální testované verzi objevil několik drobných chyb. Je jich však minimum a nemají vliv na kvalitní běh systému. Zmíním pouze některé: neúplná dokumentace, lze pouze změnit uživatele, což ale nefunguje, pokud je uživatel přihlášený jako editor. Za nevýhodu lze považovat téměř nemožnost používat znaky s diakritikou – nelze podle nich vyhledávat a systém EPrints s nimi má velké problémy (při výpisu dokumentů s diakritikou „zatuhne“ prohlížeč). Dá se předpokládat, že do budoucna by tyto nedostatky mohly být odstraněny.

Celkový dojem ze systému je však velmi kladný. Vzhled stránek je sice velmi jednoduchý, ale na druhou stranu opravdu maximálně přehledný. Prakticky okamžitě se koncový uživatel orientuje, nemá problém cokoli najít a samotné ovládání je velmi intuitivní. Systém EPrints dělá pouze „jednu“ věc, ale dělá ji dobře a přehledně. Není zde nic navíc, ale pouze a právě to, co je potřeba. Systém EPrints jednoznačně dokazuje, že v jednoduchosti je síla.

Rychlost odezvy systému EPrints je nejlepší ze všech testovaných systémů. Je to dáno zejména tím, že server pouze poskytuje statické stránky a nevykonává žádný kód a architektura je nejjednodušší možná. Systém EPrints používá skutečně jen základní HTML, práce je tak poměrně přehledná a pohodlná i v textový webových prohlížečích. Za rychlost však bohužel platíme neaktuálností.

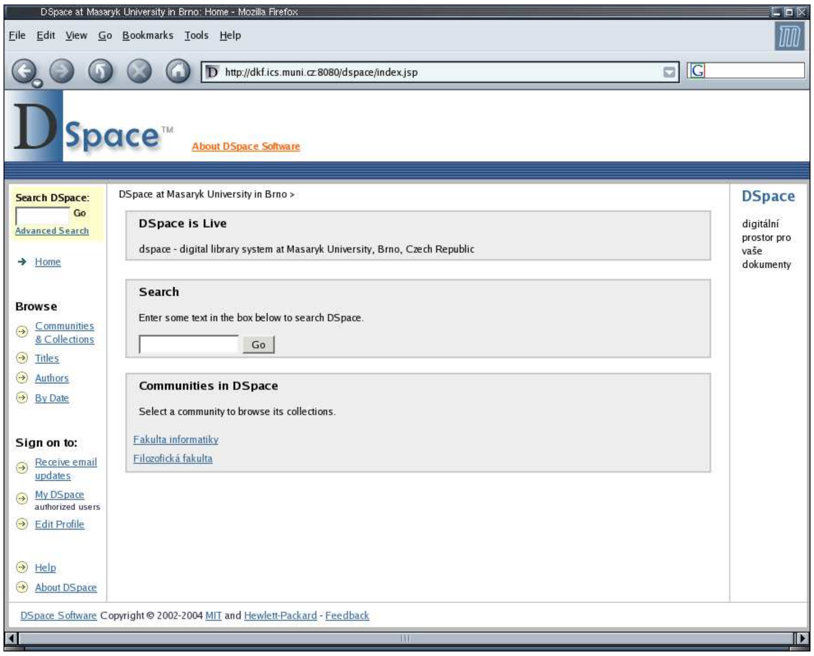
Konfigurovatelnost a možnost přizpůsobení systému je taktéž výborná (z hlediska možností). Snadnost této konfigurovatelnosti je již horší (je nutné editovat i některé zdrojové kódy v jazyce Perl). Velkým plus je i několik přidaných how-to průvodců, které usnadňují konfiguraci.

Z pohledu provozu univerzálního repozitáře, do kterého jsou vkládány různé typy dokumentů v rychlém časovém sledu a velkém objemu, se jeví systém EPrints jako systém nevyhovující.

Kapitola 6

DSpace

DSpace (Digital Archive Project) [47] je systém poskytující zázemí pro provozování digitálních knihoven (akademické) instituce a s tím spojenou základní funkcionalitu. Staví na Kahn/Wilenského architektuře. Je poskytován jako otevřený systém pod licencí BSD [48], může být volně vylepšován a rozšiřován. Je to skutečně kompletní systém, který má vše potřebné přímo v sobě – uživatelské rozhraní včetně uživatelských účtů, řešení workflow procesu atd.



Obrázek 6.1: Úvodní stránka systému DSpace.

Původním cílem projektu DSpace bylo vyvinout spolehlivý otevřený univerzální digitální knihovní systém, který bude nasazen na MIT (Massachusetts Institute of Technology). Systém však má být použitelný pro široké spektrum uživatelů, především z akademické sféry. Systém sleduje všechny nejnovější trendy, poznatky a standardy, a všech se snaží hojně používat. V současnosti se jedná o jeden z nejkomplexnějších, nejživějších, funkčních a nejrychleji se rozvíjejících projektů v oblasti volně dostupných systémů digitálních knihoven.

6.1 Historie

První oficiální zpráva [49] o systému DSpace se objevuje přibližně na přelomu dubna a května roku 2000. Firma HP a MIT Libraries oznámily informace o společném projektu spolehlivého digitálního repozitáře pro MIT, který by však zároveň sloužil jako vzor pro jiné instituce.

Na podporu projektu věnovala firma Hewlet-Packard 1,8 milionu USD, které měly vystačit na pokrytí nákladů a dvouletou práci na projektu. Samotný MIT však už v té době hledal prostředky na dlouhodobější vývoj a sliboval skutečně kvalitní digitální repozitář, postavený na ověřených a již existujících technologiích. V té době ještě nebyla vyřešena technická specifikace (jaká platforma, autorská práva). Byl proveden průzkum mezi fakultami a výzkumnými centry na MIT a myšlenka systému DSpace byla vesměs kladně přijata.

V březnu 2002 se objevuje zpráva [50] shrnující dosažené výsledky, kterých se dosáhlo za pomoci „včasných osvojitelů" – subjektů spolupracujících na beta testování DSpace – mimo jiné při ukládání textových dat, dat produkovaných geografickými systémy a digitalizovaných knih MIT Pressu.

V listopadu 2002 je na serveru SourceForge [51] zveřejněna první verze **DSpace 1.0**. Ke zkoušení a provozování systému se přidávají další univerzity, vzniká tak federace uživatelů DSpace – **DSpace Federation**. Další klíčovou verzí je **1.1.1**, která se objevuje v září 2003, přináší řadu nových vylepšení, obsahuje však ještě několik kritických chyb a nedodělaných míst. O rok později v srpnu 2004 je vystavena další klíčová verze 1.2. Tato verze odstraňuje „dětské nemoci" předchozí verze a přidává několik velmi cenných funkcí. Do budoucna autoři slibují stálou podporu a další vývoj systému ve spolupráci s otevřenou komunitou vývojářů, která se kolem systému DSpace tvoří.

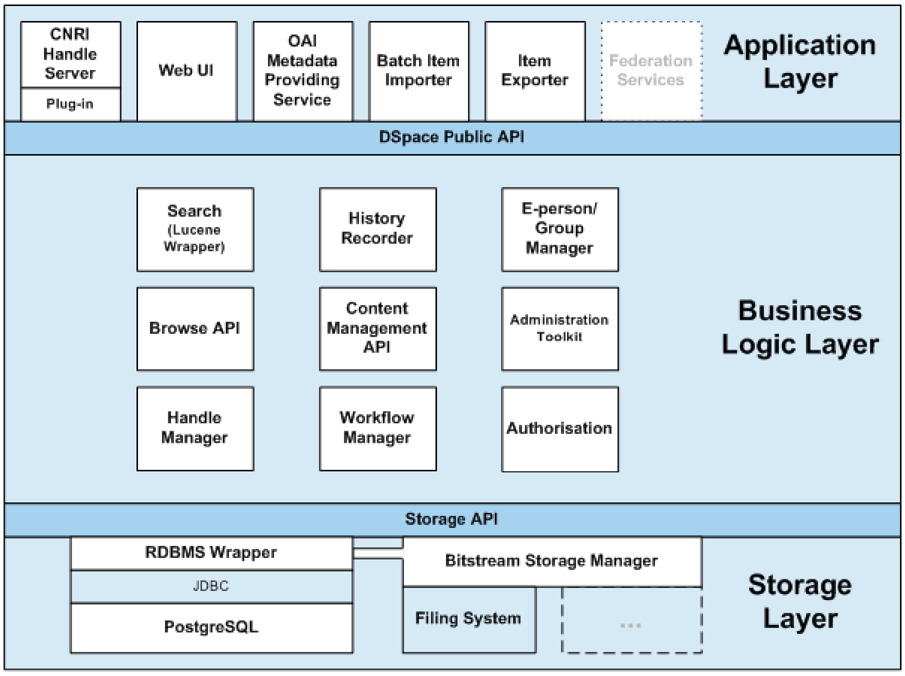
6.2 Základní vlastnosti

DSpace je vyvinut pro provoz na systémech UN*X (verze pro Windows není zatím k dispozici), veškeré použité technologie, knihovny a balíky jsou open source software a jsou volně k dispozici. Jako programovací jazyk je zvolena Java. Databázový server je PostgreSQL, jako webový server a Java servlet engine je použit Apache Tomcat. Nutné jsou také některé javovské knihovny, dostupné na stránkách firmy Sun Microsystems. Protože je DSpace open source, předpokládá se, že instalaci a provoz si zajistí provozovatel vlastními prostředky.

DSpace používá v maximální míře současných nejrozšířenějších volných standardů. Pro zaznamenávání metadat používá **Dublin Core**. Interoperabilita je zajištěna přes **OAI-PMH**, perzistence objektů a jejich jednoznačné identifikátory jsou pak řešeny pomocí **CNRI Handles**. Identifikátory „handles" (ukazatele) na objekty jsou vyjádřeny pomocí URN. DSpace podporuje i standard **OpenURL**.

6.3 Architektura

6.3.1 Systémová architektura



Obrázek 6.2: Architektura systému DSpace (převzato z dokumentace).

Architektura DSpace je postavena na třívrstvém modelu s podrobně zdokumentovanými voláními mezi jednotlivými vrstvami. Nejspodnější vrstvou (**Storage Layer**) je vrstva pro práci s databázovým serverem PostgreSQL (fyzické ukládání dat). Použito je rozhraní JDBC (Java Database Connectivity, databázové rozhraní pro jazyk Java) dodávaný spolu s PostgreSQL, nad ním je postaven RDBMS Wrapper, poskytující pro programátora API pro přístup k databázi (obaluje JDBC ovladač). Dalším modulem definujícím nízkoúrovňové volání je Bitstream Storage Manager pro přístup k fyzickým souborům (tzv. Bitstreams, viz dále), které DSpace ukládá sám na disk (mimo PostgreSQL). Výše zmíněná volání (API) tvoří dohromady rozhraní **Storage API** tvořící bránu do vyšší vrstvy.

Prostřední vrstvou je **Business Logic** vrstva – obsahuje moduly pro řízení práv, administraci, autorizaci, hledání, prohlížení uložených dat a další. Každý z těchto modulů má vlastní rozhraní a definované API, které je podrobně zdokumentováno a společně tvoří Public API, bránu do nejvyšší vrstvy. Zvláštním modulem je Search modul, který není přímo dílem vývojářů DSpace, ale je použit Jakarta Lucene search engine [52], nad kterým je naprogramováno volání pro realizaci API (tzv. wrapper).

Nejvyšší vrstvou je pak vrstva aplikační (**Application Layer**), kde je definováno uživatelské rozhraní, podpora pro servery OAI a CRNI a exportní nástroje pro METS.

Veškerá volání API jsou přehledně rozdělena do zdokumentovaných tříd a popsána pomocí JavaDoc (jak je při použití jazyka Java obvyklé).

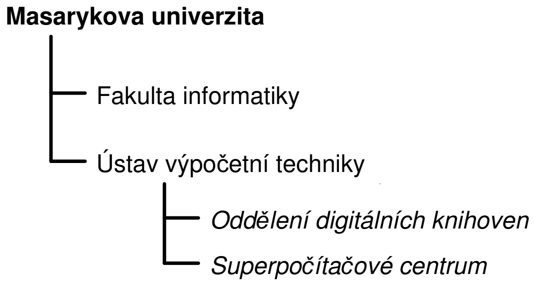
6.3.2 Digitální objekty a interoperabilita

Digitální objekt je v DSpace označován jako *položka* (Item) – jedná se o objekt ve smyslu Kahn/Wilenského architektury zapouzdřující metadata i fyzické soubory. Jako metadatový formát je volen Dublin Core. Povinnými položkami jsou pouze titul, jazyk a datum vložení/zaslání. Fyzické soubory v položce se nazývají Bitstreams (dále bitstreamy), rozlišují se dle MIME typu.

Interoperabilita je zajištěna přes OAI-PMH server, použita je implementace OCLC OAICat [53] (OCLC – Online Computer Library Center), která je naprogramována jako servlet a realizuje OAI-PMH protokol verze 2.0. Pro využití podpory OpenURL je nutné mít vlastní SFX server.

6.3.3 Struktura

Hlavní organizační jednotkou v systému DSpace je jednotka *komunita*. Reprezentuje abstraktní či konkrétní subjekt – například výzkumný ústav instituce, fakultu univerzity, apod. Komunity mohou být hierarchické a libovolně zanořené.



Obrázek 6.3: Příklad struktury zanoření v systému DSpace.

Příklad z obrázku 6.3 ukazuje model, kdy jeden systém DSpace může obsluhovat i několik univerzit (nejvyšší položkou v hierarchii je univerzita). Nejpravděpodobnější scénář nasazení systému DSpace pro potřeby jedné školy může vypadat tak, že nejvyšší komunity jsou jednotlivé fakulty univerzity, tyto fakulty si pak vytvářejí svoje subkomunity (např. katedry a laboratoře) atd.

V kterékoli komunitě lze vytvářet *kolekce*, ve kterých jsou ukládány jednotlivé *položky* (Items). Můžeme mít jak kolekce týkající se celé univerzity přímo v komunitě univerzita, tak kolekce příslušející jednotlivým fakultám a podobně. Kolekce již nelze do sebe zanořovat.

6.4 DSpace 1.2

6.4.1 Instalace

DSpace vyžaduje několik programů a knihoven (zejména pro správný běh v Java prostředí), které jsou všechny dostupné volně na Internetu. Předpokládá se znalost unixových systémů, aby byl uživatel schopen si veškeré knihovny, programy a servery nainstalovat a rozeběhnout. Veškeré informace navíc se týkají pouze problémů specifických pro DSpace. Instalace provádí do adresáře `/dspace`, pokud chcete provést instalaci jinam, přichází na řadu velmi nepohodlné přepisování cest v různých konfiguračních souborech. Podpora distribuce a podpory pro instalaci je velmi zdařilá, kompaktní a dobře dokumentovaná – mimo jiné řeší nejčastější chyby a problémy, které mohou nastat. Zkušený uživatel si s instalací poradí poměrně rychle, většinu času zaberou konfigurace různých částí systému.

6.4.2 Webové rozhraní

Hotové webové rozhraní systému DSpace má velice pěkný vzhled, je uděláno přehledně a maximálně funkčně. Uživatel, který přistupuje k systému DSpace, vidí vlevo menu s nejdůležitějšími položkami – hledání a procházení obsahu DSpace, dále odkazy na MyDSpace (uživatelova osobní stránka v DSpace), registrace nového uživatele a nápověda. Dále je na úvodní stránce (viz obrázek 6.1) seznam komunit (které jsou na vrcholu hierarchie) a informace o samotném DSpace (kde je provozován apod.). Registrace nového (neprivilegovaného) uživatele probíhá vyplněním emailu, na který je zaslán vygenerovaný náhodný řetězec pro potvrzení registrace, který uživatel zadá do DSpace. Po registraci má uživatel některé výhody proti anonymnímu brouzdání v DSpace. Uživatel si může nechat posílat emaily s informacemi o nově přidaných objektech, mohou mu být přidělena administrátorem větší práva, může se účastnit workflow procesu přidávání nových položek.

Vyhledávání

Hledání v systému DSpace je implementováno jako jednoduché nebo i složitější. Složitější vyhledávání je řešeno jako možnost zadat výrazy spojené logickými spojkami AND, OR nebo NOT. Tolik, co nabízí přímo webové rozhraní. Jako samotný výraz pro hledání však může uživatel použít celou sadu pomocných výrazů – dotaz se konstruuje podobně jako regulární výrazy a veškeré možnosti včetně příkladů jsou obsaženy v online nápovědě. Díky použití vyhledávacího stroje Jakarta Lucene jsou možnosti skutečně bohaté. Prohledávají se jak metadata, tak obsah některých typů dokumentů (viz část 6.4.4).

Administrace

Administrátor systému DSpace má k dispozici navíc další možnosti a nástroje (Administration Tools). Boční menu je v tomto případě odlišné a obsahuje odkazy na užitečné nástroje – na procházení a hledání v komunitách a kolekcích, editace uživatelů a skupin uživatelů, práce s konkrétními položkami, editace metadatových položek DC, editace typů bitstreamů (MIME typů), sledování workflow procesu (rušení jednotlivých workflow procesů), editace a přidělování přístupových práv a editace novinek (News) zobrazujících se na úvodní straně DSpace. Administrátor má také možnost vytvářet k jednotlivým kolekcím šablony položek – může tak vyplnit některé metadatové položky výchozími hodnotami.

Kontextová menu

Kontextová menu na pravé straně obrazovky se objevují registrovaným uživatelům a závisí na úrovni oprávnění, které má uživatel přiděleny. Objevují se v nich položky typu edituj práva, vytvoř kolekci, vytvoř komunitu apod. Přes tyto položky se může administrátor mimo jiné dostat do administrátorského menu.

Přístupová práva

Všichni uživatelé systému DSpace jsou děleni na *skupiny*. Výchozí skupiny má systém dvě a jsou to *Administrators* (skupina administrátorů) a *Anonymous* (všichni registrovaní uživatelé DSpace). Platí, že administrátoři mohou všechno. Administrátor může vytvářet další skupiny, přidávat do nich uživatele. Každá skupina může mít ke každému objektu (položce, kolekci, bitstreamu) různá *přístupová práva*. Tyto typy práv záleží na tom, ke kterému objektu (ve smyslu komunita, kolekce, položka, bitstream) přísluší. Práva nelze přidělovat jednotlivým uživatelům, ale pouze skupinám uživatelů, není ovšem vyloučeno, že skupina bude mít právě jednoho uživatele.

Administrátorovi se při přidělování práv zobrazují vždy pouze ta práva, která mají u daného objektu smysl. DSpace má k dispozici tyto základní typy práv: READ, WRITE, ADD a REMOVE. Význam je závislý na typu objektu, kterému se práva přidělují – např. pro kolekci ADD znamená, že skupina uživatelů může přidávat (Submit) nové položky atp. Speciálními typy práv jsou COLLECTION_ADMIN, DEFAULT_ITEM_READ a DEAFAULT_BITSTREAM_READ. Skupina uživatelů, která dostane v příslušné kolekci právo COLLECTION_ADMIN se stává administrátory kolekce. Zbylé dva speciální typy práv zajišťují přiřazování práv položkám a bitstreamům vkládaným do kolekce. Všechny skupiny, které mají u kolekce tato práva, budou mít práva pro čtení u všech položek a bitstreamů vkládaných do kolekce. Podrobně je vše popsáno v online dokumentaci přímo v systému DSpace.

Systém práv bohužel nefunguje zcela přesně jak je popsáno v dokumentaci. Ačkoli autoři deklarují řadu možností, v současné verzi systému DSpace nejsou některé z nich zcela funkční. Nefunguje přidělování práv uživatelům pro komunity a editaci složek – pouze od složek níže je již systém práv funkční. To má za následek, že struktury komunit může vytvářet pouze administrátor. Nedostatky se týkají také práv u bitstreamů (jako obyčejný uživatel nemohu měnit některá metadata u již vložených bitstreamů).

Celkově lze systém práv shrnout následovně: uživatel, není-li administrátor, může mít delegována práva pouze pro práci na úrovni kolekcí a níže. Tvořit a editovat struktury komunit a kolekcí smí pouze administrátor.

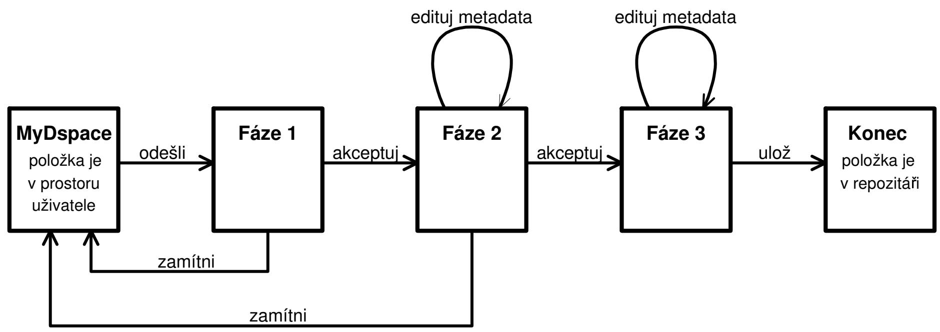
Workflow proces

Při vytváření nové kolekce se určuje, kteří uživatelé se budou účastnit workflow procesu a v jaké fázi, případně které fáze procesu se budou dané položky týkat (například nemusíme mít fáze žádné a po vytvoření bude položka okamžitě vložena do DSpace a přístupná uživatelům). Ve workflow procesu jsou možné 3 fáze, ke každé fázi lze definovat skupinu uživatelů, která má právo do dané fáze procesu zasáhnout a schválit vkládanou položku do další fáze workflow. V momentě, kdy přechází položka do nové fáze workflow, je uživatelům, kterých se to týká, zaslán email, že je třeba zkontrolovat/schválit položku. Kterýkoli uživatel ze skupiny, která je definovaná pro danou fázi workflow pak může (podle fáze) provádět s položkou dané akce a případně položku schválit a postoupit ji tak k dalšímu zpracování. Tyto tři možné fáze lze kombinovat (vynecháním některé z nich, nikoli, co se pořadí týká). Popišme si je nyní jednotlivě:

fáze 1. – pouze schvalování je-li položka způsobilá pro další fáze (pokud nejsou definovány, pak pro vložení do veřejné části DSpace);

fáze 2. – schvalování položky + editace metadat;

fáze 3. – editace metadat (předpokládá se, že položka je již schválena ke vložení do DSpace, pouze je nutné opravit metadatový záznam, akceptování v této fázi znamená vždy vložení do veřejné části DSpace).



Obrázek 6.4: Worflow proces v systému DSpace.

DSpace sám při vytváření kolekce umožňuje zvolit uživatele, kteří budou smět spravovat kolekci, automaticky vytvoří skupinu a přidělí jí příslušná práva. Totéž automaticky provede pro uživatele, kteří budou mít právo přidávat položky. Stejně tak vytvoří skupiny pro danou kolekci a příslušnou fázi workflow procesu. Tyto skupiny lze později kdykoli editovat měnit a mazat. Nárůst takto vytvořených skupin je značný (u každé kolekce cca 3-4 nové skupiny), přesto se díky jednotnému pojmenovávání dá mezi nimi dobře orientovat. Každý objekt má však při editaci speciální menu pro každou takto vytvořenou skupinu, proto není nutné používat přímo administrátorský nástroj pro nastavování práv.

6.4.3 Mapování položek

Mapování položek do jiných kolekcí se provádí pomocí nástroje **Item Mapper**. Pomocí něj lze namapovat do aktuální kolekce objekty z jiné kolekce. Fyzicky je objekt tentýž, logicky je pak ve více kolekcích. Toto mapování je velmi rychlé a snadné, bohužel lze je dělat pouze způsobem „jsem v kolekci, chci přilinkovat položku“ (navíc položku lze najít pouze tak, že vložím jméno autora a pak se mi vypíší všechny jeho položky, pouze ty mohu namapovat do kolekce). Způsob „tuto položku namapuj do kolekce xy“ není zatím možný.

6.4.4 Indexace, skripty

Kromě možnosti práce přes webové rozhraní má administrátor k dispozici několik skriptů. Jedním z nejdůležitějších je skript **dsrun**, pomocí něhož lze spouštět javové aplikace související s DSpace – například pro export položek, administrační úkony a další třídy definované tvůrci DSpace.

Důležitými skripty jsou skripty spouštějící indexaci obsahu DSpace a generování náhledů k obrázkům v DSpace. Indexace obsahu DSpace se provádí v dokumentech typu Adobe PDF, Microsoft DOC, HTML a TXT. Při zadávání dotazů pro vyhledávání v DSpace se následně hledá i uvnitř těchto typů dokumentů a nikoli jen v metadatech položek. Generování náhledu obrázku se provede tehdy, je-li bitstreamem dokument s MIME typem *image*. V konfiguračním souboru lze definovat velikost generovaných náhledů. Náhledy se pak vždy zobrazují jako ikonka vedle příslušného bitstreamu při prohlížení položky.

6.4.5 Identifikátory

DSpace má v sobě zabudovanou podporu CNRI Handles. Handle server pro resoluci identifikátorů je dodáván společně se systémem DSpace. Pro jeho provozování je nutné si zaregistrovat prefix u organizace CNRI. Handle server není dílem tvůrců DSpace a běží jako samostatná služba (komunikuje přes vlastní protokol).

6.5 Tapir

Systém DSpace je *univerzálním* digitálním repozitářem. Řešitelé projektu **Tapir** (Theses Alive Plug-in for Institutional Repositories) [54] ukázali, že vhodným programátorským zásahem do systému (vytvořením pluginu) lze tento univerzální repozitář specializovat pro použití ve zvolené úzké doméně – v tomto případě v oblasti správy vysokoškolských kvalifikačních prací.

Plugin je vyvíjen v rámci projektu **Theses Alive!** [55] pod záštitou JISC (Joint Information Systems Committee) [56]. Vývojáři nespolupracují přímo s vývojáři DSpace, ale Tapir dodávají jako samostatný balík, který je nutné v několika krocích (sestávajících z editace původních souborů DSpace, přepisu některých servletů a znovu sestavení) nainstalovat přes originální DSpace. Popišme si systém Tapir podrobněji.

6.5.1 Metadata

První změna v pluginu Tapir oproti standardnímu DSpace spočívá v přidávání nových položek. Před začátkem vkládání je uživatel dotázán na typ položky (buď Thesis/Dissertation nebo Other Research Material) a tím se určí typ vyplňovaných metadat. Zvolíme-li Thesis/Dissertationi, bude vložený objekt metadaty relevantní standardu JISC Fair Programme. Pokud zvolíme Other Research Material pak budou metadata relevantní projektu SHERPA (Securing a Hybrid Environment for Research Preservation and Access) [57]. U obou typů přibyla, proti standardnímu DSpace, metadata týkající se dohlížitele/vedoucího práce (Supervisor), jedná-li se o práci v doktorandském nebo magisterském studiu atd. (podrobně jsou metadata rozebrána v dokumentaci). Metadata jsou specializována na míru školám ve Velké Británii – stupeň dosaženého vzdělání a specializace, ke kterým se práce vztahují jsou pro české uživatele použitelné jen částečně (patrně s pomocí patřičných dohod ohledně sémantiky použitých informací). Další změna nastává v momentě přidělování licencí, kdy uživatel má možnost zvolit, v jakém časovém horizontu má být jeho práce přístupná veřejnosti, popř. jak dlouho má být přístupná instituci/škole (studentům a pracovníkům), na které je práce vypracovávána (instituce se nastavuje v konfiguračním souboru DSpace).

6.5.2 Dohlížitelé

Plugin Tapir přidává do DSpace role dohlížitelů (supervisors) nad studentskými pracemi (které studenti udržují v průběhu jejich psaní v DSpace). Tito dohlížitelé mohou studentům přes DSpace doplňovat poznámky do prací, kontrolovat průběh jejich vypracovávání atp. Samozřejmě je k dispozici možnost tvořit skupiny dohlížitelů a přidělovat jim různé typy práv ve vztahu k sledovaným studentským pracím. Každému uživateli, je-li dohlížitelem nad nějakou položkou, která je v procesu vkládání do DSpace (Submission process), se v MyDSpace objevuje nový oddíl a tím je **Workspace** – zde sleduje položky, které jsou ve stadiu rozpracování. Tapir definuje standardně tři typy dohlížitelů (dělení dle práv) – jsou jimi *Editor* (může zasahovat do položky, má stejná práva jako vlastník), *Observer* (může pouze sledovat vývoj položky) a *None* (chci-li nastavit jiná práva). To vše, spolu s možností vkládat poznámky k vkládané práci jak ze strany studenta, tak dohlížitele, dělá z pluginu Tapir dobrý komunikační nástroj. Mnohé zlepšení v této oblasti však autoři slibují zejména v budoucích verzích.

6.5.3 Závěr

Tapir je vynikajícím příkladem rozšíření DSpace pro konkrétní speciální účel. Jeho instalace bohužel „přerazí“ původní část kódu, takže zruší částečně univerzálnost původního systému. Je škoda, že autoři neponechali možnost vkládat i původní obecné typy položek (to lze, stále však je uživatel nucen vybírat si mezi Thesis/Dissertation a Other Research Material). Což je spolu s poměrně pracnou instalací jediná výtka, kterou lze k pluginu Tapir mít. Tapir především ukazuje možnou cestu modifikace systému DSpace pro speciálnější účely. Což se projevuje i tím, že jeho použitelnost ve výchozím nastavení je silně lokální – požadavky, které jsou na něj kladeny v oblasti akademických prací publikovaných ve Velké Británii, jistě účelně splňuje, pro podmínky panující na školách v České republice by musely být zavedeny modifikace. Tyto však nejsou zvláště obtížně realizovatelné a pokud uživatel hledá systém na zpracování studentských prací, pak je Tapir dobrou volbou.

6.6 Dokumentace

Dokumentace k jednotlivým verzím DSpace je distribuována ve speciálním balíku a zároveň je dostupná i online na www stránkách DSpace. Vše, co se týká instalace a administrace je popsáno výborně, nechybí příklady, malé tutoriály, přehledy a řešení možných problémů. Konfigurační soubory v sobě obsahují perfektní a podrobné komentáře včetně doporučení, jak co nastavit pro obvyklé instalace. Kolem DSpace se již utvořila poměrně rozsáhlá komunita uživatelů, kteří přispívají do mailing listů, zároveň na serveru SourceForge běží služby pro hlášení chyb, výměnu zkušeností a dalších informací s projektem souvisejících. Celé toto zázemí je velmi silné a celkově dává uživateli bohaté možnosti, jak se vypořádat s problémy nejen pomocí standardní dokumentace dodávané s DSpace a poskytuje dobré předpoklady pro další rozvoj systému.

6.7 Závěr

Systém DSpace se ukazuje být nejživějším, nejhotovějším a nejkomplexnějším z dosud testovaných systémů. V současné verzi obsahuje pouze minimální množství chyb – k těm relativně vážnějším patří především ještě ne zcela perfektně funkční systém práv a jejich delegování (z bezpečnostního hlediska však nepředstavuje riziko) a chyba při tvoření stromu komunit – nelze mít totiž dvě komunity stejného jména (jednoznačný identifikátor komunity v databázi je právě její jméno . . .). Vzhledem k pokroku a rychlosti vývoje však předpokládám, že další zásadní problémy se již neobjeví a veškeré nedostatky budou odstraněny.

Vývojáři DSpace udělali od starších verzí k verzi 1.2 obrovský skok a odvedli skutečně velký kus práce. Mnohá vylepšení a nové funkce jsou velmi vítané, zejména však odstranění počátečních chyb dělá z DSpace v praxi použitelný systém. Je tedy možné ho prakticky okamžitě nasadit do reálného provozu. Nevýhodou je snad stále jeho obecnost (i když autoři se snaží jej částečně specializovat). Vzhledem ke kvalitní podpoře a možnostem rozsáhlých úprav a přeprogramování hodnotím systém DSpace jako velmi dobrý.

Kapitola 7

Porovnání testovaných systémů

V této kapitole budou shrnuty výsledky dosažené zkoušením systémů Fedora, RIB, EPrints a Dspace a provedeme jejich vzájemné porovnání z různých pohledů. Postupně budeme hodnotit jednotlivé dříve popisované součásti a porovnávat jejich výhody a nevýhody. Vlastnosti jednotlivých systémů budou stručně prezentovány v tabulkách.

7.1 Instalace, konfigurovatelnost a správa

V porovnání samotných instalací (bez následné konfigurace) jsou na nejvyšší úrovni systémy DSpace a Fedora – oba systémy mají instalaci dobře zdokumentovanou, autoři do ní nezařadili prvky, které mohou dělat obtíže. Uživatel je sice nucen mnoho kroků provést manuálně, ale má kontrolu nad tím, co dělá. Téměř na stejnou úroveň lze postavit systém EPrints – nevýhodou je však nutnost ručního nainstalování mnoha perlových modulů a s tím spojené problémy. Naštěstí jsou všechny tyto moduly podrobně popsány v dokumentaci, takže je celý proces pouze pracný. Jako zcela nevhodná se mi jeví automatická instalace systému RIB, do které může správce systému zasahovat jen velmi obtížně. Navíc překlad všech potřebných programů včetně jazyka Perl, který je obsažen jako základ ve většině UN*Xových systémů, je velmi nešťastný. Dá se očekávat, že uživatel, který bude takový systém rozbíhat, má dostatek zkušeností, aby si s ruční instalací poradil.

Konfigurovatelnost všech systémů je dobrá, přičemž každý má své výhody a nevýhody. Konfigurace je nejlépe popsána pro systém EPrints, nejhůře pak pro systém RIB (mluvíme-li o nástrojích, které jsou v distribuci se systémem RIB). Konfigurace systému EPrints je ovšem nejméně intuitivní a nejméně příjemná. Navíc systém RIB a zejména systém Fedora jsou specifické tím, že se očekává uživatelova aktivita při vývoji uživatelského rozhraní, proto nelze tyto systémy hodnotit zcela jednoznačně – hodně záleží na zamýšleném použití. Zvláštní kapitolou jsou doprovodné skripty. Zde suverénně kraluje systém DSpace s univerzálním skriptem *dsrun* (pro spouštění konfiguračních nástrojů napsaných v Javě). Nejhůře je na tom systém RIB – žádné dokumentované skripty nemá (pouze jeden skript, nikde v dokumentaci ani v něm není napsáno, co dělá . . .). Konfigurační soubory jsou ve všech systémech dobré – s výjimkou systému RIB, který nemá žádné zdokumentované konfigurační soubory.

Všechny systémy mají velmi kvalitní administrátorské nástroje, pomocí nichž je správa repozitáře snadná a příjemná. Ve srovnání s ostatními systémy má slabší administrátorské rozhraní systém EPrints – akce, které lze v ostatních systémech dělat pohodlněji, je nutno v EPrints dělat ručně v textovém režimu editací souborů (například editace a vytváření metadatových položek, editace typů vkládaných souborů, . . .).

7.2 Struktury a objekty

Nejpropracovanější systém stromového zařazování objektů má systém RIB. Má možnost tvorby vlastních struktur bez omezení hloubky zanoření a k tomu perfektní grafickou aplikaci. Podobné třídění umožňuje i systém EPrints, který má na vytváření struktury svůj nástroj – následná instalace pomocí něj vytvořeného stromu je obtížnější a méně přehledná a lze ji aplikovat pouze na prázdný archiv. Zanoření v systému DSpace se musí dělat s využitím komunit, v současné verzi však neumožňují duplicity. Ale dá se jimi dosáhnout totéž, co v systémech RIB a EPrints. Systém Fedora žádné struktury nemá.

Nejlépe vybudovanou architekturu digitálních objektů má systém Fedora. Naprosto obecný objekt spolu s diseminátory je mezi ostatními systémy výjimečný a nemá téměř žádná omezení. Metadatové položky objektu si může uživatel snadno přidávat dle libosti. Stejně obecný je i objekt v systému DSpace, pouze nemá definovány na datech diseminátory a je na straně uživatele, jakou aplikaci pro zpracování dat použije (DSpace má jeden diseminátor, který pro obrázky zajišťuje generování náhledů – je to ovšem velmi speciální případ). Metadata lze snadno editovat pomocí administrátorského rozhraní. Digitální objekt v systému RIB má netriviální strukturu, která stojí na modelu BIDM (standard IEEE). Uživatel si vše může pomocí administrátorského nástroje předefinovat – výchozí nastavení je však poměrně omezující. Obrovskou výhodou je možnost neomezeně definovat vzájemné vztahy mezi různými typy objektů. Systém EPrints definuje digitální objekty pro potřeby prezentace a archivace STI dokumentů. Vytvořit obecný repozitář je netriviální záležitost a je nutné provádět zásahy do programového kódu (postupy, jak toho dosáhnout, jsou však dokumentovány).

Každý systém má, co se objektů a struktury týká, své výhody i nevýhody a kandidát, který má ideální vlastnosti, nebyl nalezen. Dobře vychází v těchto ohledech systém RIB, nemá však bohužel zajištěnu interoperabilitu pro nestandardní struktury a objekty. Systém EPrints má také dobrou podporu struktur, ale s objekty je to obtížnější. Celkově nejlepší se jeví systém DSpace – zejména v tom, co je okamžitě k dispozici (byť s výhradami) – má struktury i obecné objekty s podporou současných volně přístupných standardů.

7.3 Dokumentace

S výjimkou systému RIB je uživatelská dokumentace velmi dobrá. Systém RIB má nedostatky, které vyplývají především z použití proprietárních standardů. U ostatních systémů je uživatelská dokumentace v pořádku. Programátorská dokumentace je výborná ve všech systémech, pouze v systému EPrints není úplně dotažená do konce (jedná se však o malé části). Dokumentace pro správce je jednoznačně nejslabší v systému RIB, u ostatních je zcela dostačující.

Celkově vychází nejlépe systémy Fedora a DSpace. Systémy EPrints a RIB mají v oblasti dokumentace nedostatky. Navíc systém DSpace má v tomto ohledu jednoznačně nejlepší podporu díky široké komunitě uživatelů a velmi živému systému hlášení problémů a chyb, které tato komunita společně s vývojáři může řešit.

7.4 Nasazení v praxi

Funkčnost jednotlivých systémů asi nejlépe prověří jejich nasazení v praktických aplikacích. Nejvíce nasazeními se na svých stránkách chlubí systémy RIB a EPrints. Podívejme se na ně blíže.

Systém EPrints uvádí na svých stránkách aktuální počty serverů, na kterých je provozován, včetně celkového počtu digitálních objektů v nich. V listopadu 2004 se počet serverů pohyboval kolem 140 a počet digitálních objektů v nich kolem 31 tisíc. Což odpovídá tomu, že digitálními objekty v systému EPrints jsou převážně dokumenty z oblasti STI – počty prací v této oblasti nenarůstají tak rychle a tato čísla nemají patřičnou vypovídající hodnotu o vhodnosti repozitáře pro ukládání jiných typů objektů. Pro srovnání – server DKF (Digitální knihovna fotografií) [66] provozovaný na ÚVT MU pro ukládání fotografií obsahuje kolem 8 tisíc objektů. V případě, že by takových serverů bylo v provozu 140, dostáváme se někam k miliónům fotografií na těchto serverech – fotografií, nikoli všech typů objektů. Z tohoto pohledu již nejsou čísla uvedená na www stránkách systému EPrints tak přesvědčivá. Každopádně se ukazuje, že pro oblast STI je systém EPrints relativně hojně používán a nasazován v praxi. Ve srovnání se zbylými testovanými systémy vychází systém EPrints jako nejstabilnější – ale i méně dynamicky se rozvíjející.

Systém RIB má na svých stránkách také několik odkazů na své praktické nasazení – můj pocit z nich je však krajně rozporuplný, část odkazů nefunguje (RIB americké armády), fungující odkazy ukazují na repozitáře se sotva desítkou objektů . . . Nalezení RIB repozitářů přes internetové vyhledávače byl téměř nadlidský výkon – pokud již některé odkazy vypadaly slibně, byly nedostupné. Předpokládám, že pro svoji vlastní reklamu udělali vývojáři maximum a uvedli na svých www stránkách opravdu většinu provozovaných repozitářů – výsledek je ovšem velmi nepřesvědčivý.

Systém DSpace sice odkazuje pouze na svoji primární implementaci na MIT, ale tato implementace je naprosto funkční. Několik desítek dalších nasazených systémů lze nalézt přes vyhledávač Google[1] [58], zejména u partnerů, kteří na vývoji spolupracují. Jmenujme například DSpace@Cambridge [59], francouzsky lokalizovaný ALADIN [60], DSpace at Cornell University [61], DSpace at The Australian National University [62], italsky lokalizovaný DSpace a Parma [63] a další a další. Celkový počet objektů lze těžko odhadovat – mnohé instituce zatím se systémem DSpace spíše jen experimentují. Jako perspektivní systém se silnou institucí za zády je systém DSpace nasazován celosvětově a rychle se rozvíjí.

Systém Fedora je skutečně velmi speciální případ. Momentálně neexistuje žádný hotový projekt, který by představoval praktické nasazení. Systém Fedora je teprve ve druhé fázi vývoje a očekává se, že po ní se přistoupí ke skutečné aplikaci. Momentálně je možné odkázat na některá testování na různých institucích. Jejich výsledky lze nalézt na [64]. Tyto výsledky je možné shrnout do tvrzení, že proběhly a probíhají zátěžové testy s velkými počty (až desetitisíce) digitálních objektů různých typů. (Autoři uvádí na www stránkách systému Fedora, že prováděli i testy s milióny objektů!)

Když shrneme výše zmíněné skutečnosti, pak jednoznačným vítězem v praktickém nasazení v oblasti univerzální repozitář a digitální knihovní systém je DSpace. V oblasti STI je nejlepší a nejověřenější systém EPrints. Nutno vzít v úvahu, že většina nasazení testovaných systémů jsou stále spíše testovací – uživatelé zkoušejí možnosti jednotlivých systémů.

[1] Doporučuji zadat vyhledávači řetězec „DSpace at“. Výsledkem jsou odkazy na několik desítek instalací systému DSpace.

7.5 Tabulky porovnání systémů

V této části uvedeme pro přehlednost tabulky shrnující vlastnosti testovaných systémů. V jednotlivých sekcích bude vždy popsán předmět, kterého se tabulka týká, včetně vysvětlení použitých zkratek. Tabulky přehledným způsobem shrnují nebo upřesňují informace obsažené v předchozím textu diplomové práce.

7.5.1 Obecné informace

Zde uvedeme přehledově základní informace o jednotlivých systémech. Jedná se o hodnocení celkového konceptu daného předmětu.

	Fedora	**RIB**	**EPrints**	**DSpace**
Obecný repozitář	Ano	Částečně	Částečně	Ano
Interoperabilita	Dobrá	Horší	Dobrá	Výborná
Workflow	Ne	Ne	Ano	Ano
Diakritika	Částečně	Ne	Ne	Ano
Metadata	DC	BIDM	Vlastní	DC
Import/Export	Ano	Ne	Ano	Ano
Dokumentace	Výborná	Horší	Dobrá	Výborná
Velké sbírky	Ano	Částečně	Částečně	Ano

7.5.2 Protokoly, standardy

V tabulce jsou uvedeny standardy a protokoly, které jednotlivé systémy podporují – at' už jde o metadata, interoperabilitu a další.

	Fedora	**RIB**	**EPrints**	**DSpace**
DC	Ano	Ne	Ne	Ano
METS	Ano	Ne	Ne	Ano
OAI-PMH	Ano	Ne	Ano	Ano
OpenURL	Ne	Ne	Ne	Ano
CNRI-Handles	Ne	Ne	Ne	Ano
BIDM	Ne	Ano	Ne	Ne

7.5.3 Software

Informace o UN*X systémech a jejich zkratkách lze nalézt například na [65]. Zkratky použité u licencí jsou MPL – Mozilla Public Licence, UoT – softwarová licence University of Tennesse (k dispozici v distribučním balíku systému RIB), GPL – GNU Public Licence.

	Fedora	**RIB**	**EPrints**	**DSpace**
Licence	MPL	UoT	GPL	MPL
Prostředí	Java	Perl, Java	Perl	Java
WWW server	Tomcat	Apache	Apache	Tomcat
Podporované platformy				
Windows	Ano	Ano	Ne	Ne
Linux	Ano	Ano	Ano	Ano
Solaris	Ano	Ano	Ne	Ano
Jiné	MacOS	AIX, IRIX	MacOS	HP/UX
Podporované databáze				
MySQL	Ano	Ano	Ano	Ne
PostgreSQL	Ne	Ne	Ne	Ano
Oracle	Ano	Ne	Ne	Ne
Mc-Koi	Ano	Ne	Ne	Ne

7.5.4 Uživatelské prostředí

„Administrátorské rozhraní“ obsahuje stručné shrnutí podpory správy daných systémů. Položka „Aplikace“ popisuje má-li systém administrátorskou aplikaci jako zvláštní program. Položka „Web aplikace“ určuje má-li systém webové rozhraní pro správu a „Skripty“ říkají, jaká je podpora dávkových skriptů. Řádek „Knihovny“ určuje, jaká je podpora programování systému a dokumentované funkce a volání (programátorské knihovny).

Část „Uživatelské www rozhraní“ charakterizuje www rozhraní pro koncové uživatele, pro přístup, hledání a procházení. Položka „Typ generování“ může mít tyto hodnoty: Stat=statické (částečné generování statických stránek) a Dyn=dynamické (dynamicky generované stránky). Vlastnost „Hledání“ říká jsou-li prohledávána jen metadata (Metadata) nebo i obsah objektů (Obsah).

	Fedora	**RIB**	**EPrints**	**DSpace**
Administrátorské rozhraní				
Aplikace	Ano	Částečně	Ne	Ne
Web aplikace	Ne	Částečně	Ano	Ano
Typ aplikace	Java	Applet	Web	Web
Skripty	Dobré	Nemá	Výborné	Výborné
Programátorské rozhraní				
Knihovny	Ano	Ano	Ne	Částečně
Typ volání	WSLD	Perl, C, Java	–	Java
Typ dokumentace	Java Docs	www	–	Java Docs
Zdrojové kódy	Ano	Částečně	Ano	Ano
Uživatelské www rozhraní				
Kvalita	Velmi chudé	Chudé	Dobré	Výborné
Typ	Stat	Stat	Stat	Dyn
Uživ. účty	Ne	Ne	Ano	Ano
Přístup. práva	Ne	Ne	Částečně	Ano
Hledání	Metadata	Metadata	Obsah	Obsah
Subskripce	Ne	Ne	Ano	Ano

7.5.5 Celkové hodnocení

V následující přehledové tabulce jsou celkově hodnoceny vybrané vlastnosti testovaných systémů. Hodnocení je prováděno stupnicí 1 až 5, přičemž nejlepší je 1 a nejhorší je 5. Význam jednotlivých hodnocení je tento:

1 – výborný (systém nemá v dané oblasti nedostatky, případně je jednoznačně nejlepší vzhledem k ostatním systémů).

2 – velmi dobrý (systém je v dané oblasti velmi dobrý, má plnou podporu, obsahuje však některé nedostatky).

3 – dobrý (systém deklaruje danou vlastnost/funkcionalitu, její zpracování je však zatíženo chybami – dosažení uspokojivého stavu je možné, ale náročné).

4 – špatný (systém deklaruje vlastnost/podporu, ve skutečnosti ji však plně neimplementuje nebo se závažnými chybami – daný stav je prakticky nemožné vlastními silami změnit).

5 – zcela nedostatečný (daná oblast je v daném systému zcela nepoužitelná nebo není vůbec podporována).

Hodnocení je většinou komplexní a zahrnuje celou škálu poznatků, které byly o systémech zjištěny.

„Podpora databází“ určuje, jak se který systém umí vyrovnat s novými verzemi databázových strojů, které jsou doporučeny pro daný systém jako výchozí[2].

Řádek „Velké sbírky“ hodnotí, jak se systémy jsou schopny vypořádat s velkými objemy objektů (statisíce až miliony digitálních objektů).

„Snadnost obsluhy“ je subjektivní hodnocení příjemnosti práce se systémem.

	Fedora	**RIB**	**EPrints**	**DSpace**
Obecnost	1	2	3	1
Interoperabilita	1	4	1	1
Dokumentace	1	3	2	1
Diakritika	3	5	3	1
Praktické nasazení	4	3	1	1
Import/Export	1	5	2	1
Podpora databází	4	5	1	1
Instalace	2	5	1	1
Uživatelské rozhraní	4	3	2	1
Struktury zanoření	4	1	2	2
Vyhledávání	4	3	2	1
Rozhraní pro správu	1	1	2	1
Administrační skripty	2	5	1	1
Snadnost obsluhy	1	2	3	1
Programátorské rozhraní	1	1	3	2
Stupeň vyzrálosti (dokončenosti)	4	4	1	1
Velké sbírky	1	3	3	2

7.6 Závěr

Společný nedostatek, kterým trpí všechny testované systémy, je nepřítomnost horizontálního provázání, tj. možnost provádět vazby mezi digitálními objekty (nikoli ve smyslu, že dané objekty leží v jedné kolekci, ale ve

[2] Systém Fedora (verze 1.2.x) má problémy s vyššími verzemi databáze MySQL, bez problémů si poradí jen se staršími verzemi 3. Pro rozběhnutí systému spolu s databází MySQL verze 4.x je potřeba změnit ovladač JDBC. I novější ovladač si však nedokáže poradit s nejnovější verzí MySQL 4.6.x. Z těchto důvodů má systém Fedora v oblasti databází špatné hodnocení.

smyslu vazby objektu na jiný objekt v jiné kolekci nebo složce). Částečně má horizontální provázání pouze systém RIB, ale ten má bohužel i několik základních nedostatků, které tato výhoda nevyváží.

Žádný ze systémů není obecný v tom smyslu, že by umožňoval ukládání všech typů objektů a přitom dovoloval rozlišovat jejich typy a podle těchto typů poskytoval příslušné rozhraní a chování vůči uživateli.

Na základě provedené analýzy jsem vybral pro implementaci typů objektů systém DSpace. V současnosti se jedná o projekt, který nejlépe splňuje požadavky kladené na univerzální digitální knihovnu. Má nejlepší uživatelské rozhraní a podporuje nejvíce standardů. Zároveň obsahuje potřebné informace pro programátora a je nejperspektivněji a nejrychleji se rozvíjejícím softwarem. Spolu se silnou institucí (MIT) v pozadí a rostoucí komunitou uživatelů se zdá být stabilním řešením i do budoucna.

Kapitola 8

Přehled dalších systémů

Kromě systémů, detailně otestovaných a popsaných v předešlých kapitolách, existuje ještě několik desítek dalších systémů, které se více či méně přibližují digitální knihovně. Tato kapitola stručně uvede přehled těch systémů, které byly testovány na Ústavu výpočetní techniky Masarykovy univerzity v Brně.

8.1 Bricolage

Systém **Bricolage** [67] je komplexní publikační software. Slouží především jako správce obsahu www stránek. Obsahuje kompletní www rozhraní s propracovaným systémem přístupových práv, uživatelů a skupin. Typické použití systému je v oblasti publikování elektronických časopisů a novin.

8.2 Zope/Plone

Systém **Zope** [68] je aplikační server určený pro vytváření a správu webových portálů a aplikací. Poskytuje sadu aplikačních funkcí pro vytváření vlastních projektů. Nad ním je s podporou aplikačního rozhraní Zope CMF (Content Management Framework) postavená aplikace **Plone** [69]. Plone je systém pro správu dokumentů (content management system, dále jen CMS). Poskytuje uživatelské rozhraní i systém práv. V době testování (červen 2004) nebylo zřejmé budoucí profilování systému a jeho zaměření (zejména v oblasti řešení problému editace a správy metadat).

8.3 PHP Nuke

Systém **PHP Nuke** [70] je systém pro správu dokumentů (CMS). Integruje nástroje pro tvorbu webových portálů, vhodný je zejména pro e-learning, e-komerci, různé agendy a další. Postaven je na principu modulů – několik hotových modulů je předem k dispozici (moduly pro encyklopedie, recenze, ...). Nemá možnosti vnořování a správy metadat.

8.4 Postnuke

Systém **Postnuke** [71] vychází ze systému PHP Nuke, ale je zaměřen především na větší bezpečnost. Vlastnosti má téměř shodné s PHP Nuke, umožňuje navíc jednu úroveň vnoření objektů a celkově je stabilnější. Bohužel ve srovnání s PHP Nuke je méně robustní a má slabší podporu.

8.5 Midgard Project

Midgard Project [72] se skládá ze dvou částí – **Midgard Framework** a **Midgard CMS**. Midgard Framework je programátorská knihovna poskytující nástroje pro tvorbu systémů CMS. Midgard CMS je systém pro správu dokumentů postavený na systému Midgard Framework. Filozofie projektu je podobná jako u systému Zope, celkově se však celý projekt jevil slabší, zejména kvůli výskytu chyb, slabé dokumentaci a nepřehlednosti.

8.6 OpenCMS

OpenCMS [73] je systém pro vytváření a správu obsahu webu. Uživatel vytváří stránky editorem typu Microsoft Word. Obsahuje správu verzí a proces vkládání dokumentů (workflow). Práce je řešena pomocí webového rozhraní, které je však plně funkční pouze v prohlížeč Microsoft Internet Explorer.

8.7 TikiWiki

Jedná se o víceúčelový balík aplikací **Tiki CMS/Groupware** [74] (zkráceně nazývaný TikiWiki), který může být také použit pro tvorbu webových portálů a aplikací. Je to modulární systém, k dispozici jsou hotové moduly pro tvorbu článků, galerií obrázků, diskusní fór apod. Má podporu přístupových práv, uživatelů i skupin. Objekty lze řadit do kategorií, které lze libovolně zanořovat. Systém je vhodný spíše pro publikační činnost.

8.8 CDSware

CDSware [75] je systém pro správu dokumentů. Deklaruje podporu OAI, jako metadatový standard používá MARC (tento standard je používán v klasickém knihovnickém prostředí pro popis tištěných materiálů). Systém byl v době testování ve vývoji a obsahoval velké množství nedostatků a chyb (nepodařilo se uspokojivě celý systém zprovoznit). Do budoucna se však jeví jako zajímavý projekt – zejména z toho pohledu, že je používán institutem CERN ve Švýcarku. Proto se dá předpokládat budoucí silná podpora tohoto systému.

8.9 phpWebThings

Podobně jako systém PHPnuke je i systém **phpWebThings** [76] určen pro webové publikování a správu dokumentů (systém CMS). Je jednoduchý, přehledný a intuitivní – pro práci nabízí kompletní uživatelské rozhraní. Jeho nedostatky spočívají opět v chudosti metadat a nemožnosti přidávat metadatové položky.

8.10 Další systémy

Mimo výše zmíněné systémy existuje i celá řada projektů, které nebyly testovány. Zejména z důvodu, že se jedná o komerční systémy nebo systémy vyžadující komerční software. Z komerčních produktů byl Ústavem výpočetní techniky zakoupen a testován systém Aipsave. Z ostatních již jen stručně výpisem: openacs, Amo, i-tor, mycore, DLXS, Lenya a další.

8.11 Shrnutí

Všechny výše uvedené systémy se deklarují jako systémy pro správu a archivaci digitálního materiálu. I když mnohdy nepoužívají terminologii užívanou v oblasti digitálních knihoven, použité technologii částečně digitální knihovny implementují. S výjimkou Zope/Plone a CDSware však nemají v současné době podporu protokolů a standardů běžných v oblasti digitálních knihoven (systém Plone má tuto podporu jen nepřímo pomocí dalších dodaných programů). Výše uvedené systémy svým zaměřením spadají do oblasti systémů pro správu dokumentů (CMS) a lze je částečně používat i jako digitální knihovny.

Kapitola 9

Rozšíření systému DSpace

V této techničtěji zaměřené kapitole popíšeme původní návrh a realizaci specializace systému DSpace, která byla vytvořena autorem této diplomové práce. Bude uveden návod, jak přidávat vlastní typy specializovaných komunit a kolekcí a jaké úpravy bylo pro toto přizpůsobení potřeba implementovat. Byl zvolen postup, který doporučují autoři systému DSpace – minimální nutné zásahy přímo do jádra (do tříd a servletů) a libovolné úpravy stránek JSP (Java Server Pages). Uživatel má velkou svobodu v interpretaci údajů, může je snadnou změnou stránek JSP zobrazovat v požadovaném tvaru a manipulovat s nimi. Zároveň nemusí provádět veliké zásahy při přechodu na novou verzi systému DSpace.

Vnitřní jádro a návrh systému DSpace je sám o sobě na dobré úrovni a pro implementaci vlastních typů kolekcí a komunit lze použít současnou architekturu. Tato nabízí větší možnosti, než současné naprogramované rozhraní distribuované se systémem DSpace využívá. Záleží pak na uživateli, jak dokáže tyto možnosti uplatnit. Jeví se jako velmi vhodné využít stávající definovaný model a systém volání kvůli vzájemné kompatibilitě vzhledem k budoucím verzím systému DSpace.

V některých pasážích textu budou uvedeny i postupy, jak zasahovat hlouběji do systému. Je na zvážení uživatele-programátora, zda-li se rozhodne tyto zásahy provádět. Doporučení je maximálně se jim vyhnout a použít je pouze v nezbytných případech – a pokud to je možné, pak provádět zásahy pouze přidáváním nových funkcí a nikoli přímou úpravou stávajícího kódu.

Je nutné předeslat, že pro dobré pochopení následujícího textu jsou nutné znalosti některých síťových technologií spojených s programováním v jazyce Java. Týká se to znalosti značkovacího jazyka HTML, programovacího jazyka Java (minimálně na úrovni schopnosti používat základní volání metod a řídící konstrukce jazyka) a technologie JSP – s těmito znalostmi si programátor vystačí při využití již hotové podpory pro přidávaní nových typů do systému DSpace (bude-li chtít provádět pouze základní změny).

Pro sofistikovanější programátorské úpravy typů je nutná znalost programování Java servletů, uživatelských tagů JSP a vytváření dynamických webových systémů (naprostým minimem je částečná znalost protokolu HTTP, standardních metod `doGet` a `doPost` a metod tříd **HttpServletRequest** a **HttpServletResponse**). Orientace v programátorské dokumentaci systému DSpace je také nutná, protože vývojáři dali k dispozici mnoho vysokoúrovňových volání, která odstiňují některé základní metody standardně poskytované prostředím J2SDK (kompilátor a interpret jazyka Java). Tato volání usnadňují modifikaci systému a zajišťují bezpečnější programování, proto je vhodné jich maximálně využívat. Systém DSpace definuje rozhraní pro přístup k databázi, dále sám řeší problém sezení protokolu HTTP a autentizaci uživatele, takže není nutné znát práci přímo s databázovým rozhraním JDBC ani práci se sezeními (**HttpSessions**).

9.1 Značení v textu práce

V textu je používáno poměrně velké množství názvů balíků, souborů, příkazů, metod, tříd, servletů, objektů a databázových tabulek. Pro snadnější orientaci jsou balíky značeny kurzívou (*balík*), standardní metody, příkazy, cesty k souborům a soubory neproporcionálním písmem (`metoda`), nově přidané metody neproporcionálním tučným písmem (**`nová_metoda`**), servlety a třídy tučným písmem (**třída**), objekty tučnou kurzívou (***objekt***), názvy tabulek tučným skloněným písmem (***tabulka***) a jména sloupců[1] v tabulkách skloněným písmem (*sloupec*).

Dále v textu bude odkazováno na některé soubory, které se týkají implementace systému DSpace. V této souvislosti bude užíváno proměnných `DSPACE-SRC` a `TOMCAT`. Proměnná `DSPACE-SRC` reprezentuje cestu k zdrojovým souborům systému DSpace (například `/usr/src/`) a proměnná `TOMCAT` cestu k instalaci serveru Tomcat (například `/opt/tomcat/`). Veškeré soubory pak budou odkazovány vzhledem k těmto cestám (například `TOMCAT/webapps/dspace.war`). Výjimku tvoří soubory reprezentující třídy jazyka Java, které budou identifikovány svým balíkem (jsou uloženy v adresáři `DSPACE/src`). Kvůli zvýšení přehlednosti a zkrácení výpisů je

[1] Z formálního hlediska je „sloupec“ *jméno atributu* relačního schématu a tomuto jménu je přiřazená *doména* (množina hodnot, kterých může atribut nabývat). Pojem *atribut* používáme tehdy, zabýváme-li se relacemi obecně a teoreticky. V praxi jsou ovšem většinou relace a relační schémata určována (jednoznačně) tabulkou – *jméno atributu* je jméno sloupce tabulky a *doména* je typ sloupce tabulky (například typ `integer` – množina celých čísel). V následujícím textu bude uváděn (i v odborné literatuře užívaný) pojem „sloupec“ tabulky (resp. jméno sloupce tabulky) ve smyslu výše uvedeného formálního pojmu atribut (jméno atributu). Přidání sloupce do tabulky pak odpovídá rozšíření relačního schématu o daný atribut.

zavedena proměnná `JSP`, která reprezentuje cestu `DSPACE-SRC/jsp/`.

9.2 Zavedení typů

V rámci praktické části diplomové práce byly v systému DSpace provedeny změny, které umožňují typovat komunity a složky. Každá komunita a kolekce má svůj typ a podle něj může uživatel navrhovat speciální rozhraní – například kolekce může mít typ „fotografie" a podle toho je veškerý její obsah prezentován uživateli.

Pomocí modifikovaných stránek JSP a několika nových metod je nyní při vytváření komunity nebo kolekce uživatel tázán na jejich typ. Informace o tomto typu má pak uživatel-programátor k dispozici a může je využívat při modifikaci stránek JSP. Tato modifikace nesnižuje nijak obecnost vlastního systému DSpace – všechny komunity a kolekce mají výchozí typ nastaven tak, aby se vzhledem k nim systém DSpace choval naprosto standardním způsobem. Ale navíc k tomuto standardnímu modelu lze vytvářet vlastní specializované komunity a kolekce a jejich obsah prezentovat vlastním zvoleným způsobem.

Vlastní typ má i položka (Item). Tento typ koncový uživatel systému DSpace nenastavuje. Položka zdědí typ své mateřské kolekce (kolekce, ve které byla vytvořena). Toto rozšíření je zavedeno z několika důvodů. Předně položka se může objevit ve více kolekcích různých typů. Uživatel může odkazovat přímo na položku aniž by se k ní propracoval přes nějakou kolekci (nelze rozpoznat jakým způsobem se uživatel k položce dostal – zda přímým odkazem nebo při procházení systémem DSpace). Informace o mateřské kolekci není nikde uchovávána, navíc mateřská kolekce může být smazána a položka v systému zůstává. Současný návrh systému DSpace neumožňuje zjistit, ve které kolekci si uživatel položku prohlíží.

9.2.1 Implementace

Prvním krokem bylo provedení zásahu do databáze – modifikace tabulek ***collection***, ***community*** a ***item***. K těmto tabulkám byl přidán sloupec, který určuje typ. Pro tabulku ***collection*** to je sloupec *col_type*, pro tabulku ***community*** sloupec *com_type* a pro tabulku ***item*** sloupec *item_type*.

Dále proběhla modifikace tříd **Community** a **Collection** (obě jsou umístěny v balíku *org.dspace.content*). Byly přidány metody, které vracejí typ příslušného objektu (komunity nebo kolekce) a realizují uložení typu do databáze. Metody **`getComType()`**, **`setComType()`** a **`getComTypeName()`** příslušejí třídě Community. Metody ve třídě Collection jsou **`setColType()`**,

`getColType()` a **`getColTypeName()`**. Do třídy **Item** byly přidány metody **`setItemType()`** a **`getItemType()`**. Všechny výše zmíněné metody budou popsány dále v textu (bude ukázáno jejich použití).

Bylo nutno také doprogramovat příslušné části kódu zajišťující nastavování typů z uživatelského rozhraní. Upraveny byly příslušné stránky JSP realizující vkládání nových komunit a kolekcí a servlety, které toto vkládání zajišťují (**CollectionWizardServlet** a **EditCommunitiesServlet** – oba z balíku *org.dspace.app.webui.servlet.admin*). Vkládání se v systému DSpace realizuje v několika krocích. Příslušné nastavení typu je vždy prováděno jako úplně první krok, proto je možné další kroky (typ metadat apod.) modifikovat dle tohoto zadaného typu. Upraven byl i **SubmitServlet**, který provádí nastavení typu položky v databázi (děděním od mateřské kolekce).

Byl zvolen model, v němž jsou typy komunit a kolekcí na sobě naprosto nezávislé. Důvodem je ponechat větší volnost uživateli-programátorovi při implementaci vlastních typů. Nelze totiž dopředu odhadovat, jaké typy bude chtít uživatel v DSpace uchovávat, a svázání typů komunit a kolekcí by mohlo být omezením. Zůstává tedy pouze na programátorovi, na kolik využije možnosti typovat jak kolekce, tak komunity, či zda zavést jejich závislost obecně nebo jen pro určité typy. V dalších částech bude ukázáno jak využití typů u kolekcí (pro typ *Image*), tak využití typů u komunit (pro typ *Magazines*).

9.2.2 Instalace

V této části budou popsány kroky nutné pro nainstalování podpory typů v systému DSpace[2]. Celá tato instalace vyžaduje již funkční a nainstalovaný systém DSpace. Popis jeho instalace lze nalézt na [77]. Pro následnou instalaci rozšíření je nutné použít jako výchozí verzi systém DSpace *1.2.1beta2*.

Změny tabulek v databázi

Ke každá kolekci (Collection), komunitě (Community) a položce (Item) musí být přidán v databázi sloupec, který určuje její typ. Toto nastavení se provede těmito příkazy jazyka SQL na databázi *dspace*:

```
alter table community add com_type int;
alter table collection add col_type int;
alter table item add item_type int;
```

[2]Uvedeným postupem se kompletně nainstaluje i podpora typy *Image*, *Magazines* a *Magazines-Year*. Podorbně je instalace popsána v souboru `INSTALL` na CD, které je přiloženo k práci.

Kvůli podpoře typu *Magazines* je nutné rozšířit tabulku komunit o sloupec *metadata_id* tímto příkazem (opět v databázi *DSpace*):

```
alter table community add metadata_id text;
```

Pokud provádíme instalaci do systému DSpace, ve kterém jsou již kolekce a komunity, tak je nutné tyto hodnoty ve sloupcích vyplnit. Nastavují se na hodnotu **0**, která určuje, že tento typ kolekce, komunity nebo položky je původní typ systému DSpace. Nastavení lze provést příkazy

```
update community set com_type = 0;
update collection set col_type = 0;
update item set item_type = 0;
```

Přeinstalování systému

Pro následnou instalaci rozšíření systému DSpace je nutné aplikovat přiložený patch. Patch musí být aplikován na systém DSpace verze *1.2.1beta2*. Aplikace patche se provede příkazem

```
patch -p1 < ../Dspace-typesextension.patch
```

provedeným v adresáři `DSPACE-SRC`.

Nyní je již možné systém DSpace přeinstalovat. Postup instalace je naprosto shodný s postupem uvedeným v dokumentaci k DSpace. Přeinstalování se provede spuštěním příkazu

```
DSPACE-SRC/ant build_wars
```

a nakopírováním souboru `dspace.war` do adresáře `TOMCAT/webapps`. Poté je nutné smazat adresář `TOMCAT/dspace` a restartovat www server Tomcat. Tímto je modifikovaný systém nainstalován. Přeinstalování pomocí příkazu `DSPACE-SRC/ant build_wars` a následné kroky je nutné provádět vždy, když dojde k změně typů v systému DSpace.

9.3 Tvorba vlastního typu

V této části bude popsán základní proces vytváření a použití vlastního typu. Založen je pouze na modifikaci základních stránek JSP a volání již připravených metod.

9.3.1 Přidání typu

Celý proces proběhne ve dvou krocích – editací metod, které vracejí jméno typu, a úpravou stránek JSP, které typ nastavují.

Novému typu nejdříve abstraktně přidělíme číslo (typu Java Integer). Následně je nutné změnit statické metody **`String getColTypeName(int type)`** (uložena ve třídě **Collection**, v balíku *org.dspace.content*) a **`String getComTypeName(int type)`** (třída **Community** v tomtéž balíku). Tyto metody v podstatě přiřazují příslušnému typu (jeho číselné hodnotě) sémantiku. Modifikace se provede přidáním větve do příkazu

```
switch (type) {
case 0: return "Classic Dspace";
case 1: return "Magazines";
case 2: return "Images";
...
}
```

Například větev `case 3: return "New type"` přidá typ reprezentovaný číslem **3** s názvem **New type**. Typy v obou metodách mohou být zcela nezávislé a lze přidat typ pouze komunitě nebo kolekci – není nutné měnit obě metody – jejich vzájemná korespondence je ponechána na programátorovi a cíleném využití.

Posledním krokem při přidávaní nového typu je editace souborů JSP, které poskytují uživatelské rozhraní pro určení typů. Jedná se o soubory `create-community.jsp` a `wizard-questions.jsp` uložené v adresáři `DSPACE-SRC/jsp/dspace-admin/`. Výběr typu je realizován pomocí tagu `<select>` a na příslušném místě v těchto souborech. Nový typ se přidá vložením řádky

```
<option value="3">New type</option>
```

Takto je vložen nový typ s názvem **New type**. Zbývá znovu přeinstalovat DSpace (postup uveden v části 9.2.2 v odstavci s názvem Přeinstalování systému).

9.3.2 Použití vlastního typu

Vlastí typ lze použít na stránkách JSP tak, že se přidá (nejlépe na začátek stránky) direktiva, která zaručí zpřístupnění tříd **Collection** nebo **Community** (případně obou z nich). Příslušné direktivy vypadají takto:

```
<%@ page import="org.dspace.content.Collection" %>
<%@ page import="org.dspace.content.Community" %>
```

Pokud jsou na stránce tyto direktivy, pak lze příslušné typy získat voláním metod **`int getComType()`** pro komunitu a **`int getColType()`** pro kolekci. Tyto metody vracejí číselný typ. Pro získání jména typu lze použít metody **`String getColTypeName()`** a **`String getComTypeName()`**. Tyto metody vracejí řetězec, který slovně určuje daný typ. Všechny tyto metody jsou dynamické, je tedy nutné pro jejich volání mít k dispozici objekt typu Community nebo Collection.

K dispozici jsou také dvě statické metody, které jsou ekvivalentem metod **`getColTypeName()`** a **`getComTypeName()`**. Jedná se o metody **`String get[Col|Com]TypeName(int type)`**. Jejich vstupním parametrem je číselný typ. Tyto metody obvykle voláme, pokud jsme již typ získali – jsou statické a jejich volání je rychlejší než volání jejich dynamických ekvivalentů.

Získání objektu typu Community nebo Collection závisí na typu stránky JSP, kterou upravujeme. Nejčastějším případem je, když původní stránka již příslušné objekty využívá – autoři DSpace je pojmenovávají jednotně ***collection*** a ***community***. Při úpravě stránek JSP je pravděpodobné, že bude nutné upravovat i stránky, které se týkají vkládání položek (v závislosti na typu kolekce upravíme metadata apod.). U těchto souborů je k dispozici objekt ***si*** typu **SubmissionInfo**, z kterého lze příslušný typ kolekce, do které se vkládá získat takto:

```
int colType =
       si.submission.getCollection().getColType();
```

Pokud je typ již znám, lze snadnou konstrukcí typu `if-the-else` (nebo, je-li již typů více, je vhodnější `switch-case`) oddělit příslušné bloky HTML kódu, který pracuje s konkrétními daty. Vzhledem k přehlednosti je rozumné volit použití direktivy

```
<%@ include file="manage_my_type.jsp" %>
```

která vkládá do kódu soubor `manage_my_type.jsp`. V něm je uložen kód, který obsluhuje nový typ.

9.4 Užití typů kolekce – kolekce fotografií

Nyní na praktickém příkladu ukážeme použití předem zavedené podpory *typů kolekcí* – zavedeme kolekci typu *Image*, která bude uchovávat fotografie. Celý postup sleduje soubory, které je nutné editovat – jejich kód je opatřen poznámkami, které zároveň formou tutoriálu vedou programátora obsluhou vlastního typu. Kvůli budoucí distribuci kódu jsou tyto podrobné komentáře v anglickém jazyce.

Programátor se může spokojit s předem definovanými metadaty – pouze je jinak interpretovat. Postup úpravy je pak poměrně snadný, nedává však tak bohaté možnosti, jako poskytují sofistikovanější zásahy. V následujícím textu bude ukázáno, jak provádět tyto sofistikovanější úpravy, například přidávání vlastních (povinných i nepovinných) metadatových položek, změny typu zobrazování položky apod. V těchto případech je nutné tyto novinky ošetřit doprogramováním příslušných částí kódu do servletů. Uvedený postup není univerzální, a proto bude pravděpodobně nutné dělat pro určitý typ konkrétní zásahy hlouběji do kódu. Bohužel v současné verzi systému DSpace jsou některé části kódu nedotažené do konce a obsahují chyby (na některé bude v textu upozorněno), které je nutné řešit individuálně. Níže uvedený postup je zvolen tak, aby v minimální nutné míře docházelo k vlastnímu přepisování původního programu DSpace, a aby se co nejvíce využívalo stránek JSP – mnohdy i za cenu redundance, která však později usnadní přechod na novější verze systému DSpace.

V následujících odstavcích si postupně uvedeme, jak využít nově vytvořeného typu *Image*. S pomocí tohoto typu vytvoříme model, který je podobný modelu, který používá systém DKF (Digitální knihovna fotografií) [66]. V tomto modelu je uživatel při vkládání do kolekce typu *Image* oproštěn od vyplňování irelevantních metadat. Každá položka obsahuje právě jeden soubor s obrázkem, při zobrazení položky se tento obrázek přímo zobrazuje a navíc se zobrazují pouze ta metadata, která mají u této položky význam. Stránka se zobrazenou kolekcí automaticky ukáže fotografie, které jsou v kolekci obsaženy – ukáže jejich náhledy a popis.

Typu *Image* byla přidělena hodnota **2**. V následujícím textu se nejdříve se seznámíme s postupem, jak upravit vkládání položek – jak upravovat metadata, tvořit vlastní metadata a celkově ovlivňovat celý proces vkládání. Poté bude uveden postup řešení speciálního prohlížení kolekce a položky na základě jejího typu.

Následující úvahy předpokládají, že je již typ *Image* zaveden (pomocí postupu, který je popsán v části 9.3).

9.4.1 Úprava stránek JSP – vkládání položky

Prvním krokem, který se provede poté, co uživatel zvolí v systému DSpace možnost vložit novou položku, je vyvolání servletu **SubmitServlet** z balíku *org.dspace.app.webui.servlet* (v tomto balíku jsou uloženy všechny servlety).

Tento servlet zavolá JSP stránku `JSP/submit/initial_questions.jsp`, která realizuje získávání základních informací o vkládané položce. Konkrétně se ptá uživatele na to, zda položka bude mít alternativní názvy, zda již byla dříve někde publikována a jestli do ní bude uživatel chtít uložit více souborů. Pro typ *Image* je vhodné ponechat první dvě možnosti výběru na uživateli. Ale třetí možnost je (v tomto modelu) nadbytečná, protože položka bude obsahovat jeden obrázek, tedy jeden soubor, který bude uživatel vkládat. Proto jsou ponechány dotazy na první dvě otázky. Odpověď na třetí otázku je známa – uživatel bude vkládat pouze jeden soubor – proto bude nutné stránku změnit. Nejdříve vložíme na původní stránku kód, který zjistí typ a jméno kolekce. Tento kód vypadá následovně

```
<%@ page import="org.dspace.content.Collection" %>

<%
Collection collection = si.submission.getCollection();
int colType = collection.getColType();
String colTypeName =
              Collection.getColTypeName(colType);
%>
```

V proměnné `colType` je nyní uložen typ kolekce a jméno typu v proměnné `colTypeName`. Následuje příkaz `switch`, který oddělí původní kód systému DSpace a obslouží kód pro nový typ:

```
<% switch (colType) {
       default: {
%>

  [ Původní kód DSpace ]

<% break; }
   case 2: {
%>

<%@ include file="initial-questions-image.jsp" %>

<% break;} } %>
```

Zde je využito příjemné vlastnosti příkazu `switch`[3] – část `default` zajistí, že se bude, pokud je typ kolekce neznámý, provádět původní kód systému DSpace. Dále je zajištěno pomocí direktivy `include` vložení souboru s kódem, který obsluhuje typ *Image*. Jméno vkládaného souboru je v tomto případě zvoleno jako jméno originální stránky JSP doplněné řetězcem „`-image.jsp`". Tato konvence bude použita i u všech dalších souborů JSP. Na tomto místě je dobré připomenout, že celý kód editované stránky JSP typu *Image* je okomentován tak, aby formou tutoriálu provedl uživateleprogramátora celým nastavením. Protože se daný postup opakuje i u dalších stránek JSP, nebude již podrobně znovu zmiňován (ani komentáře v souborech nebudou tak bohaté).

Soubor `initial-questions-image.jsp` má již zřejmou strukturu – kód realizující získání odpovědí na příslušné otázky je stejný (případně může být pozměněn) jako kód v souboru `initial-questions.jsp`. Předání informace o tom, že bude vkládán pouze jeden soubor, se zařídí pomocí tagu

```
<input type="hidden"
       name="multiple_files" value="false">
```

Protože neodesíláme žádné nové informace, není v této fázi nutné dopisovat obsluhující kód do servletu **SubmitServlet**.

Ještě je vhodné diskutovat umístění příkazu `switch` v původním souboru JSP. Jeho umístění je voleno tak, aby se nemusela pro každý typ měnit celá stránka JSP, ale pouze její část, protože některé části kódu zůstávají konstantní nezávisle na typu (tlačítka pro odeslání požadavku atp.). Zvolená míra kódu, který příkaz `switch` obsluhuje, je taková, aby se musela měnit jen formulářová data. Tento způsob je vhodným kompromisem mezi kompletním zahrnutím celé stránky JSP na jedné straně a rozdrobeným kouskováním stránky původní (mnoha podmínkami na potřebných místech kódu) na straně druhé.

Po odeslání dat z výše popsané stránky JSP se řízení předává zpět servletu **SubmitServlet**, který po zpracování předaných informací a jejich vyhodnocení volá JSP stránku `JSP/submit/edit-metadata-1.jsp`.

9.4.2 Zpracování metadat

Vyplňování popisných metadat (v této části se budou pod pojmem metadata rozumět popisná metadata, nebude-li řečeno jinak) u vkládané po-

[3] Je potřeba poznamenat, že každá větev příkazu `switch` je samostatným blokem (uzavřena v { ... }) – nedochází tak k duplicitním definicím proměnných.

ložky je v systému DSpace rozděleno na dvě části. V první se vyplňují některé povinné údaje, jako je název položky apod. Obsloužení této fáze provádí JSP stránka `JSP/sumbit/edit_metadata-1.jsp`, která zpracované údaje opět odesílá servletu **SubmitServlet**, který je, v závislosti na tom v jaké fázi vkládání se požadavek uskutečnil, zpracuje.

Výchozí metadata systému DSpace jsou pro typ *Image* nevhodná, například vkládání identifikátoru ISBN, typu položky (Dublin Core typ, nikoli vnitřní typ DSpace), apod. Pro obrázek a fotografii budeme povinně vyžadovat pouze položku **Title** a položku **Publishing by**. Provedeme patřičné úpravy původního souboru se stránkou JSP (viz část 9.4.1) a připravíme si soubor `JSP/sumbit/edit_metadata-1-image.jsp`. U metadatových položek, které chceme ponechat stejné, jako v původním DSpace, původní kód pouze zkopírujeme (případně upravíme). Veškeré metadatové položky, u kterých nepotřebujeme, aby je uživatel vyplňoval (a jsou na původní stránce DSpace), pošleme skrytě (pomocí tagu `<input>` typu *hidden*) s hodnotou, kterou je prázdný řetězec. Toto je nutné provést, protože metody, které parametry zpracovávají obsahují pravděpodobně chybu, která způsobí, že při odeslání hodnoty `null` (tedy neodeslání příslušných parametrů) je vyvolána výjimka ***NullPointerException***.

Nová metadatová položka

Případ, kdy potřebujeme přidat, následně odeslat a zpracovat, vlastní položku je komplikovanější a vyžaduje přidání kódu do servletu **SubmitServlet**. Pro typ *Image* bude přidána metadatová položka, která určí, kdo danou fotografii či obrázek publikuje. Přes uživatelské rozhraní správce v systému DSpace (Dublin Core Registry) přidáme novou metadatovou položku *contributor.publisher*. Na část stránky JSP obsluhující typ *Image* vložíme příslušný HTML formulář, který bude odesílat parametr `contributor_publisher` (tento tvar kódování typů Dublin Core používá originální DSpace, proto je vhodné ho dodržovat). Tímto končí editace stránky JSP. Nyní je ale potřeba, aby odeslaná data parametru `contributor_publisher` zpracoval servlet **SubmitServlet**. V tomto servletu vyhledáme metodu, která zpracovává data odeslaná stránkou JSP. Metody jsou obvykle pojmenovány podle příslušné fáze, ve které je momentálně přidávání položky (dá se určit i podle jména dané stránky JSP, ne však vždy). Metoda zpracovávající data, zadaná uživatelem ve fázi editace základních metadat, se jmenuje `processEditMetadataOne` (konvenci `process<SubmissionStep>` dodržují téměř všechny metody v servletu **SubmitServlet**). Na vhodném místě v této metodě ošetříme parametr `contributor_publisher` vložením následujícího kódu[4]

```
if (subInfo.submission.getCollection().getColType()==2)
{
  String contributorPublisher =
      request.getParameter("contributor_publisher");
item.clearDC("contributor", "publisher", Item.ANY);
item.addDC("contributor", "publisher",
           "", contributorPublisher);
}
```

Povinná položka

Budeme-li chtít, aby daná položka byla povinná (aby uživatel musel vložit nějakou hodnotu), pak musíme udělat změny jak ve stránce JSP, která parametr odesílá, tak i v kódu servletu, který jej zpracovává. V případě první fáze vkládání metadat doplníme část kódu, který vypíše uživateli upozornění v případě, že se pokusil odeslat prázdný formulář. Konstrukce provedená v souboru `JSP/submit/edit-metadata-1-image.jsp` je následující

```
if (si.missing &&
    si.jumpToField != null &&
    "contributor\_publisher".equals(si.jumpToField)) {

  [ Vypiš upozornění ]

}
else {

  [ Vypiš standardní nápovědu formuláře ]

}
```

a po ní následuje samotný kód HTML s formulářem.

V servletu je situace komplikovanější. Narážíme totiž na ne zcela obecně zpracované části systému DSpace (ty se vyskytují častěji – někdy jsou označeny komentářem *FIXME: . . .*). V první řadě doplníme kód, který zpracovává parametr `contributor_publisher` o podmínku

[4]Není zcela lhostejné na jaké místo danou část kódu vložit – obecně však platí i dále, že vždy před jeden z posledních příkazů metody, kterým bývá `context.complete();`. Vložení kódu až za tento příkaz způsobí, že data nebudou zapsána do databáze!

```
if ( contributorPublisher == null ||
     contributorPublisher.equals("") ) {
               publisherMissing = true;
}
```

Tato podmínka nastavuje proměnnou `publisherMissing`. Tuto proměnnou musíme definovat mimo část kódu, který zpracovává náš parametr, a nastavit na hodnotu `false`. Nyní je potřeba opravit část kódu původního DSpace – ošetření stavu, kdy hodnota nebyla zadána. Ve stávající verzi systému DSpace je toto uděláno nevhodně – je možné buď velkou část servletu přeprogramovat nebo dopsat příslušné ošetření na správné místo kódu. V případě parametru `contributor_publisher` byla zvolena druhá možnost. Uživatel si může příslušnou část kódu vyhledat. Toto dočasné řešení je nevyhovující v případě, že bude přibývat povinných metadatových položek více, a do budoucna bude nutné zvážit, jak se s tímto problémem vypořádat.

Předvyplněná metadata

Může se stát, že některé metadatové položky jsou již předvyplněny (uživatel se vrací k rozpracovanému vkládání položky apod.). Je vhodné tato data načíst a zobrazit ve formuláři na stránce JSP. Načtení dat se provádí pomocí metody `getDC()` třídy **Item**. Tato třída vrací pole typu *DCValue[]*, v němž jsou uložena příslušná metadata dané položky. Například načtení metadat pro položku *contributor.publisher* se provede následujícím způsobem

```
DCValue[] cP
    = item.getDC("contributor", "publisher", Item.ANY);
String contributorPublisher
    = (cP.length > 0 ? cP[0].value : "");
```

Protože víme, že *contributor.publisher* může být nejvýše jeden, není nutné procházet celé pole.

9.4.3 Dokončení procesu vkládání

Postup u dalších souborů při úpravě procesu vkládání je obdobný, jako u předchozích popsaných. Proto již jen uvedu výčtem soubory, které realizují zbytek procesu vkládání, a které byly pro typ *Image* upraveny. Jsou to soubory

```
edit-metadata-2.jsp
choose-file.jsp
show-uploaded-file.jsp
review.jsp
```

U souboru `edit-metadata-2.jsp` byla změněna metadata a byla přidána další nová metadata týkající se fotografií a obrázků (rozlišení, autor fotografie, ...). Soubor `review.jsp` byl modifikován tak, aby se zobrazovaly pouze informace relevantní typu *Image*.

Soubory, které realizují vložení obrázku uživatele (upload), byly předělány jen tak, aby zobrazovaly správné popisky. Bohužel se ukázalo, že tato část DSpace je ještě nedodělaná, a celé rozhraní stahování souborů působí, jako by autoři zatím nevěděli, jak rozumně celý potenciál navržené architektury využít – příliš často se objevují poznámky typu *FIXME: ...* – týká se to zejména možnosti, kdy uživatel může vkládat více souborů do jedné položky. Zde je dobré zvážit, zda celý kód přepisovat nebo vyčkat na další verzi systému DSpace. Závěrem nutno poznamenat, že je samozřejmě možné provádět další modifikace, často se však nelze vyhnout přepisování původního kódu systému DSpace, což vede k větší pracnosti při přechodu na nové verze.

9.4.4 Zobrazování položky dle typu

V navrženém modelu požadujeme, aby se položka typu *Image* zobrazovala způsobem odlišným od typického zobrazování položky. Zobrazování položky řeší JSP stránka `JSP/display-item.jsp`. Samotné zobrazení položky je však řešeno v programátorem definovaném tagu JSP[5]. Je proto vhodné držet se zavedeného modelu a napsat vlastní tag pro zobrazení položky daného typu. Soubor `JSP/display-item.jsp` pak vypadá podobně jako ostatní změněné soubory JSP (tedy obsahuje příslušný `switch`), ale místo tagu pro vkládání obsahuje nový definovaný JSP tag. Pro typ *Image* vypadá tento tag následovně

```
<dspace:image item="<%= item %>"
              collections="<%= collections %>"
              style="<%= displayStyle %>" />
```

[5]Uživatelem definované tagy JSP jsou technologií, která umožňuje psát vlastní tagy jazyka JSP. Využívá se při tom pravidel, které zavádí technologie Java Beans. Uživatel v XML souboru definuje popis tagu, jeho parametrů a místo, kde je uložena třída, ve kterém si uživatel nový tag v podstatě naprogramuje.

Jedná se o zcela nový tag, jehož parametry určují položku, dále kolekce, ve kterých je obsažena, a styl zobrazení (zkrácený nebo plný záznam). Tento tag je třeba nadefinovat v souboru `JSP/WEB-INF/dspace-tags.tld`. V tomto souboru především určíme, kde se bude nacházet třída s definovaným tagem. Třída tagu bude mít název **ImageTag** a uložena bude v balíku *cz.muni.ics.jsptag*.

Třída **ImageTag** má několik povinných metod a především metodu `renderDefault()`. V této metodě určíme, která metadata se budou u dané položky zobrazovat. Autoři systému DSpace připravili kód velmi kvalitně, takže lze pouze nadefinovat, které metadatové položky se mají zobrazovat. V případě typu *Image* se oproti obecnému typu DSpace některé metadatové položky odstraní a především se přidá *contributor.publisher* – do metody vložíme

```
fields.add(new String[]
          {"Published by", "contributor", "publisher"}
);
```

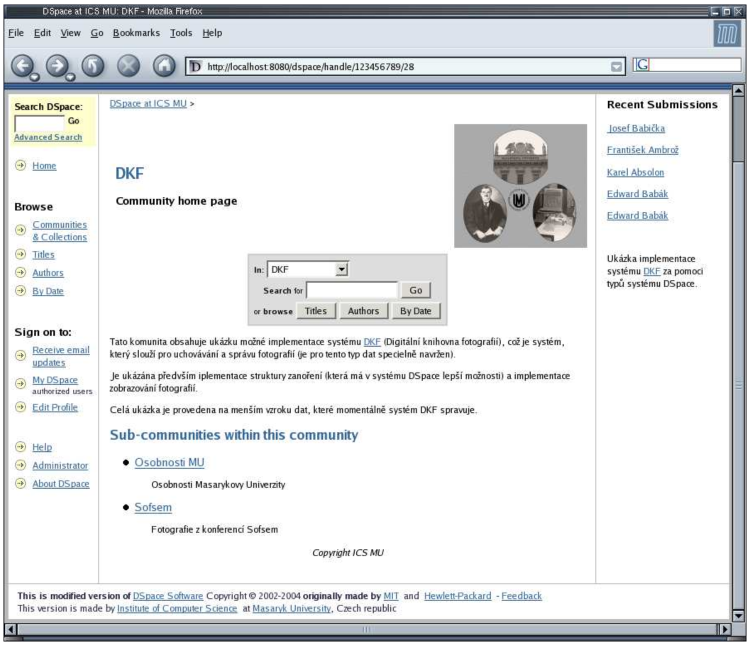
Druhou důležitou metodou je `listBitstreams()`. V ní se určuje, jak se zobrazí příslušná položka. V případě typu *Image* zjistíme, zda-li je vygenerován náhled obrázku a pokud ano, zobrazíme tento náhled (náhled je aktivní a pokud na něj uživatel klikne, otevře se mu původní obrázek). Pokud není náhled k dispozici, načteme obrázek a zobrazíme ho zmenšený tak, aby jeho šířka byla 500 bodů (obrázek je stejně jako u náhledu aktivní). V kódu třídy jsou uvedeny komentáře, které podrobně provedou důležitými pasážemi zdrojových textů.

9.4.5 Zobrazování kolekce dle typu

U některých kolekcí může být žádoucí, aby bylo zobrazování jejich obsahu závislé na typu metadat. V případě, že kolekce má typ *Image*, budeme požadovat, aby se při otevření kolekce zobrazily náhledy všech fotografií, které obsahuje, spolu s jejich popisy. Stránka JSP, která se stará o zobrazování kolekcí, je uložena v souboru `JSP/collection-home.jsp`. Za použití volání již hotových metod lze získat všechny položky, které tato kolekce obsahuje. Konstrukce uvedená spolu s komentáři v souboru `JSP/collection-home-image.jsp`, provede výpis všech fotografií s jejich náhledy. Z důvodu jednotnosti kódu jsou všechny potřebné třídy importovány již v souboru `JSP/collection-home.jsp`.

9.4.6 Praktická ukázka typu *Image*

Pro praktické ukázání zavedeného typu *Image* jsem zvolil přenesení malé části systému DKF do DSpace. Protože systém DKF v současnosti nepodporuje export do žádného standardního formátu, byly ukázkové kolekce a složky vytvořeny ručním vložením. Počet vložených fotografií proto nepřesahuje tři desítky. Jako ukázka praktické realizace však postačuje.



Obrázek 9.1: Ukázka implementace typu *Image* (DKF).

Pokud by mělo dojít ke kompletnímu přenosu systému DKF do systému DSpace, musel by se vyvinout filtr pro transformaci exportního formátu systému DKF na formát METS (formát, který podporuje systém DSpace), a zároveň vyřešit přenesení metadat, která jsou specifická pro systém DKF (toto není problém z technického hlediska, je to problém spíše z pohledu otázky, na jaké metadatové položky systému DSpace mapovat informace ze systému DKF, jak vyřešit problematiku práv a přístupu apod.).

9.5 Využití typů komunit – časopisy

V této části práce bude ukázáno využití *typů komunit*. Současná verze systému DSpace neumožňuje zanořování kolekcí a zanořovat lze pouze komunity. Nemožnost zanořovat kolekce lze obejít tak, že se komunitám přiřadí speciální sémantika. Pak lze budovat libovolné stromové struktury, kde listy budou v podstatě kolekce. Toto řešení není úplně ideální, je však jediné možné v případě, že nechceme přepisovat velkou část celého systému (včetně jádra). Zavedením typů komunit tak získáváme v manipulaci s nimi nové možnosti.

Typy komunit lze využít například i u archívu obrázků typu *Image* – vystačíme si však i s původními (netypovanými) komunitami (i když se dá předpokládat, že při dalším rozvoji se typování komunit uživatel nevyhne). Kritičtějším typem dokumentů, u kterého se pravděpodobně bez typování komunit neobejdeme, jsou například časopisy. Zde si lze živě představit případ, kdy instituce vydává časopisy či sborníky. Jedno z možných řešení, jak přehledně tyto časopisy (vydávané elektronicky, jednotlivé články časopisu například ve formátu PDF) ukládat v DSpace, je vytvořit si vhodnou strukturu. Na jejím vrcholu může být komunita nazvaná Časopisy – může to být klasická komunita DSpace nebo pro drobné úpravy při zobrazování můžeme již zde sáhnout k typům. Tato komunita bude obsahovat komunity s názvy časopisů, které daná instituce vydává (tyto komunity je již vhodné typovat kvůli zobrazení správných informací o daném časopisu). Komunita každého takového časopisu bude obsahovat komunity reprezentující jednotlivé ročníky daného časopisu. Kolekcí pak může být konkrétní číslo časopisu v daném ročníku, položky budou jednotlivé články. Takto navržená struktura je na jednu stranu komplikovanější, dává však do budoucna velké možnosti v ukládání článků časopisu, které mohou být pořizovány v různém formátu (například jednotlivé články jsou naskenované obrázky po jednotlivých stránkách apod.)[6].

Aby bylo možné dosáhnout výše popsaného modelu v praxi, je třeba nejdříve zavést nové typy *Magazines* a *Magazines-Year* (postup zavedení nového typu je uveden v části 9.3). Typu *Magazines* bude přidělen identifikátor **1**, typu *Magazines-Year* identifikátor **3**. Typ *Magazines* zavedeme jak pro komunity, tak pro kolekce. Typ *Magazines-Year* zavedeme pouze pro kolekce a bude reprezentovat ročník časopisu.

Pokud je již typ vytvořen, lze přistoupit k modifikaci metadat a modifikaci zobrazování komunit.

[6]Bohužel současná verze DSpace má značná omezení v ukládání samotných souborů a jejich dělení do zvláštních „přihrádek" (v DSpace zvaných Bundles). Napevno „zadrátované" tři druhy těchto Bundles jsou zcela nedostačující a celé pasáže kódu týkající se jejich využití jsou naprogramovány provizorně, což autoři sami přiznávají – zdá se, že ještě není definitivně rozhodnuto, jak se s Bundles v budoucnu naloží. Provést přeprogramování by znamenalo značnou námahu a navíc je z poznámek v samotném kódu zřejmé, že v budoucích verzích systému DSpace bude kompletní podpora Bundles zahrnuta.

9.5.1 Modifikace metadat

Změna metadat komunit, ale stejně tak změna metadat kolekcí, je složitější než změna metadat u položek. Položky mají metadata uložena ve formátu Dublin Core a principiálně není jejich rozsah omezen (uživatel si smí přidávat vlastní metadatové položky dle libosti). Metadata komunit a kolekcí jsou však pevně uložena v tabulce v databázi. Chceme-li přidávat metadata, je nutné rozšířit příslušnou tabulku (***community*** nebo ***collection***) o nový sloupec, ve kterém bude metadatová informace uložena.

Jednodušší scénář se nabízí v případě, že nechceme využívat některá metadata, která kolekce nebo komunita nabízí – pak je možné upravit pouze stránky JSP a formuláře pro odesílání dat a při výpisu dat z databáze pouze tato metadata vhodně interpretovat. To znamená, že pokud nepotřebujeme využívat například metadata, která jsou ukládána v sloupci *copyright_text* tabulky ***community***, ale potřebujeme ukládat například nějaký identifikátor komunity, můžeme tento identifikátor (jeho textovou podobu) uložit do databáze do sloupce *copyright_text*[7]. Komunita určitého typu pak bude data z tohoto sloupce interpretovat jinak, než to provádí originální DSpace. Postup při úpravě příslušné stránky JSP je podobný jako při úpravě stránek pro zobrazování kolekce nebo položky (viz část 9.4.4) – zjistíme typ kolekce nebo komunity, připravíme příslušný `switch` a ošetříme jednotlivé jeho větve. Metadata pro kolekci nebo komunitu získáme voláním metody (pro kolekci i komunitu) `getMetadata()`, jejímž jediným argumentem je textový řetězec, který určuje, která metadata chceme. Například získání metadat ze sloupce *copyright_text* tabulky ***community*** pro komunitu `comm` (třída **Community**) se provede následujícím příkazem

```
String copyright = comm.getMetadata(”copyright_text”);
```

Řetězec v proměnné `copyright` pak můžeme zobrazit na stránce JSP libovolným způsobem.

Pokud nám však dostupná data nedostačují, musíme přidat sloupec do příslušné tabulky v databázi SQL, přidat na stránky JSP formuláře, které zajistí odeslání informací servletu, a v servletu zajistit provedení metod, které tyto hodnoty uloží do databáze. Všechny tyto činnosti ukážeme podrobně na konkrétním příkladu – rozšíření tabulky komunit o sloupec *metadata_id*. Tento sloupec bude univerzální a bude uchovávat textové řetězce. V případě

[7] Je nutné zkontrolovat typ sloupce v databázi, aby nedocházelo k přetečení. Například obsah sloupce *short_description* je v databázi omezen na 512 znaků a ukládání delších textů může vést k problémům.

typu *Magazines* do něj bude ukládán identifikátor ISSN (International Standard Serial Number) [78], který je používán pro jednoznačnou identifikaci časopisů a jiných periodik. Při manipulaci se sloupci pro ukládání textových řetězců lze s výhodou použít univerzálních metod `setMetadata()` a `getMetadata()` (obě jsou dostupné jak ze třídy **Community**, tak ze třídy **Collection**, balík *org.dspace.content*). V případě, že je vhodnější ukládat metadata v jiném než textovém typu, je nutné dopsat i příslušné metody do dané třídy (viz například metody **`setComType()`** nebo **`getComType()`** v části 9.2.1).

Pokud již máme vložen sloupec *metadata_id* v tabulce, můžeme přikročit k úpravě JSP stránky `edit-community.jsp`. Tato stránka je uložena v adresáři `JSP/dspace-admin/`. Standardní konstrukcí (`switch` a `include`) oddělíme zpracování typů. Metadata pro typ *Magazines* obslouží JSP stránka v souboru `edit-community-magazines.jsp`, který je uložen v adresáři `JSP/dspace-admin/`. Odeslání informací o ISSN provedeme zařazením příslušného formuláře, například:

```
<input type="text" name="metadata_id"
       value="<%= metadataId %>" size=15>
```

Formulář používá jako výchozí hodnotu obsah proměnné `metadataId`. Tuto proměnnou získáme vložením kódu

```
metadataId = community.getMetadata("metadata_id");
```

do stránky JSP. Pokud užijeme typu, pro jehož zpracování nelze použít metody `[get|set]Metadata()`, pak se musí do třídy **Community** dopsat metoda, která zajistí uložení nové hodnoty do databáze, a metoda, která zajistí vrácení této hodnoty.

Posledním krokem je editace servletu **EditCommunitiesServlet** v balíku *org.dspace.app.webui.servlet.admin*. V něm je nutné zajistit zpracování přijatých dat. Do metody `processConfirmEditCommunity()` se přidá následující kód[8]:

```
String metadataId =
            request.getParameter("metadata_id");
if (metadataId == null) { metadataId = ""; }
community.setMetadata("name", metadataId);
```

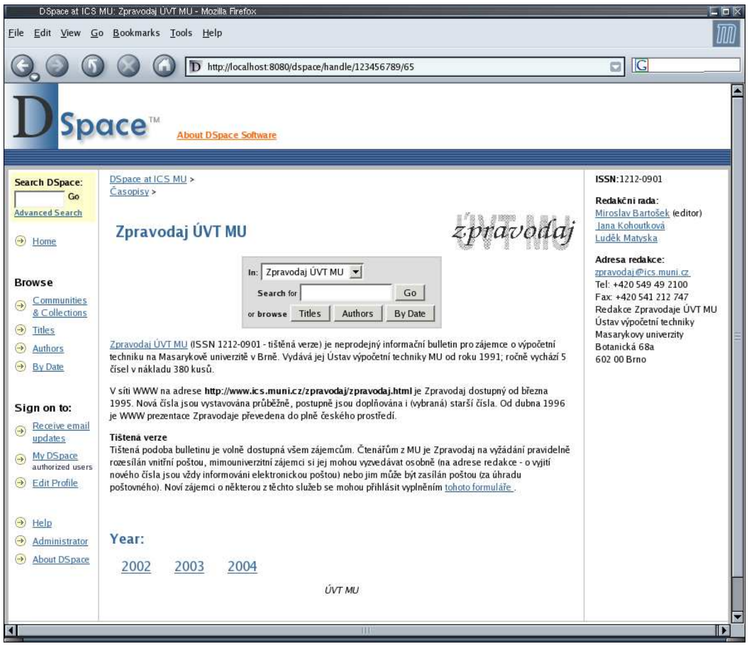
[8]Tento kód musí být před příkazem `context.complete();`, který provádí zapsání do databáze.

Tento kód zajistí, že data budou zpracována a následně uložena do databáze. Tímto je zpracování metadat u komunity ukončeno. Podobně lze postupovat pro libovolnou jinou komunitu a typ.

9.5.2 Zobrazování komunity

Pro zobrazování komunity určitého typu platí tatáž pravidla, jako pro zobrazování kolekcí a položek. Zobrazování stránky komunity zajišťuje JSP stránka uložená v souboru `JSP/community-home.jsp`. Její úpravou lze docílit potřebného vzhledu pro daný typ komunity.

9.5.3 Praktická ukázka typu *Magazines*



Obrázek 9.2: Ukázka implementace typu *Magazines* (Zpravodaj ÚVT MU).

Typ *Images* byl prakticky použit pro ukládání Zpravodajů Ústavu výpočetní techniky Masarykovy univerzity [79]. Byly použity navržené typy a struktura. Archiv zpravodajů není kompletní, na ukázku bylo do systému DSpace uloženo pouze několik zpravodajů. V tomto případě však bylo především cílem ukázat budování struktur a speciálních komunit.

9.6 Shrnutí

Praktická implementace typů ukázala možnosti systému DSpace. Zavedená podpora typů dovoluje specializovat systém pro různé typy dat – v současném stavu řešení je však nutná znalost jazyka Java a programování stránek JSP. Praktické implementace konkrétních jednoduchých typů však i méně zkušený programátor s pomocí poskytnutého návodu snadno zvládne. Pro složitější data je pak nutná hlubší znalost systému a dobré porozumění programování webových aplikací v jazyce Java.

Kapitola 10

Závěr

Cílem práce bylo prozkoumat současné volně dostupné systémy pro podporu digitálních knihoven a repozitářů, provést jejich analýzu a na základě této analýzy vybrat systém, který by byl vhodný jako digitální repozitář pro potřeby Masarykovy univerzity a na příkladech ukázat jeho praktické nasazení.

Podrobně byly analyzovány čtyři volně dostupné systémy – Fedora, RIB, EPrints a DSpace. Provedená analýza ukázala, že oblast praktických repozitářů a různých typů software pro digitální knihovny je velmi mladá a stále se (velmi dynamicky) rozvíjí.

Na základě získaných informací byl jako perspektivní a vhodný pro digitální knihovnu Masarykovy univerzity vybrán systém DSpace. Systém DSpace přesvědčil především svojí celkovou koncepcí – ať již architekturou, která je kvalitně navržená a dovoluje další rozvoj, podporou standardů a okamžitou praktickou použitelností, nebo i celkovým přístupem vývojářů, který slibuje další podporu a rozvoj systému DSpace i do budoucna.

I přes všechny své klady má současná verze systému DSpace některé nedostatky – především díky své univerzálnosti dovoluje pouze jednotnou prezentaci a jednotné zacházení se všemi typy digitálních objektů. Praktické požadavky uživatelů jsou však směrovány na specializaci práce s různými typy digitálních objektů při zachování univerzálnosti celého systému. Proto byla v rámci této práce do systému DSpace navrhnuta a úspěšně implementována podpora pro typy digitálních objektů. Pomocí této jednoduché ale silné nadstavby lze docílit specializace celého systému, aniž by byla snížena jeho schopnost uchovávat jakékoli typy digitálních objektů. Na praktických příkladech bylo předvedeno využití naprogramované podpory typů. Zároveň s praktickým předvedením byl podrobně popsán i návod, jak postupovat při vytváření vlastních speciálních typů.

Díky časové náročnosti, která přímo vyplynula z nedostatků systému DSpace, se nepodařilo podporu typů dotáhnout do ideálního stavu a stále jsou zde patrné nedostatky – zejména v oblasti ukládání souborů a následné manipulace s nimi. Nebylo rovněž implementováno uživatelské rozhraní, které by umožnilo snadnější přidávání nových typů. Celá praktická implementace však především ukázala i současné nedostatky a možnosti systému DSpace a umožnila získat cenné zkušenosti související s praktickou realizací systémů digitálních knihoven, které budou využity při budoucím vývoji (nejen) systému DSpace. Bezprotředně se jeví jako vhodný postup prosazení implementované podpory typů i do oficiální distribuce DSpace.

Vývoj systémů pro digitální knihovny je náročnou a dlouhodobou prací. Doufám, že tato diplomová práce se stane dobrým výchozím bodem v celé etapě praktického zavádění digitálních knihoven a repozitářů a získané poznatky se ukáží užitečnými i v delším časovém horizontu.

Literatura

[1] Miroslav Bartošek. *Digitální knihovny*. Proc. Datakon 2001. http://www.ics.muni.cz/mba/dl-datakon01.pdf

[2] Jaroslav Pokorný. *Digitální knihovny: Principy a problémy*. Automatizace knihovnických procesů (8), duben 2001. http://knihovny.cvut.cz/akp/clanky/03.pdf

[3] Miroslav Bartošek. Prezentace ke přednášce PV070 Digitální knihovny, Fakulta informatiky, Masarykova univerzita. http://www.ics.muni.cz/mba/

[4] Dublin Core. http://dublincore.org/

[5] Library of Congress, Core Metadata Elements. http://www.loc.gov/standards/metadata.html

[6] RDF, Resource Description Framework. http://www.w3.org/RDF/

[7] METS, Metadata Encoding & Transmission Standard. http://www.loc.gov/standards/mets/.

[8] URIs, URLs, and URNs: Clarifications and Recommendations 1.0. http://www.w3.org/TR/uri-clarification/

[9] PURL, Persistent URL. http://purl.oclc.org/

[10] CNRI, Corporation for National Research Initiatives. Handle System. http://www.handle.net/

[11] DOI, The Digital Object Identifier System. http://www.doi.org/

[12] ARK, Archival Resource Key. http://www.cdlib.org

[13] OAI-PMH, Open Archives Initiative – Protocol for Metadata Harvesting. http://www.openarchives.org/OAI/openarchivesprotocol.html

[14] Z39.50. `http://www.loc.gov/z3950/agency/`

[15] OpenURL. `http://library.caltech.edu/openurl/`

[16] Robert Kahn, Robert Wilensky. *A Framework for Distributed Digital Objects Services*. May 13, 1995. cnri.dlib/tn95-01. `http://www.cnri.reston.va.us/home/cstr/arch/k-w.html`

[17] William Y. Arms. *Key Concepts in the Architecture of the Digital Library*. July 1995. cnri.dlib/july95-arms `http://www.dlib.org/dlib/July95/07arms.html`

[18] DNS, Domain Name System. `http://www.dns.net/dnsrd/`

[19] Vannevar Bush. *As We May Think*. Atlantic Monthly, July 1945 `http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm`

[20] Ditital Library Initiative. `http://www.dli2.nsf.gov/`

[21] Fedora, Flexible Extensible Digital Object and Repository Architecture. `http://www.fedora.info/`

[22] Mozilla Public Licence. `http://www.mozilla.org/MPL/MPL-1.1.txt`

[23] Sandra Payette, Carl Lagoze. *Flexible and Extensible Digital Object and Repository Architecture.* `http://www.cs.cornell.edu/payette/papers/ECDL98/FEDORA.html`

[24] CORBA, Common Object Request Broker Architecture `http://www.corba.org`

[25] Web Services. `http://www.w3.org/2002/ws/`

[26] Web Services Description Language. `http://www.w3.org/TR/wsdl`

[27] SOAP, Simple Object Access Protocol. `http://www.w3.org/TR/soap12-part1/`

[28] Java. `http://java.sun.com/`

[29] MySQL Database Machine `http://www.mysql.com/`

[30] Apache Ant. `http://ant.apache.org/`

[31] Jakarta Tomcat. http://jakarta.apache.org/tomcat/

[32] MIME Media Types. http://www.iana.org/assignments/media-types/

[33] Marty Hall. *Java servlety a stránky JSP*. Neocortex, 2001. ISBN 80-86330-06-0

[34] FOP, Formatting Objects Processor. http://xml.apache.org/fop/

[35] XSL-FO, XSL Formatting Objects. http://www.w3.org/Style/XSL/

[36] ImageJ, Image Processing and Analysis in Java. http://rsb.info.nih.gov/ij/

[37] The Fedora Project, An Open-source Digital Object Repository Management System http://www.dlib.org/dlib/april03/staples/04staples.html

[38] RIB, Repository In a Box. http://icl.cs.utk.edu/rib/

[39] IEEE, The Institute of Electrical and Electronics Engineers. http://www.ieee.org/

[40] NHSE, National HPCC Software Exchange. http://www.nhse.org/

[41] DTD Tutorial. http://www.w3schools.com/dtd/default.asp

[42] XML DTD - An Introduction to XML Document Type Definitions. http://www.xmlfiles.com/dtd/

[43] RIB API, RIB Application Programmer's Interface http://icl.cs.utk.edu/rib/api/RIBAPI.html

[44] EPrints. http://software.eprints.org/

[45] GNU is Not Unix Organization. http://www.gnu.org

[46] EPrints 2 Relational Diagram. http://software.eprints.org/docs/php/structure.php

[47] DSpace, Digital Archive Project. http://dspace.org

[48] BSD licence. http://www.opensource.org/licenses/bsd-license.php

[49] DSpace Project. MIT Faculty Newsletter, Vol. XII No. 4. April/May 2000 http://dspace.org/news/articles/dspace-project.html

[50] DSpace Project Hits Milestone: Early Adopter Phase Begins. http://libraries.mit.edu/about/news/early-dspace.html

[51] SourceForge. www.sf.net

[52] Jakarta Lucene Search Engine. http://jakarta.apache.org/lucene/docs/index.html

[53] OAICat Open Source Software. http://www.oclc.org/research/software/oai/cat.htm

[54] The Tapir for DSpace. http://www.thesesalive.ac.uk/dsp_home.shtml

[55] Theses Alive!. http://www.thesesalive.ac.uk/

[56] JISC, Joint Information Systems Committee. http://www.jisc.ac.uk/

[57] SHERPA, Securing a Hybrid Environment for Research Preservation and Access. http://www.sherpa.ac.uk/

[58] Google http://www.google.com

[59] DSpace@Cambridge. http://www.dspace.cam.ac.uk/

[60] ALADIN, Accs Libre aux Archives du Dépôt Institutionnel Numérique de la Maison des Sciences de l'Homme-Alpes. https://dspace.msh-alpes.prd.fr/index.jsp

[61] DSpace at Cornell University. http://dspace.library.cornell.edu/index.jsp

[62] DSpace at The Australian National University. http://dspace.anu.edu.au/

[63] DSpace a Parma. http://dspace-unipr.cilea.it:8080/index.jsp

[64] Fedora Testbeds. http://www.fedora.info/testbed.shtml

[65] Unix Guide. http://www.unixguide.net/

[66] DKF, Digitální knihovna fotografií na ÚVT MU. http://dkf.ics.muni.cz/

[67] Bricolage. http://www.bricolage.cc/

[68] Zope. http://zope.org/

[69] Plone. http://plone.org/

[70] PHP Nuke. http://phpnuke.org/

[71] Postnuke. http://www.postnuke.com/

[72] Midgard Project. http://www.midgard-project.org/

[73] OpenCMS. http://www.opencms.org/opencms/en/

[74] TikiWiki, Tiki CMS/Groupware. http://tikiwiki.org/tiki-index.php

[75] CDSware, CERN Document Server Software. http://cdsware.cern.ch/

[76] phpWebThings. http://www.phpdbform.com/

[77] DSpace System Documentation: Installation. http://dspace.org/technology/system-docs/install.html

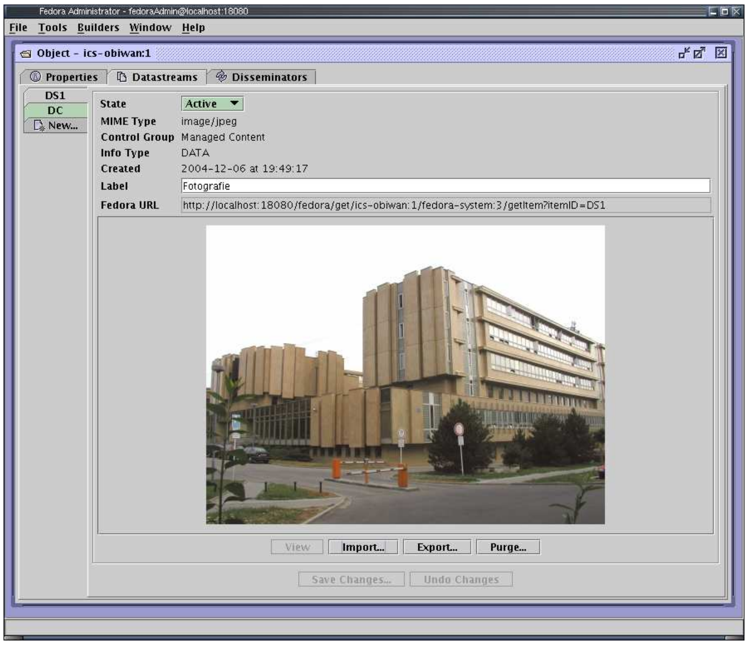
[78] ISSN, International Standard Serial Number. http://www.issn.org/

[79] Zpravodaj ÚVT MU http://www.ics.muni.cz/zpravodaj/

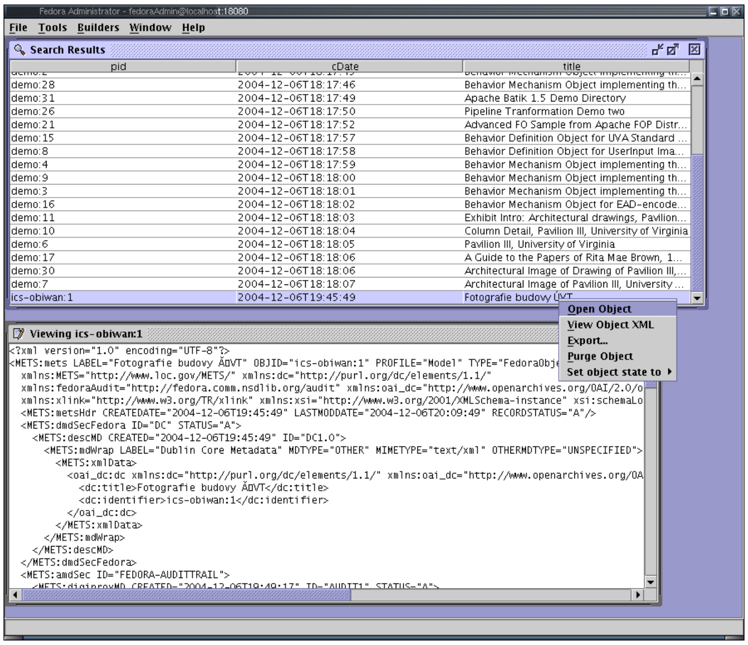
Příloha A

Ukázky z testovaných systémů

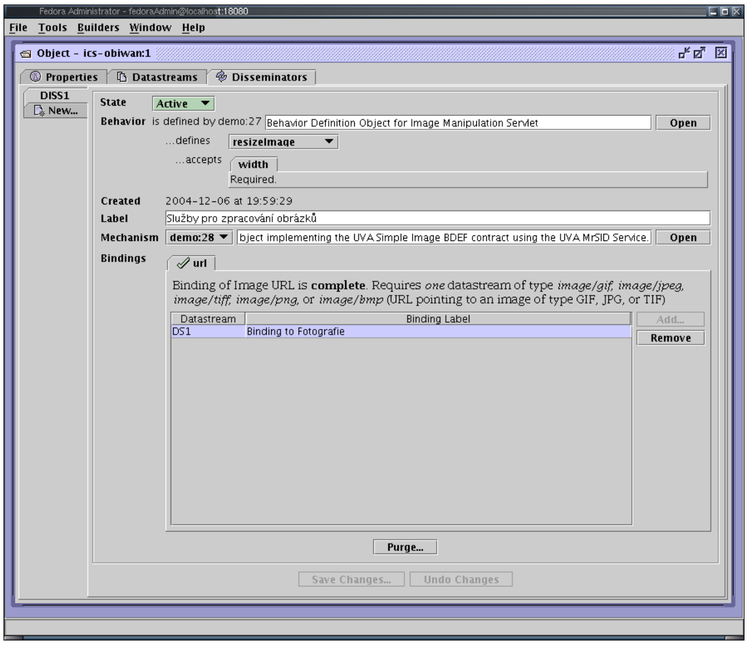
A.1 Fedora



Obrázek A.1: Fedora Administrative Client – práce s objektem.

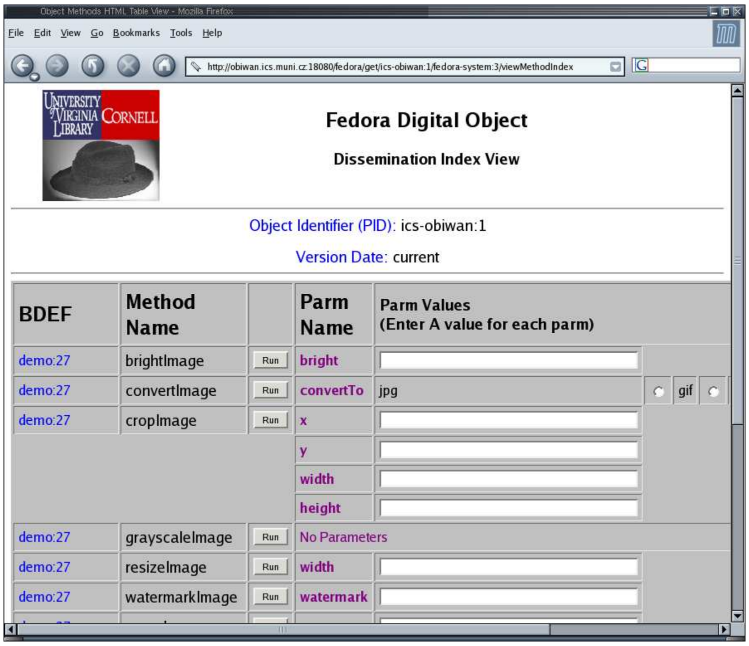


Obrázek A.2: Fedora Administrative Client – procházení repozitáře.



Obrázek A.3: Fedora Administrative Client – nastavení diseminátorů.

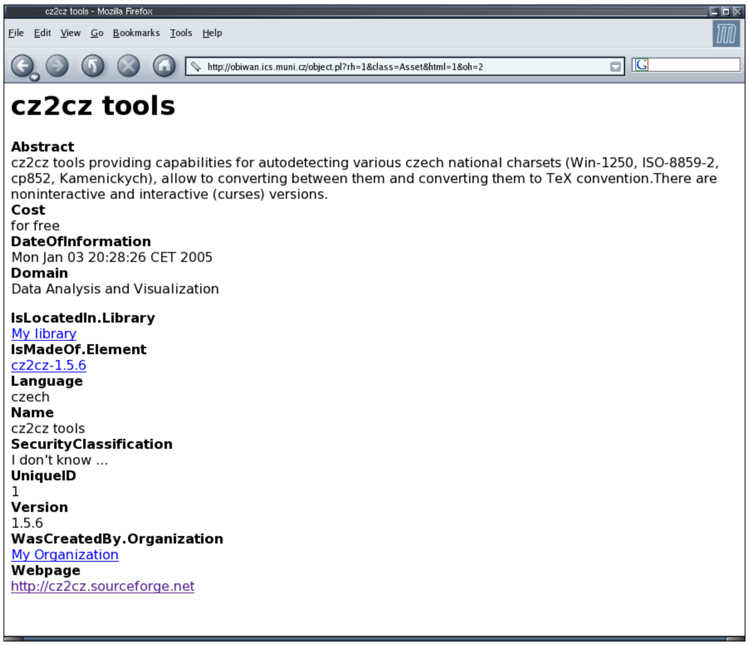
Obrázek A.4: Webové rozhraní – prohlížení objektu.



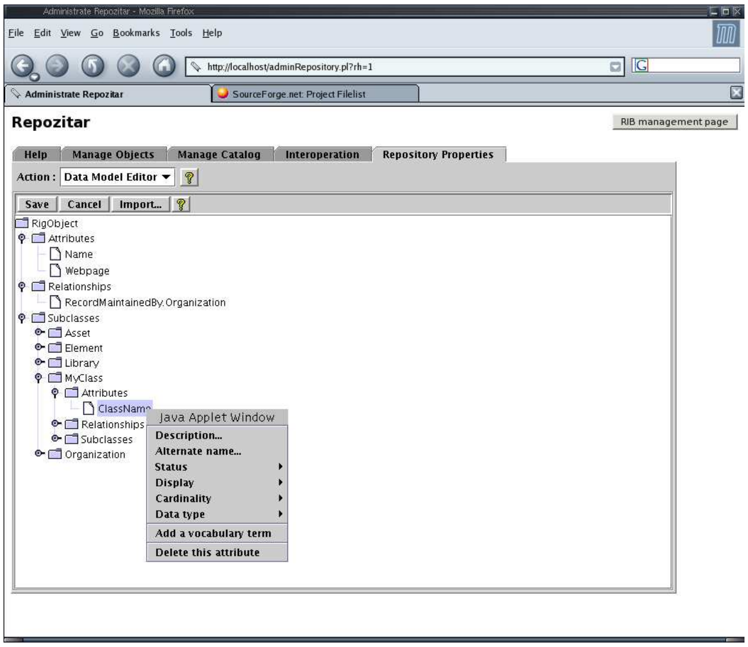
Obrázek A.5: Webové rozhraní – diseminátory objektu.

A.2 RIB

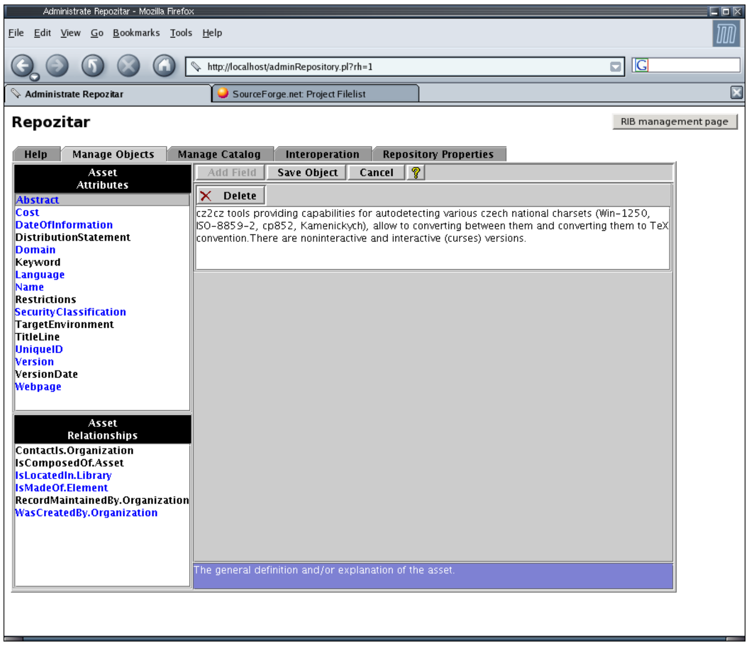
Obrázek A.6: Procházení repozitářem (katalog).



Obrázek A.7: Prohlížení objektu v katalogu.

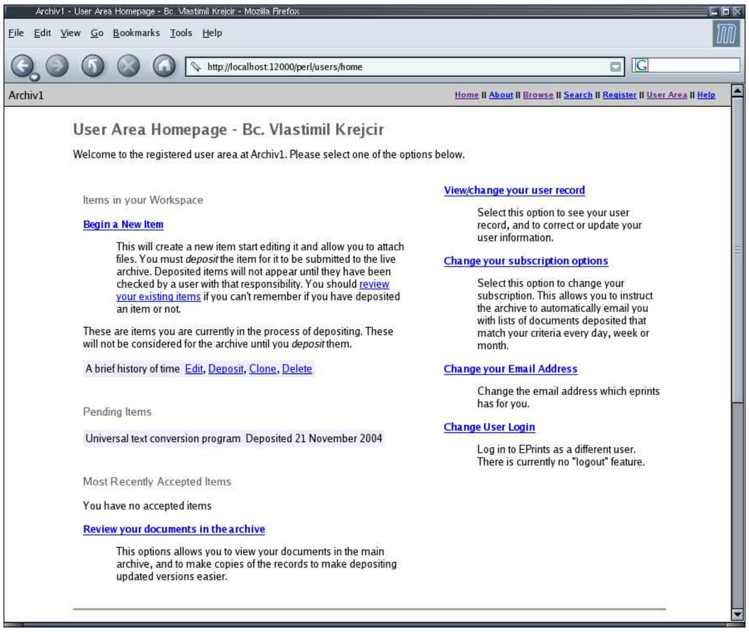


Obrázek A.8: Data model editor.



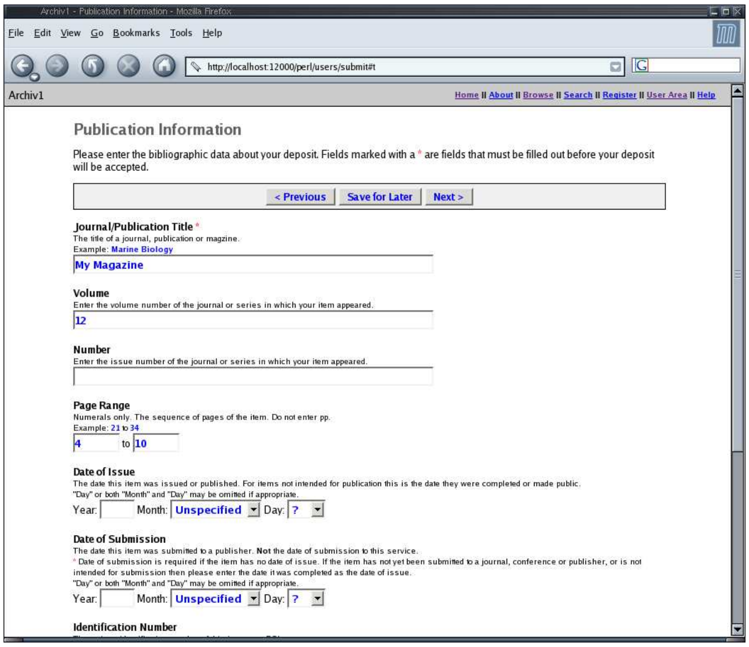
Obrázek A.9: Editování objektu.

A.3 EPrints

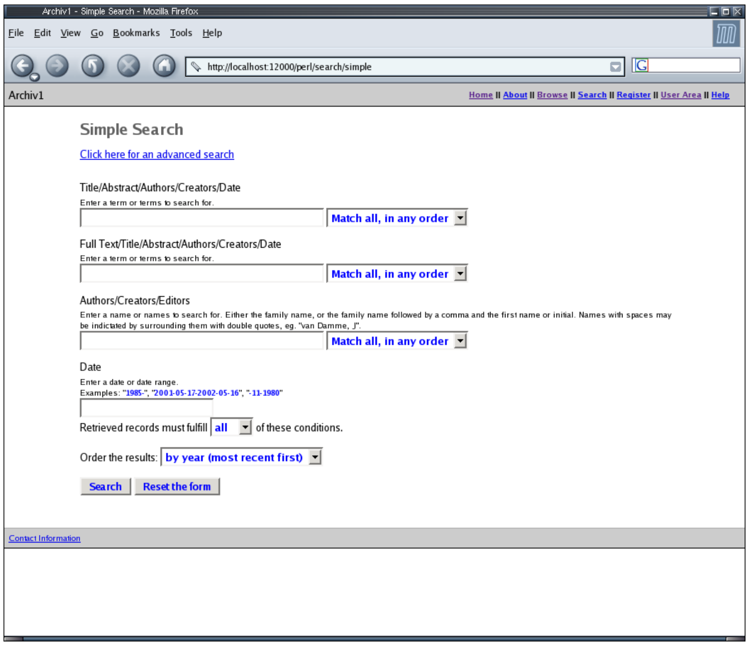


Obrázek A.10: Stránka přihlášeného uživatele.

Obrázek A.11: Prohlížení položky v archivu.

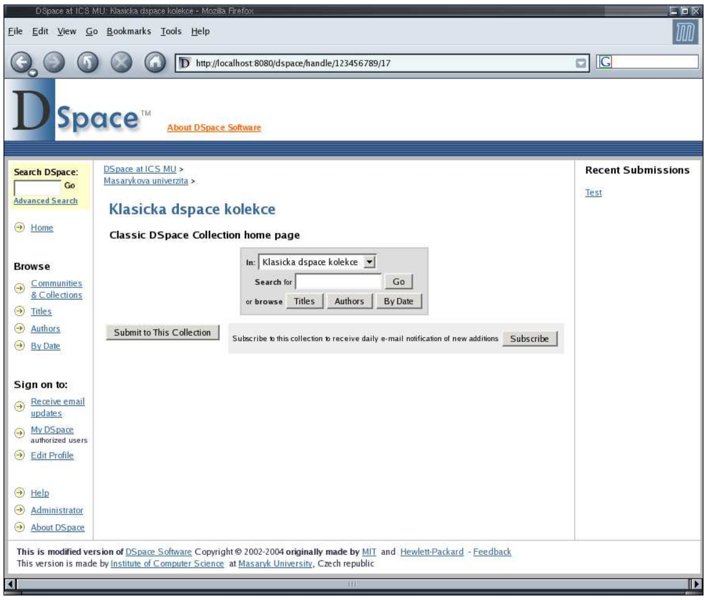


Obrázek A.12: Vkládání položky – fáze vyplňování popisných metadat.

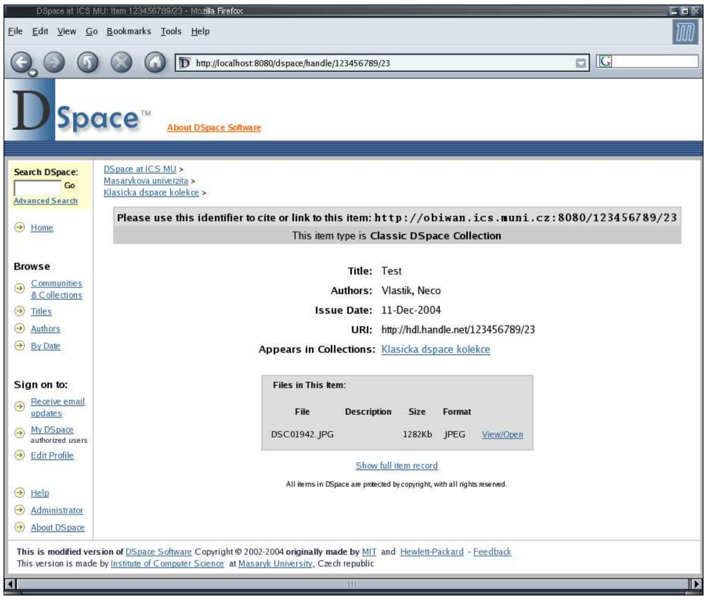


Obrázek A.13: Vyhledávání.

A.4 DSpace

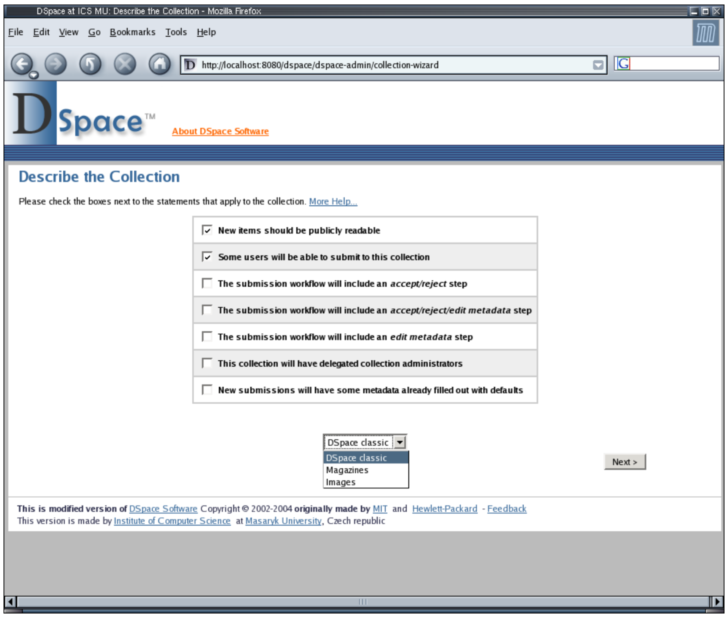


Obrázek A.14: Klasická kolekce (standardní DSpace).



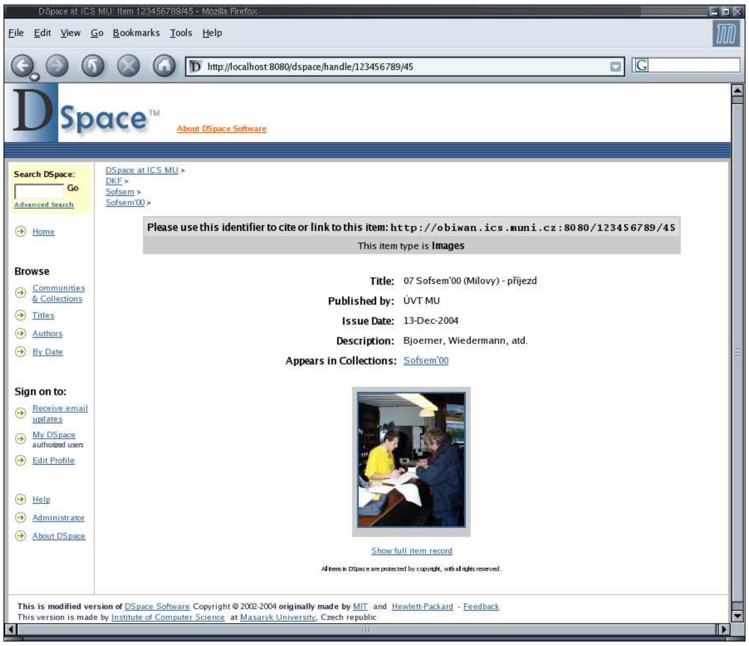
Obrázek A.15: Prohlížení položky (standardní DSpace).

Obrázek A.16: Úvodní stránka upraveného systému DSpace.

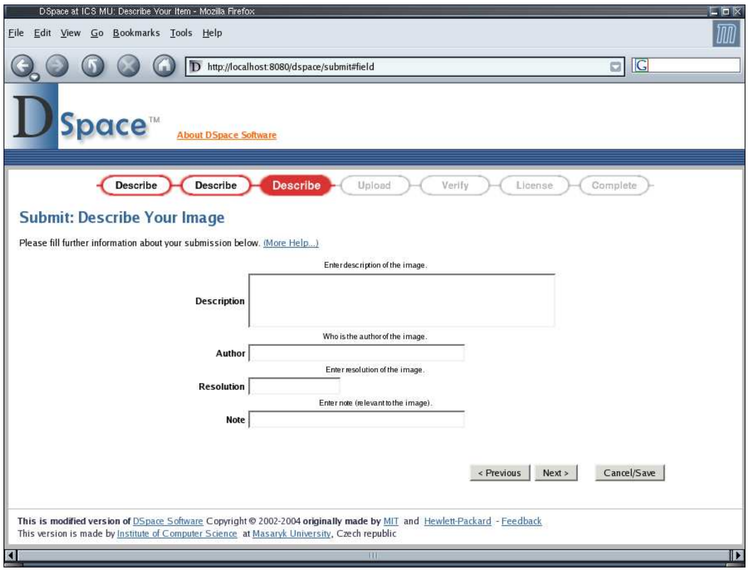


Obrázek A.17: Vytváření kolekce, výběr jejího typu.

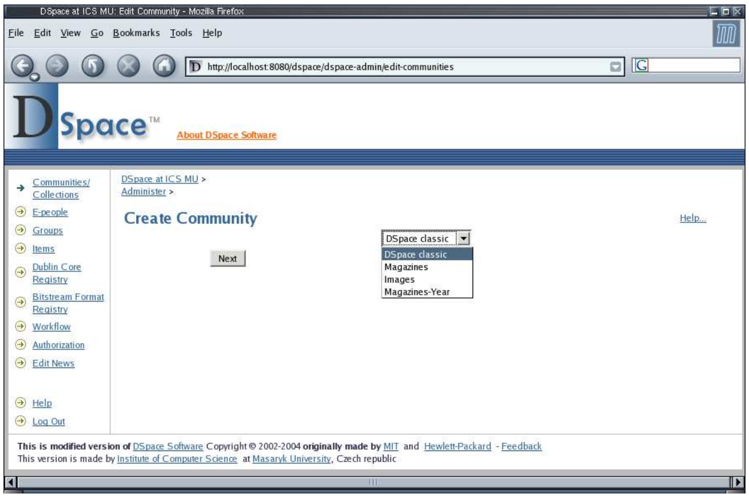
Obrázek A.18: Prohlížení kolekce typu *Image*.



Obrázek A.19: Prohlížení položky typu *Image*.



Obrázek A.20: Přidávání nové položky typu *Image* – vkládání metadat.



Obrázek A.21: Vytváření komunity, výběr jejího typu.