Małgorzata Janiak
Instytut Informacji Naukowej i Bibliotekoznawstwa
Uniwersytet Jagielloński
malgorzata.janiak@uj.edu.pl

OPROGRAMOWANIE FLOSS W KSZTAŁCENIU AKADEMICKIM BIBLIOTEKARZY

Słowa kluczowe: oprogramowanie free libre, open source, kształcenie bibliotekarzy

Streszczenie: W tym przeglądowym rozdziale zaprezentowano zarówno oprogramowanie free libre / open source wykorzystywane w kształceniu akademickim bibliotekarzy, jak i opinię studentów, przyszłych pracowników bibliotek o tym typie oprogramowania. Analizy przeprowadzono w Instytucie Informacji Naukowej i Bibliotekoznawstwa Uniwersytetu Jagiellońskiego.

1. Wprowadzenie

Zaprezentowany materiał badawczy zgromadzono w toku wywiadów ze studentami, realizowanych przez kilkanaście lat w trakcie zajęć dotyczących systemów informacyjno-wyszukiwawczych. Podczas prezentowania kolejnych programów bibliotecznych oraz systemów zintegrowanych rozmawiano o wadach i zaletach różnorodnych aplikacji, o ich możliwościach implementacyjnych, funkcjonalności, perspektywach rozwoju oraz o trudnościach związanych ze zmianami organizacyjnymi i strukturalnymi jednostek, które decydują się na konkretne oprogramowanie. W sondzie udział wzięli studenci studiów stacjonarnych i niestacjonarnych, pierwszego i drugiego stopnia, oraz jednolitych magisterskich, a także słuchacze studiów podyplomowych. Różnorodność rozmówców spowodowana była czasem trwania kwerendy oraz zmianami dokonywanymi w programach studiów. Podkreślić w tym miejscu jednak należy, iż taki wybór dyskutantów był elementem pozytywnym, gdyż studenci studiów niestacjonarnych oraz podyplomowych, pracujący także w bibliotekach, wnieśli do rozmów własne doświadczenie zawodowe i przemyślenia związane z pracą z konkretnym oprogramowaniem.

2. Oprogramowanie FLOSS

Termin FLOSS (*Free Libre/Open Source Software*) łączy w sobie dwa inne: *Free Software* (wolne oprogramowanie) oraz *Open Source* (otwarte oprogramowanie).

Zgodnie z definicją Free Software Foundation, *Free Software* to ruch propagujący wolne oprogramowanie oraz wszelkiego rodzaju aplikacje, które mogą być uruchamiane, kopiowane, rozpowszechniane, analizowane, zmieniane i poprawiane przez użytkowników [Free Software Foundation]. Najważniejsze cechy takiego oprogramowania, które świadczą o jego „wolności" (zgodnie z definicją z 1989 r.), to „wolność kopiowania programu i dzielenia się nim ze swoimi przyjaciółmi i współpracownikami; po drugie, wolność modyfikowania programu wedle własnego uznania, dzięki pełnemu dostępowi do kodu źródłowego" [Definicja wolnego oprogramowania]. Obecnie przedstawia się cztery stopnie „wolności", wedle których klasyfikuje się software:

1. wolność uruchamiania programu, wykorzystywania go w dowolnym celu (wolność 0),
2. wolność analizowania programu oraz dostosowywania go do swoich potrzeb (wolność 1),
3. wolność rozpowszechniania kopii programu (wolność 2),
4. wolność udoskonalania programu i publicznego rozpowszechniania nowych wersji, dzięki czemu może z nich skorzystać cała społeczność Internetu (wolność 3).

Wolności 1 i 3 mogą być spełnione tylko wtedy, gdy dostępny jest kod źródłowy oprogramowania. Aby oprogramowanie mogło być nazwane oprogramowaniem wolnym, musi spełnić wszystkie cztery wolności (0–3). Jeśli nie spełnia choć jednej z nich, jest oprogramowaniem zamkniętym.

Open Source (otwarte oprogramowanie) to odłam ruchu *Free Software*, którego celem jest istnienie swobodnego dostępu do oprogramowania dla wszystkich jego uczestników. Oficjalna definicja otwartego oprogramowania została opracowana w 1997 r. przez organizację Open Source Initiative [Open Source Initiative]. Otwarte oprogramowanie nie musi być bezpłatne. Poza sprzedażą producenci mogą osiągać zyski np. przez prowadzenie szkoleń, wsparcie klienta w trakcie implementacji czy też przez stworzenie dodatkowych modułów na prośbę konkretnej jednostki itp. Mogą także proponować bezpłatną wersję *open source*, aby zachęcić klientów do kupna bardziej rozbudowanej wersji komercyjnej, lub darmowo dostarczyć sam kod źródłowy, a pobierać opłaty za biblioteki programu lub np. jego spolszczenie.

Dla celów pragmatycznych przedstawiciele obu nurtów występują często wspólnie. Z tego też względu omówiono wytwory obydwu prądów razem, stosu-

jąc głównie termin „oprogramowanie FLOSS". Najważniejsze właściwości programów wytwarzanych przez przedstawicieli tych trendów są bowiem w większości przypadków zbieżne. Dla celów analiz aplikacji wykorzystywanych w trakcie kształcenia akademickiego bibliotekarzy wybrano najważniejsze ich cechy (przedstawiane także w wielu definicjach oprogramowania FLOSS): 1) niezawodność, związaną z faktem upublicznienia kodu źródłowego, sprawdzanego i poprawianego na bieżąco przez internautów, 2) nieskrępowany i wartki rozwój oprogramowania, powiązany z liczbą osób dbających o jego ciągły progres, 3) zmniejszenie kosztów systemu przez brak opłat licencyjnych (co nie niweluje kosztów wdrożenia, szkolenia personelu oraz rozbudowy systemu), 4) niezależność od jednego usługodawcy/firmy. Te właściwości były także wyszczególniane przez studentów informacji naukowej i bibliotekoznawstwa UJ jako podstawowe dla wolnego/otwartego oprogramowania. Natomiast fenomen współpracy wielu internautów dbających o dalsze istnienie „ulubionego" software'u był cechą często przez nich wymienianą, uznawaną za pozytywną właściwość aplikacji, które przez to mogą się rozwijać bardziej dynamicznie. Każdy nowy twórca może bowiem, przynajmniej potencjalnie, wnieść nowe pomysły, ulepszyć interfejs, dodać nowe usługi, zadbać o większe bezpieczeństwo danych itp.

3. Oprogramowanie FLOSS w ocenie studentów

W trakcie prowadzenia zajęć o różnorodnych systemach informacyjnych autorka przeprowadzała wywiady ze studentami, które miały na celu zebranie opinii o prezentowanym oprogramowaniu, m.in. o oprogramowaniu FLOSS. Sondaż opierał się na pytaniach zadawanych w trakcie zajęć. Dotyczyły one funkcjonalności systemów, ich użyteczności, wad i zalet wykorzystywania konkretnych aplikacji oraz reakcji studentów, jako użytkowników, na stawiane im w trakcie ćwiczeń zadania do wykonania. Zadania te dotyczyły podstawowych funkcji: wyszukiwania informacji, wprowadzania i modyfikowania danych, tworzenia raportów, opcji wypożyczania dokumentów oraz zmian administracyjnych: dostosowywaniu wyglądu interfejsu, ustawiania zasadniczych kryteriów (grup użytkowników, przedstawianych baz, ustawiania zasad dostępu do danych, drukowania sprawozdań itp.), przygotowywania specyficznych raportów itp.

Na podstawie odpowiedzi studentów przygotowano poniższe zestawienie najważniejszych cech aplikacji, które były oceniane zarówno pozytywnie, jak i negatywnie. Cechy te pogrupowano w kategorie: ekonomiczne, czasowe (w znaczeniu: dotyczące czasu) i funkcjonalne, chociaż nie jest to podział rozłączny. Część właściwości systemów mogła znaleźć się we wszystkich kategoriach (np. opracowane nowe funkcjonalności, na których przygotowanie trzeba mieć czas i pieniądze), lecz

dla spójności rozdziału omówione zostały tylko raz. Na zakończenie analizy przestawiono też wady i zalety implementacji dla jednej instytucji oraz dla konsorcjum. Jako pierwsze przedstawione zostaną kategorie poszczególnych cech.

3.1. Kategorie

3.1.1. Kategorie ekonomiczne

Ważną cechą, zawsze podkreślaną przy analizie oprogramowania FLOSS jest jego „taniość". Oczywiście ten brak kosztów odnosi się wyłącznie do braku opłat licencyjnych, a nie do braku opłat w ogóle. Studenci zauważali mechanizmy pobierania opłat np. za dodatkowe biblioteki oprogramowania wolnego, tworzenie struktur baz, dostosowywanie interfejsów itp. Stwierdzali także, iż instytucja, która chciałaby zaimplementować takie aplikacje, musi przeprowadzać wiele zmian, także kosztownych. Po pierwsze musi zatrudnić lub doszkolić informatyków, którzy będą pracować przy wdrożeniu i rozbudowie oprogramowania (np. stworzeniu nowych modułów, spolszczeniu itp.). Może się też okazać, iż trzeba zatrudnić nowych bibliotekarzy, którzy będą służyć pomocą merytoryczną oraz będą wprowadzać do systemu ogólne i szczegółowe dane (słowniki, pojedyncze informacje o zbiorach, użytkownikach itp.). Biblioteka na pewno powinna też wszystkich pracowników przeszkolić, co oznacza kolejne wydatki.

Do tych uwag studenci dodawali kolejne. Część bibliotek już skomputeryzowanych wykorzystuje oprogramowanie wyłącznie do tworzenia opisów bibliograficznych o strukturze nierelacyjnej, część tworzy relacje np. w związku z wykorzystywaniem rekordów KHW. Niektóre jednostki opracowują bardzo różnorodne typy dokumentów, nie tylko książki i czasopisma, co oznacza wielorakie struktury danych. Do tego dochodzą instytucje posiadające zautomatyzowaną wypożyczalnię, a więc dysponujące bazami z rekordami użytkowników. Trudno więc *a priori* założyć wszystkie możliwe procedury, które należy przygotować dla tak wielu różnorodnych obiektów. Można założyć ogólny model, który za każdym razem wymaga dopracowania i spersonalizowania. A to oznacza dalsze koszty.

Wnioskiem z tych uwag jest stwierdzenie, iż aplikacje FLOSS, jak każde inne, wymagają dostosowania do potrzeb konkretnej instytucji. W sytuacji gdy biblioteka otrzymuje kod źródłowy, jej zadaniem staje się przygotowanie adekwatnego dla niej i jej użytkowników interfejsu (przede wszystkim zadbanie o funkcjonalność i estetykę, np. przez dostosowanie go do ogólnie przyjętej identyfikacji jednostki). Przy wielu programach konieczne jest ich spolszczenie. Do tego dochodzą kolejne koszty zainstalowania systemu, np. analiza istniejących struktur, ich rozbudowa lub stworzenie od nowa, spolszczenie nazw pól, opracowanie relacji itp. Ważne jest przygotowanie słowników, które obowiązywać będą w konkretnym systemie

(haseł wzorcowych, słów kluczowych itp.). Istotną kwestią są również możliwości aplikacji względem importów i eksportów danych z własnych oraz ogólnie dostępnych systemów. W pewnych przypadkach trzeba cały taki moduł stworzyć od nowa, aby dobrze pobierać rekordy np. z baz centralnych. Jest to zarówno problem aplikacyjny, jak i merytoryczny: dostosowania danych między sobą. Trzeba więc przede wszystkim dobrze „zmapować" pola. Do tego dochodzi cena przygotowania plików pomocowych dla użytkowników, aby mogli oni sami wykorzystywać nowe aplikacje. Może się też okazać (i często tak się okazuje), iż trzeba zakupić nowy sprzęt albo go zmodernizować. Pamiętać też należy o niezbędnych nakładach na stałą eksploatację bazy (utrzymanie serwera, informatyka, opłaty za prąd itp.) oraz o jej zabezpieczeniu, zwłaszcza jeśli instytucja przechowuje dane użytkowników (firewall itp.).

Studenci, zwłaszcza studiów niestacjonarnych i podyplomowych, podkreślali, iż małych instytucji nie stać na takie wydatki. Nie zawsze też łatwo takim instytucjom starać się o grant. Stąd wiele małych bibliotek, jeśli wchodzą w jakąś sieć, wybiera aplikacje proponowane przez jednostki nadrzędne lub próbuje tworzyć własne stowarzyszenia, aby spróbować otrzymać dofinansowanie dla całej grupy.

Podsumowanie najważniejszych zalet i problemów związanych z kryteriami ekonomicznymi można przedstawić w następującej tabeli:

Tabela 1. Kryteria ekonomiczne

Zalety zakupu oprogramowania	**Koszty implementacji**
cena oprogramowania	zatrudnienie informatyków, bibliotekarzy; koszty szkoleń pracowników
	koszt dostosowania programu do wymagań biblioteki (w tym spolszczenia, importów itp.), zainstalowania, utrzymywania, rozbudowy
	niezbędne nakłady na stałą eksploatację bazy oraz jej zabezpieczanie
	zakup nowego sprzętu lub jego modernizacja

Źródło: opracowanie własne.

3.1.2. Czas potrzebny na wdrożenie systemu

Kolejnym elementem, który należy brać pod uwagę przy analizie oprogramowania, jest czas potrzebny na: dostosowanie programu do wymagań bibliotek, jego zainstalowanie, przyuczenie personelu i czytelników oraz wprowadzenie danych.

Koszty czasowe są – poza finansowymi – często wskazywane przez studentów. Część studentów studiów niestacjonarnych i podyplomowych wskazywała wręcz na niemożliwe do spełnienia życzenia władz jednostek: natychmiastowe urucho-

mienie i uzupełnienie danymi nowego oprogramowania. A przecież czas na zainstalowanie i dostosowanie systemu do potrzeb wszystkich użytkowników bywa dość długi: może trwać np. rok. Natomiast „zapełnianie" danymi jest procesem wręcz nieskończonym. Można go oczywiście skracać poprzez różnego rodzaju importy. Lecz żaden import wielu danych nie został jak na razie przeprowadzony „bezstratnie". Dopracowanie mapowania pól, ustalenie wzajemnych relacji, podjęcie decyzji, jak poprzedni sposób katalogowania zastąpić nowym, wymaga także czasu. Do tego dochodzi okres „akceptacji" nowego systemu przez użytkowników bibliotek, ich przyuczenia, a ze strony jednostki – czas przygotowania pomocy/helpów dla czytelników. Natomiast w sytuacji gdy konkretna biblioteka przyzwyczaiła swoich użytkowników do specyficznych usług, np. przygotowywania specyfikacji, bibliografii tematycznych itp., dodać jeszcze należy okres implementacji tych usług, np. poprzez stworzenie nowych modułów w systemie.

Oczywistą zaletą jest jednakże zaoszczędzony przez czytelników i pracowników czas, który dla przekazywania różnorodnych informacji skraca się, niekiedy bardzo znacząco. Nowe funkcjonalności pozwalają użytkownikom systemu przygotowywać także nowe opracowania i analizy, które wcześniej nie były po prostu możliwe. Skraca się więc okres tworzenia nowej wiedzy.

Jak widać z przedstawionych uwag, poprzednie tezy wygłaszane przez studentów przy analizie finansowej odnieść można także do czasu trwania wdrożenia oraz eksploatacji aplikacji. Stąd podsumowanie kryteriów dotyczących czasu tworzy dość podobną zakresowo tabelę.

Tabela 2. Kryteria dotyczące czasu

Zalety zaimplementowania oprogramowania	**Czas implementacji**
czas dostępu do danych zaoszczędzony przez czytelników i pracowników instytucji	dostosowanie programu do wymagań bibliotek i jego zainstalowanie
nowe funkcjonalności systemu = poszerzenie usług = krótszy czas tworzenia się nowej wiedzy	przyuczenie personelu
	opracowanie plików pomocy
	przygotowanie czytelników
	wprowadzenie danych

Źródło: opracowanie własne.

3.1.3. Funkcjonalność

Potencjalne problemy, które w stosunku do implementacji wolnego oprogramowania zgłaszali studenci, odnosiły się także do cech funkcjonalnych. Podkreślali oni przede wszystkim konieczność takiego dopracowania aplikacji, aby można było ją ocenić jako przyjazną dla czytelnika i bibliotekarza. Oczywiście część oprogramowań ma przygotowane podstawowe funkcjonalności: wprowadzanie, wyszukiwa-

nie i prezentację danych. Bardziej specjalistyczne, zaawansowane możliwości administrator bazy musi jednak w większości systemów przygotować samodzielnie. Nie jest to tylko kwestia czasu i pieniędzy, ale także umiejętności i wiedzy merytorycznej. Aby informatyk napisał oprogramowanie, specjalista musi przygotować algorytm i zdefiniować najważniejsze czynności, konieczne do wykonania zaplanowanych zadań. Poza kosztami finansowymi i czasowymi wymaga to wiedzy i umiejętności, zwłaszcza przy dokładnym rozdzieleniu modułów czytelnika, bibliotekarza i administratora. Ważne dane wymuszają stworzenie kilku interfejsów oraz aplikacji zapewniających bezpieczeństwo. Praca może więc zostać „pomnożona przez 3" w związku z trzema interfejsami i aplikacjami wspomagającymi konkretne elementy systemu.

Kolejne segmenty funkcjonalne problematyczne dla przyszłych i obecnych bibliotekarzy to cechy związane z ulokowaniem nowego wolnego/otwartego oprogramowania w istniejącym systemie bibliotecznym. Wchodzi w to możliwość wykorzystania istniejących baz danych (czyli np. importu danych do nowych baz nowego systemu), wymiany informacji z innymi systemami oraz dostosowanie oprogramowania do baz istniejących w poszczególnych bibliotekach (czyli ze zmianą aplikacji, aby istniejące bazy można było wykorzystywać). Studenci na przestrzeni lat podkreślali, iż wiele instytucji rozpoczęło katalogowanie dokumentów w formacie MARC BN. Niektóre mniejsze biblioteki jeszcze nie dokonały pełnych konwersji do formatu MARC 21. Do tego dochodzi problem baz o strukturach całkowicie odmiennych od struktur MARC-owskich. Zaliczyć do nich można np. bazy tworzone w bibliotekach publicznych czy kościelnych: o osobach związanych z regionem, baz cytatów, baz pełnotekstowych itp. Nie należy też zapominać o dodatkowych polach, które zostały stworzone w poszczególnych jednostkach, np. dla własnych klasyfikacji, dodatkowych opisów, notatek, streszczeń itp. Nawet jeśli możemy wykorzystać bazy przygotowane w podobnej instytucji, to może okazać się, iż musimy dodać nowe pola, relacje, stworzyć mapowania oraz dokonać nowych importów.

Studenci rozważali także cechy określone w definicjach i opisach systemów FLOSS. Zapisana wśród nich niezawodność oparta na współpracy internautów nie była do końca przekonująca dla studentów, zwłaszcza studiów niestacjonarnych czy podyplomowych. Instytucja publiczna, jaką jest biblioteka, nie może liczyć tylko na „pospolite ruszenie" internautów. Dostęp dla czytelników musi być zapewniony zawsze. W sytuacjach trudnych, awaryjnych trudno oczekiwać natychmiastowego odzewu społeczności sieci. Nie oznacza to, iż taki nie nastąpi. Problem jest jednak z czasem, który upływa od wystąpienia awarii do ponownego pełnego uruchomienia systemu.

Kolejna cecha – nieskrępowany i wartki rozwój oprogramowania – wydała się studentom „fascynująca". Uznali taki nurt współdziałania za fenomen sieci, cho-

ciaż pojawiały się obawy o „znudzenie programem", „wytworzenie nowego produktu, nie do końca kompatybilnego", z przerzuceniem na instytucję dbałości o rozwój aplikacji, co niesie za sobą koszty finansowe i czasowe. Studenci wskazali wręcz, iż niezależność od jednego usługodawcy/firmy dla małej jednostki nie jest zawsze cechą pozytywną. Poczucie wsparcia oraz samo wsparcie jest istotne zarówno dla pracowników, jak i użytkowników biblioteki. Wiązali je często z dobrze funkcjonującą firmą, stabilną, rozwijającą się, tworzącą niedrogie aplikacje.

Podsumowując cechy związane z funkcjonalnością, można stworzyć następującą tabelę:

Tabela 3. Cechy funkcjonalne systemu

Zalety oprogramowania	**Funkcjonalność – problemy**
możliwość wykorzystania istniejących baz danych oraz stworzenia nowych	dostosowanie oprogramowania do baz istniejących w poszczególnych bibliotekach (struktury inne niż MARC 21; dodatkowe pola itp.)
możliwość wymiany informacji z innymi systemami	rozwój systemu
przyjazność dla czytelnika; nowe usługi	reakcja na sytuacje awaryjne
przyjazność dla bibliotekarza; nowe usługi	
potencjalny dynamiczny rozwój (wielu twórców)	

Źródło: opracowanie własne.

3.2. Jednostki implementujące

Kolejny podział zalet i wad oprogramowania dotyczy liczby jednostek implementujących oprogramowanie: jednej lub wielu, głównie zrzeszonych w konsorcja. W zależności od liczby współpracujących bibliotek inaczej rozkłada się bowiem praca przy implementacji i użytkowaniu systemu. Tę część rozpoczyna analiza dla jednej instytucji.

3.2.1. Jedna instytucja implementująca

Do zalet samodzielnej implementacji studenci zaliczyli przede wszystkim dokładne dostosowanie oprogramowania do własnych potrzeb biblioteki – przy możliwości wykorzystywania rozwiązań instytucji, które zrobiły to wcześniej. Wiąże się to oczywiście z większymi kosztami finansowymi i czasowymi. Do wad takiego rozwiązania zaliczono: trudności przy instalacji, które niekoniecznie pracownicy konkretnej instytucji mogą pokonać, oraz konieczność wykonania samodzielnych prac implementacyjnych, zmieniających i poszerzających działanie oprogramowania. Może to doprowadzić w ostateczności do wytworzenia „podprogramów", które w porównaniu z innymi implementacjami mogą okazać się wręcz nowymi aplikacjami, mało kompatybilnymi. W takiej sytuacji istnieje też możliwość zaistnienia wielu struktur, nawet jeśli zbudowane są one na zaakceptowanym formacie.

Podsumowując najważniejsze parametry oprogramowań, przedstawić można następującą tabelę:

Tabela 4. Zalety i wady samodzielnej implementacji

Zalety samodzielnej implementacji	**Trudności powstające przy samodzielnej implementacji**
wykorzystywanie rozwiązań innych instytucji	trudności przy instalacji; konieczność samodzielnych prac implementacyjnych – wytworzenie „podprogramów", możliwość zaistnienia wielu struktur
stworzenie programu dla potrzeb konkretnej instytucji	

Źródło: opracowanie własne.

3.2.2. Konsorcjum

W porównaniu z samodzielną pracą przy implementacji oprogramowania FLOSS konsorcjum wydawało się studentom bardziej atrakcyjne. Do zalet stworzenia grupy współpracujących instytucji zaliczono: rozłożenie prac między wiele jednostek, rozłożenie opłat, opracowanie wspólnej polityki instalacji i importów danych oraz większy nacisk na utrzymywanie przyjętej struktury danych. Podkreślano też większą pomysłowość wielu pracowników, która może spowodować wytworzenie nowych usług i lepszą funkcjonalność aplikacji.

Nie zabrakło jednak i wad, do których zaliczono kolejność prac nad adaptacją programu dostosowywaną do potrzeb najliczniejszej grupy zainteresowanej danym problemem, co może oznaczać, iż biblioteka posiadająca np. dodatkowe pola może długo czekać na dopracowanie aplikacji specjalnie dla niej. Podkreślano też, iż rozwój systemu często zgodny jest z polityką tworzoną przez lidera, najmocniejszego w konsorcjum.

Zebrane w tabelę przymioty aplikacji zaprezentować można następująco:

Tabela 5. Zalety i wady implementacji konsorcjalnej

Zalety implementacji konsorcjalnej	**Trudności powstające przy implementacji konsorcjalnej**
rozłożenie prac między wiele instytucji	kolejność prac adaptacyjnych dostosowana do potrzeb najliczniejszej grupy zainteresowanej danym problemem (lub zgodna z polityką tworzoną przez lidera)
opłaty rozłożone między wiele instytucji	
opracowywana wspólna polityka instalacji i importów danych	
utrzymywanie struktur baz	
nowe funkcjonalności	

Źródło: opracowanie własne.

Na końcu rozdziału przedstawiono wykaz programów FLOSS wykorzystywanych w trakcie akademickiego kształcenia bibliotekarzy w INIB UJ.

4. Oprogramowanie wykorzystywane w INIB UJ

Po przeanalizowaniu aplikacji wykorzystywanych w INIB UJ ustalono, iż część z nich zaliczyć można do systemów *open* lub *free*. Należą do nich system Linux oraz system e-learningowy Moodle. Wiele programów ułatwiających pracę w Internecie także zaklasyfikować można to tego typu software'u. Są to oprogramowania dla połączeń internetowych: WinSCP czy Putty, a także systemy zarządzania treścią: Joomla oraz Liferay. Wymienić należy także aplikacje wspomagające tworzenie plików html (np. KompoZer) czy rejestrowanie działań dokonywanych na komputerze (CamStudio). Wiele osób korzysta także z programów do zarządzania plikami znajdującymi się na dysku komputera np. z Unreal Commandera. Najliczniejszą grupę wolnych/otwartych systemów stanowią programy graficzne i tekstowe: Blender, Hugin, ScanTailor, OpenJPEG, RAW, Therapee, XNView, Scribus, PDF Creator czy PDFSAM i inne.

Jeśli chodzi o bazy danych, to w trakcie zajęć omawiane są: CDS ISIS oraz Koha. Przedstawiane są także bazy napisane w MySQL. Nie jest to duża liczba bazodanowych programów FLOSS, ale też w bibliotekach częściej wykorzystywane są programy komercyjne. Takie też prezentowane są na ćwiczeniach. Wymienić tu można programy: MAK, MAK+, Mol, Patron, Libra, Molik, Prolib, Sowa oraz systemy: VTLS/Virtua, Aleph. W poprzednich latach omawiano też Millenium, Horizon oraz TINLIB.

5. Uwagi końcowe

Z przedstawionego zestawienia programów FLOSS wykorzystywanych w trakcie akademickiego szkolenia bibliotekarzy wynika, że nie są to najczęściej prezentowane systemy bazodanowe. Jest to rezultat ogólnej sytuacji bibliotek, które w większości przypadków zdecydowały się na oprogramowanie komercyjne.

Odnosząc się do opinii studentów o programach wolnych/*free*, należy podkreślić, że pomimo wielu uwag do oprogramowania FLOSS docenili oni jego możliwości. W trakcie rozważań o poszczególnych cechach aplikacji oraz ich implementacji i wykorzystywaniu studenci wykazali się wiedzą, a czasem także doświadczeniem wyniesionym z pracy z różnorodnymi aplikacjami. Głównym wnioskiem wyciągniętym z wywiadów jest jednakże stwierdzenie, iż nie ma systemów idealnych, a praca nad kolejnymi ulepszeniami jest nieskończona.

Bibliografia

Definicja wolnego oprogramowania. http://pl.wikipedia.org/wiki/Definicja_wolnego_oprogramowania [odczyt: 12.10.2014].

Free Software Foundation. http://www.fsf.org/ [odczyt: 12.10.2014].

Open Source Initiative. http://opensource.org/ [odczyt: 12.10.2014].