Rozwijanie dużego systemu w warunkach akademickich

Jacek Sroka

Instytut Informatyki, Instytut InformatykiUniwersytet Warszawski

Droga prosta jest najkrótsza, ale na ogół wymaga najwięcej czasu, aby dojść nią do celu.
Georg Christoph Lichtenberg (1742 - 1799)

Streszczenie

Rozwijanie dużego systemu w warunkach akademickich oraz w warunkach komercyjnych różni się pod kilkoma aspektami. Poza głównym celem, jakim jest dostarczenie użytkownikom wysokiej jakości produktu spełniającego ich wymagania, w pracach prowadzonych na wyższej uczelni duży nacisk stawia się na walory edukacyjne i badawcze przedsięwzięcia. Różnice dotyczą również organizacji zespołu. W warunkach komercyjnych dąży się do specjalizacji poszczególnych osób. W warunkach akademickich każdy uczestnik przedsięwzięcia powinien zdobyć jak najszerszy zakres wiedzy, a osiągnięcie specjalizacji nie jest możliwe ze względu na krótki okres uczestnictwa w projekcie, porównywalny z okresem tworzenia pracy magisterskiej. Celem rozdziału jest ocena przydatności języka UML do zastosowania w pracach nad projektem USOS. USOS to duży system bazodanowy do obsługi spraw studiów, rozwijany w środowisku akademickim. Opisane wyniki zostały zebrane podczas rozwoju obecnie wdrażanego podsystemu Akademiki.

Wprowadzenie

Uniwersytecki System Obsługi Studiów USOS2003 został przedstawiony na KKIO'2001 [JMD2001] i KKIO'2002 [JMD2002]. Jego rozwojem steruje Uniwersyteckie Centrum ds. Informatyzacji, zrzeszające 15 polskich uniwersytetów. Prace projektowe i programistyczne prowadzone są przez pracowników oraz studentów Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego (WMIM UW). Projekt trwa od grudnia 1999. Brało w nim dotychczas udział kilkudziesięciu studentów, z których około 15 zrealizowało projekty magisterskie dotyczące USOS.

Prace rozwojowe prowadzone są przez studentów, w ramach zajęć dydaktycznych, bądź projektu magisterskiego. Zazwyczaj każdy uczestnik opracowuje od początku do końca nowy podsystem. Nadzór nad jego postępami sprawują nauczyciele akademiccy

WMIM posiadający doświadczenia zebrane podczas realizacji dotychczasowej funkcjonalności systemu. Przy projekcie zatrudnionych jest też dwóch programistów, którzy wykonują prace standaryzacyjne, dostrajają bazę danych, poprawiają znalezione błędy oraz dodają niezwłocznie potrzebną funkcjonalność.

Początkowo wydawało się, iż USOS będzie systemem średniej wielkości, składającym się z luźno powiązanych podsystemów. Dlatego prace prowadzone były zgodnie z metodą *Fast Track* [CDM2000] należącą do *Custom Development Method* - opracowanego przez firmę *Oracle* sposobu wytwarzania oprogramowania. Wykorzystywano przy tym zalecane narzędzia z pakietu *Oracle Designer*. Główną zaletą takiego podejścia jest możliwość szybkiego reagowania na życzenia użytkowników dzięki często weryfikowanym prototypom. Okazuje się jednak, że w skutek ciągłego rozszerzania funkcjonalności systemu dalsza jego realizacja w ten sposób staje się utrudniona. USOS przekształca się w system duży. Nowe podsystemy zaczynają być silnie powiązane z tymi już wdrożonymi, a nawet innymi rozwijanymi równolegle. Szybkie dostarczanie prototypów przestaje być atutem przy uwzględnianiu ewentualnych błędów, jakie mogą być przy okazji wprowadzone do reszty systemu. Rośnie za to waga dokładnej analizy i dobrego projektu.

Co więcej ewolucji systemu nie towarzyszy wzrost doświadczenia zespołu. Pomyślne ukończenie kolejnego podsystemu pociąga za sobą obronienie pracy magisterskiej przez jego autora i odejście z zespołu osoby najlepiej znającej dane zagadnienie. Pozostawiona dokumentacja staje się głównym źródłem wiedzy dla kolejnych ochotników, którzy odbywają przy okazji ważną, praktyczną lekcję inżynierii oprogramowania.

Podczas realizacji podsystemu Akademiki zdecydowano się wprowadzić do projektu język *Unified Modeling Language* (UML), który stał się standardem zaakceptowanym przez środowiska akademickie oraz przemysł i uważany jest za jeden ze skuteczniejszych sposobów zarządzania zmianami w projekcie. Jeden z pionierów metodologii obiektowej Grady Booch uważa, że "jeśli aplikacja zawiera kilkadziesiąt klas lub abstrakcji, to zespół nie jest w stanie zapamiętać wszystkich zależności pomiędzy nimi". Wykorzystanie intuicyjnej, graficznej notacji w trakcie prac analitycznych, projektowych i tworzenia dokumentacji sprawia, iż powstałe w ten sposób artefakty oraz zależności między nimi są zrozumiałe nawet dla nowych członków zespołu. Jednocześnie ich znaczenie nie budzi wątpliwości, gdyż wszystkie elementy są dobrze zdefiniowane (por. [UML2001]). Inne istotne cechy UML to możliwość komputerowego wspomagania walidacji opracowywanych modeli, dostępność materiałów edukacyjnych oraz dobre wsparcie narzędziowe.

Prowadzone przez półtora roku prace pozwoliły zbadać sposoby wykorzystania języka UML we wszystkich etapach rozwoju nowego podsystemu. Jako punkt wyjścia wybrano dostosowany do warunków komercyjnych proces wytwarzania oprogramowania *Rational Unified Proces* (RUP). Przy pomocy zakupionego oprogramowania *Rational Rose Enterprise Edition* opracowano model podsystemu Akademiki. Powstała także praca magisterska [SRO2003], która może posłużyć za szablon dla innych uczestników projektu. W ramach eksperymentu, dla przetestowania jakości rozwiązań, część prac programistycznych została wykonana jedynie na podstawie przygotowanego projektu,

przez osobę nieuczestniczącą w pracach analitycznych ani projektowych.

Wypracowane rozwiązania oraz ich ocena przedstawione są w kolejnych punktach.

Wykorzystanie UML w projekcie

Prace nad podsystemem Akademiki prowadzone są od początku roku akademickiego 2001/2002. Rozpoczęto od zapoznania się ze sposobem działania zespołu USOS i wykorzystywanymi narzędziami. Następnie przeanalizowano architekturę USOS i dostępną dotychczas funkcjonalność.

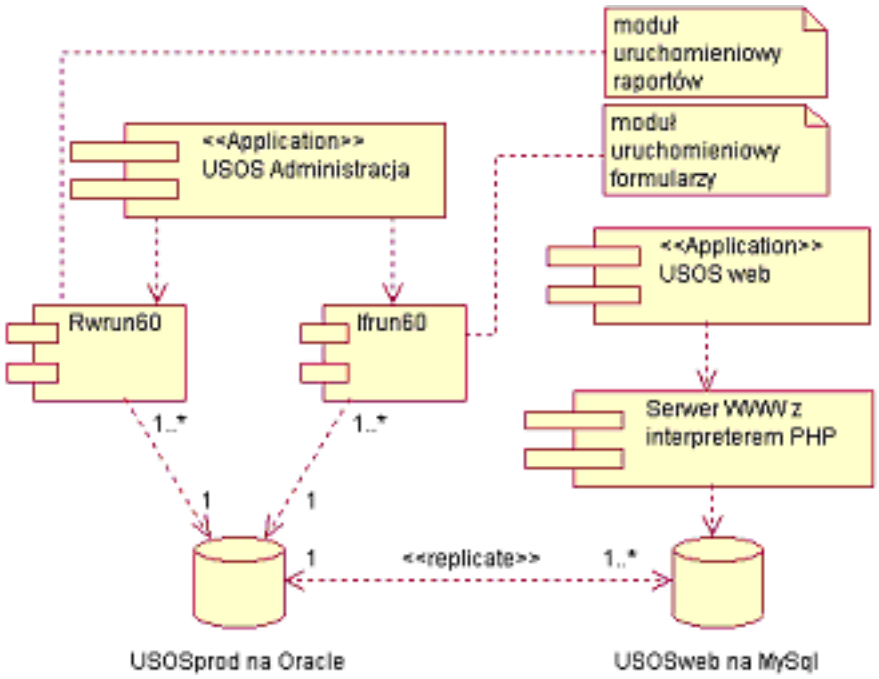


Diagram komponentów - architektura systemu USOS

Głównym źródłem wiedzy były prace magisterskie wcześniejszych uczestników projektu oraz dokumentacja elektroniczna stworzona przy pomocy pakietu *Oracle Designer*. Narzędzie to sprawdza się przy projektowaniu bazy danych oraz formularzy i raportów. Istotne jest przy tym operowanie dobrze zrozumiałymi przez studentów diagramami *Entity Relation Diagram* (ERD) oraz opracowaną specjalnie do tego celu graficzną notacją umożliwiającą przedstawianie organizacji formularzy i raportów. Istotną zaletą jest również możliwość wygenerowania na podstawie opracowanego projektu skryptów *Data Definition Language* (DDL) tworzących bazę danych oraz prototypów formularzy i raportów, które spełniają przyjęte w projekcie standardy. Niestety możliwości modelowania architektury, zależności pomiędzy podsystemami oraz funkcjonalności są niedostępne bądź niewystarczające i większość uczestników projektu wykonywało te prace przy pomocy wybranych, a czasami nawet opracowanych przez siebie notacji i narzędzi. Prowadziło to do powstawania niejasności co do znaczenia elementów tworzonych modeli.

Zdecydowano się zatem na wykorzystanie języka UML. Do przedstawienia architektury wykorzystano diagramy komponentów (por. rysunek 1). Zależności pomiędzy pod-

systemami pokazano na diagramach klas, a do analizy funkcjonalności zdecydowano się zastosować przypadki użycia. Efekty pracy były obiecujące, a pomysł zyskał poparcie kierownictwa projektu.

Wykorzystanie UML ma duże znaczenie dydaktyczne. Język ten jest obecnie podstawą kursów dotyczących projektowania systemów informacyjnych. Jego wykorzystanie w USOS może zachęcić studentów do podejmowania się projektów badawczych z dziedziny inżynierii oprogramowania mogących znaleźć praktyczne zastosowanie. Daje to większą satysfakcję oraz czyni zajęcia bardziej interesującymi.

Kontakty z użytkownikami i analiza wymagań

Każde przedsięwzięcie programistyczne rozpoczyna się od analizy wymagań. Błędy popełnione w tej fazie trwania projektu są bardzo kosztowne. Głównym źródłem ich powstawania jest niedostateczne zrozumienie potrzeb przyszłych użytkowników. Powodem mogą być problemy z komunikacją, brak odpowiedzialności za ustalenia, a nawet nie zrozumienie własnych potrzeb przez samych użytkowników.

Problemy z komunikacją

W kontaktach z użytkownikami należy posługiwać się zrozumiałym dla obu stron językiem. Niezależnie od przyjętej metodyki powinien zostać sporządzony słownik niejasnych pojęć zawierający objaśnienia terminów zarówno z zakresu modelowanej dziedziny, jak i tworzonego systemu informatycznego. Umożliwi on zrozumienie powstających dokumentów przez osoby nieuczestniczące w rozmowach i ma szczególne znaczenia dla zespołu programistycznego o systematycznie uzupełnianym, bądź zmienianym składzie.

Nowoczesne procesy wytwarzania oprogramowania, jak RUP, zalecają rozpoczęcie projektu od analizy biznesowej. Jej wynikiem powinno być sporządzenie dokumentów obrazujących sposób działania przedsiębiorstwa/organizacji zlecającej wytworzenie oprogramowania. Podczas realizacji podsystemu Akademiki szczególnie przydatne okazały się diagramy przepływu czynności. Na przykład na rysunku 2. przedstawiono nadzorowany przez Biuro Spraw Studenckich (BSS) proces rozdzielania akademików na UW. Poza udostępnieniem informacji dla pozostałych członków zespołu takie diagramy pozwalają w wygodny sposób komunikować się z użytkownikami.

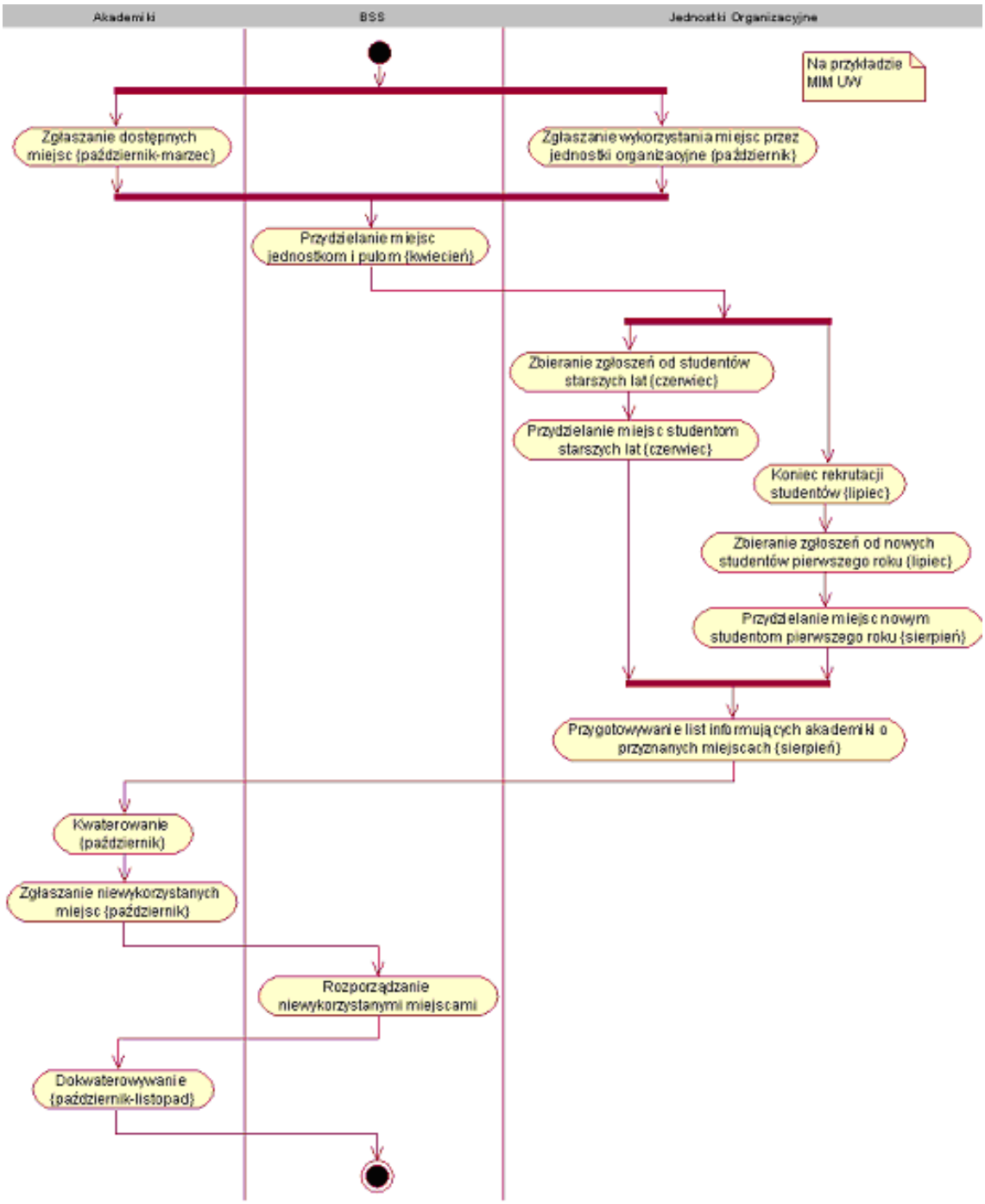
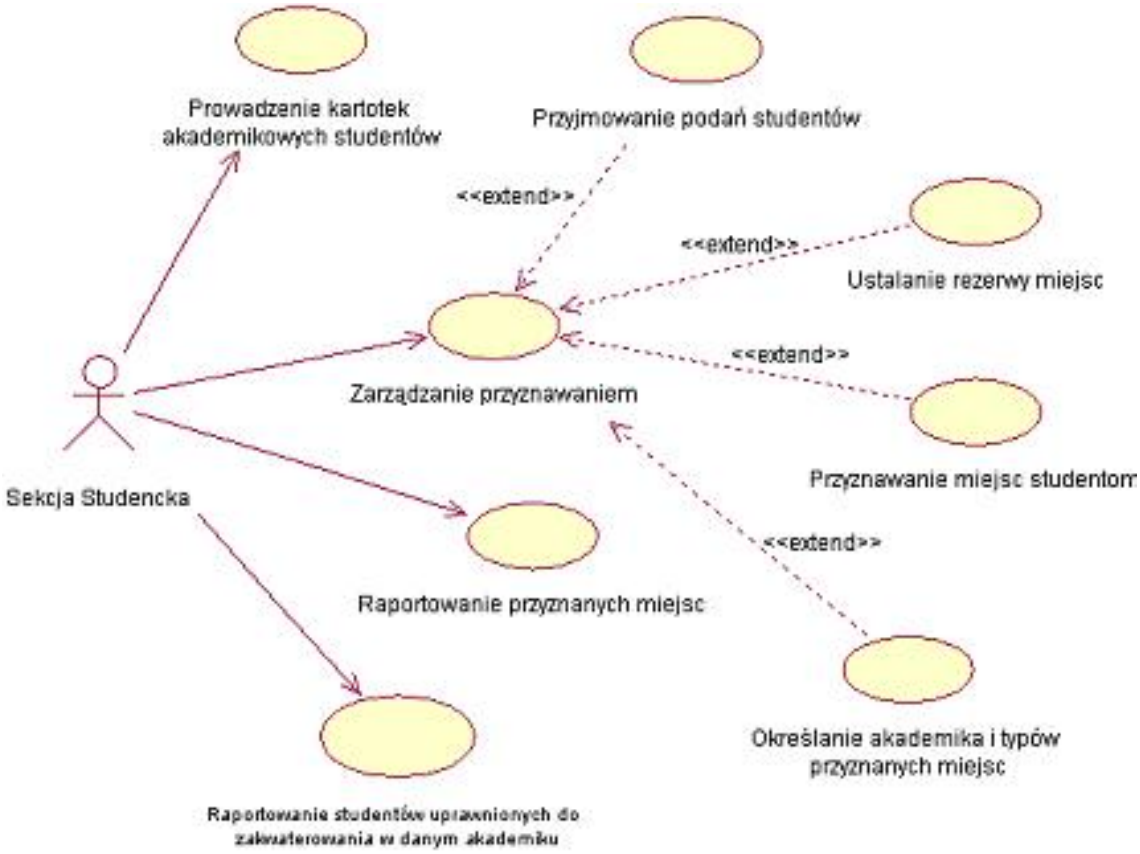


Diagram przepływu czynności - rozdzielanie akademików na UW

Ponieważ USOS ma działać na wielu uniwersytetach nie mogąc ograniczać ich autonomii, konieczne jest wypracowanie modelu biznesowego pasującego do rozwiązań stosowanych przez nie wszystkie. Rozproszenie geograficzne ogranicza często komunikację do wymiany dokumentów. Doświadczenie pokazało, że generowane przez studentów diagramy są zrozumiałe dla użytkowników, co nie zawsze miało miejsce z opisami słownymi.

Odpowiedzialność za ustalenia

W komercyjnych przedsięwzięciach część dokumentów sporządza się po to, aby umożliwić nadzór nad postępami projektu. Niezbędna jest też możliwość wykazania, że gotowy produkt spełnia wynegocjowane wymagania. Ze względu na niekomercyjny charakter USOS wydawać by się mogło, że spisywanie wymagań jest zbędne, ponieważ uczestnikami projektu jak i przyszłymi użytkownikami kierują dobre chęci. Tak jednak nie jest, a doświadczenie tego na własnej skórze jest doskonałą szkołą inżynierii oprogramowania. W przeciwieństwie do projektów wykonywanych na zajęciach, dla których wymagania są dobrze określone, podczas realizacji systemu produkcyjnego użytkownicy nie zawsze wiedzą, czego dokładnie im potrzeba. Skoro jednak prace programistyczne prowadzone są do momentu zaakceptowania stworzonego podsystemu cel musi być widoczny od samego początku.



Przypadki użycia dla aktora Sekcja Studencka

Należy założyć, że ustalenia ustne nie są wiążące. Zgoda użytkowników może wynikać z ich niechęci do przyznania się, iż nie rozumieją, bądź nie są pewni. Sytuację może utrudniać obecność ich przełożonych na spotkaniach, a czasami skrępowanie wiekiem studentów. Opracowanie i utrzymywanie dokumentów zawierających uzgodnione wymagania jest więc dobrą praktyką. Użytkownicy powinni po każdym spotkaniu dostać czas na zapoznanie się z dokumentami i być zachęcani do zadawania pytań. Istnienie spisanych wymagań stwarza większą presję do zastanowienia się nad nimi zanim podsystem zostanie stworzony, a wprowadzenie jakichkolwiek zmian będzie dużo bardziej pracochłonne.

Do określenia granic i zakresu odpowiedzialności systemu w UML wykorzystuje się

aktorów i przypadki użycia. Ogólne zależności przedstawiane są na diagramach przypadków użycia (por. rysunek 3). Szczegółowy opis można wykonać zgodnie z jednym z wielu dostępnych szablonów. W pracy [SRO2003] wykorzystano szablon dostarczony z [RUP2002]. Przy określaniu przypadków użycia należy zachować wiele uwagi. Powinny one obejmować wszystkie kluczowe aspekty działania systemu bez wchodzenia w zbędne szczegóły.

Niezrozumienie swoich potrzeb przez użytkowników

Podstawowym celem systemu USOS jest usprawnienie prac administracyjnych związanych z obsługą dydaktyki na wyższej uczelni. Jednak czasami użytkownicy na początku nie postrzegają wprowadzenia systemu jako ułatwienia. Jego siła objawia się po poznaniu możliwości i opanowaniu sposobu obsługi. Bardzo istotne jest zachęcenie użytkowników do współpracy przez ich przełożonych. Prace rozwojowe nad nowym podsystemem powinny należeć do zakresu ich obowiązków, a wykazywanie inicjatywy powinno być motywowane.

Należy się spodziewać, iż użytkownicy mogą nie znać technologii komputerowej w stopniu dostatecznym do określenia swoich potrzeb. Cześć ich pomysłów może być niemożliwa do zrealizowania. Z drugiej strony mogą nie być świadomi, iż niektóre wykonywane przez nich czynności da się znacznie uprościć poprzez zastosowanie odpowiednich rozwiązań. Spisywanie wymagań może mieć dodatkowy efekt porządkujący. Skoro trzeba spisać, to trzeba najpierw przemyśleć, uprościć, może nawet zreorganizować. Jest to dobry moment na zastanowienie się, czy nie występują jakieś przypadki szczególne. Możliwe, że ich wykrycie spowoduje rewizję procedur organizacyjnych.

Trzeba być jednak świadomym, iż mimo dobrych chęci i odpowiedzialnego podejścia formalnie spisane wymagania oraz jasny projekt nie zawsze wystarczą. Konieczne jest częste weryfikowanie postępów poprzez przedstawianie użytkownikom prototypów oraz zastosowanie iteracyjnego sposobu pracy.

Projekt bazy

Przy podejściu strukturalnym do projektowania relacyjnej bazy danych wykorzystuje się diagramy ERD. Technika ta stanowi podstawę każdego kursu baz danych. Diagramy ERD są skutecznie wykorzystywane podczas prac nad systemem USOS. W UML do modelowania bazy danych wykorzystuje się diagramy odpowiednio stereotypowanych klas. Kluczową różnicą tych podejść jest szczegółowość. Diagramy ERD mają charakter pojęciowy. Przedstawia się na nich encje, a nie tabele. Encje posiadają jedynie atrybuty je opisujące. Natomiast w UML przedstawia się tabele, które fizycznie mogą istnieć w bazie danych. Tabele zawierają dodatkowe kolumny niezbędne do realizacji więzów referencyjnych. Z tabelami mogą być związane wyzwalacze, które w UML modeluje się jako metody.

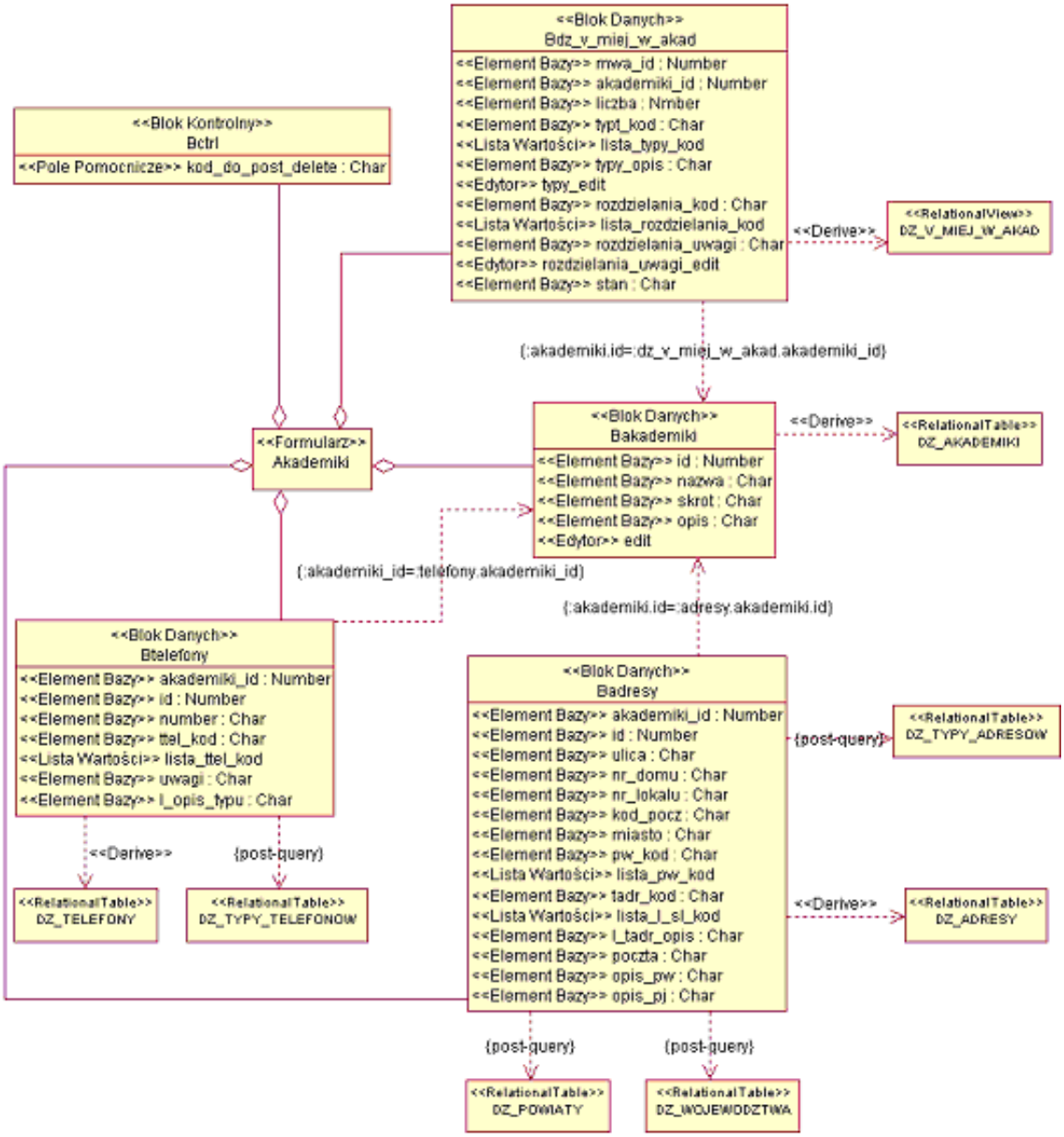
Może się wydawać, iż ze względu na wyższy poziom abstrakcji diagramy ERD powinny być wygodniejsze w użyciu. Jednak przy dużym projekcie, w którym baza dany-

ch ulega dynamicznemu rozwojowi istotnym czynnikiem jest możliwość wykorzystania inżynierii wprzód (ang. *forward enginering*) i inżynierii wstecz (ang. *backward engineering*). W obu wypadkach inżynieria wprzód jest możliwa, dla encji po sprecyzowaniu kryteriów przekształcenia do tabel, a dla klas po doprecyzowaniu wszystkich szczegółów. Jednak pełna inżynieria wstecz jest możliwa jedynie w przypadku UML. Dlatego dodatkowy nakład pracy potrzebny na stworzenie projektu o wyższym poziomie szczegółowości wydaje się procentować przy uwzględnieniu łatwości utrzymania modelu bazy. Ponadto opłacalne wydaje się podejście iteracyjne, w którym w zależności od wygody zmiany wprowadza się raz w modelu, a raz w schemacie bazy danych i przy pomocy inżynierii wprzód lub inżynierii wstecz uzgadnia się ich stan.

Projekt modułów

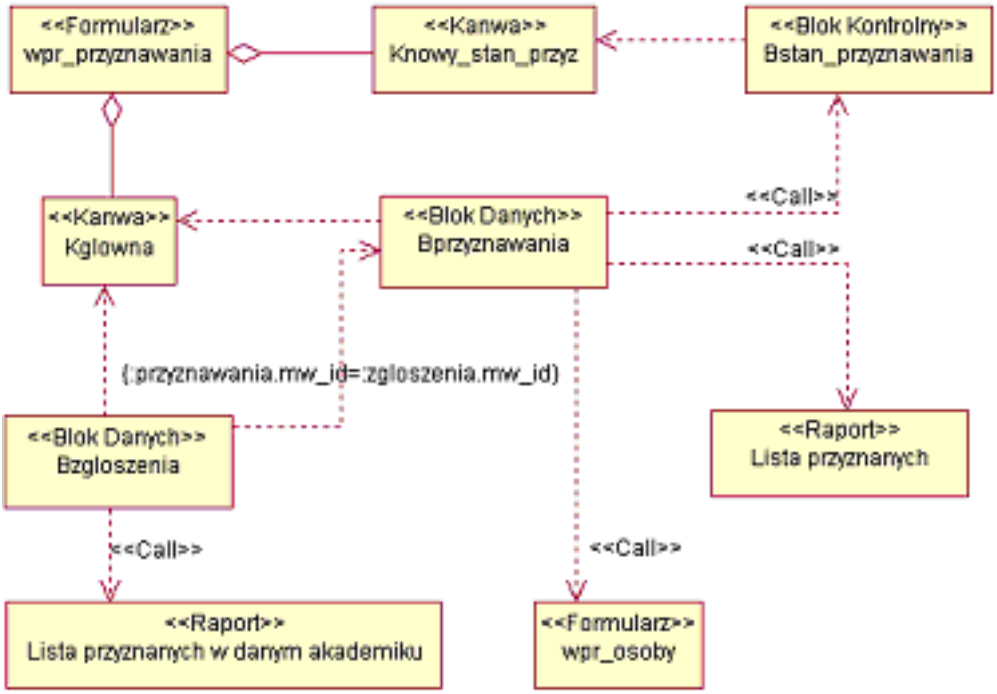
Konstrukcja systemu USOS, ze względu na jego złożoność, nie byłaby możliwa bez wyodrębnienia mniejszych podsystemów. Aby projekt zachował jednorodność, wszystkie prace powinny być prowadzone zgodnie z uprzednio ustalonymi standardami. Dotyczy to zarówno architektury systemu, rozwiązań programistycznych w nim przyjętych, jak i interfejsu użytkownika.

Funkcjonalność części administracyjnej systemu opiera się na formularzach i raportach przygotowywanych za pomocą narzędzi z pakietu *Oracle Developer*. W ramach pracy standaryzacyjnej [GOLD2003] spisano zalecane rozwiązania oraz przedstawiono szablony umożliwiające szybkie generowanie zgodnych ze standardami prototypów. Wykonywane są również czynności mające na celu ujednolicenie wdrożonych już podsystemów. Niestety samo przestrzeganie standardów nie gwarantuje jeszcze jednolitości rozwiązań. Przykładowo niektóre dane wykorzystywane są przez kilka podsystemów. Dla wygody użytkowników dostęp do nich powinien być możliwy z kontekstu różnych formularzy. Należy zadbać o to, aby w miarę możliwości wykorzystano analogiczne rozwiązania. Wymaga to dodatkowych rozwiązań w zakresie organizacji pracy zespołu, ponieważ prace rozwojowe prowadzone są przez osoby nieznające istniejącej funkcjonalności USOS. Nawet wykorzystanie często weryfikowanych prototypów prowadzi w niektórych przypadkach do niepotrzebnego rozwiązywania tych samych problemów na nowo, a czasami nawet do marnowania wykonanej pracy. Jakkolwiek nie da się przecenić wagi prototypów przy współpracy z użytkownikami, to oceny planowanych rozwiązań warto dokonać wcześniej, w obrębie zespołu. Konieczne jest, więc sporządzenie projektu i jego weryfikacja. Ze względu na wymuszone wygodą użytkowników dążenie do tworzenia modułów jak najprostszych, nakład prac konieczny do uzyskania projektu o zadowalającej szczegółowości może być niewielki. Analiza projektu pozwoli nauczycielom akademickim nadzorującym projekt spostrzec analogie z dotychczas wdrożonymi podsystemami i zasugerować ewentualne poprawki przed dokonaniem żmudnej organizacji interfejsu użytkownika w pierwszych prototypach.



Model danych formularza Akademiki

UML najlepiej sprawdza się w pracach nad obiektowymi systemami informacyjnymi. Został jednak przewidziany wygodny mechanizm rozszerzania samego języka o specyficzne pojęcia, dotyczące modelowanej dziedziny. W pracy [SRO2003] przedstawiono proponowany sposób modelowania formularzy i raportów *Oracle Developer* przy pomocy języka UML oraz zdefiniowanych specjalnie do tego celu stereotypów i metek. Na rysunku 4. pokazano model danych formularza Akademiki, a na rysunku 5. elementy składowe, zależności między nimi oraz powiązani z innymi modułami formularza Przyznawania. Rozwiązanie to jest ściśle dopasowane do sposobu konstruowania modułów wykorzystywanych w USOS. Przedstawienie przy jego pomocy konstrukcji odmiennych może być niewygodne. Ten efekt jest zamierzony. Wspomaga się tym samym powstawanie projektów o analogicznej strukturze, i ogranicza bałaganiarskie programowanie. Jednocześnie ogólność UML gwarantuje możliwość wprowadzenia rozszerzeń, gdyby w przyszłości zaistniała taka potrzeba.



Organizacja i powiązania formularza Przyznawania

Dodatkową korzyścią z opracowania projektu modułów jest zwiększenie jakości tworzonego kodu. System opracowany na podstawie nawet prostego projektu jest zazwyczaj zdecydowanie solidniej wykonany niż analogiczny powstały metodą programowania ewolucyjnego.

Dokumentacja techniczna i dokumentacja użytkownika

Działający system nie powinien być jedynym efektem prac zespołu programistycznego. Należy również opracować solidną dokumentację techniczną oraz dokumentację użytkownika. Podstawowe cech języka UML przemawiające za wykorzystaniem go w tych celach to:

- zastosowanie łatwej do zrozumienia graficznej notacji,
- możliwość przedstawienia powiązań pomiędzy elementami należącymi do różnych podsystemów,
- użycie tego samego formalizmu co w poprzednich fazach przedsięwzięcia,
- duże możliwości wspomagania komputerowego, w tym automatycznej kontroli spójności modeli.

Podczas rozwoju podsystemu Akademiki wykazano, że nakład pracy niezbędnej do utrzymywania aktualności artefaktów powstałych w wyniku analizy i projektowania jest niewielki w porównaniu z metodami dotychczas stosowanymi. Dodatkowo powstające diagramy doskonale nadają się do wykorzystania w różnego rodzaju publikacjach, np. pracach magisterskich opracowywanych po wdrożeniu kolejnych podsystemów. Poznanie UML przez uczestników projektu ma również pozytywny wpływ na jakość powstającej dokumentacji użytkownika. Należy pamiętać, iż przejrzysty diagram może przeka-

zywać więcej informacji niż kilkustronicowy opis.

Podsumowanie

Metody inżynierii oprogramowania ulegają stałej ewolucji. Uczestnictwo w projekcie rozwijanym w warunkach akademickich jest doskonałym sposobem przetestowania ich w praktyce. Dzięki niezależności od czynników komercyjnych istnieje tu duże pole do eksperymentów.

Specyfika dużego, ciągle ewoluującego, projektu akademickiego sprawia, że duży nacisk kładzie się na organizację pracy w zespole i wspomaganie wymiany doświadczeń pomiędzy jego członkami. W warunkach systematycznie zmieniającego się zespołu jednolity sposób wymiany informacji jest niezmiernie istotny. Język UML idealnie nadaje się do tego celu. Prace prowadzone nad rozwojem podsystemu Akademiki potwierdziły jego przydatność. Wykazano poprawę kontaktów z użytkownikiem. Wzrosły możliwości kontroli nad prowadzonymi pracami przez osoby nadzorujące projekt. Wymiana informacji pomiędzy członkami zespołu stała się łatwiejsza.

Pomyślnie został oceniony sposób rozszerzenia UML o stereotypy umożliwiające projektowanie modułów wytwarzanych przy pomocy narzędzi z pakietu *Oracle Developer*. Jakość powstałego projektu potwierdziło wykonanie części pracy programistycznej przez osobę nieuczestniczącą w pracach analityczno-projektowych, jedynie na podstawie dostarczonych dokumentów.

Zniechęcający może się wydawać dodatkowy nakład pracy konieczny do poznania nowego formalizmu. Jednak ze względu na powszechność wykorzystania UML w przemyśle informatycznym posiadanie praktycznego doświadczenia w jego stosowaniu wydaje się znaczącą zaletą. Warto przypomnieć, że pomysł użycia UML został wysunięty przez samych uczestników projektu, a nie narzucony im przez osoby nadzorujące zespół.

Należy zaznaczyć, iż zainteresowanie metodami inżynierii oprogramowania wśród studentów jest duże i przy dobrym umotywowaniu ich przez nauczycieli akademickich projekt USOS może dostarczyć wiele ciekawych wyników z tej dziedziny.

Bibliografia

[GOLD2003] B. Goldman, *Uniwersytecki System Obsługi Studiów. Standardy*, Instytut Informatyki Uniwersytetu Warszawskiego, Praca magisterska (w przygotowaniu).

[JMD2001] J. Mincer-Daszkiewicz, *Tworzenie produkcyjnego oprogramowania w środowisku akademickim*, 225-236, 2001, Oracle Corporation, III Krajowa Konferencja Inżynierii Oprogramowania - KKIO'2001.

[JMD2002] J. Mincer-Daszkiewicz, *Tworzenie produkcyjnego oprogramowania w środowisku akademickim*, 299-314, 2002, Oracle Corporation, IV

Krajowa Konferencja Inżynierii Oprogramowania - KKIO'2002.

[SRO2003] J. Sroka, *Uniwersytecki System Obsługi Studiów. Akademiki*, Instytut Informatyki Uniwersytetu Warszawskiego, Praca magisterska (w przygotowaniu).

[UML2001] G. Boch, J. Rumbaugh i I. Jacobson, *UML przewodnik użytkownika*, 2001, WNT.