

Programowanie zespołowe

Janusz Jabłonowski

Institut Informatyki, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

Janusz@mimuw.edu.pl

Gdzie kucharek sześć, tam nie ma co jeść.

Prysłowie ludowe

Streszczenie

W niniejszym rozdziale podsumowano doświadczenia uzyskane przez Autora w czasie ostatnich sześciu lat przy prowadzeniu zajęć z Zespołowego Projektu Programistycznego (ZPP) w Instytucie Informatyki UW oraz omówiono przyjęte przy planowaniu tego przedmiotu założenia metodologiczne, przedstawiono krótki opis przeprowadzonych zajęć w poszczególnych latach i przeanalizowano wprowadzane podczas realizacji tego przedmiotu zmiany.

1. Wstęp

W niniejszym rozdziale podsumowano doświadczenia uzyskane przez autora w czasie ostatnich sześciu lat przy prowadzeniu zajęć z Zespołowego Projektu Programistycznego (ZPP) w Instytucie Informatyki UW. Najpierw przedstawiono powody i krótką historię powstania tego przedmiotu (sekcja 1.1). W podrozdziale 2. krótko uzasadniono konieczność prowadzenia takich zajęć na studiach informatycznych. Zasadniczą częścią jest podrozdział 3., w którym omówiono postawione cele i przyjęte założenia przy tworzeniu zajęć z programowania zespołowego. W kolejnym podrozdziale pokazano organizację tych zajęć na naszych studiach. W następnym podrozdziale przedstawiono przegląd zajęć z ZPP w poszczególnych latach - zaprezentowano przykładowe programy i omówiono wprowadzane sukcesywnie zmiany. Ostatni podrozdział zawiera podsumowanie i uwagi końcowe.

Uczenie programowania zespołowego ma w naszym instytucie długą tradycję. Niemalże od początku prowadzenia studiów z informatyki wprowadzano zajęcia laboratoryjne, w ramach których studenci (również autor niniejszej pracy) tworzyli większe programy w zespołach. Zajęcia te przechodziły wiele zmian, bywało nawet, że zanikały na krótkie okresy.

Autor opisuje projektowanie i późniejszą realizację zajęć z Zespołowego Projektu Programistycznego. Ten przedmiot pojawił się w planie studiów dopiero w 1997 roku. To, że autor skupia się jedynie na nim, nie wynika w najmniejszym stopniu z tego, że

poprzednie podejścia uważa za mniej udane - autor niniejszego rozdziału uczestniczył w nich zarówno jako student i jako prowadzący i w obu rolach uważał (i nadal uważa) je za niezwykle wartościowe. Powód skoncentrowania się jedynie na ZPP jest taki, że ten przedmiot stanowi pewną zamkniętą całość, a piszący te słowa miał przyjemność nie tylko stworzyć pierwotny projekt tych zajęć, lecz także nadzorować je przez cały czas ich trwania, aż do chwili obecnej. Oczywiście nie do pominięcia jest tu inspirująca rola dyrekcji Instytutu Informatyki UW, która nie tylko postawiła zadanie stworzenia projektu tych zajęć, ale także aktywnie wspomagała ich tworzenie i dalsze modyfikowanie.

Nieco historii

Na początku lat 90-tych przeprowadzono w Instytucie Informatyki UW istotną reorganizację planu studiów. Jednym z jej celów było ujednoczenie czasu trwania poszczególnych przedmiotów do jednego semestru (przedtem niektóre przedmioty były jednosemestralne, niektóre jednoroczne, co oczywiście komplikowało zarządzanie procesem studiowania). Jednym z efektów tej reorganizacji było zlikwidowanie jednorocznych pracowni programowania na drugim roku studiów, w ramach których studenci pisali programy w zespołach. Aby wypełnić powstałą lukę, na początku roku 1997-ego autorowi niniejszej pracy postawiono zadanie przygotowania projektu zajęć z programowania zespołowego. Miały to być zajęcia jednosemestralne i w przeciwieństwie do poprzednich podejść poświęcone przede wszystkim programowaniu zespołowemu. W poprzednich wydaniach programowanie zespołowe pojawiało się jako jeden z elementów realizacji ogólniejszych zadań, np. nauki nowego języka programowania (np. Moduli-2), czy nauki metod programowania w ogólności. W tym wydaniu punkt ciężkości przesunięto na programowanie zespołowe, czego odzwierciedleniem jest także nazwa przedmiotu.

Jednym z założeń było oparcie tego przedmiotu na programowaniu obiektowym. Jest to dość naturalne, gdyż po pierwsze programowanie obiektowe jest obecnie dominującym nurtem programistycznym, a po drugie jego zalety stają się tym bardziej widoczne z im większym i bardziej złożonym zadaniem ma się do czynienia. Zgodnie z dotychczasowymi doświadczeniami ZPP zaplanowano na drugi rok studiów, a tu na drugi semestr. Jest jasne, że pierwszy rok studiów jest jeszcze zbyt wczesny dla takiego przedmiotu - wtedy studenci jeszcze po prostu za mało wiedzą i mają za mało doświadczenie w programowaniu. Drugi semestr drugiego roku wybrano dlatego, że na pierwszym odbywa się wykład z programowania obiektowego, więc połączenie tych dwu przedmiotów w jednym roku studiów wydawało się bardzo rozsądne. Trzeci rok studiów nie był brany pod uwagę, ze względu na dużą liczbę odbywających się tam laboratoriów. Wreszcie umieszczenie tego przedmiotu na roku czwartym lub piątym oznaczałoby, że studenci kończący studia na poziomie licencjackim nie przechodziliby przez ZPP. A to byłoby niepożądane ze względu na naturę tego przedmiotu i studiów. Studenci kończący je na poziomie licencjackim są tymi, którzy właśnie jak najszybciej chcą zacząć pracę (głównie jako programiści) i nie czują powołania do zgłębiania bardziej teoretycznych obszarów informatyki. Zatem umieszczenie ZPP na jednym z dwu ostatnich lat studiów oznaczałoby, że stracą ten przedmiot ci studenci, którym jest on najbardziej potrzebny.

Zanim przejdziemy do dokładnego omówienia założeń i celów ZPP, krótko podsumujmy powody, dla których uważamy, że jest to niezwykle ważny przedmiot dla studentów.

Dlaczego programowanie zespołowe?

Niezwykle szybki rozwój wymagań stawianych przed inżynierią oprogramowania i samymi programistami oraz ciągle poszerzanie obszaru zastosowań informatyki w naszym codziennym życiu powodują nie tylko to, że ciągle powstaje dużo nowych programów, ale także to, że wielkość i złożoność tych programów stale wzrasta. Typowa współczesna aplikacja musi:

- mieć ładny, wygodny w użyciu, graficzny interfejs (GUI),
- mieć zaawansowany, kontekstowy system podpowiedzi, często z różnorodnymi kreatorami dla typowych zadań,
- móc pracować w środowisku sieciowym, a coraz częściej po prostu jest aplikacją rozproszoną,
- zapewniać ochronę danych, zaczynając od poziomu prostego systemu kont użytkowników z różnymi zakresami praw działania, a kończąc na zaawansowanych protokołach szyfrowania informacji,
- itd., itp.

Wszystkie te żądania wymagają dużego wkładu pracy na poziomie analizy, projektowania i programowania (np. stworzenie dobrego interfejsu użytkownika, takiego który byłby intuicyjny, wygodny i bezpieczny wymaga bardzo starannego zaprojektowania). Zrealizowanie wszystkich tych zadań przez jedną osobę w akceptowalnym terminie (czy w ogóle) jest praktycznie niemożliwe.

Różnorodne zadania składające się na realizację projektu programistycznego spowodowały naturalny podział twórców oprogramowania na analityków, projektantów i programistów. Oczywiście każdy projekt powinien też mieć swojego szefa. Często liczba ról w projekcie jest jeszcze większa (na przykład przy dużych projektach tworzy się oddzielne grupy pracowników zajmujących się utrzymaniem programu, czy obsługą i szkoleniami klientów, itp.). Ale konieczność pracy zespołowej nie wynika tylko z tego, że do wykonania są różne grupy zadań. Każdy z wymienionych rodzajów zadań (może poza kierowaniem) wymaga przy dużym projekcie więcej niż jednej osoby, aby zdążyć z wykonaniem całości na czas. Zatem istnieje nie tylko potrzeba współpracy pomiędzy pracownikami zajmującymi się różnymi zadaniami, ale także pomiędzy pracownikami zajmującymi się tymi samymi bądź podobnymi zadaniami. Ten drugi przypadek oznacza zwykle znacznie bliższą, intensywniejszą i dłużej trwającą współpracę.

Z powyższej dyskusji jasno wynika, że umiejętność pracy w zespole jest niezwykle ważna dla programistów (dla analityków i projektantów oczywiście też). Jest ona ważna także w wielu innych dziedzinach życia, ale to już osobny temat. Oczywiście programowanie zespołowe nie pojawiło się dopiero ostatnio, jak mogłyby sugerować dotychczasowe rozważania. Programy tworzone przez duże zespoły powstawały niemalże od zainicjowania informatyki, ale obecnie konieczność pracy zespołowej jest dużo bardziej oczywi-

sta.

Skoro już wiemy czego chcemy nauczyć, pozostaje jeszcze pytanie: jak to zrobić? Niestety nie jest łatwo dać odpowiedź na to pytanie.

Pierwsza trudność polega na tym, że jest to zagadnienie typowo praktyczne, a nie teoretyczne. To czyni uczenie go dużo trudniejszym. Nauczanie przedmiotu o solidnej teoretycznej podbudowie właściwie sprowadza się do wyboru odpowiednio bogatego zestawu definicji i twierdzeń, który by obejmował istotny podzbiór danej dziedziny. Również weryfikowanie postępów w nauce studentów jest w takiej sytuacji znacznie prostsze. W przypadku programowania zespołowego takie podejście nie jest możliwe. Istnieje oczywiście obszerna literatura na ten temat (choć najczęściej rozszkana po literaturze dotyczącej inżynierii oprogramowania w ogólności), ale nie zawiera ona z oczywistych powodów jednej, zgrabnej teorii, lecz raczej opis doświadczeń autorów i stosowanych bądź zalecanych przez nich heurystyk, dotyczących organizacji pracy w zespole. Płyne stąd wniosek, że najlepszą formą uczenia programowania zespołowego nie są wykłady (choć oczywiście pewien wstępny zasób wiedzy jest konieczny), lecz praktyczne ćwiczenia w postaci laboratoriów lub rozbudowane studia przypadków.

Drugi problem polega na tym, że trudności związane z programowaniem zespołowym stają się tym większe, a więc bardziej interesujące i pouczające, im większy jest tak zespół jak i samo zadanie. To wymaganie oczywiście nie pasuje do struktury studiów akademickich. Jest praktycznie niemożliwe zorganizowanie zajęć trwających dłużej niż rok (chociażby ze względu na co najwyżej roczny system rozliczania studentów). Również zorganizowanie zajęć częściej niż raz na tydzień, przy bardzo obciążonym tygodniowym planie zajęć studentów, jest bardzo trudne, a z kolei praca dużego zespołu nad dużym projektem nie może się toczyć w rytmie 90-cio minutowych spotkań raz na tydzień.

Przyjrzyjmy się więc rozwiązaniu przyjętemu w ramach ZPP. W kolejnych rozdziałach przedstawiono cele, które zostały postawione przy projektowaniu tego przedmiotu, następnie omówiono organizację zajęć, a na koniec zaprezentowano krótki przegląd wprowadzanych zmian.

Przyjęte założenia i postawione cele

Projektując nowy przedmiot (w tym przypadku nie był on zupełnie nowy) staje się przede wszystkim przed zadaniem dokładnego ustalenia celów, które chce się uzyskać prowadząc projektowane zajęcia. Trzeba też określić założenia dotyczące wstępnej wiedzy studentów przyjmowanych na ten przedmiot. Oto lista sformułowanych w 1997 roku celów. Oprócz opisu każdego z nich podano też opisy sposobów realizacji poszczególnych celów. W większości przypadków opisane tu sposoby realizacji zostały zaprojektowane od razu w podanej postaci w roku 1997, czasami następowały drobne ich modyfikacja, jak na przykład rozszerzanie zestawu omawianych podczas zajęć narzędzi, co nie jest już specjalnie zaznaczane w opisach.

Przekonanie studentów o znaczeniu programowania zespołowego

Opis: Tak jak już wspomniano we wstępnej części tej pracy programowanie zespołowe jest obecnie niezwykle istotne dla każdego, kto chciałby zajmować się pracą związaną z programowaniem. Nie jest to jednak wiedza, jaką studenci wynoszą z zajęć pierwszego roku studiów. Wręcz przeciwnie, laboratoria na pierwszym roku są poświęcone samodzielnemu pisaniu małych programów, w celu zaznajomienia studentów z wybranym językiem programowania, strukturami danych z i algorytmami. Nie ma tam ani potrzeby ani możliwości pokazywania zalet pracy zespołowej - po prostu zadania z pierwszego roku są zbyt łatwe, by było warto rozwiązywać je zespołowo.

Realizacja: Należy więc jakoś przekonać studentów o konieczności zapoznania się z pracą zespołową. Zapewne najlepszym sposobem jest przydzielenie zadania na tyle dużego, żeby zrealizowanie go w pojedynkę w wyznaczonym czasie było zniechęcające. Kłopot polega tylko na tym, że musi to być jednocześnie zadanie realizowalne w tym samym czasie przez zespół uczący się wspólnej pracy. Dlatego oprócz samego zadania stosowane są różne formy wstępnej agitacji do programowania zespołowego, poprzez informacje o przedmiocie umieszczone w sieci i dostępne dla naszych studentów, czy też podczas specjalnego spotkania informacyjnego. Po ostatnich zmianach w planie studiów (i przeniesieniu ZPP na trzeci rok) ten problem stał się mniej istotny, gdyż to zadanie powinien realizować przedmiot Inżynieria Oprogramowania na drugim roku.

Pokazanie studentom problemów związanych z programowaniem zespołowym

Opis: Co najmniej tak samo ważnym celem jak poprzedni jest pokazanie studentom, że programowanie zespołowe, aczkolwiek konieczne, nie jest łatwe. Po samodzielnym napisaniu na pierwszym roku kilku/kilkunastu programów studenci wiedzą o problemach związanych z:

- środowiskiem, które nie zawsze jest tak przyjazne, jak by to wynikało z opisu,
- kompilatorem, który nie zawsze daje czytelne komunikaty o błędach,
- językiem, który często ma swoje pułapki,
- algorytmami, które często są trudne, a czasami też kłopotliwe w implementacji.

Lecz studenci nie wiedzą, jakie trudności wiążą się z pracą zespołową.

Realizacja: Dlatego na pierwszym spotkaniu podkreśla się głębokie różnice pomiędzy pisaniem programów samodzielnie, a w zespole. Podkreśla się, że z tego, iż jedna osoba może napisać jakiś program w rok nie wynika, że dwanaście osób napisze go w miesiąc. Zwraca się uwagę na problemy z porozumiewaniem się między członkami zespołu, podkreśla się rolę wspólnych konwencji zapisu projektów czy wspólnych standardów kodowania. Uczula się studentów na niezwiązane z informatyką problemy pracy w zespole, takie jak to, że dobrze być w zespole z solidnymi kolegami, takimi na których można polegać, a na przykład nie jest dobrze być w zespole z więcej niż jednym wybitnym współpracownikiem.

Niestety ta część doświadczeń związanych z programowaniem zespołowym jest też najboleśniejszą - o tych problemach najlepiej się przekonują ci studenci, którzy nie mieli szczęścia trafić do dobrego zespołu. Tym nie mniej, lepiej przejść przez takie, nawet

niemię, doświadczenie podczas studiów i być przygotowanym na te problemy potem w pracy zawodowej.

Pokazanie studentom narzędzi ułatwiających programowanie w zespołach

Opis: Gdy studenci już wiedzą, że praca w zespole nie jest usiana różami, jest najlepszy moment na pokazanie narzędzi tę pracę ułatwiających. Oczywiście jest ich dużo i nie jest możliwe pokazanie ich wszystkich. Co więcej wybór zależy od wybranego języka i środowiska programistycznego. Tym nie mniej co roku przedstawiany jest pewien podstawowy zestaw takich narzędzi:

- Przede wszystkim jakieś narzędzie do zarządzania wersjami. Przy programowaniu w zespole możliwość odtwarzania poprzednich wersji, chociażby po to, żeby się przekonać, czy nowy błąd w programie pochodzi z nowego własnego modułu, czy kolegi jest niezwykle cenna. Na naszych zajęciach standardowo wybierany jest CVS [CVS2003].
- Jakieś środowisko programistyczne. Jego ważnym elementem musi być wsparcie programowania wizualnego, czyli łatwego tworzenia interfejsów użytkownika wraz ze szkieletem aplikacji. Wybór zależy oczywiście od samego języka programowania (np. Sun ONE [Sun2003] dla Javy).
- Ponieważ kładziony jest bardzo duży nacisk na tworzenie dokumentacji, to oczywiście pokazywane są jakieś narzędzia ułatwiające jej tworzenie (np. JavaDoc [JDoc2003] dla tworzenia dokumentacji technicznej w Javie).
- na ile to jest możliwe, zwraca się uwagę na problemy utrzymywania i pielęgnowania kodu, dlatego pokazywane są narzędzia do zarządzania informacjami o zgłaszanych błędach i uwagach (kto zgłosił, opis, kto jest odpowiedzialny za obsłużenie, jaki jest obecny stan, itp.), np. Bugzilla [Bugz2003].

Oczywiście ten zestaw w poszczególnych latach jest rozszerzany o dodatkowe narzędzia (np. do tworzenia dokumentacji projektowej).

Realizacja: w tym przypadku sprawa jest prosta, z mniejszym lub większym powodzeniem wymuszane jest stosowanie przedstawianych narzędzi.

Pokazanie studentom w możliwie najszerszym zakresie pełnego cyklu rozwoju oprogramowania

Opis: Realizacja tego celu jest oczywiście najtrudniejsza do zorganizowania podczas studiów. Dla przykładu: niesłuchanie istotną część życia udanego projektu programistycznego (może nawet najistotniejsza) zaczyna się po przekazaniu końcowej wersji aplikacji klientom. Klienci przekazują swoje uwagi, idee rozszerzeń, zgłaszają błędy, krytykują przyjęte rozwiązania. Na tej podstawie modyfikuje się aktualną wersję lub buduje następną. Jest oczywiste, że w ramach studiów nie ma możliwości dostarczenia studentom dostatecznie dużej liczby klientów, ani odpowiednio dużo czasu na ich obsługę. Mimo wszystko można spróbować uczynić te zajęcia jak najbliższe rzeczywisto-

ści.

Realizacja: Po pierwsze zajęcia nie zaczynają się od zadania studentom gotowej specyfikacji wymagań. Na wstępnych zajęciach przekazywana jest zgrubna informacja o tematach realizowanych w poszczególnych grupach. Na pierwszych zajęciach odbywają się rozmowy ze studentami o tym, co klient chciałby dostać, ale to studenci mają wydobywać od prowadzących informacje, a nie prowadzący przekazywać je studentom. Przez następny tydzień lub dwa studenci mogą zadawać prowadzącym pytania (podczas konsultacji, pocztą elektroniczną, na korytarzu), a następnie mają przedstawić specyfikację wymagań (być może najpierw wizję, to już zależy od realizowanej metodologii projektowej, czego już nie omawia się w tej pracy). Otrzymany dokument jest oceniany przez prowadzącego (i ewentualnie wprowadzane są do niego modyfikacje i uzupełnienia). Wymagana jest pełna dokumentacja projektowa, a każdy dokument jest oceniany przez prowadzącego.

Innym elementem urealnienia pracy nad projektem jest jego końcowa modyfikacja. Po zakończeniu prac nad projektem i **po** jego ocenieniu przez prowadzącego zajęcia zespół otrzymuje zadanie zmodyfikowania gotowej aplikacji. Oczywiście z góry (na jednych z pierwszych zajęć) ustalane jest, jakiego rodzaju mogą to być zmiany. Zmiany są tak dobierane, by przy dobrym zaprojektowaniu aplikacji nie wymagały więcej niż jednego dnia pracy zespołu. Celem tego zabiegu jest uczulenie studentów na konieczność łatwego modyfikowania programów i dbania o to już na poziomie projektu.

Nauczenie studentów przedstawiania swoich rozwiązań i tego czego się nauczyli

Opis: To również jest rzecz, której nie da się nauczyć inaczej niż przez ćwiczenie. A jednocześnie jest to niezwykle cenna (nie tylko dla informatyków) umiejętność. Prezentacja swoich osiągnięć to coś, co studenci będą musieli robić przez całe swoje życie zawodowe. Z kolei prezentowanie tego, czego się nauczyli ma szczególne znaczenie w pracy zespołowej. Gdy zespół dostaje zadanie poznania nowej technologii czy środowiska, to jest sensowne, by zajęli się tym najpierw wybrani członkowie zespołu, a następnie przekazali swoją wiedzę pozostałym. Z tego względu w planie zajęć ZPP od samego początku położono bardzo duży nacisk na prezentacje.

Realizacja: Wszystkie wspomniane wcześniej narzędzia programistyczne są prezentowane przez samych studentów (poszczególne zespoły przygotowują kolejne prezentacje). W ciągu zajęć każdy zespół przygotowuje od 2 do 5 (zależnie od roku) prezentacji.

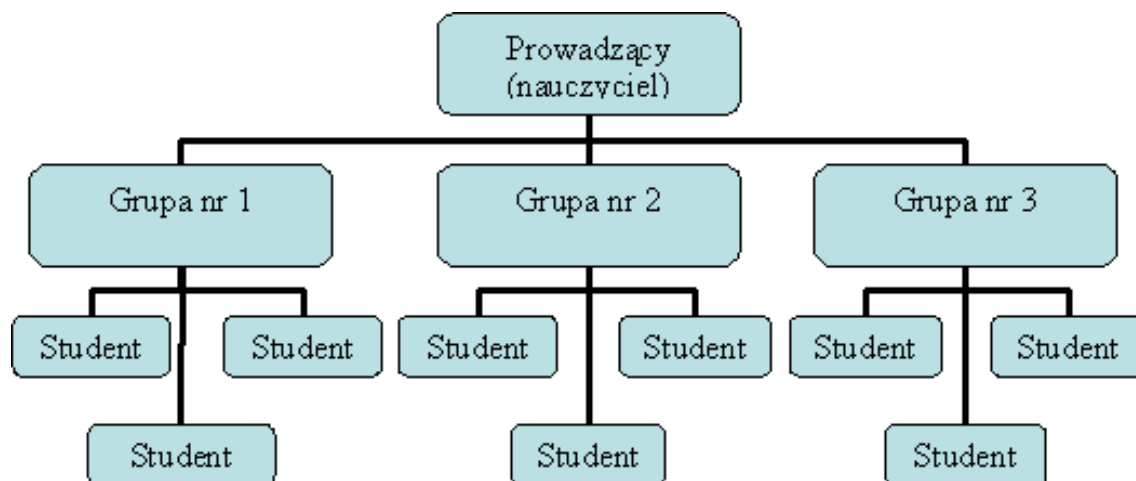
Założenia

W przypadku założeń o początkowej wiedzy studentów sytuacja była bardzo prosta - wynikały one z ułożenia ZPP w ramach planu studiów. Studenci zaczynając te zajęcia muszą mieć wysokie praktyczne umiejętności programowania w małej skali oraz dobrą wiedzę z zakresu projektowania i programowania obiektowego.

Organizacja zajęć

Jedną z trudniejszych decyzji jest ustalenie, z ilu studentów powinien składać się jeden zespół. Im większa ta liczba, tym bardziej realistyczny i ciekawszy może być projekt. Ale z drugiej strony będzie też bardziej niebezpieczny dla biorących w nim udział studentów - z założenia nie mają oni jeszcze doświadczeń w pracy zespołowej.

Aby zapewnić sensowność zajęć i bezpieczeństwo ich zaliczania ustalono liczebność zespołów na poziomie 3-4 osób (początkowo było to 3, obecnie jest 4, zwiększenie wynikało ze zmian w organizacji przedmiotu). Wszyscy studenci (jeden rok, zwykle nieco ponad 100 osób) są podzieleni na grupy laboratoryjne. Każda grupa ma swojego prowadzącego - pracownika dydaktycznego. Każda grupa składa się z 3-5 zespołów. Zwykle wszystkie zespoły w grupie mają to samo zadanie. Prezentacje poszczególnych zespołów są wykonywane w obrębie jednej grupy. Rysunek 1 przedstawia strukturę przykładowej grupy laboratoryjnej.



Rys. 1. Przykładowa struktura jednej grupy laboratoryjnej

Przegląd prowadzonych zajęć z ZPP w latach 1997-2003

W tym rozdziale przedstawiono krótką historię ZPP i zmian w jego strukturze. Podano też tematy przykładowych programów realizowanych w ramach zajęć.

- **1997/1998:** Pierwszy rok prowadzenia ZPP. W różnych grupach były różne zadania, np. napisanie edytora strukturalnego sparametryzowanego składnią i regułami formatowania języka programowania.
- **1998/1999:** Drugi rok ZPP. Tym razem wszystkie grupy otrzymały to samo zadanie: stworzenie gry o bogatym interfejsie graficznym i rozbudowanej hierarchii obiektów biorących udział w grze.
- **1999/2000:** Trzeci rok ZPP i jednocześnie ostatni przed dużymi zmianami. Zadanie postawione przed studentami było tym razem bardzo praktyczne: napisanie podsystemu wydziałowego systemu wspomagającego zarządzanie studiami. Zadaniem podsystemu było rejestrowanie studentów na ćwiczenia i zajęcia laboratoryjne.

- **2000/2001:** W związku ze zmianami w strukturze planu studiów, w tym roku nie było ZPP. Zostało ono przeniesione na trzeci rok. Zaletą tego rozwiązania jest to, że obecnie ZPP jest po zajęciach z Inżynierii Oprogramowania odbywających się na drugim roku. Wymagało to przesunięcia jeszcze kilku zajęć (żeby nie było zbyt wielu pracowni na trzecim roku). Zaszły również zmiany w samym ZPP: stało się ono przedmiotem składającym się z dwu semestralnych części (formalnie ZPP I i ZPP II), za to zajęcia odbywają się teraz co dwa tygodnie. Pierwszy semestr jest obecnie poświęcony tworzeniu dokumentacji projektowej, drugi zaś implementacji. Wprowadzono końcowe, publiczne prezentacje projektów. Dzięki wydłużeniu ogólnego czasu trwania zajęć uznano, że można bezpiecznie zwiększyć liczbę studentów w zespole do 4.
- **2001/2002:** Pierwszy rok nowego ZPP. Każda grupa realizuje inne zadanie, np. napisanie systemu wspomagającego wyprowadzanie dowodliwie poprawnych programów.
- **2002/2003:** w trakcie realizacji.

Podsumowanie

Nie ma wątpliwości co do tego, czy należy uczyć programowania zespołowego. Jest oczywiste, że tak - autor podał przemawiające za tym argumenty. Lecz pozostaje otwartym pytanie jak to robić. Przedstawiono tu kilka problemów związanych z realizacją takiego kształcenia. Pokazano też, jak te problemy próbuje się rozwiązywać w Instytucie Informatyki Uniwersytetu Warszawskiego. Z pewnością nie są to rozwiązania idealne - jak widać z załączonej historii ZPP, struktura tego przedmiotu wciąż ewoluuje. Wydaje się jednak, że te rozwiązania są już na tyle ciekawe, by warto je było przedstawić publicznie. Autor niniejszego rozdziału jest głęboko przekonany, że zajęcia z tego przedmiotu są nie tylko interesujące dla studentów, ale że, przede wszystkim, ułatwią im one pomyślną pracę zawodową w przyszłości.

Bibliografia

- [Bugz2003] URL: <http://www.bugzilla.org>.
[CVS2003] URL: <http://www.cvshome.org>.
[JDoc2003] URL: <http://java.sun.com/javadoc>.
[Sun2003] URL: www.sun.com/software/sunone.