



Wstep

W życiu wielokrotnie miałem szczęście. Jednym ze szczęśliwych zrządeń mojego losu było to, że znalazłem się we właściwym miejscu i miałem właściwą wiedzę, aby w 1997 roku napisać pierwsze wydanie tej książki. Przedtem chaotyczny świat modelowania obiektowego dopiero zaczynał się przekształcać w zunifikowany język modelowania (UML). Od tego czasu UML stał się standardem graficznego projektowania oprogramowania, i to nie tylko obiektowego. Moje szczęście polega też na tym, że książka ta była najpopularniejszą pozycją na temat UML-a i sprzedała się w liczbie ponad ćwierci miliona egzemplarzy.

No tak, to niewątpliwie przyjemne dla mnie, ale czy Wy powinniście kupić tę książkę?

Chciałbym podkreślić, że jest to książka krótka. Jej celem nie jest szczegółowe wyjaśnienie każdego aspektu UML-a, który przez lata bardzo się rozrósł. Moją intencją było znalezienie najbardziej przydatnej części UML-a i opowiedzenie tylko o niej. Obszerniejszy podręcznik da więcej informacji, ale jego przeczytanie zajmie też więcej czasu. A czas to Wasza największa inwestycja, którą poczynicie, czytając tę książkę. Dbając o to, aby książka była mała, musiałem poświęcić wiele czasu, aby wybrać najlepsze fragmenty i oszczędzić Wam konieczności dokonywania tego wyboru samodzielnie.

Jednym z argumentów za kupieniem tej książki jest to, że można się z niej nauczyć języka UML. Ponieważ jest to książka krótka, szybko pozwoli wdrożyć się w najistotniejsze elementy języka. Po opanowaniu tej wiedzy będzie można pójść dalej, korzystając z bardziej szczegółowych podręczników, takich jak [7] lub [39].

Niniejsza książka może też służyć jako poręczny przewodnik po najczęściej używanych elementach UML-a. Chociaż nie ma w niej wszystkiego, to jednak mało waży i znacznie wygodniej nosić ją przy sobie, niż większość innych książek o UML-u.

Uczciwie ostrzegam, że jest to książka tendencyjna. Z obiektami pracowałem przez wiele lat i mam pewne własne przemyślenia na temat tego, co dobrze działa, a co nie. Każda książka odzwierciedla opinie autora i ja też nie zamierzam ukrywać swoich. Jeśli szukacie książki o większym stopniu obiektywności, to być może będziecie musieli poszukać innej pozycji.

Wiele osób powiedziało mi, że ta książka jest dobrym wstępem do obiektów, ale nie napisałem jej z takim zamiarem. Jeśli poznaliście podstawy projektowania obiektowego, to sugeruję książkę Craiga Larmana [30].

Wiele osób zainteresowanych UML-em używa różnych narzędzi. W tej książce skoncentrowałem się na standardowym i konwencjonalnym użyciu UML-a i nie opiszę szczegółów na temat obsługi poszczególnych narzędzi. Chociaż UML rozwiązał problem mnogości różnych notacji, to jednak wciąż pozostaje wiele denerwujących rozbieżności między tym, co pokazują i na co pozwalają narzędzia podczas rysowania diagramów UML.

Nie napisałem tu wiele o architekturze wyprowadzanej z modelu, czyli MDA (ang. Model Driven Architecture). Chociaż wiele osób uważa UML i MDA za to samo, to jednak często twórcy oprogramowania używają UML-a, nie interesując się MDA. Jeśli chcecie nauczyć się czegoś więcej o MDA, to radzę zacząć od tej

książki, aby najpierw uzyskać ogólny obraz UML-a, a następnie przejść do książki omawiającej temat MDA bardziej szczegółowo.

Głównym tematem tej książki jest oczywiście UML, ale dodałem też trochę materiału o kartach CRC, które są bardzo wartościowym narzędziem w modelowaniu obiektowym. UML jest tylko częścią tego, czego potrzeba do pracy z obiektami i myślę, że przedstawienie również innych technik jest bardzo ważne.

W tak krótkiej książce nie jest możliwe szczegółowe omówienie, jak ma się UML do kodu źródłowego, w szczególności dlatego, że nie istnieje żadna standardowa metoda utworzenia takiego powiązania. Postarałem się jednak pokazać najczęstsze techniki kodowania przy implementowaniu fragmentów UML-a. Moje przykłady są w Javie i C#, ponieważ uznałem, że te języki są ogólnie znane. Nie są to jednak moje ulubione języki, bo za dużo pracowałem w Smalltalku, żeby tak było!

## Po co zawracać sobie głowę UML-em

Graficzne języki projektowania istnieją już jakiś czas. Dla mnie ich podstawowa wartość tkwi w łatwości ich rozumienia i w ich możliwościach komunikacyjnych. Dobry diagram często może pomóc w przekazaniu idei projektu, szczególnie wtedy, gdy trzeba zrezygnować z wielu szczegółów. Diagramy mogą też ułatwić zrozumienie jakiegoś systemu informatycznego lub procesu ekonomicznego. Podczas dyskusji w zespole, diagramy pomagają zarówno zrozumieć pomysły, jak i przenieść to zrozumienie na cały zespół. Chociaż nie zastąpią one, przynajmniej na razie, tekstowych języków programowania, to są one dla nich pomocnym uzupełnieniem.

Wiele osób wierzy, że w przyszłości graficzne techniki odegrają dominującą rolę w rozwijaniu oprogramowania. Ja jestem bardziej sceptyczny, ale na pewno warto wiedzieć, na co te techniki pozwalają, a na co nie.

Znaczenie UML-a wśród innych notacji graficznych bierze się z jego powszechnego użycia w społeczności twórców programowania obiektowego. UML stał się też popularną techniką poza tym kręgiem.

## Struktura książki

Rozdział 1. stanowi wstęp do UML-a — mówi o tym, czym jest UML, jakie może mieć zastosowania i jak powstał.

Rozdział 2. omawia proces tworzenia oprogramowania. Mimo że jest on całkowicie niezależny od UML-a, to myślę, że zasadniczą sprawą jest zrozumienie procesu w celu dostrzeżenia kontekstu dla takiego narzędzia, jak UML. W szczególności ważne jest zrozumienie roli iteracyjnego tworzenia oprogramowania, które było podstawowym podejściem do procesu dla większości twórców oprogramowania obiektowego.

Pozostałe rozdziały książki przeznaczyłem na opis poszczególnych typów diagramów w UML-u. Rozdziały 3. i 4. opisują dwie najbardziej użyteczne części UML-a: diagramy klas (zbiór podstawowy) i diagramy sekwencji. Mimo że książka

jest cienka, to wierzę, że techniki opisane w tych rozdziałach, okażą się najbardziej przydatne. UML jest wielkim i wciąż rosnącym tworem, ale nie potrzebujemy go całego.

Rozdział 5. szczegółowo omawia mniej podstawowe, ale równie użyteczne części diagramów klas. Rozdziały od 6. do 8. opisują trzy przydatne diagramy, które rzucają więcej światła na strukturę systemu: diagramy obiektów, diagramy pakietów i diagramy wdrożenia.

Rozdziały od 9. do 11. zawierają trzy dalsze techniki związane z zachowaniem: przypadki użycia systemu, diagramy stanów (choć oficjalnie są zwane diagramami stanów maszyn, to ogólnie nazywa się je diagramami stanów) i diagramy czynności. Rozdziały od 12. do 14. dotyczą mniej ważnych diagramów, podałem więc tylko po krótkim przykładzie z wyjaśnieniem.

Na specjalnej wkładce znajduje się streszczenie najbardziej użytecznych fragmentów notacji. Często słyszałem opinię, że te diagramy są najbardziej wartościową częścią mojej publikacji — warto je zawsze mieć pod ręką.

## Zmiany w trzecim wydaniu

Jeśli macie wcześniejsze wydania tej książki, to pewnie zastanawiacie się, jakie zmiany zaszły w nowym wydaniu i, co najważniejsze, czy w ogóle warto je kupić.

Podstawowym czynnikiem, który zdecydował o przygotowaniu trzeciego wydania było pojawienie się wersji 2.0 języka UML. W UML 2.0 dodano wiele nowych narzędzi, w tym kilka nowych typów diagramów. Nawet znane już diagramy zyskały mnóstwo nowych elementów, takich jak ramki interakcji w diagramach sekwencji. Aby poznać te zmiany, bez przedzierania się przez specyfikację (czego zdecydowanie nie polecam!), wystarczy uważnie przeczytać niniejszą książkę — stanowi ona dobry ich przegląd.

Przy okazji postanowiłem gruntownie przerehabilitować większość książki, uaktualniając tekst i przykłady. Zawarłem dużą część swojej wiedzy, którą zdobyłem ucząc i używając UML-a przez pięć ostatnich lat. Zatem duch ultracienkiej książki o UML-u pozostał nietknięty, ale większość słów została napisana od nowa.

Przez te ostatnie lata włożyłem wiele wysiłku w to, aby ta książka była wciąż bardzo aktualna. Zrobiłem wszystko, aby nadażyć za zmianami zachodzącymi w UML-u. Niniejsza książka jest oparta na szkicach języka UML 2.0, które zostały zaakceptowane przez stosowny komitet w czerwcu 2003 roku. Nie jest prawdopodobne, aby zaszły jeszcze jakieś zmiany w następnych bardziej formalnych głosowaniach komitetu, więc uważam że UML 2.0 jest już wystarczająco stabilny, aby moja książka mogła trafić do druku. Informacje na temat dalszych uaktualnień będą zamieszczać na swojej stronie WWW (<http://martinfowler.com>).

## Podziękowania

Do sukcesu tej książki przyczyniło się wiele osób. Moje pierwsze podziękowania należą się Carterowi Shanklinowi i Kendallowi Scottowi. Carter był redaktorem w Addison-Wesley, który zasugerował mi napisanie tej książki. Kendall Scott po-

mogł mi przygotować dwa pierwsze wydania, pracując nad tekstem i stroną graficzną. Obaj dokonali niemożliwego, wydając pierwszą wersję w niesłychanie krótkim czasie, utrzymując przy tym wysoką jakość, z której słynie Addison-Wesley. A przecież na początku istnienia UML-a, gdy jeszcze nie było mowy o jakiegokolwiek stabilności, zmian trzeba było dokonywać bez przerwy.

Jim Odell przez długi czas był moim mentorem i przewodnikiem na początku mojej kariery. Był też głęboko zaangażowany w zagadnienia techniczne i sprawy personalne i bardzo się przyczynił do tego, aby zadufani w siebie metodologowie poszli na kompromis i zgodzili się na wspólny standard. Jego wkład w tę książkę jest bardzo duży, ale trudny do zmierzenia i myślę, że jest podobnie z jego wkładem w UML.

UML jest tworem standardowym, a ja osobiście mam alergię na standardy. Aby wiedzieć co się dzieje, potrzebuję więc siatki agentów, którzy informują mnie na bieżąco o wszelkich knowaniach komitetów. Bez tych agentów, wśród których są Conrad Bock, Steve Cook, Cris Kobryn, Jim Odell, Guus Ramackers i Jim Rumbaugh, nic bym nie wskórał. Oni wszyscy dawali mi bardzo pomocne rady i cierpliwie odpowiadali na głupie pytania.

Grady Booch, Ivar Jacobson i Jim Rumbaugh są znani jako „Trzej muszkietery”. Mimo rubasznych docinków, które im posyłałem przez wiele lat, uważam, że dali mi ogromne wsparcie i zachętę do napisania tej książki. Pamiętajcie, że te docinki były zwykle wyrazem głębokiego uznania.

Kluczem do jakości książki są recenzenci. Nauczyłem się od Cartera, że tych nigdy za wielu. Recenzentami poprzednich wydań tej książki byli Simmi Kochhar Bhargava, Grady Booch, Eric Evans, Tom Hadfield, Ivar Jacobson, Ronald E. Jeffries, Joshua Kerievsky, Helen Klein, Jim Odell, Jim Rumbaugh i Vivek Salgar.

Trzecie wydanie również może się poszczycić pokaźną grupą recenzentów:

Conrad Bock	Craig Larman
Andy Carmichael	Steve Mellor
Alistair Cockburn	Alan O’Callaghan
Steve Cook	Jim Odell
Luke Hohmann	Guus Ramackers
Pavel Hruby	Jim Rumbaugh
Jon Kern	Tim Seltzer
Cris Kobryn	

Wszystkie te osoby poświęciły swój cenny czas na czytanie rękopisu i każda z nich znalazła przynajmniej jeden poważny błąd. Składam im wszystkim szczerze podziękowania. Za wszelkie błędy, które pozostały, całkowitą odpowiedzialność ponoszę tylko ja. Gdy znajdę jeszcze jakieś usterki, wówczas zamieszczę w serwisie *martinfowler.com* stosowną erratę.

Najważniejsza grupa ludzi, która zaprojektowała i napisała specyfikację UML-a składała się z następujących osób: Don Baisley, Morgan Björkander, Conrad Bock, Steve Cook, Philippe Desfray, Nathan Dykman, Anders Ek, David Frankel, Eran Gery, Oystein Haugen, Sridhar Iyengar, Cris Kobryn, Birger Moller-Pedersen, James Odell, Gunnar Övergaard, Karin Palmkvist, Guus Ramackers, Jim Rumbaugh, Bran Selic, Thomas Weigert i Larry Williams. Bez nich nie miałbym o czym pisać.

Pavel Hruby przygotował doskonałe szablony Visio, których intensywnie używam do diagramów UML. Można je pobrać ze strony <http://phruby.com>.

Wiele osób przekazywało mi przez Internet i osobiście pewne sugestie, zadawało pytania i wytykało błędy. Nie byłem w stanie pamiętać Was wszystkich, ale szczerze Wam dziękuję.

Sprzedawcy w mojej ulubionej księgarni technicznej SoftPro w Burlington w stanie Massachusetts pozwalali mi spędzać u siebie wiele godzin i poszukiwać książek o tym, jak ludzie stosują UML w praktyce. Przy okazji częstowali mnie bardzo dobrą kawą.

Redaktorem prowadzącym trzeciego wydania był Mike Hendrickson. Kim Arney Mulcahy zarządzała projektem, ale też zajmowała się porządkowaniem diagramów i ich układem. John Fuller w Addison-Wesley był redaktorem technicznym, a Evelyn Pyle i Rebecca Rider pomogły w przepisywaniu pracy i korekcie. Dziękuję im wszystkim.

Cindy była przy mnie cały czas, gdy nieprzerwanie pisałem książki. Dochody z tych książek zasadziła w naszym ogrodzie.

Moi rodzice zadbali o moje wykształcenie, a to z niego wynikło wszystko inne.

Martin Fowler  
Melrose, Massachusetts  
<http://martinfowler.com>