Jarosław Żeliński

ANALIZA BIZNESOWA

Praktyczne modelowanie organizacji

one press
Helion

Redaktor prowadzący: Barbara Gancarz-Wójcicka
Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Fotografia na okładce została wykorzystana za zgodą shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *onepress@onepress.pl*
WWW: *http://onepress.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://onepress.pl/user/opinie?sfomod
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-4880-1

Printed in Poland.

Spis treści

Rozdział 1.

Po co nam pan analityk?

Przecież analizę zrobimy sami, a jak nie — to zrobi to dostawca. W ten sposób często rozpoczyna się tak zwana droga do klęski. Jednym z typowych listów inicjujących współpracę z wieloma moimi klientami był ten:

> *Planujemy wdrożenie nowego oprogramowania, jednak chcemy tym razem uniknąć presji ze strony dostawcy oprogramowania. Poprzednim razem dostawca oprogramowania upraszczał sobie pracę już na etapie analizy, którą wykonywał sam. Analiza wymagań polegała na wywiadach i warsztatach grupowych, a skończyła się zwykłym opisem, tabelkami i kilkoma nieczytelnymi schematami. Podczas analizy i wdrożenia stale wmawiano nam, że „tego nie można", „to należy uprościć" albo „tak się nie robi", my zaś nie potrafiliśmy tego ocenić. Wiele naszych potrzeb odkrywaliśmy dopiero podczas instalacji kolejnych prototypów, zaś dostawca unikał wprowadzania zmian i obiecanych dostosowań. Dostaliśmy w efekcie nieprzydatne i niezgodne z biznesowymi potrzebami oprogramowanie. Czy może nam Pan tym razem pomóc?*

Czy to propaganda? Niestety nie. W czym tkwi problem? A mianowicie w metodzie prowadzenia analizy i potem w sposobie formułowania specyfikacji wymagań. Łatwo powiedzieć: zebrać wymagania. Powszechnie uważa się, że analiza wymagań wobec oprogramowania to prosty, ale pracochłonny proces prowadzenia wywiadów i skrzętnego zapisywania tego, czego oczekuje przyszły użytkownik. Nic bardziej błędnego — ten sposób jest najgorszy.

Wśród wielu tego typu metod są te najprostsze i najbardziej ryzykowne, a mianowicie: wywiady, ankiety, sesje JAD (*Joint Application Development*, metoda polegająca na stałym aktywnym udziale zamawiającego w tworzeniu oprogramowania). Sama ich istota zakłada, że klient wie, czego chce, należy to tylko z niego wyciągnąć i uporządkować.

Być może czasami tak jest, ale stosowanie tego rodzaju metod z reguły kończy się klasycznym stwierdzeniem klienta, wygłaszanym na koniec projektu:

> *Tak, dostarczyli nam państwo dokładnie to, co chcieliśmy, ale zupełnie nie to, czego potrzebujemy.*

Dlaczego tak się dzieje, choć wielu (analityków) tak bardzo się stara? Ta książka, zbiór esejów z mojego bloga, jest próbą opisu tego, jak może wyglądać skuteczna analiza i specyfikacja wymagań. Zebrałem tu wybrane opisy sformalizowanych metod analizy. Ale po co tyle zachodu? Dlaczego upieram się przy tych śmiesznych formalizmach, skoro można po ludzku zapisać, co powiedział zamawiający, a potem zamienić to na wypunktowane listy lub wiersze ładnej tabeli? Najlepiej jeszcze, gdy liczą sobie najmniej kilkaset pozycji. Jednak tak właśnie pracuje „analityk dyktafon".

Dobra specyfikacja wymagań dla oprogramowania to wynik analizy i modelowania organizacji, kompromis pomiędzy możliwościami technologii a celami biznesowymi wdrożenia. Nie ma tu znaczenia, czy będzie to gotowy system ERP, czy dedykowane rozwiązanie.

Czym grozi wykonywanie analizy w sposób, w jaki robi to „analityk dyktafon"?

Poniżej kilka statystyk oraz mój komentarz co do przyczyn stwierdzonych problemów. A żeby zdać sobie sprawę, że sprawa jest poważna, wystarczy wiedzieć, że:

> *Błędy w wymaganiach to jedna trzecia wszystkich defektów w projektach.*
>
> (Źródło: Dean Leffingwell, Don Widrig, *Managing Software Requirements*, 1999)

Produktem analizy wymagań jest specyfikacja wymagań i, jak już wspomniałem, można je zbierać metodami „rejestracyjnymi" (odpytywanie klientów) oraz „badawczymi". Te drugie to kolejno: analiza i model organizacji klienta, identyfikacja celów biznesowych i czynników wpływających na ich osiągnięcie, projekt rozwiązania: co ma zostać dostarczone, opis projektu, czyli specyfikacja produktu. Przeprowadzenie wywiadów jest łatwe, a pełna analiza trudna. Nietrudno się więc domyślić, które metody stosuje się najczęściej. I mamy główną przyczynę problemu, to właśnie:

> *Procesy związane ze zbieraniem wymagań są źródłem większości poważnych problemów z jakością.*
>
> (Źródło: Gerald Weinberg, *Quality Software Management*, 1997)

Jakość specyfikacji wymagań wyraża się między innymi w jej kompletności, spójności i niesprzeczności. Gdy dokument wymagań jest niskiej jakości, to typowym objawem jest tak zwane odkrywanie wymagań w trakcie trwania projektu, czyli zmiany zakresu projektu, a:

> *Zmiany zakresu są jednym z najczęstszych źródeł dodatkowych kosztów w projektach i często prowadzą do porzucenia projektu.*
>
> (Źródło: Capers Jones, *Assessment and Control of Software Risks*, 1994)

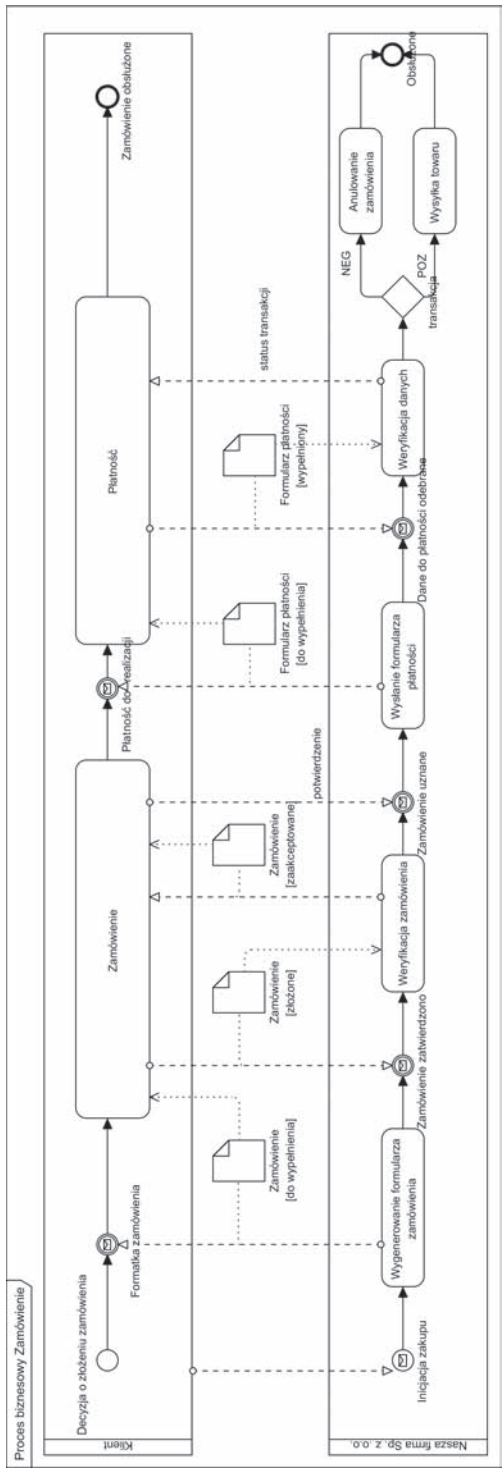
Rozdział 12.

Przypadki użycia i granice systemu

Pora opisać przypadki użycia i granice systemu. Często stosowane do opisu wymagań tak zwane *user story* (w polskim piśmiennictwie zwane *historyjkami użytkownika*) stwarza ryzyko niejednoznaczności opisu. Narzędziem niwelującym to ryzyko jest model procesu biznesowego (jaki opisał *Zamawiający* metodą *user story*). Tworzenie modelu ma dwa zadania: weryfikację spójności i kompletności opisu *Zamawiającego* oraz stworzenie podstawy do określenia zakresu projektu i wyspecyfikowania wymagań funkcjonalnych, czyli tak zwanych przypadków użycia. Czy tak się zawsze postępuje? Ja z zasady, zgodnie z moją metodyką opracowaną na bazie doświadczeń i literatury, traktuję KAŻDY opis dostarczony przez *Zamawiającego*, nawet w postaci strukturalnej (tabele, listy itp.), jako takie właśnie ryzykowne *user story*.

Czy to oznacza brak zaufania do *Zamawiającego*? Przecież ten system jest przeznaczony dla niego, więc *Zamawiający* powinien umieć określić, czego potrzebuje. Hm... Każdy z nas potrafi powiedzieć, jaki chce mieć samochód i do czego jest mu potrzebny, ale czy to znaczy, że potrafi wyspecyfikować konstrukcję tego samochodu? (Mój mechanik jest bardziej dosadny, zawsze mówi: nie ma nic gorszego od faceta, któremu wydaje się, że zna się na samochodach). Jeśli chcemy kupić gotowe oprogramowanie o standardowej, powszechnie stosowanej funkcjonalności, w zasadzie żaden analityk nie jest nam potrzebny. Jeżeli jednak chcemy coś, co choć troszkę ma być dostosowane do specyfiki naszego biznesu i nie jest trywialne, należy do tego podejść z większa rezerwą. Wracając do wcześniejszego przykładu: nie mówimy już bowiem o przeciętnym samochodzie, ale o samochodzie sportowym, którym wystartujemy w zawodach. Ten sport to wolny rynek i starcie z konkurencją. Może nie zawsze jest to Formuła 1, ale też prawie nigdy nie jest to też jazda spacerowa.

Wróćmy do naszego modelu procesów. Ten powstały na bazie opisu *Zamawiającego* pozwolił znaleźć luki i niespójności, jest jednak zbyt szczegółowy. To rzadki u mnie przypadek analizy *bottom-up* (od szczegółu do ogółu), jednak pierwsza sztywna zasada analityka mówi: nie istnieją sztywne zasady analizy i projektowania. Po drobnych poprawkach i dokonaniu uogólnień model wygląda tak (rysunek 12.1):



Rysunek 12.1. Model procesów

Wprowadzone zmiany polegają na dwóch uogólnieniach w obszarze klienta: *Zamówienie* oraz *Płatność* (szczegółowe scenariusze stały się procedurami z serią czynności, a tego nie chcemy, gdyż kończymy na zasadzie: jeden cel procesu — jeden proces). Dlaczego tak? Na poziomie opisu *user story* najczęściej opisy zawierają szczegóły związane z aktualnym (znanym *Zamawiającemu* z autopsji) sposobem pracy. W zakresie projektu jednak pozostanie jeden proces (czynność): *Zamówienie*, oznaczony stereotypem *przypadek użycia* (stereotypy nie są częścią notacji BPMN, stosuję je do wskazywania powiązań pomiędzy obiektami w modelach BPMN i UML).

Warto tu podać pewne wyjaśnienie. Z teorii komunikacji wiadomo, że zasadniczo nie można (niewprawiony i nieświadomy zjawiska człowiek praktycznie nie jest w stanie) uniknąć skażenia każdego swojego przekazu własnym doświadczeniem, czyli znaną sobie historią. Dlatego pierwszym krokiem jest uogólnienie treści *user story* do poziomu abstrakcji: co i po co jest robione, gdyż metody analizy bazujące na faktach zamiast na opisie *Zamawiającego* są wolne od tej przypadłości. Można je jednak stosować tylko w pewnych warunkach (o tym w dalszej części książki).

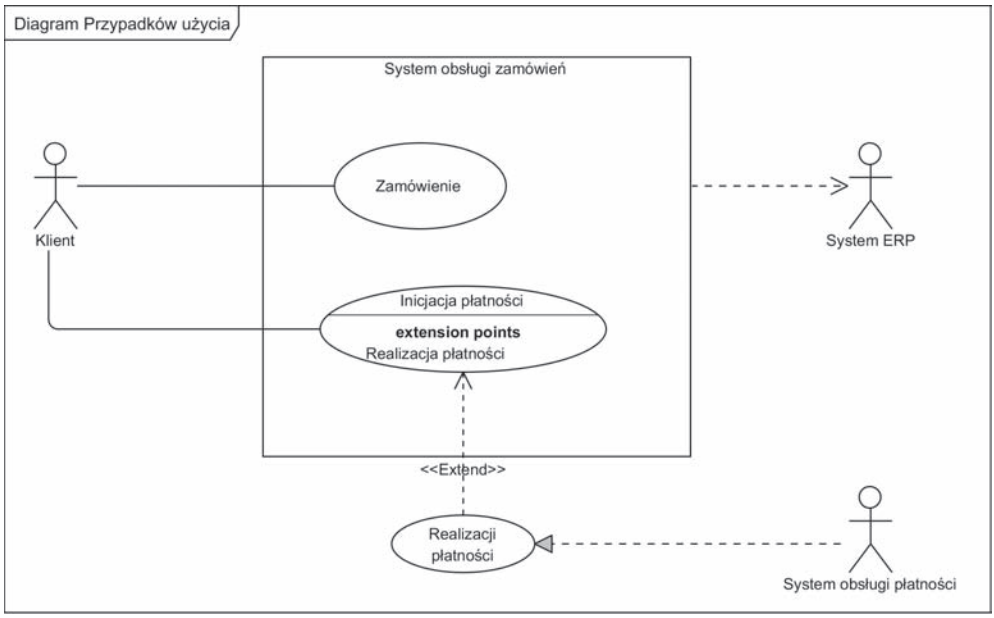
Po stronie *Zamawiającego* (użytkownika, dalej zwanego także aktorem) zastosowano uogólnienie bazujące na definicji procesu biznesowego: proces biznesowy to czynność lub logicznie powiązany łańcuch czynności przekształcający produkt na swoim wejściu w produkt (jego stan) na wyjściu. Różnica pomiędzy tymi produktami stanowi wartość biznesową wnoszoną przez dany proces.

Po stronie *Klienta* mamy teraz tylko dwa takie procesy (czynności): *Zamówienie* oraz *Płatność*. Czynności po stronie dostawcy (Nasza Firma Sp. z o.o.) zaniedbujemy, gdyż system ma wprowadzać automatyzację, więc wdrożenie oprogramowania wyeliminuje z obsługi *Zamówień elektronicznych* człowieka.

Kolejną sprawą są granice systemu. Przyjmijmy dwa założenia, które wydają się bardzo rozsądne: firma posiada system ERP oraz płatności elektroniczne będą przedmiotem outsourcingu. Tak więc wymagania funkcjonalne dla systemu to:

1) przyjmowanie zamówień,
2) przyjmowanie płatności drogą elektroniczną,
3) integracja z ERP: system ten odpowiada za księgowanie faktur i zarządzanie magazynem,
4) wykorzystanie zewnętrznego operatora płatności elektronicznych.

Powyższe wydaje się rozsądne i bywa często spotykane (staram się, by ten przykład nie odbiegał przesadnie od rzeczywistości ;)). Warto, aby wymagania były sformułowane zwięźle, ale i nie wykraczały poza opis działań, które powinny być możliwe dzięki nowemu systemowi. Tak więc diagram przypadków użycia, jaki utworzyłem, wygląda tak (rysunek 12.2):



Rysunek 12.2. Diagram przypadków

Czytelnikowi należy się kilka słów na temat tego diagramu. W zasadzie mamy jeden przypadek użycia: *Zamówienie*. Płatność jest realizowana w ramach outsourcingu przez zewnętrzny system (aktor *System obsługi płatności*, strzałka z pełnym grotem skierowana w stronę przypadku użycia: *Operator płatności* realizuje ten przypadek użycia). Płatność jest inicjowana (*Extend*) w *Systemie*. System ERP odpowiada za faktury i magazyn (*Zamówienie* zależy od tego ERP — strzałka przerywana skierowana do ERP). Przypadek użycia *Płatności* leży poza granicami naszego systemu (czyli poza zakresem projektu!). Każdy przypadek użycia powinien zostać udokumentowany co najmniej trzema elementami:

1) opisem stanu początkowego (warunków początkowych),
2) scenariuszem,
3) opisem oczekiwanego stanu końcowego.

Kontekst stanowi model procesów, więc nie musimy tu pisać, czym są poprzedzane i jakich mają następców poszczególne przypadki użycia (diagram ten nie służy do modelowania procesów!). Wystarczy w modelach zachować tak zwane *traceability* (śladowanie). Osobiście stosuję prostą zasadę: nazwa przypadku użycia jest taka sama jak czynności w modelu procesów, z której został wyprowadzony. Nazwy te są unikalne. Nie musimy także tworzyć diagramów scenariuszy nadrzędnych łańcuchów przypadków użycia, bo zastępuje je właśnie model BPMN. Na koniec jedna uwaga: przypadek użycia powinien być samowystarczalny, innymi słowy, musi mieć sens jako sam jeden (sam z siebie służy do wykonania jakiejś logicznej czynności dającej przydatny produkt).

Scenariusz przypadku użycia najprościej i najskuteczniej jest opisywać w postaci dialogu *Aktor <—> System*. Ja najczęściej stosuję prostą tabelę z kolumnami *Aktor* i *System*:

AKTOR	SYSTEM
Inicjuje pracę.	Wyświetla listę produktów.
Zaznacza wybrane produkty i zatwierdza przyciskiem „OK".	Wyświetla treść zamówienia.
Potwierdza zamówienie.	Wyświetla dane o płatności.
Wybiera system płatności i zatwierdza przyciskiem „OK".	Kończy i ewentualnie przekazuje sterowanie.

Coraz częściej spotykam (i sam stosuję) też następującą konwencję:

1. Inicjacja pracy.

2. SYSTEM wyświetla listę produktów.

3. Zaznaczenie wybranego produktu i kliknięcie przycisku „OK".

4. SYSTEM wyświetla treść zamówienia.

5. Potwierdzenie zamówienia.

6. SYSTEM wyświetla dane o płatności.

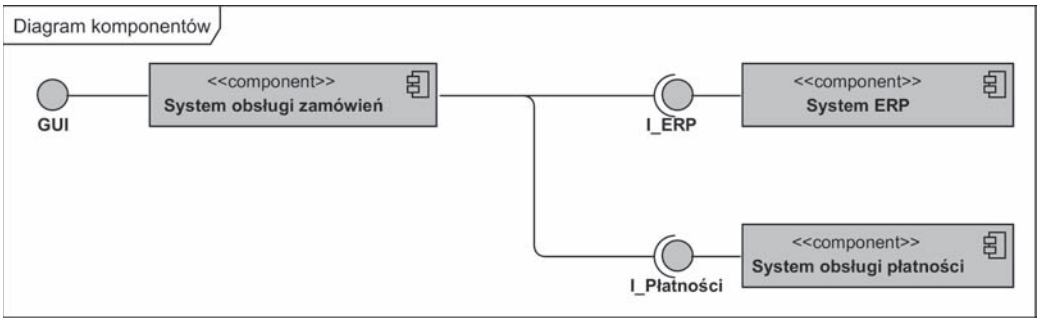
7. Wybór systemu płatności i zatwierdzenie przyciskiem „OK".

8. SYSTEM kończy i ewentualnie przekazuje sterowanie.

Ostatni krok to ewentualne przekazanie sterowania do systemu obsługi płatności elektronicznych, jeśli *Aktor* zaznaczył taką opcję (alternatywą jest tylko wydruk formularza i klasyczny przelew bankowy).

Tworzenie diagramów przypadków użycia w zasadzie warto traktować tylko jako specyfikację koncepcji, zachowując ich zrozumiałość dla laika. „Bąbelki" przypadków użycia powinny być tylko specyfikacją funkcjonalną i niczym ponadto. Ten diagram i tak nie zastąpi diagramu klas, sekwencji czy tym bardziej opisu procesu (tu był BPMN). Jak nietrudno chyba już zauważyć, dokumentacja zmierza w kierunku kilku prostych modeli (diagramów) zamiast jednego spaghetti-modelu. Często spotykam się z dokumentami, w których autor analizy wymagań wszystko udokumentował na diagramie przypadków użycia (lub przynajmniej starał się to zrobić). Takie dokumentacje bywają straszne.

Na zakończenie przedstawię diagram komponentów obrazujący nasz projektowany system oraz te systemy, z którymi będzie zintegrowany (rysunek 12.3).



Rysunek 12.3. Diagram komponentów

Diagram ten zawiera poza komponentami także klasy interfejsów I_ERP i I_Płatności, klasy te są wykorzystywane w diagramach interakcji.

Przy okazji warto wspomnieć o *Cloud Computing*: komponenty integrowane mogą być dostępne „w chmurze". W kolejnym rozdziale opiszę model dziedziny systemu i model sekwencji dla przypadku użycia. Nadmienię też, że wciąż nie przedyskutowaliśmy problemu wymagań niefunkcjonalnych!

Zrozum, zanim zaproponujesz rozwiązanie

Analiza biznesowa, przeprowadzana przez wykonawcę rozwiązań IT dla biznesu, nie polega na zwykłym spisaniu problemów zgłaszanych przez klienta. Zapis przebiegu spotkań, wypunktowanie najważniejszych wymagań stawianych przed oprogramowaniem i prezentacja ich klientowi w formie tabeli to jeszcze nie analiza biznesowa! Wykonawca oprogramowania otrzymuje w ten sposób jedynie wnioski dotyczące efektów wadliwego działania aktualnego systemu zarządzania, ale dalej nie ma pojęcia o przyczynach tego stanu rzeczy. Jak zatem może zaproponować zleceniodawcy lepsze rozwiązanie?

Prawidłowo przeprowadzona analiza i modelowanie biznesowe opierają się na pewnych ściśle określonych zasadach i korzystają z przeznaczonych do tego celu notacji (języków). Na naszym rynku jest dostępnych wiele podręczników szczegółowo opisujących teorię poszczególnych notacji: UML, BPMN, BMM, SysML itd. Ich autorzy skupiają się na definicji pojęć, słownictwie i powiązaniach występujących w poszczególnych metodach zapisu, a teorie uzupełniają prostymi przykładami. Autor tej książki postanowił podejść do tematu z całkiem innej strony — wychodzi od faktycznych problemów, z jakimi spotkał się w swojej wieloletniej praktyce, i podaje przykłady ich poprawnego, analitycznego opisu z użyciem takiej notacji, która jest najwłaściwsza w danej sytuacji biznesowej. Dzięki temu masz szansę poznać i zrozumieć, kiedy, po co i jak stosować dany język analityczny.

Jarosław Żeliński — niezależny ekspert. Ukończył Wydział Elektroniki Wojskowej Akademii Technicznej. Od 1991 roku pracuje w branży IT. Uczestniczy w projektach z zakresu wdrożeń systemów IT, analiz systemowych i doradztwa organizacyjnego. Od 1998 roku prowadzi niezależną działalność doradczą, publikuje eseje, autorskie prace analityczne i opracowania branżowe w prasie specjalistycznej i gospodarczej oraz na własnym portalu IT-Consulting.pl. Od 2003 roku jest członkiem Stowarzyszenia Doradców Gospodarczych, a od 2005 roku współpracownikiem i wykładowcą Katedry Systemów Informacyjnych na Wydziale Przedsiębiorczości i Towaroznawstwa Akademii Morskiej w Gdyni. Od 2008 roku pracuje także na Politechnice Warszawskiej. Od dwóch lat prowadzi wykłady i laboratoria na kierunku Informatyczne techniki zarządzania w Wyższej Szkole Informatyki Stosowanej i Zarządzania pod auspicjami Polskiej Akademii Nauk. Prowadzi samodzielne studia i badania naukowe w zakresie zastosowań teorii poznania i języków formalnych w analizach systemowych organizacji. Jest członkiem International Institute of Business Analysis.

książkiklasybusiness

Nr katalogowy: 26395

Księgarnia internetowa:
http://onepress.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

one press

Helion

Sprawdź najnowsze promocje:
http://onepress.pl/promocje
Książki najchętniej czytane:
http://onepress.pl/bestsellery
Zamów informacje o nowościach:
http://onepress.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: onepress@onepress.pl
http://onepress.pl

ISBN 978-83-246-4880-1

Cena: 39,90 zł