Współczesna Gospodarka

Contemporary Economy
Electronic Scientific Journal
www.wspolczesnagospodarka.pl

Vol. 2 Issue 1 (2011) 39-48
ISSN 2082-677X

OD SPECYFIKACJI WYMAGAŃ DO GENEROWANIA KODU ŹRÓDŁOWEGO: TRANSFORMACJE W JĘZYKU UML I METODYCE RUP

Stanisław Wrycza, Bartosz Marcinkowski

Streszczenie

Polska literatura związana z językiem UML doczekała się znacznej liczby publikacji związanych ze specyfikacją języka UML. Nieliczne spośród nich koncentrują się na praktycznych aspektach użytkowania tego języka w kontekście zastosowania narzędzi CASE oraz metodyki RUP. Artykuł poświęcono prezentacji ścieżki przejścia od założeń wyrażonych przypadkami użycia do strukturalnego kodu źródłowego systemu informatycznego. Punkt 1 podsumowuje podstawowe zagadnienia związane z językiem UML oraz metodyką RUP. Punkt 2 przedstawia zakres transformacji. Punkt 3 ilustruje przejście od wymagań do kodu na przykładzie systemu wspomagania realizacji projektów informatycznych.

Słowa kluczowe: programowanie, językUML, narzędzia CASE, metodyka RUP

1. Iteracyjno-przyrostowy cykl życia systemu a diagramy UML

Problemy użytkowania metodyk tworzenia systemów informatycznych wzbudzały zawsze znaczne zainteresowanie środowiska informatycznego – akademickiego [22] i biznesowego; zwolenników podejścia strukturalnego, społecznego i obiektowego [2]. Dyskusja ta nie została nigdy jednoznacznie rozstrzygnięta. Podkreślić jednak należy, że wiele podejść autorskich i firmowych okazało się bardzo użytecznymi w całym klasycznie pojętym cyklu życia systemu, jak i poszczególnych jego fazach – planowaniu, analizie, projektowaniu, wdrażaniu, użytkowaniu i modyfikacji.

Aktualnie sferę tworzenia systemów i wytwarzania oprogramowania zdominowały takie pojęcia i podejścia jak SOA (*Service-Oriented Architecture*), MDA (*Model-Driven Architecture*), wzorce projektowe, język UML (*Unified Modeling Language*) [13], jak również związana z nim metodyka RUP (*Rational Unified Process*) [14]. Współzależności pomiędzy tymi dwoma ostatnimi kategoriami stanowią tematykę niniejszego artykułu.

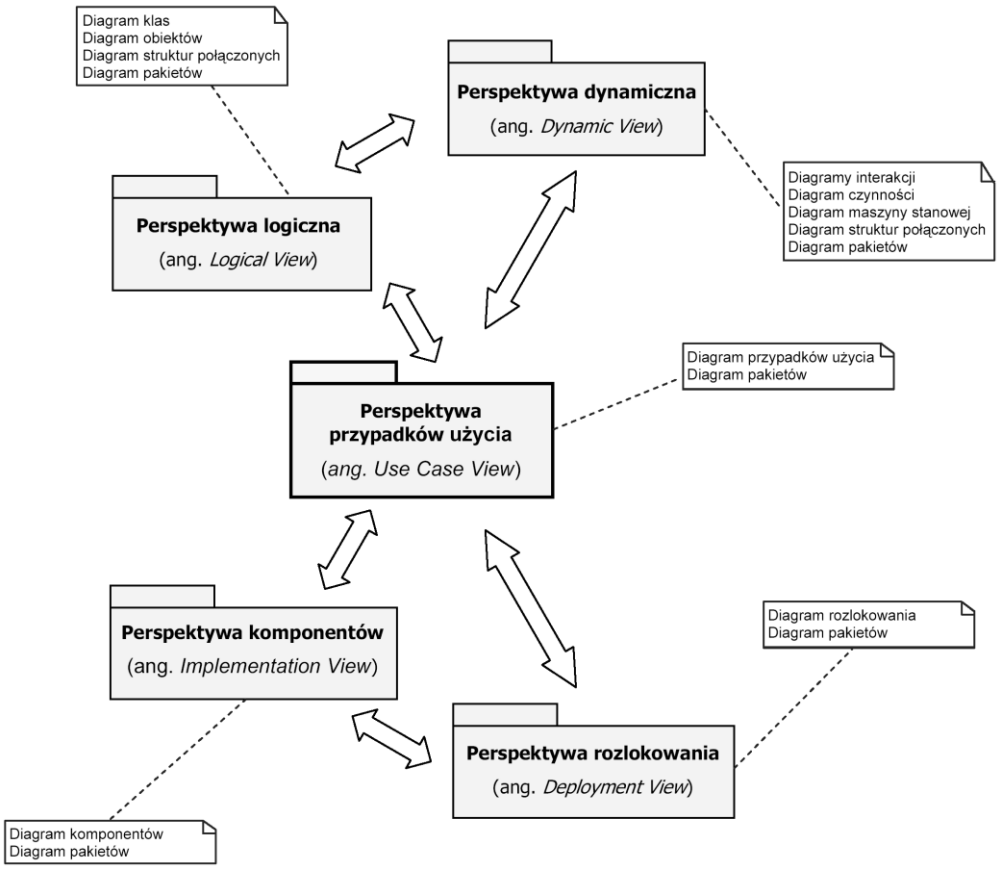
Język UML 2 (jego aktualnie zgłoszona wersja 2.1) został spopularyzowany w Polsce i na świecie za pośrednictwem licznych podręczników [17] i książek [16]. W skład standardu 2.1 wchodzi 13 rodzajów diagramów:
- diagram przypadków użycia,
- diagram klas,
- diagram obiektów,
- diagram pakietów,
- diagram struktur połączonych,
- diagram czynności,
- diagram sekwencji,
- diagram komunikacji,
- diagram harmonogramowania,
- diagram sterowania interakcją,
- diagram maszyny stanowej,
- diagram komponentów,
- diagram rozlokowania.

Aprobatę udziałowców (ang. *stakeholders*) projektów i systemów IT – menedżerów, klientów, analityków, projektantów, programistów pozyskał iteracyjno-przyrostowy cykl życia systemu, mający postać uzupełnianej w kolejnych iteracjach, a w konsekwencji przyrostowej, macierzy zależności pomiędzy czterema fazami a dziewięcioma dyscyplinami. Uwzględniając kryterium czasu tworzenia typowego systemu informatycznego przyjmuje się uśrednione szacunki [9]:
- faza rozpoczęcia: 10% czasu,
- faza opracowania: 30% czasu,
- faza budowy: 50% czasu,
- faza przekazania: 10% czasu.

Z kolei dyscypliny (często mylone przez początkujących projektantów z fazami) obejmują dyscypliny podstawowe, stanowiące rdzeń procesu:
- modelowanie biznesowe,
- specyfikację wymagań,
- analizę i projektowanie,
- programowanie,
- testowanie,
- wdrożenie;

oraz trzy dyscypliny uzupełniające:
- zarządzanie konfiguracjami i zmianami,
- zarządzanie projektem,
- przygotowanie środowiska.

Iteracja poszczególnych fazach przybiera postać mini-cyklu życia systemu, w którym następuje wykonanie ze zróżnicowaną intensywnością poszczególnych dyscyplin. Elastyczność języka UML i metodyki RUP w tworzeniu obiektowego systemu informatycznego wyraża się w możliwości przyjęciu kilku perspektyw systemu, jego architektury, zaproponowanych przez Kruchtena. Zmodyfikowaną wersję układu perspektyw i najpowszechniej stosowanych w ich ramach diagramów UML oraz technik uzupełniających zamieszczono na rys. 1.



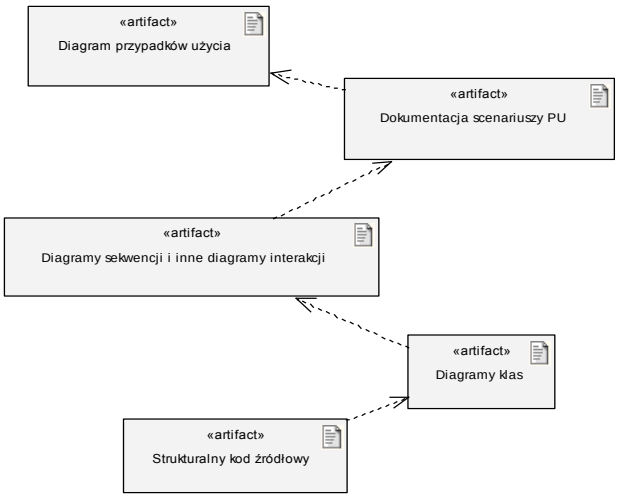
Rys. 1. Perspektywy architektury systemu i diagramy UML najczęściej stosowane w ich ramach

Źródło: [17]

Dominującą rolę pełni zatem perspektywa przypadków użycia. Stanowi ona punkt wyjścia dla modelowania systemu informatycznego. Konsekwentnie na jej bazie rozbudowywane są perspektywy logiczna oraz dynamiczna, a z kolei na bazie jej wyników perspektywa komponentów oraz rozlokowania. Pomiędzy wymienionymi na rys. 1 diagramami istnieją silne współzależności, sformalizowane w punktach 2 i 3.

2. Współzależności pomiędzy diagramami UML w iteracjach RUP

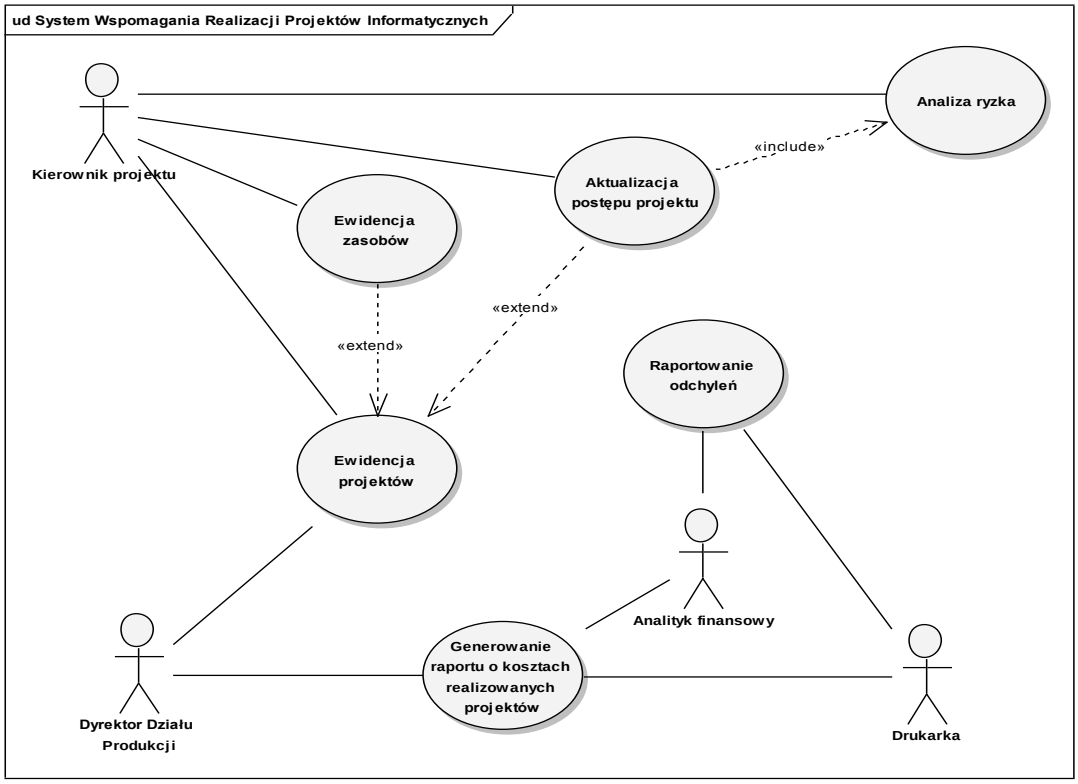
Literatura na temat języka UML i metodyki RUP jest bogata w różnorodne opracowania i książki będące bądź systematycznymi podręcznikami [17, 3] bądź publikacjami analizującymi szczegółowo wybrane aspekty specyfikacji [5]. Znacznie mniej prac poświęcono współzależnościom i transformacjom poszczególnych elementów dokumentacji projektowej jak również poszczególnych podstawowych i zaawansowanych kategorii modelowania języka UML. Jest to problem istotne ze względu na problem szybkości i precyzji przejścia od zdefiniowania i specyfikacji potrzeb użytkownika, wymagań systemowych do konkretnej specyfikacji na poziomie klas systemowych czy wręcz generowania szkieletu kodu na podstawie tych specyfikacji. Zaproponowana w niniejszym artykule ścieżka przejścia od założeń wyrażonych przypadkami użycia do kodu strukturalnego została przedstawiona na rys. 2.



Rys. 2. Ścieżka przejścia od założeń wyrażonych przypadkami użycia do kodu strukturalnego

3. Transformacje pomiędzy diagramami UML i technikami uzupełniającymi

Jak zasygnalizowano w poprzednim punkcie, naturalną inicjację diagramowej dokumentacji systemowej stanowią przypadki użycia. Jako, że symbolizują one kompletne usługi systemowe, udostępniające aktorowi nową, istotnią funkcjonalność, istnieje realne niebezpieczeństwo zbytniego skupienia się na szczegółach przez projektanta systemu. Na rys. 3 zaprezentowano diagram przypadków użycia systemu wspomagania realizacji projektów informatycznych.



Rys. 3. Diagram przypadków użycia systemu wspomagania realizacji projektów informatycznych

Wymogi co do jednoznaczności i precyzyjności odwzorowania wymagań użytkownika, jak również znaczny odsetek popełnianych na tym etapie błędów oraz koszt ich usuwania

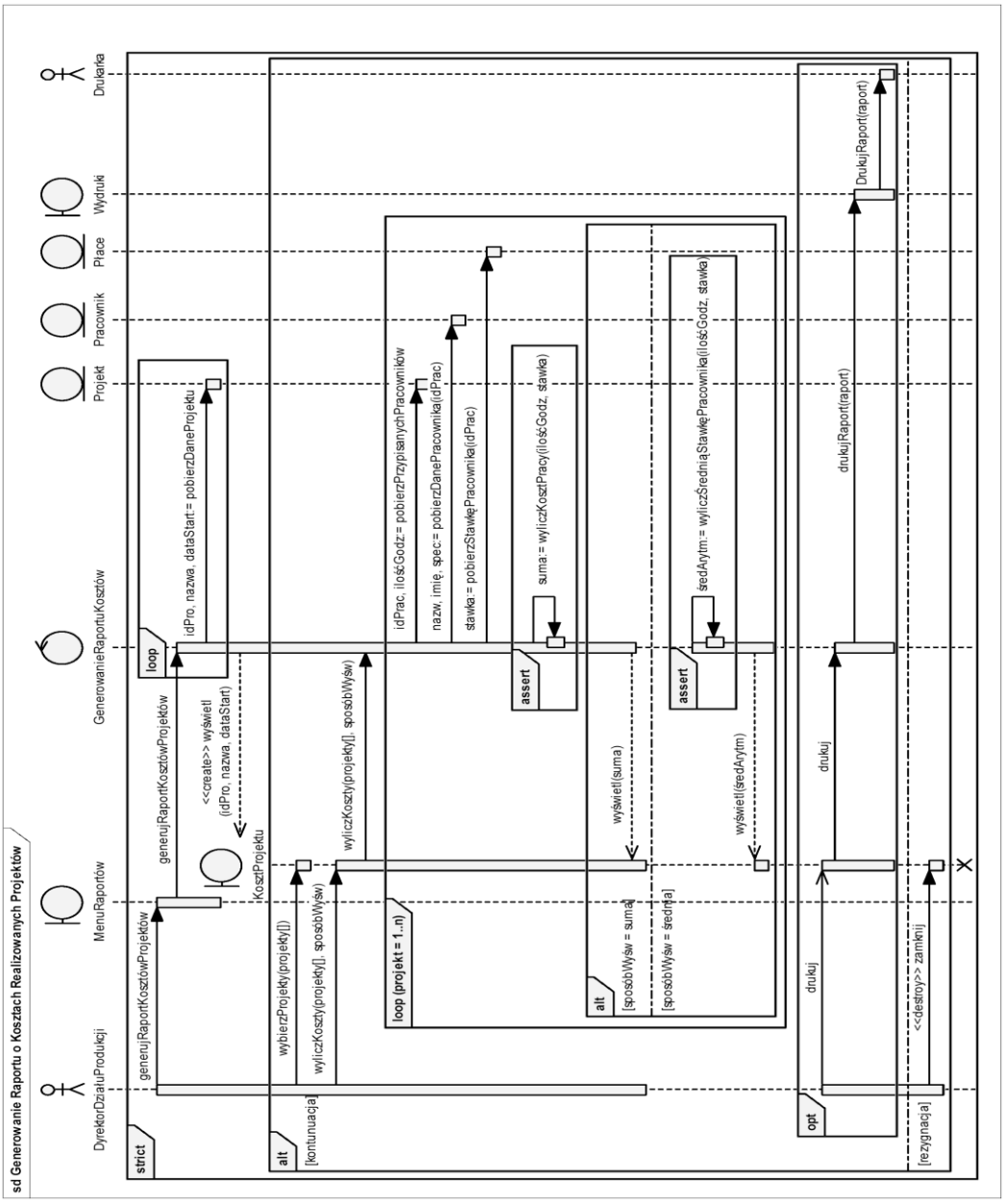
uzasadniają rozbudowanie dokumentacji systemowej o specyfikację scenariuszy poszczególnych przypadków użycia. Istnieje wiele form tego typu specyfikacji, począwszy od nieformalnego opisu tekstowego, aż po specyfikację w formie diagramowej (zwłaszcza w formie diagramów czynności). Zasadne wydaje się zastosowanie formy pośredniej w postaci półformalnej, tabelarycznej specyfikacji scenariuszy. Z jednej strony forma taka byłaby przystępna dla odbiorcy nie dysponującego przygotowaniem technicznym, z drugiej nie pozostawiałaby wątpliwości co do szczegółowego zakresu informacji, jaki powinien być zawarty w dokumentacji przypadku użycia. Liczba interakcji aktor-system ujętych w głównym oraz alternatywnych przepływach zdarzeń jest przy tym dobrą miarą poziomu szczegółowości przypadku użycia. Tabela 1 przedstawia przykładową dokumentację przypadku użycia „Generowanie raportu o kosztach realizowanych projektów".

Tabela 1. Dokumentacja przypadku użycia

Nazwa:	**Generowanie raportu o kosztach realizowanych projektów**
Numer:	4
Twórca:	Krzysztof Zaremba, analityk
Poziom ważności:	Ważny
Typ PU:	Ogólny, średnio istotny
Aktorzy:	Dyrektor Działu Produkcji, Drukarka
Krótki opis:	PU umożliwia wyliczenie kosztów sumarycznych, jak i średnich
Warunki wstępne:	Dyrektor działu produkcji zautentykowany systemie Minimum jeden projekt o statusie „w toku" lub „zrealizowany"
Warunki końcowe:	Wyliczenie raportu
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję „generuj raport kosztów projektów" 2. System wyświetla formatkę z listą dostępnych projektów 3. Użytkownik zaznacza jeden lub kilka projektów, dla których ma być przeprowadzona analiza i wybiera opcję wyświetlania kosztów sumarycznych 4. System wyświetla koszty wybranych projektów 5. Użytkownik zleca opcję drukowania 6. System powraca do wyświetlania listy projektów
Alternatywne przepływy zdarzeń:	3a) Użytkownik po zaznaczeniu projektów decyduje się na wyliczenie średnich kosztów przypadających na pracownika – krok 4 3b) Użytkownik nie zaznacza żadnego projektu – krok 2 5a) Użytkownik pomija drukowanie raportu kosztów – krok 2 5b) Użytkownik wybiera opcję rezygnacji – wyjście z PU
Specjalne wymagania:	brak
Notatki:	brak

Kolejnym etapem przygotowywania dokumentacji systemowej jest przygotowanie diagramów dynamiki UML na bazie scenariuszy przypadków użycia. Ogniwo pośrednie w postaci

dokumentacji scenariuszy przypadków użycia okazuje się szczególnie użyteczne podczas przygotowywania diagramów interakcji.



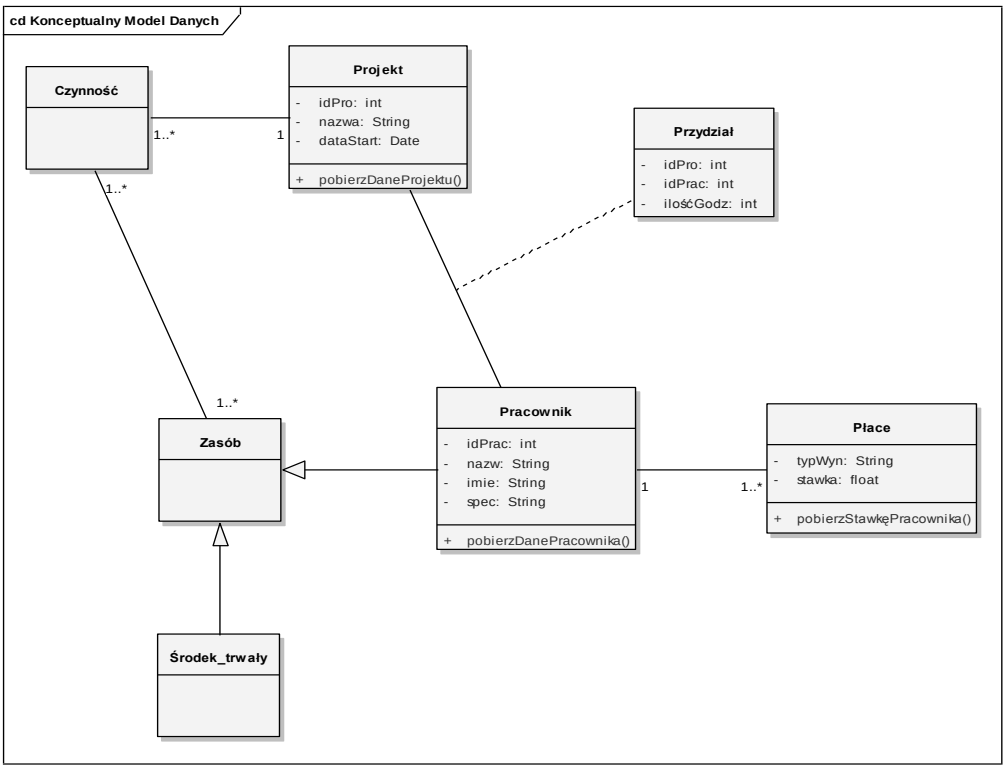
Rys. 4. Diagram przypadków użycia systemu wspomagania realizacji projektów informatycznych

Źródło: [17]

Konsekwentne opieranie się na scenariuszach w zasadzie wyklucza możliwość pominięcia funkcji szczegółowej podczas obrazowania interakcji systemu na poziomie konkretnych klas oraz metod wraz z parametrami i wartościami zwrotnymi. Rys. 4. przedstawia najczęściej stosowany w praktyce spośród wszystkich diagramów interakcji, tj. diagram sekwencji, przygotowany na podstawie dokumentacji scenariuszy przypadku użycia „Generowanie raportu o kosztach realizowanych projektów".

Specyfikacje dynamiki systemu w postaci diagramów interkacji (bądź diagramów czynności UML uwzględniających przepływy danych) stanowią dobrą podstawę do wysokopoziomowej specyfikacji struktury danych w systemie. Należy zwrócić uwagę, że

wszystkie analityczne klasy sterujące, zaprezentowane na rys. 4 (*Projekt*, *Pracownik*, *Płace*) można bezpośrednio przenieść na diagram klas i uzupełnić je o operacje wywoływane za pośrednictwem komunikatów na źródłowym diagramie sekwencji, jak również przekazywane pomiędy klasami atrybuty. Uzupełnienie diagramu o klasy, atrybuty oraz operacje niezbędne do zrealizowania funkcjonalności pozostałych przypadków użycia (a tym samym zaczerpnięcie danych z innych diagramów interakcji) prowadzi do przygotowania konceptualnego modelu danych w systemie (rys. 5).



Rys. 5. Konceptualny diagram klas opracowany na podstawie diagramu sekwencji oraz DPU

Funkcjonalność narzędzi CASE umożliwia w większości przypadków wygenerowanie strukturalnego kodu źródłowego w jednym z oferowanych przez narzędzie języków. W zależności od poziomu szczegółowości diagramu klas wyręcza to programistę w realizowaniu wstępnych etapów programowania oraz zmniejsza liczbę błędów technicznych. Wybrane klasy z rys. 5, tj. klasy *Projekt*, *Pracownik* oraz *Płace* zostały przeniesione na kod źródłowy w języku JAVA z wykorzystaniem narzędzia Enterprise Architect firmy Sparx Systems [7].

```
public class Projekt {

  private int idPro;
  private String nazwa;
  private Date dataStart;
  public Pracownik m_Pracownik;
  public Czynność m_Czynność;

  public Projekt(){

  }

  public void finalize() throws Throwable {
```

```
    }
    public pobierzDaneProjektu(){

    }

  }

  public class Pracownik extends Zasób {

    private int idPrac;
    private String nazw;
    private String imie;
    private String spec;
    public Płace m_Płace;

    public Pracownik(){

    }

    public void finalize() throws Throwable {
        super.finalize();
    }

    public pobierzDanePracownika(){

    }

  }

  public class Płace {

    private String typWyn;
    private float stawka;

    public Płace(){

    }

    public void finalize() throws Throwable {

    }

    public pobierzStawkęPracownika(){

    }
  }
```

Zakończenie

Użytkowanie tak rozbudowanego instrumentarium, jakim jest język UML 2.x, stanowi poważne wyzwanie dla analityków i projektantów systemów informatycznych. O ile język UML stanowi w istocie zbiór artefaktów, które mogą być stosowane podczas tworzenia systemu informatycznego w zależności od aktualnych potrzeb, wspomaganie pracy twórcy systemu sprawdzonymi schematami postępowania oraz transformacjami przy zastosowaniu narzędzi

CASE w istotny sposób przyczynia się do zmniejszenia liczby błędów popełnionych w procesie wytwórczym oprogramowania oraz zmniejsza czas poświęcony na realizację poszczególnych dyscyplin RUP. W punkcie 2 niniejszego artykułu zaproponowano schemat transformacji pomiędzy diagramami UML oraz technikami uzupełniającymi, umożliwiający efektywne przejście od wymagań użytkownika, wyrażonych za pośrednictwem przypadków użycia, aż do strukturalnego kodu źródłowego systemu. W punkcie 3 zilustrowano zaproponowane podejście na przykładzie systemu wspomagania realizacji projektów informatycznych.

LITERATURA

[1] BOEHM B., TURNER R., *Balancing Agility and Discipline*, New York, Pearson Education, 2004.
[2] BOOCH G., RUMBAUGH J., JACOBSON I., *UML – Przewodnik użytkownika*, Warszawa, WNT,2001.
[3] BOOCH G., RUMBAUGH J., JACOBSON I., *The UML Reference Manual 2nd Edition*, Boston, Addison-Wesley, 2004.
[4] ERL T., *Service Oriented Architecture. Concepts, Technology and Design*, New York, Prentice Hall, 2005.
[5] FOWLER M., SCOTT K., *UML w kropelce*, Warszawa, Oficyna Wydawnicza LPT, 2002.
[6] FRANKEL D.S., *Model Driven Architecture*, Wiley Publishing Inc., Indianapolis 2003.
[7] http://www.sparxsystems.com.au, stan na dzień 05.09.2005.
[8] KROLL P., KRUCHTEN P., *The Rational Unified Process Made Easy*, Boston, Addison-Wesley, 2003.
[9] KRUCHTEN P., *Rational Unified Process od strony teoretycznej*, Warszawa, WNT, 2007.
[10] LARMAN C., *Agile and iterative development*, New York, Pearson Education, 2004.
[11] MARCINKOWSKI B., *Isomorphism of Interaction Diagrams in UML 2*. In: Proceedings of 8th International Conference on Business Information Systems, W. Abramowicz (ed.), Poznań, Poznań University of Economics Press, 2005.
[11] MARCINKOWSKI B., *Relevance of Use-Case Scenarios' Descriptions in System Requirements Specification*. In: Information Management, B.F. Kubiak, A. Korowicki (eds.), Gdańsk, Gdańsk University Press, 2005.
[13] Object Management Group; *UML 2.0 Superstructure FTF Convenience Document*, http://www.omg.org/docs/ptc/04-10-02.pdf, stan na dzień 15.07.2005.
[14] Rational Software White Paper, *Rational Unified Process. Best Practices for Software Development Teams*, ftp://ftp.software.ibm.com/software/rational/web/whitepapers/ 2003/rup_bestpractices.pdf, stan na dzień 05.09.2005.
[15] SUBIETA K., *Obiektowość w projektowaniu i bazach danych*, Warszawa, Oficyna Wydawnicza PLJ, 1998.
[16] ŚMIAŁEK M., *Zrozumieć UML 2.0. Metody modelowania obiektowego*, Gliwice, Helion, 2005.
[17] WRYCZA S., MARCINKOWSKI B., WYRZYKOWSKI K., *Język UML 2.0 w modelowaniu systemów informatycznych*, Gliwice, Helion, 2005.
[18] WRYCZA S., MARCINKOWSKI B., WYRZYKOWSKI K., *Rola fragmentów wyodrębnionych w języku UML 2*. W: Bazy danych: Modele, Technologie, Narzędzia. Architektura, metody formalne, bezpieczeństwo, S. Kozielski, B. Małysiak, P. Kasprowski, D. Mrozek (red), Warszawa, Wydawnictwo Komunikacji i Łączności, 2005.
[19] WRYCZA S., MARCINKOWSKI B., WYRZYKOWSKI K., *Rola i funkcje diagramów harmonogramowania w modelowaniu systemów informatycznych z wykorzystaniem języka UML 2*. W: A. Nowakowski (red), Infobazy '2005. Bazy danych dla nauki, Gdańsk, Centrum Informatyczne TASK, 2005.
[20] WRYCZA S., MARCINKOWSKI B., WYRZYKOWSKI K., *Timing Diagram Functionalities in Information Systems Modeling with UML 2*. In: Information Management, B.F. Kubiak, A. Korowicki (eds.), Gdańsk, Gdańsk University Press, 2005.
[21] WRYCZA S., MARCINKOWSKI B., *UML 2 Teaching at Postgraduate Studies – Prerequisites & Practice*, Proceedings of ISECON 2005, Volume 22, the 22nd Annual Conference for Information Systems Educators, Chicago, AITP Foundation for IT Education, 2005.
[22] WRYCZA S., *Analiza i projektowanie systemów informatycznych zarządzania*; Warszawa: PWN, 1999.

FROM SYSTEM REQUIREMENTS TO SOURCE CODE: TRANSITIONS IN UML AND RUP

Summary

There are many manuals explaining language specification among UML-related books. Only some of books mentioned concentrate on practical aspects of using the UML

language in effective way using CASE tools and RUP. The current paper presents transitions from system requirements specification to structural source code, useful while developing an information system.

Keywords: programming language, UML, CASE tools, methodology RUP

prof. zw. dr hab. Stanisław Wrycza
dr Bartosz Marcinkowski
Uniwersytet Gdański
Wydział Zarządzania
81-864 SOPOT, ul. Piaskowa 9