Diagramy wdrożeniowe w modelowaniu systemów informatycznych

Stanisław Wrycza

Bartosz Marcinkowski

Jacek Maślankowski

Nieodłącznym elementem dokumentacji projektowej systemów informatycznych są diagramy prezentujące infrastrukturę sprzętową i oprogramowania tworzonych systemów. Przykładem są diagramy wdrożeniowe, tworzone w języku UML. Można wyróżnić dwie konkretne odmiany takich diagramów – diagramy komponentów i rozlokowania, wykorzystywane odpowiednio do modelowania oprogramowania i zasobów sprzętowych, na których to oprogramowanie jest fizycznie alokowane. Celem niniejszego artykułu jest ocena możliwości stosowania diagramów wdrożeniowych do modelowania systemów informatycznych (MSI), które wykorzystują współczesne rozwiązania technologiczne. Po omówieniu miejsca i roli diagramów wdrożeniowych w MSI zdefiniowano ich funkcjonalność oraz kategorie pojęciowe i konwencje graficzne. W dalszej części artykułu przedstawiono kwestie teoretyczne dotyczące diagramów oraz ich praktyczne zastosowania na przykładach współczesnych aplikacji z zakresu gospodarki elektronicznej. Są to kolejno: system informatyczny do sterowania ruchem pojazdów w miastach, wspomagana komputerowo obsługa magazynu oraz wykorzystanie urządzeń mobilnych do zakupu biletów elektronicznych.

Modelowanie systemów informatycznych często kojarzone jest z graficzną reprezentacją funkcjonalności tworzonego oprogramowania[1]. Dotyczy to w szczególności specyfikacji wymagań funkcjonalnych – a rzadziej pozafunkcjonalnych, z uwagi na brak powszechnie uznanego standardu modelowania tego typu wymagań[2]. Wymagania wobec usług systemów informatycznych definiowane są na poziomie koncep-

tualnym[3], niemniej dla fazy wdrożenia systemu w jego cyklu życia ważne są aspekty techniczne, związane z infrastrukturą sprzętu komputerowego i sieciowego, a także oprogramowania. Aspekty architektoniczne systemu informatycznego – oprogramowania i sprzętu – modelowane są w języku UML, w postaci ściśle ze sobą powiązanych diagramów, tj. odpowiednio:

- diagramów komponentów (*component diagrams*),
- diagramów rozlokowania (*deployment diagrams*).

Z uwagi na silne powiązanie merytoryczne i praktykę wykorzystania wymienionych typów diagramów autorzy niniejszego artykułu formalnie uogólniają je do postaci abstrakcyjnego pojęcia diagramów wdrożeniowych (*implementation diagrams*, por. rysunek 1).

Diagramy komponentów pozwalają nie tylko na modelowanie tworzonego i użytkowanego oprogramowania[4], ale również na specyfikację wybranych aspektów modeli biznesowych systemu. Diagramy rozlokowania z kolei identyfikują sprzęt informatyczny użytkowany w systemie, na którym wyspecyfikowane wcześniej komponenty oprogramowania systemu są zainstalowane[5]. Nawiązując do cyklu życia systemu, zgodnie z metodyką RUP (*Rational Unified Process*) komponenty systemu mogą być udoskonalane w kolejnych iteracjach tworzenia oprogramowania. Zakłada się, że diagramy rozlokowania będą prezentować finalną postać infrastruktury sprzętowej, niezbędną do implementacji systemu zgodnie z przyjętymi wymaganiami funkcjonalnymi i pozafunkcjonalnymi.

[1] P. Shoval, J. Kabeli, *Special Theme of Research in Information Systems Analysis and Design – II. Data Modeling or Functional Modeling – Which Comes First? An Experimental Comparison*, „Communications of the Association for Information Systems" 2005, Vol. 16, pp. 725–769.

[2] J. Zou, C. Pavlovski, *Control case approach to record and model non-functional requirements*, „Information Systems & E-Business Management" 2008, Vol. 6, No. 1, pp. 49–67.

[3] S. Wrycza, B. Marcinkowski, *Specyfikacja wymagań systemowych w języku SysML*, [w:] J. Goliński, A. Kobyliński, A. Sobczak (red), *Technologie informatyczne w administracji publicznej i służbie zdrowia*, „Monografie i Opracowania Szkoły Głównej Handlowej w Warszawie" 2010, nr 569, s. 323–337; S. Wrycza, B. Marcinkowski, *SysML Requirement Diagrams: Banking Transactional Platform Case Study*, „Lecture Notes in Business Information Processing" 2011, No. 93, pp. 15–24.

[4] S. Al-Fedaghi, A. Abdullah, *A New Approach to Component-Based Development of Software Architecture*, „International Review On Computers & Software" 2013, Vol. 8, No. 1, pp. 1–10.

[5] J. Holt, *UML for Systems Engineering: Watching the Wheels*, wyd. 2, The Institution of Engineering and Technology, 2007.

Celem niniejszego artykułu jest identyfikacja roli, jaką pełnią diagramy wdrożeniowe języka UML w modelowaniu systemów informatycznych. Choć informatyka jest jedną z najszybciej zmieniających się dziedzin działalności naukowej i biznesowej, język UML – w tym kwestie związane z modelowaniem aspektów wdrożeniowych systemu – zachowuje wyjątkową stabilność, pomimo że powstał kilkanaście lat temu. Nadal całkowicie spełnia oczekiwania analityków i projektantów systemów informatycznych we współczesnych tworzonych i użytkowanych technologiach ICT, w tym mobilnych. W artykule zaprezentowano również szczegółowo zasady skutecznego stosowania diagramów wdrożeniowych, tj. diagramów komponentów i diagramów rozlokowania.

Pomimo stabilności języka UML w literaturze naukowej stale publikowane są prace teoretyczne, w których proponuje się dokonywanie modyfikacji, adaptacji bądź uzupełnienia języka UML w celu rozwiązania specyficznych problemów czy złożonych zagadnień dotyczących np. systemów czasu rzeczywistego (ang. *UML-RT – Real-Time*)[6] lub modelowania struktury hurtowni danych[7].

Miejsce i rola diagramów wdrożeniowych

Język UML[8], a aktualnie również notacja BPMN[9], stały się standardami modelowania systemów informatycznych powszechnie wykorzystywanymi w biznesie[10] i dydaktyce akademickiej[11]. Język UML to język graficzny wizualizacji, specyfikowania, tworzenia i dokumentowania systemów informatycznych[12]. W praktyce jego użycie polega na umiejętnym wyborze (spośród aktualnych 14 rodzajów) i stosowaniu w procesie modelowania diagramów UML do rozwiązywania problemów informatycznych firmy czy instytucji.

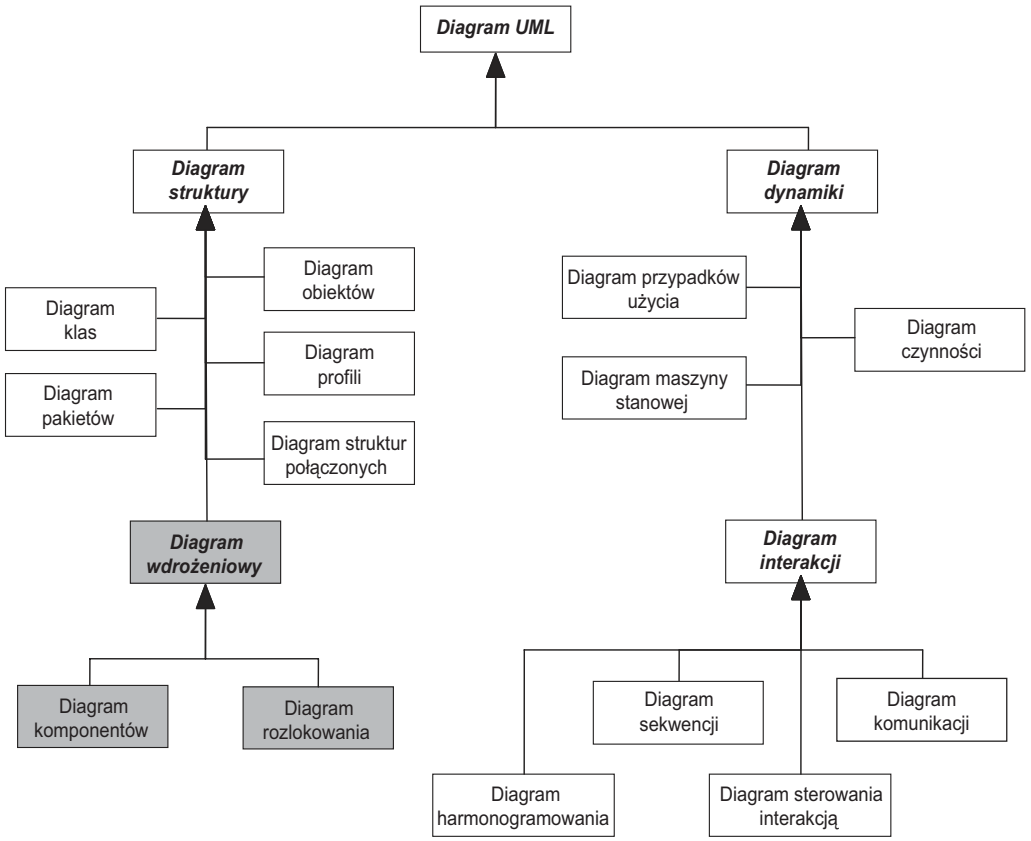
Ogólnie diagramy UML dzieli się na diagramy struktury i diagramy dynamiki – każda z tych grup zawiera aktualnie po 7 rodzajów diagramów. Z ich pełnego zestawu projektanci wybierają i selektywnie stosują odpowiednie dla danego projektu rodzaje diagramów w poszczególnych fazach procesu tworzenia systemów informatycznych. W grupie diagramów struktury najistotniejsze i jednocześnie najstarsze (zaproponowane i stosowane jeszcze przed opracowaniem języka UML) są diagramy klas, a poza nimi diagramy obiektów, pakietów, profili oraz właśnie konkretne odmiany diagramów wdrożeniowych: diagramy komponentów i rozlokowania. Z kolei w grupie diagramów dynamiki podstawową rolę odgrywają diagramy przypadków użycia (DPU), w opinii wielu analityków najważniejsze spośród 14 rodzajów diagramów. Opracowanie DPU inicjuje realizację procesu tworzenia systemu informatycznego. Definiują one bowiem założenia i wymagania systemów. Poza DPU do diagramów dynamiki zalicza się diagramy: czynności, maszyn stanowych oraz – najbardziej rozbudowane – cztery odmiany diagramów interakcji: diagramy sekwencji, komunikacji, harmonogramowania i sterowania interakcją.

Zarysowaną klasyfikację diagramów UML, będącą rozwinięciem oficjalnej klasyfikacji UML w wersji 2.4.1[13], obrazuje rysunek 1. Ta wersja języka UML została zaadaptowana również jako standard ISO/IEC 19505-1 oraz 19505-2[14].

Różne rodzaje diagramów stosowane są przez analityków i projektantów systemów na różnych poziomach zaawansowania projektu informatycznego. Kierunek włączania diagramów UML do procesu projektowania przebiega od diagramów ogólnych do diagramów pozwalających na identyfikację szczegółowych elementów systemu. Tezę tę potwierdzają badania przeprowadzone na reprezentatywnej grupie respondentów – znawców i użytkowników języka UML – przez J. Ericksona i K. Siau[15], jak również badania

[6] V. Bianco, L. Lavazza, M. Mauri, G. Occorso, *Towards UML-based formal specifications of component-based real-time software*, [in:] *Fundamental Approaches to Software Engineering, 6th International Conference*, Springer Verlag, 2003.
[7] S. Luján-Mora, J. Trujillo, I. Song, *A UML profile for multidimensional modeling in data warehouses*, „Data & Knowledge Engineering" 2006, Vol. 59, No. 3, pp. 725–769.
[8] S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML w modelowaniu systemów informatycznych*, Helion, Gliwice 2005.
[9] B. Gawin, B. Marcinkowski, *Symulacja procesów biznesowych. Standardy BPMS i BPMN w praktyce*, Helion, Gliwice 2013.
[10] S. Wrycza, B. Marcinkowski, *Towards a Light Version of UML 2.x: Appraisal and Model*, „Organizacija" 2007, No. 4, pp. 171–179.
[11] S. Wrycza, *The UML 2 Academic Teaching Challenge: An Integrated Approach*, [in:] A. Bajaj, S. Wrycza (eds.), *Systems Analysis and Design for Advanced Modeling Methods. Best Practices*, IGI Global, 2009; S. Wrycza, B. Marcinkowski, *UML 2 Academic Course – Methodological Background and Survey Benchmarking*, [in:] *Proceedings of the 23rd Annual Conference for Information Systems Educators,* AITP Foundation for Information Technology Education, Dallas 2006; S. Wrycza, B. Marcinkowski, *UML 2 Teaching at Postgraduate Studies – Prerequisites and Practice*, [in:] D. Colton, T. Janicki (ed.), *Proceedings of the 22nd Annual Conference for Information Systems Educators,* AITP Foundation for Information Technology Education, Columbus 2005.
[12] S. Wrycza, B. Marcinkowski, K. Wyrzykowski, dz.cyt., s. 20.
[13] OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, sierpień 2011, s. 694; S. Wrycza, B. Marcinkowski, J. Maślankowski, *UML 2.x. Ćwiczenia zaawansowane*, Helion, Gliwice 2012, s. 14.
[14] Information technology – Object Management Group Unified Modeling Language (OMG UML), Superstructure, 2012.
[15] J. Erickson, K. Siau, *Theoretical and Practical Complexity of Modeling Methods*, „Communications Of The ACM" 2007, Vol. 50, nr 8, pp. 46–51.

Rysunek 1. Miejsce diagramów wdrożeniowych w hierarchii diagramów UML



Źródło: S. Wrycza, B. Marcinkowski, J. Maślankowski, dz.cyt., Helion, Gliwice 2012.

autorów niniejszego opracowania[16]. Wynikiem tych badań był tzw. lekki, szybki kanon UML, czyli jego wersja ograniczona do minimum najbardziej użytecznych i najintensywniej użytkowanych diagramów – ważnych w procesie modelowania przy zastosowaniu języka UML. Zarówno badanie amerykańskie, jak i polskie, doprowadziły do tego samego wniosku – do „lekkiego" UML należy zdecydowanie zaliczyć cztery rodzaje diagramów: przypadków użycia, klas, czynności oraz sekwencji. Dominują więc diagramy dynamiki, pozwalające modelować procesy biznesowe, algorytmy oprogramowania i interakcje systemu. Z kolei jedyne spośród diagramów struktury w tej wersji – diagramy klas – dokumentują architekturę przyszłej struktury danych i konsekwentnie bazy danych systemu informatycznego. DPU pozwalają na identyfikację założeń i wymagań modelowanego systemu na poziomie konceptualnym, podczas gdy na poziomie logicznym – na precyzyjną identyfikację klas, czynności oraz interakcji w systemie. Pozostałe rodzaje diagramów w opinii respondentów – menedżerów, analityków i projektantów – spełniają pomocnicze, ale nie absolutnie niezbędne role i zadania dla zaprojektowania skutecznego i sprawnego systemu informatycznego.

Szczególne znaczenie w kontekście MSI mają diagramy wdrożeniowe, jak wspomniano – mniej intensywnie wykorzystywane przez twórców systemów. Pełnią one specyficzną i unikalną rolę na etapie wdrażania systemu, bowiem pozwalają na dokładne modelowanie architektury i zależności wszystkich elementów sprzętu i oprogramowania projektowanego systemu. Jest to rozwiązanie wyjątkowe w metodach i technikach analizy oraz projektowania systemów informatycznych. Popularne w poprzednich dekadach metodyki projektowania strukturalnego takich możliwości nie oferują, dominują tu diagramy przepływu danych czy diagramy związków encji, przydatne w modelowaniu wysokopoziomowym.

Funkcjonalność diagramów wdrożeniowych

Zgodnie z sekwencją cyklu życia systemu – zarówno liniowego, spiralnego, jak i iteracyjno-przyrostowego – najpierw powinny zostać zidentyfikowane wymagania funkcjonalne i pozafunkcjonalne systemu, tak aby możliwe stały się: jego analiza, projektowanie i implementacja. Dotyczy to zarówno oprogramowania, jak i infrastruktury sprzętowej. Pierwszy

[16] S. Wrycza, B. Marcinkowski, *A Light Version of UML 2: Survey and Outcomes*, [in:] *Proceedings of the 2007 Computer Science and IT Education Conference*, University of Technology, Mauritius, 2007.

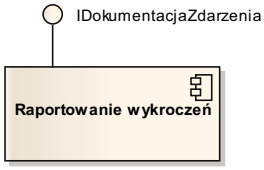
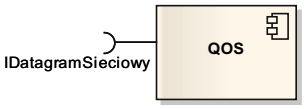
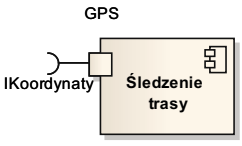
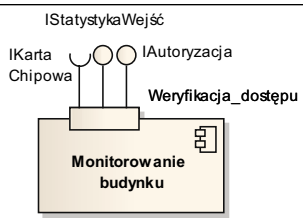
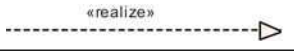
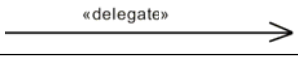
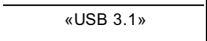
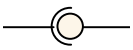
rodzaj wymagań w zakresie infrastruktury programowo-sprzętowej opisywany jest przez diagramy komponentów, a drugi przez diagramy rozlokowania.

Diagram komponentów to rodzaj diagramu wdrożeniowego, który wskazuje organizację komponentów i zależności między nimi. Podstawowymi kategoriami pojęciowymi i graficznymi diagramów komponentów są[17]: komponent, interfejs udostępniający, interfejs pozyskujący, port, port złożony, a w zakresie związków – zależność, realizacja, konektor delegowany i konektor składany. Interfejsy zawierają zestaw operacji, które wyznaczają usługi oferowane przez klasę lub komponent. Klasy ujmują przede wszystkim zestaw atrybutów i operacji. Komponent w praktyce może być implementowany w postaci programów wykonywalnych, bibliotek programów, baz danych, tabel danych, podsystemów czy też usług systemowych[18].

Z kolei diagram rozlokowania to rodzaj diagramu wdrożeniowego, który przedstawia sieć węzłów połączonych ścieżkami komunikowania (węzły reprezentują przede wszystkim sprzęt komputerowy i sieciowy), z osadzonymi na nich artefaktami. Pod pojęciem artefaktu rozumieć należy każdy sztucznie wytworzony produkt – w szczególności implementacje komponentów i pliki/dokumenty niezbędne do ich poprawnego funkcjonowania, wyspecyfikowane uprzednio na diagramach komponentów, a rezydujące na odpowiednich węzłach diagramów rozlokowania. Węzeł to fizyczny lub logiczny zasób przetwarzający, na którym osadzone są artefakty użytkowanego systemu. Może on przyjmować formę określonego rodzaju sprzętu ICT, jak również platform użytkowania systemu, którymi mogą być m.in. systemy operacyjne, platformy e-learningowe (np. Moodle, BlackBoard, WebCity, Sakai i inne), systemy ERP, CRM, WFM, systemy zarządzania bazami danych. Rozróżnienie typów węzłów osiągane jest dzięki zastosowaniu stereotypów, których bogaty zestaw ujmuje sam standard UML – a profesjonalista użytkujący ten standard może we własnym zakresie stosować własne tekstowe i graficzne propozycje tego typu rozszerzeń.

O ile diagramy rozlokowania mogą być konstruowane z wykorzystaniem stosunkowo wąskiego zakresu podstawowych kategorii modelowania, podkreślić należy, iż artefakty często stanowią implementację komponentów systemowych – co wiąże się z selektywnym ujęciem funkcjonalności diagramów komponentów oraz wprowadzeniem stereotypowanych związków zależności i opcjonalnych specyfikacji rozlokowania. Notacje podstawowych kategorii modelowania dla diagramów komponentów i rozlokowania wymienionych i zdefiniowanych powyżej zawarto w tabeli 1.

Tabela 1. Podstawowe kategorie pojęciowe i graficzne diagramów wdrożeniowych

Diagramy komponentów		Diagramy rozlokowania	
Pojęcie	**Notacja**	**Pojęcie**	**Notacja**
Komponent		Węzeł	
Interfejs udostępniający			
Interfejs pozyskujący			
Port		Artefakt	
Port złożony			
Zależność		Ścieżka komunikowania	
Realizacja			
Konektor delegowany			
Konektor składany			

Źródło: opracowanie własne.

[17] OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, sierpień 2011, rozdz. 8; S. Wrycza, B. Marcinkowski, K. Wyrzykowski, dz.cyt., rozdział 11.

[18] K.Z. Ahmed, C.E. Umrysh, *Developing Enterprise Java Applications with J2EE and UML*, Addison-Wesley, 2002.

Zastosowania diagramów komponentów i rozlokowania

Przedstawione założenia teoretyczne diagramów wdrożeniowych można zobrazować praktycznymi przykładami. W tym celu wybrano określone aplikacje z zakresu współczesnej gospodarki cyfrowej:

- dla diagramu komponentów – inteligentny system informacji transportowej;
- dla diagramu rozlokowania – system obsługi magazynu wspomagany narzędziami teleinformatycznymi;
- dla diagramu rozlokowania, ujmującego elementy diagramu komponentów – zastosowanie urządzeń mobilnych do zakupu biletów elektronicznych.

Analizie poddano przede wszystkim złożone aspekty modelowania oprogramowania, specyficzne dla tego rodzaju systemów informatycznych.

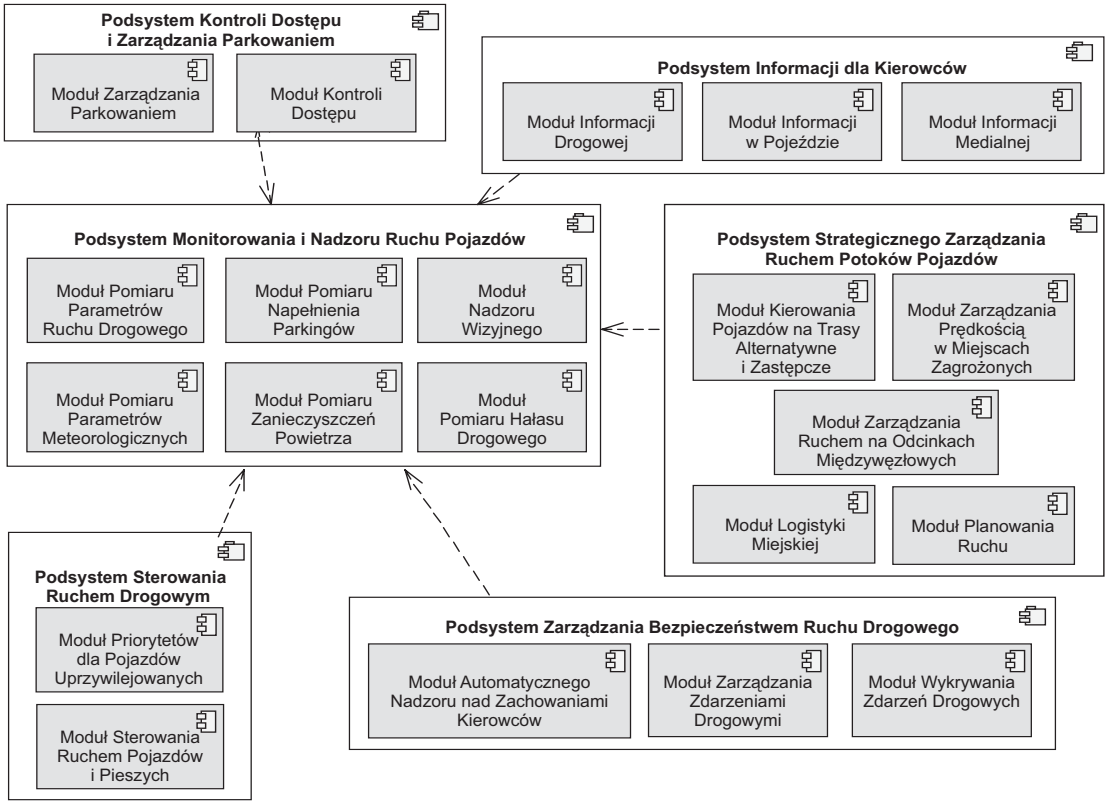
Inteligentny system informacji transportowej

Istniejąca w centrach dużych miast infrastruktura zwykle ogranicza możliwość rozbudowy sieci dróg, po których poruszają się pojazdy mechaniczne. Aby zapobiec nadmiernemu powstawaniu zatorów w mieście, wykorzystuje się wiele rozwiązań. Należy do nich m.in. pobieranie opłat za wjazd do centrum czy też nakładanie kar finansowych za nieuprawniony wjazd do niego. Innym rezultatem wzrostu liczby samochodów w miastach jest wytyczanie specjalnych pasów dla pojazdów komunikacji miejskiej oraz pojazdów uprzywilejowanych. Remedium na powstające w miastach zatory są też inteligentne systemy sterowania ruchem, należące do grupy systemów informatycznych zwanych inteligentnymi systemami transportowymi (*Intelligent Transport Systems* – ITS).

Inteligentne systemy transportowe mają na celu przede wszystkim usprawnienie ruchu pojazdów poprzez sterowanie sygnalizacją świetlną zależnie od natężenia ruchu. Jednak spektrum ich zastosowania jest dużo szersze i uwzględnia również mechanizmy zarządzania bezpieczeństwem ruchu, w tym powiadamianie o wypadkach, jak również ekrany LED wyświetlające ważne informacje ostrzegawcze dla kierowców.

Na rysunku 2 przedstawiono diagram komponentów dla koncepcji systemu Tristar[19], wdrażanego obecnie w Trójmieście. W dalszej części opracowania zamieszczono opis funkcjonalności tego systemu z użyciem głównych kategorii diagramów komponentów.

Rysunek 2. Diagram komponentów dla systemu informacji transportowej na przykładzie koncepcji systemu Tristar



Źródło: opracowanie własne na podstawie K. Jamroz, R. Krystek, W. Kustra, dz.cyt.

[19] K. Jamroz, R. Krystek, W. Kustra, *Koncepcja zintegrowanego systemu zarządzania ruchem na obszarze Gdańska, Sopotu i Gdyni, Część I. Koncepcja ogólna Zintegrowanego Systemu Zarządzania Ruchem TRISTAR*, Urząd Miejski w Gdańsku, Gdańsk 2007.

Związki przedstawione na rysunku 2 wskazują, że centralnym komponentem jest *Podsystem Monitorowania i Nadzoru Ruchu Pojazdów* (w opisach aplikacji kursywą oznaczono nazwy wszystkich kategorii graficznych występujących na danym diagramie). Podsystem ten pobiera za pomocą detektorów dane, które są wykorzystywane przez pozostałe pięć podsystemów. Są to podsystemy: *Sterowania Ruchem Drogowym*, *Strategicznego Zarządzania Ruchem*, *Zarządzania Bezpieczeństwem Ruchu*, *Zarządzania Parkowaniem* oraz *Informowania Kierowców i Pasażerów*. Struktura poszczególnych modułów została wyraziście przedstawiona na diagramie komponentów na rysunku 2.

Jak wspomniano wyżej, centralnym komponentem inteligentnego systemu informacji transportowej jest *Podsystem Monitorowania i Nadzoru Ruchu Pojazdów*. Podsystem jest rozbudowany hierarchicznie w modułach, w związku z tym pozostałe podsystemy są powiązane z nim z wykorzystaniem związku zależności. Stanowi on jeden z podstawowych rodzajów związków w diagramach UML i oznacza taką relację pomiędzy dwoma elementami modelowania, w której zmiana jednego z nich – niezależnego, wpływa na drugi element (w tym przypadku komponent) – zależny.

System obsługi magazynu, wspomagany narzędziami teleinformatycznymi

Odpowiednim przykładem nowoczesnego systemu informatycznego wykorzystywanego w gospodarce cyfrowej jest system zarządzania inteligentnym magazynem, zwanym również e-magazynem. Taki system przydziela zadania związane z obsługą magazynu poprzez wysyłanie poleceń do osób zajmujących się obsługą magazynu, najczęściej do operatorów wózków widłowych. Po otrzymaniu polecenia pracownik ma za zadanie wykonać zawarte w nim instrukcje przemieszczenia towaru. Zadanie uznaje się za zakończone, gdy zeskanowany zostanie kod kreskowy towaru, którego dotyczyło polecenie. Po zeskanowaniu kodu wydanie bądź zmiana miejsca przechowywania towaru są odnotowywane w systemie bazodanowym. Taki system znacząco usprawnia pracę dużych magazynów, gdzie na dużej powierzchni magazynowej składowane są setki różnych towarów.

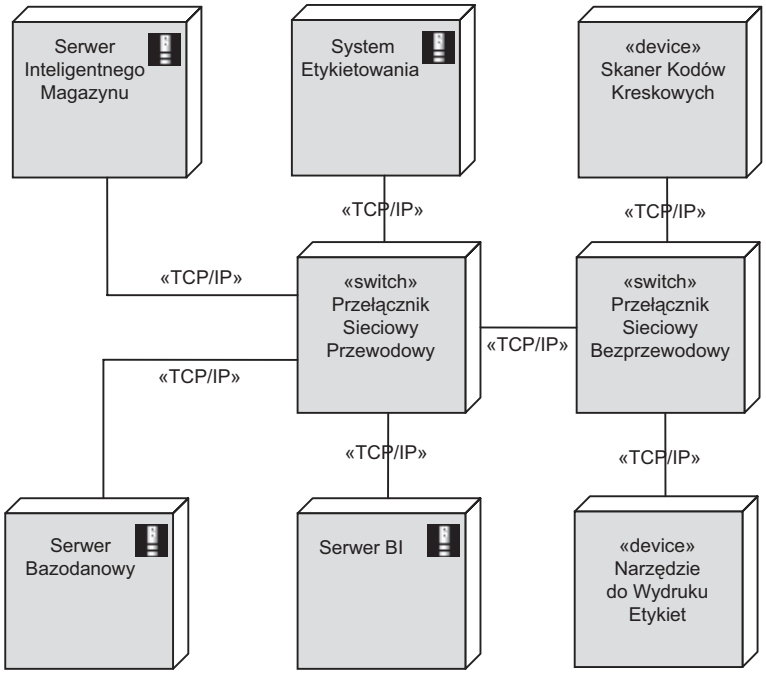
Zgodnie z koncepcją przedstawioną na rysunku 3 system składa się z następujących węzłów:

- *Serwer Inteligentnego Magazynu*,
- *Serwer Bazodanowy*,
- *Skaner Kodów Kreskowych*,
- *Serwer Business Intelligence*,
- *System Etykietowania*,
- *Narzędzie do Wydruku Etykiet*,
- *Przełącznik Sieciowy Przewodowy*,
- *Przełącznik Sieciowy Bezprzewodowy.*

System inteligentnego magazynu jest zatem złożonym systemem, którego funkcjonalność realizowana jest przez wymienione powyżej węzły. Jego rdzeniem jest *Serwer Inteligentnego Magazynu*, który zarządza komunikacją pomiędzy pięcioma węzłami: *Serwerem Bazodanowym*, *Serwerem BI*, *Systemem Etykietowania*, *Skanerem Kodów Kreskowych* oraz *Narzędziem do Wydruku Etykiet*.

Węzeł o nazwie *Serwer Bazodanowy* jest odpowiedzialny za przechowywanie informacji o zaplanowanych zadaniach, zaplanowanych i wykonanych poleceniach, pracownikach wykonujących polecenie oraz towarach znajdujących się w magazynie, jak również tych, które zostały wydane z magazynu.

Rysunek 3. Diagram rozlokowania dla systemu obsługi magazynu wspomaganego narzędziami teleinformatycznymi



Źródło: opracowanie własne.

Kolejnym węzłem jest urządzenie mobilne operatora – *Skaner Kodów Kreskowych*, który pozwala pracownikowi ewidencjonować wykonanie zaplanowanych wcześniej zadań. Jednocześnie służy on do wyświetlania zadań pozostających do wykonania. Informacja z systemu bazodanowego jest przekazywana do narzędzia klasy *Business Intelligence*, które reprezentowane jest przez węzeł *Serwer BI*. Dzięki temu narzędziu analizowane są: czas pozostawania towaru w magazynie i czas oczekiwania na wykonanie zadania od otrzymania polecenia do wydania towaru, co pozwala na optymalizowanie wydajności pracy na poszczególnych stanowiskach.

Węzeł *System Etykietowania* przekazuje pracownikowi informacje o nowym towarze przyjętym do magazynu. Gdy taka informacja dociera do pracownika, ma on możliwość oznaczenia nowego towaru etykietą, korzystając z *Narzędzia do Wydruku Etykiet*. Oznaczenia na etykietach, które pracownik drukuje przy pomocy tego narzędzia, są dostarczane przez *System etykietowania* i obejmują m.in. następujące informacje: kod i nazwę towaru, dane dostawcy oraz datę przyjęcia towaru, a także identyfikator pracownika przyjmującego towar.

Łączność pomiędzy urządzeniami odbywa się przy pomocy *Przełączników Sieciowych Przewodowego* i *Bezprzewodowego*. Zastosowanie *Przełącznika Sieciowego Bezprzewodowego* jest niezbędne, gdyż zarówno *Narzędzie do Wydruku Etykiet*, jak również *Skaner Kodów Kreskowych* to urządzenia mobilne, z którymi pracownik porusza się w obrębie magazynu.

Zastosowanie urządzeń mobilnych do zakupu biletów elektronicznych

Obecnie coraz częściej spotyka się możliwość uiszczania opłat za różne usługi z wykorzystaniem telefonu komórkowego. Jednym z przykładów jest sprzedaż biletów elektronicznych umożliwiających przejazd pojazdami komunikacji miejskiej. W typowym rozwiązaniu tego typu użytkownik, po bezpłatnym zarejestrowaniu się w serwisie usług mobilnych, ma możliwość pobrania na telefon aplikacji umożliwiającej kasowanie biletów. W zależności od wybranej opcji pieniądze za skasowane bilety są pobierane z konta *prepaid* lub z rachunku bankowego. Aby skasować bilet, użytkownik musi wybrać w aplikacji mobilnej m.in. środek transportu (autobus, tramwaj itp.) wraz z numerem linii, którą ma zamiar się poruszać, oraz rodzaj biletu (czasowy, jednorazowy). W przypadku kontroli biletu użytkownik powinien pokazać na ekranie swojego telefonu komórkowego informację o skasowanym bilecie, która jest weryfikowana przez kontrolera biletów.

Na rysunku 4 zaprezentowano diagram rozlokowania, na którego węzłach umieszczono artefakty systemu obsługiwane przez dany węzeł. Rysunek przedstawia system wykorzystujący urządzenia mobilne w celu zakupu biletów oraz udostępniania innych funkcjonalności pozwalających na pisanie odwołań czy weryfikację stanu konta. Przykład ujmuje także komponenty implementowane przez poszczególne artefakty oraz odzwierciedla kluczowe zależności, które zachodzą w tym systemie.

Na diagramie przedstawionym na rysunku 4 użytkownik wykorzystuje *Urządzenie mobilne pasażera* (najczęściej jest to telefon komórkowy), które ujmuje dwa artefakty: *GUI.jar* (*Graphical User Interface*) oraz usługę rejestracji – *Rejestracja.class*. W modelowanym systemie artefakt *GUI.jar* to archiwum zawierające funkcjonalność webową używaną w celu zalogowania się do *Serwera aplikacyjnego*, pozwalającego na *Zakup biletów*, *Raportowanie* stanu konta oraz informacji o zakupionych biletach czy też na pisanie *Odwołań* w przypadku, gdy zachodzi taka konieczność. Komunikacja pomiędzy *Urządzeniem mobilnym pasażera* a *Serwerem aplikacyjnym* odbywa się z wykorzystaniem protokołu TCP/IP. Oznacza to, że zakup biletu jest możliwy poprzez sieć Wi-Fi lub GSM (np. 3G, 4G). Logowanie użytkownika do *Serwera aplikacyjnego* jest możliwe za pośrednictwem usługi *Autoryzacja.svc*. Ten artefakt jednocześnie wykorzystuje artefakt *Log* w celu zapisywania w pliku dziennika zdarzeń informacji o przebiegu procesu uwierzytelniania użytkownika. Drugi z artefaktów, zainstalowany na węźle *Urządzenie mobilne pasażera*, tj. *Rejestracja.class*, umożliwia zarejestrowanie się w bazie danych użytkowników.

Wszelkie informacje na temat użytkowników, zakupionych biletów, reklamacji i odwołań są przechowywane na *Serwerze bazodanowym*, który zawiera platforma użytkowania systemu *BD EBilet*. Baza danych gromadzi dane m.in. o *Klientach*, *Karach*, *Biletach*, *Kontrolerach* i *Odwołaniach*. Zarządzanie dostępem do środowiska bazy danych jest integralną funkcjonalnością artefaktu *e-Bilet.war*, implementującego m.in. komponent *Obsługa BD*, co podkreślono zależnością stereotypowaną «manifest».

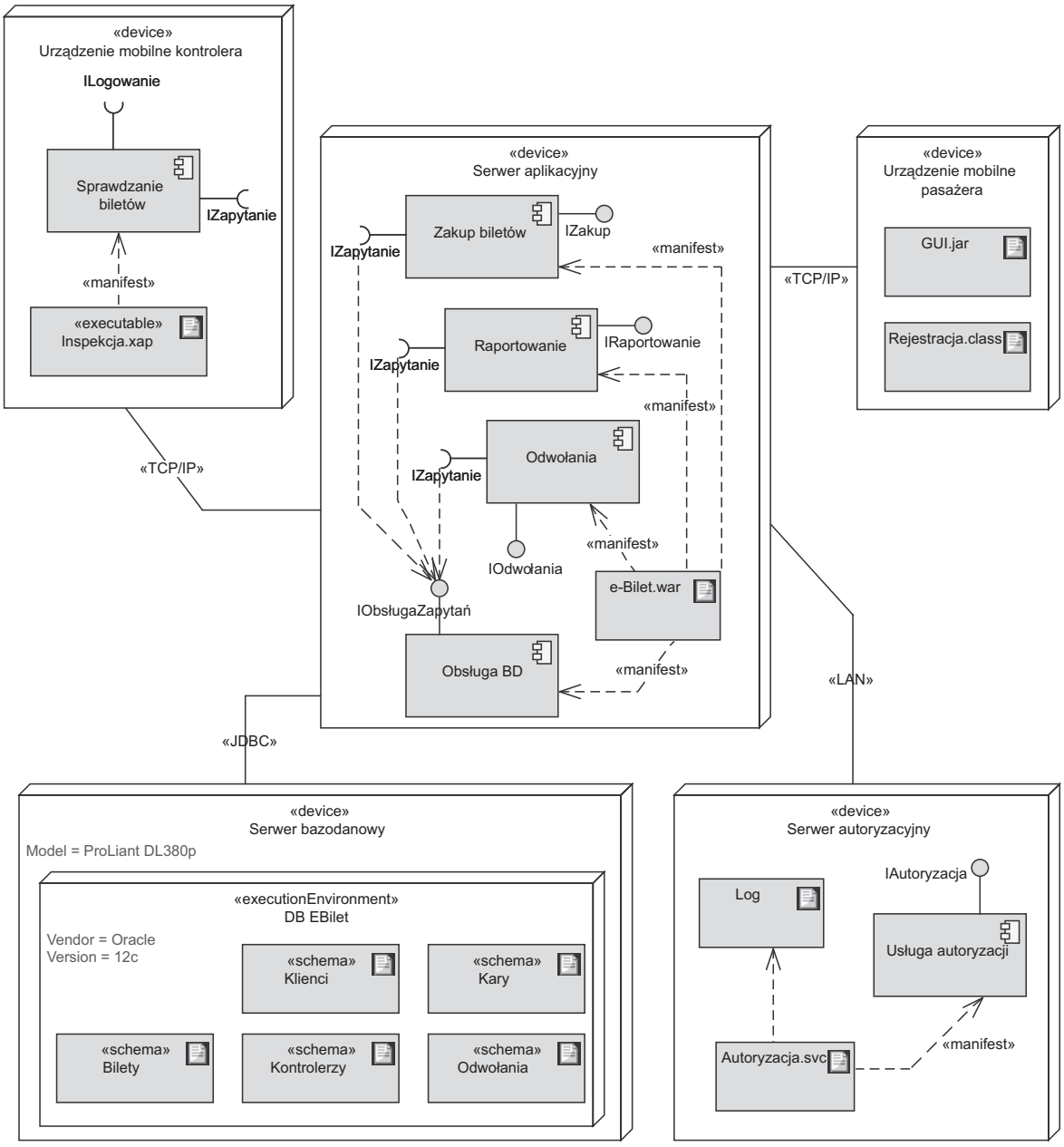
W przypadku konieczności weryfikacji biletów, kontroler wykorzystuje węzeł *Urządzenie mobilne kontrolera*, aby za pomocą artefaktu wykonywalnego *Inspekcja.xap* zalogować się do systemu – a następnie zweryfikować, czy bilety zostały skasowane, poprzez wysłanie zapytania (interfejs *IZapytanie* komponentu *Sprawdzanie biletów*) do *Serwera bazodanowego*.

Pomimo że nie jest to obligatoryjne, to jednak na diagramach rozlokowania zasadne jest sprecyzowanie sposobu transmisji danych pomiędzy węzłami. I tak, pomiędzy *Serwerem bazodanowym* a *Serwerem aplikacyjnym* występuje połączenie obsługiwane przez protokół JDBC (*Java DataBase Connectivity*). Pomiędzy *Serwerem aplikacyjnym* a *Urządzeniem mobilnym kontrolera*, jak również *pasażera* zastosowano protokół TCP/IP (*Transport Control Protocol*/*Internet Protocol*), który standardowo wykorzystywany jest w internecie. Jednocześnie oznacza to, że transmisja pomiędzy tymi urządzeniami może wykorzystywać istniejącą sieć.

Podsumowanie

Niniejszy artykuł podejmuje ważną tematykę związaną z modelowaniem systemów informatycznych, jaką jest modelowanie infra struktury sprzętu i oprogramowania. W celu identyfikacji potrzeb

Rysunek 4. Zakup biletu elektronicznego przez urządzenia mobilne – diagram rozlokowania z osadzonymi artefaktami



Źródło: opracowanie własne.

projektantów wynikających z dynamicznie zmieniających się technologii informatycznych wykorzystano diagramy wdrożeniowe języka UML: diagramy komponentów i rozlokowania.

Przedstawione w opracowaniu przykłady: inteligentnego systemu informacji transportowej, systemu obsługi magazynu wspomaganego narzędziami teleinformatycznymi oraz zastosowania urządzeń mobilnych do zakupu biletów elektronicznych, to nowoczesne formy wykorzystania informatyki w rozwiązaniach biznesowych gospodarki cyfrowej. Zaprezentowane diagramy wraz z opisami pozwalają na sformułowanie wniosku, że język UML – poza wspieraniem wysokopoziomowego modelowania procesów i interakcji – w pełni umożliwia modelowanie architektury sprzętu i oprogramowania tworzonego systemu.

Pomimo iż historia i ewolucja języka UML liczy już kilkanaście lat, to jego diagramy, w tym diagramy wdrożeniowe, wciąż nie tracą na aktualności i są szeroko stosowane, dzięki czemu stały się klasyką projektowania współczesnych systemów informatycznych. Przyczyniają się do tego liczne opracowania naukowe, wzbogacające język UML o rozszerzenia oraz modyfikacje pozwalające na modelowanie złożonych i zmieniających się aplikacji systemów informatycznych w biznesie i administracji.

Bibliografia

K.Z. Ahmed, C.E. Umrysh, *Developing Enterprise Java Applications with J2EE and UML*, Addison-Wesley, 2002.

S. Al-Fedaghi, A. Abdullah, *A New Approach to Component-Based Development of Software Architecture*, „International Review On Computers & Software" 2013, Vol. 8, No. 1, pp. 1–10.

V. Bianco, L. Lavazza, M. Mauri, G. Occorso, *Towards UML-based formal specifications of component-based real-time software*, [in:] *Fundamental Approaches to Software Engineering*, *6th International Conference*, Springer Verlag, 2003.

J. Erickson, K. Sian, *Theoretical and Practical Complexity of Modeling Methods*, „Communications Of The ACM" 2007, Vol. 50, No. 8, pp. 46–51.

B. Gawin, B. Marcinkowski, *Symulacja procesów biznesowych. Standardy BPMS i BPMN w praktyce*, Helion, Gliwice 2013.

J. Holt, *UML for Systems Engineering: Watching the Wheels, 2nd Edition*, The Institution of Engineering and Technology, 2007.

K. Jamroz, R. Krystek, W. Kustra, *Koncepcja zintegrowanego systemu zarządzania ruchem na obszarze Gdańska, Sopotu i Gdyni, Część I. Koncepcja ogólna Zintegrowanego Systemu Zarządzania Ruchem TRISTAR*, Urząd Miejski w Gdańsku, Gdańsk 2007.

P.B. Kruchten, *The 4+1 View Model of Architecture*, [in:] „IEEE Software" 1995, Vol. 12, No. 6, pp. 42–50.

S. Luján-Mora, J. Trujillo, I. Song, *A UML profile for multidimensional modeling in data warehouses*, „Data & Knowledge Engineering" 2006, Vol. 59, No. 3, pp. 725–769.

P. Shoval, J. Kabeli, *Special Theme of Research in Information Systems Analysis and Design – II. Data Modeling or Functional Modeling – Which Comes First? An Experimental Comparison*, „Communications of the Association for Information Systems" 2005, Vol. 16, pp. 831–847.

S. Wrycza, B. Marcinkowski, *A Light Version of UML 2: Survey and Outcomes*, [in:] *Proceedings of the 2007 Computer Science and IT Education Conference*, University of Technology Mauritius, 2007.

S. Wrycza, B. Marcinkowski, J. Maślankowski, *UML 2.x. Ćwiczenia zaawansowane*, Helion, Gliwice 2012.

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML w modelowaniu systemów informatycznych*, Helion, Gliwice 2005.

S. Wrycza, B. Marcinkowski, *Specyfikacja wymagań systemowych w języku SysML*, [w:] J. Goliński, A. Kobyliński, A. Sobczak (red.), *Technologie informatyczne w administracji publicznej i służbie zdrowia*, „Monografie i Opracowania Szkoły Głównej Handlowej w Warszawie" 2010, nr 569, pp. 323–337.

S. Wrycza, B. Marcinkowski, *SysML Requirement Diagrams: Banking Transactional Platform Case Study*, „Lecture Notes in Business Information Processing" 2011, No. 93, pp. 15–24.

S. Wrycza, B. Marcinkowski, *Towards a Light Version of UML 2.x: Appraisal and Model*, „Organizacija" 2007, No. 4, pp. 171–179.

S. Wrycza, B. Marcinkowski, *UML 2 Academic Course – Methodological Background and Survey Benchmarking*, [in:] *Proceedings of the 23rd Annual Conference for Information Systems Educators*, AITP Foundation for Information Technology Education, Dallas 2006.

S. Wrycza, B. Marcinkowski, *UML 2 Teaching at Postgraduate Studies – Prerequisites and Practice*, [in:] D. Colton, T. Janicki (eds.), *Proceedings of the 22nd Annual Conference for Information Systems Educators*, AITP Foundation for Information Technology Education, Columbus 2005.

S. Wrycza, *The UML 2 Academic Teaching Challenge: An Integrated Approach*, [in:] A. Bajaj, S. Wrycza (eds.), *Systems Analysis and Design for Advanced Modeling Methods. Best Practices*, IGI Global, 2009.

J. Zou, C. Pavlovski, *Control case approach to record and model non-functional requirements*, „Information Systems & E-Business Management" 2008, Vol. 6, No. 1, pp. 49–67.

Implementation diagrams in IT systems modeling

An inherent element of the documentation of IT system projects are the diagrams, which present the hardware and software infrastructure of the created systems. An example of such diagrams are implementation diagrams in the UML. There can be distinguished two types of implementation diagram, namely Component diagram and Deployment diagram. The aim of the present paper is to evaluate the possibilities of using the implementation diagrams for modeling of IT systems which use modern technology solutions. After discussing the role of implementation diagrams in IT systems modeling, the authors present also their functions and graphical conventions. Then, theoretical issues concerning diagrams and their practical uses have been examined and illustrated by the such examples of modern applications in the range of e-economy as: IT system of city traffic control, electronic warehouse management system and using mobile devices for purchase of e-tickets.

Stanisław Wrycza jest profesorem, kierownikiem Katedry Informatyki Ekonomicznej Uniwersytetu Gdańskiego. Współpracuje z czasopismami naukowymi: „Information Systems Management Journal" (IF2011=1.318, Senior Editor), „Information Systems Journal" (IF2011=2.067, Editorial Board) oraz „Journal of Database Management" (IF2011=0.875, Editorial Review Board).

Bartosz Marcinkowski jest adiunktem w Katedrze Informatyki Ekonomicznej Uniwersytetu Gdańskiego oraz certyfikowanym instruktorem Cisco i Object Management Group. Problematyką szkoleń e-learningowych oraz w formule blended learningu zajmuje się aktywnie od 2003 roku. Zainteresowania z zakresu analizy i projektowania systemów informatycznych (UML, BPMN) oraz architektury sieci komputerowych udokumentował licznymi publikacjami naukowymi.

Jacek Maślankowski jest doktorem nauk ekonomicznych, adiunktem w Katedrze Informatyki Ekonomicznej na Wydziale Zarządzania Uniwersytetu Gdańskiego. Jego zainteresowania naukowe skupiają się wokół szeroko pojętych zastosowań baz danych, w tym hurtowni danych, oprogramowania Business Intelligence oraz systemów *Big Data*. Autor licznych artykułów naukowych z tego zakresu, publikowanych zarówno w wydawnictwach krajowych, jak i zagranicznych.