POLSKIE TOWARZYSTWO INFORMACJI PRZESTRZENNEJ
ROCZNIKI GEOMATYKI 2007 ❍ TOM V ❍ ZESZYT 2

MODELOWANIE SYSTEMÓW INFORMACJI PRZESTRZENNEJ Z ZASTOSOWANIEM UOGÓLNIONEJ KLASY OBIEKTÓW

THE MODELING OF SPATIAL INFORMATION SYSTEMS WITH THE USE OF GENERALIZED CLASS OF OBJECTS

Edward Kołodziński[1], Grzegorz Betliński[2]

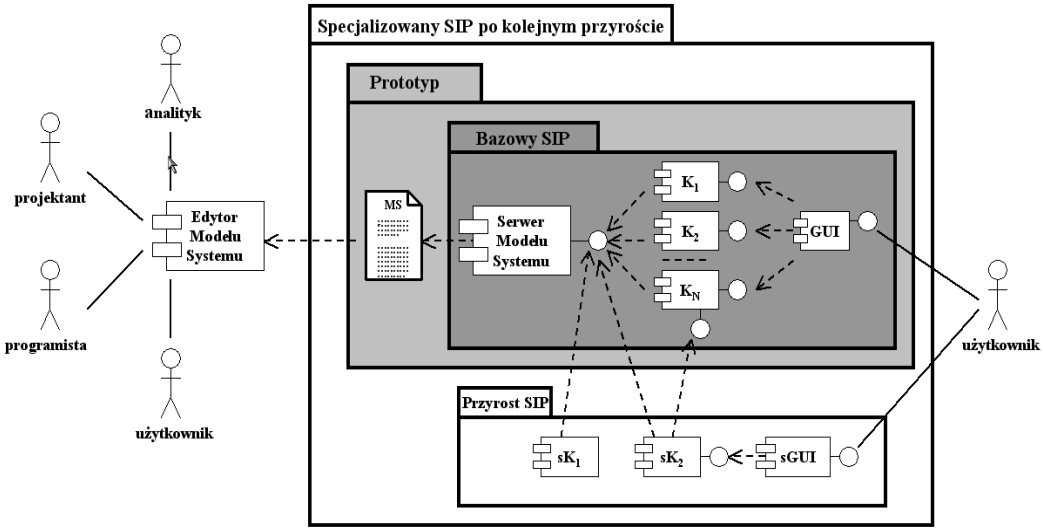
[1]Regionalne Centrum Informatyczne, Wydział Matematyki i Informatyki
Uniwersytet Warmińsko-Mazurski w Olsztynie
[2]Przedsiębiorstwo Projektowo-Wdrożeniowe INFOKART S.A.

Słowa kluczowe: inżynieria oprogramowania, metadane, model systemu, architektura systemu
Keywords: software engineering, metadata, system model, system architecture

Wstęp

W przypadku systemów informacji przestrzennej (SIP) jednym z najbardziej istotnych oczekiwań docelowych użytkowników jest **możliwie szybkie uzyskiwanie działającego prototypu dedykowanego oprogramowania użytkowego,** zapewniającego realizację, zazwyczaj długotrwałego, procesu ładowania bazy danych, który w większości przypadków obejmuje zarówno pozyskiwanie danych z istniejących już źródeł, np. cyfrowej mapy, jak i tworzenie całkowicie nowych danych, które dotychczas nie istniały w formie elektronicznej, takich jak np. opisowe dane dotyczące poszczególnych obiektów z założenia mających być przetwarzanymi w budowanym systemie. Spełnienie powyższego możliwe jest przy wykorzystaniu ogólnej architektury SIP pokazanej na rysunku 1.

SIP składa się tu z dwóch zasadniczych części: ***Edytora Modelu Systemu*** i pakietu ***Bazowy SIP***. ***Edytor Modelu Systemu*** (EMS) jest narzędziem CASE wykorzystywanym przez wszystkich uczestników procesu wytwarzania SIP, a więc również i przez jego przyszłego użytkownika. Rezultatem działania EMS jest elektroniczna wersja tzw. **Modelu Systemu**, na który składają się, m.in.: definicje klas obiektów, schemat bazy danych oraz opis graficznego interfejsu użytkownika. Definicje te dostępne są w ***Bazowym SIP*** za pośrednictwem specjalnie dla tego celu przeznaczonego komponentu **Serwer Modelu Systemu**. Działanie wszystkich pozostałych komponentów ***Bazowego SIP*** jest z założenia całkowicie uzależnione od tych definicji. Komponenty te (K_1, K_2, ... , K_N) udostępniają interfejsy zapewniające realizację wszystkich podstawowych funkcji SIP. Interfejsy te za pośrednictwem specjalnie dla



Rys. 1. Ogólna architektura SIP zapewniającego szybkie uzyskiwanie prototypu

tego celu przeznaczonego komponentu **GUI** *(Graphical User Interface)* udostępniane są użytkownikowi SIP. Zakładając zatem, że dysponujemy binarnymi wersjami wszystkich wymienionych wyżej komponentów programowych uzyskujemy tym samym w pełni funkcjonalny prototyp SIP składający się w istocie z ***Bazowego SIP***, którego działanie sparametryzowane jest całkowicie **Modelem Systemu**. Tak uzyskany prototyp udostępniany jest użytkownikowi, który może rozpocząć proces wypełniania bazy danych systemu i jednocześnie rozpoczynane są prace nad wytwarzaniem oprogramowania użytkowego docelowej wersji dedykowanego SIP. W trakcie realizacji prac związanych z kolejnymi przyrostami funkcjonalności wszyscy uczestnicy procesu wytwarzania SIP zajmują się przede wszystkim rozszerzaniem i/lub modyfikowaniem **Modelu Systemu**, a w wymagających tego przypadkach wytwarzane są dodatkowe i/lub modyfikowane uprzednio już wykonane komponenty programowe. W rezultacie przedstawionego postępowania budowany system w coraz większym stopniu zaczyna spełniać oczekiwania docelowego użytkownika – zarówno te wcześniej określone (przyjęte wprost bądź zmodyfikowane) jak całkiem nowe, powstałe z uświadomienia sobie rzeczywistych potrzeb w wyniku zapoznawania się z właściwościami wytwarzanego SIP. Nawet zakładając, że proces analizy wymagań został przeprowadzony najlepiej jak to było możliwe, to jednakże „najlepiej" dotyczy okresu, w którym analiza ta była realizowana. Po pewnym okresie eksploatacji praktycznie każdy system podlega naturalnej ewolucji, podczas której powstają nowe wymagania, ulegają zmianie wymagania uprzednio sformułowane, a niekiedy stają się całkowicie nieaktualne (często jest to rezultatem zmian w wykorzystywanych technologiach). Należy tu zauważyć, że komponenty programowe wchodzące w skład prototypu nie podlegają żadnym modyfikacjom – prace programistyczne z założenia obejmują jedynie oprogramowanie dodatkowych komponentów specjalizowanych (sK_1, sK_2, ... , sGUI), przy czym ich działanie, podobnie jak w przypadku komponentów ***Bazowego SIP***, musi być również całkowicie uzależnione od definicji **Modelu Systemu**, udostępnianych im za pośrednictwem interfejsów **Serwera Modelu Systemu**. Specjalizo-

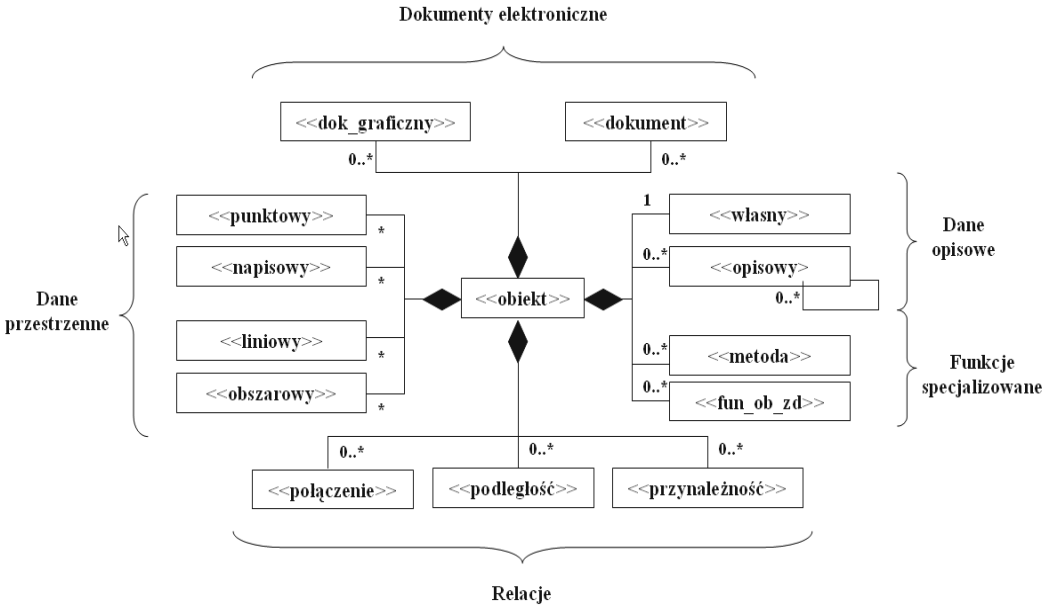
wane komponenty powinny również w możliwie największym stopniu wykorzystywać specjalnie dla tego celu przeznaczone interfejsy komponentów bazowych.

Uogólniona klasa obiektów

Modelowanie klas obiektów

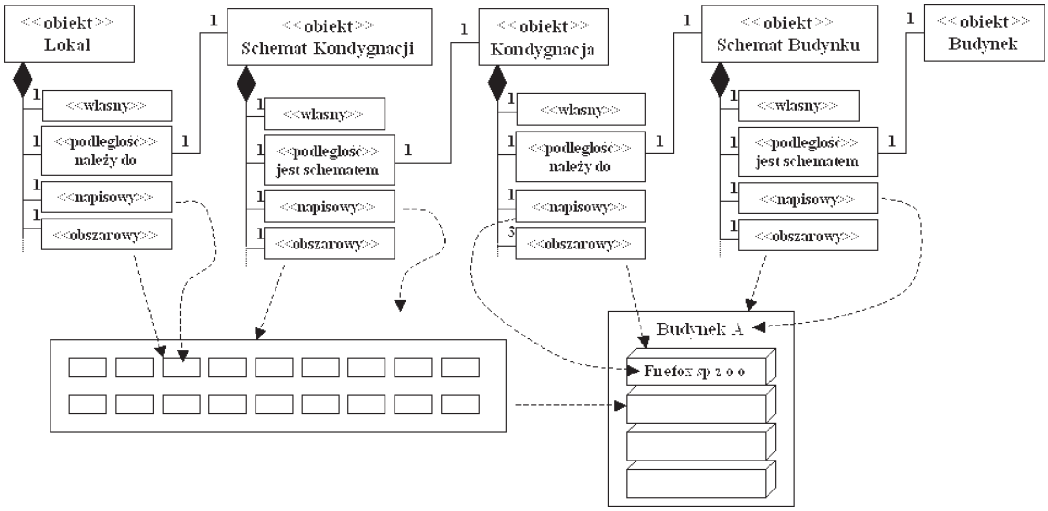
Oczywiste jest, że wykorzystanie w praktyce przedstawionej we wstępie koncepcji wytwarzania SIP wymaga opracowania metadanych, w oparciu o które określane będą wszystkie niezbędne definicje, w możliwie najwyższym stopniu opisujące budowany SIP.

W artykule przedstawione zostały metadane, których najistotniejszym elementem jest tzw. **Uogólniona Klasa Obiektów**, dzięki której SIP może być traktowany jako system przetwarzający w istocie jedną, uogólnioną klasę obiektów, której diagram przedstawiony jest na rysunku 2. Klasy obiektów definiowane są w tym przypadku zawsze za pośrednictwem całkowitej agregacji, a zatem jej elementy składowe nie określają niezależnie istniejących w systemie bytów i w związku z tym nie są one nazywane obiektami, lecz **specjalizowanymi komponentami obiektowymi**. Należy tu zwrócić uwagę na to, że specjalizowane komponenty obiektowe nie reprezentują określonych, ściśle zdefiniowanych, ustalonych zestawów atrybutów i metod, lecz w istocie odwzorowują one, wszystkie zidentyfikowane przez autora rodzaje (stereotypy) takich zestawów. Inaczej mówiąc przedstawiona na diagramie klasa uogólnionego obiektu, stereotypowana jako <<obiekt>>, jest w istocie szablonem klas obiektów, na podstawie którego definiowane są za pomocą ***Edytora Modelu Systemu*** klasy rzeczywistych obiektów przetwarzanych w budowanym systemie. Przykład modelowania klasy Budynek za pośrednictwem Uogólnionej Klasy Obiektów przedstawiono na rysunku 9.



Rys. 2. Uogólniona Klasa Obiektów

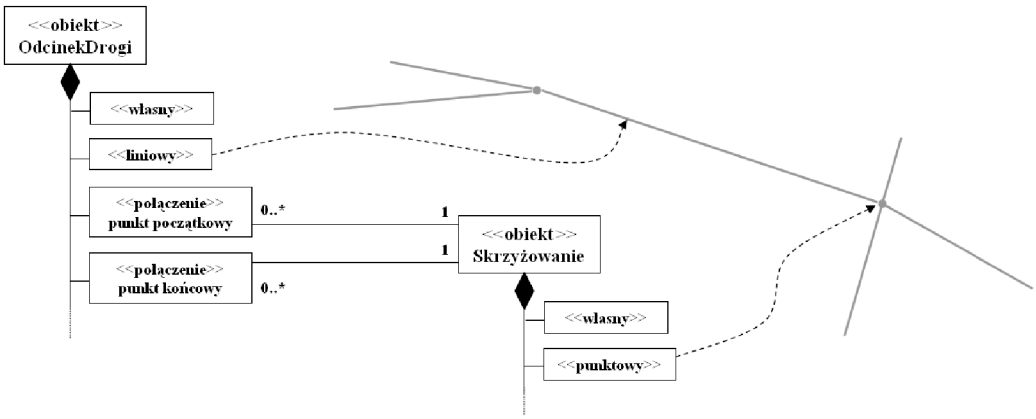
Komponent **<<własny>>** służy do opisywania charakterystycznych danych klasy obiektów, które mogą być wykorzystywane np. w celu identyfikacji poszczególnych obiektów. Komponent **<<opisowy>>** określa inne dane opisowe, które mogą być związane z obiektami danej klasy. Dane takie mogą być „zagnieżdżane" umożliwiając tym samym modelowanie złożonych struktur (w przykładowej klasie Budynek komponent Lokal jest zagnieżdżony w komponencie Kondygnacja). Komponenty graficzne (**<<liniowy>>, <<punktowy>>, <<obszarowy>>** i **<<napisowy>>**) określają sposób zobrazowania obiektu na mapie (wyznaczają lokalizację przestrzenną i graficzną reprezentację obiektu). Należy tu zauważyć, że diagram z rysunku 2 jest ze względów redakcyjnych uproszczony, gdyż w istocie komponenty graficzne posiadają swoje specjalizacje, np. specjalizacjami komponentu <<obszarowy>> mogą być: <<koło>>, <<prostokąt>>, <<kwadrat>>, <<trójkąt równoboczny>> itp., natomiast istotną w SIP specjalizacją komponentu <<liniowy>> jest <<odcinek>> (linia łamana o dwóch wierzchołkach). W przypadku klasy Budynek potrzebne są jedynie dwa komponenty graficzne. Jeden z nich <<obszarowy>> reprezentuje obrys budynku, natomiast drugi – <<napisowy>>, nazwę umieszczaną na mapie w jego pobliżu. Komponent <<**dokument>>** zapewnia możliwość wiązania z obiektem tzw. elektronicznych dokumentów, a więc plików (tekstowych, graficznych, arkuszy, baz danych, multimedialnych itp.) wytworzonych przez aplikacje działające w środowisku systemu operacyjnego, tzw. aplikacje własne dokumentów. Tak jak to jest pokazane na diagramie, z obiektami klasy Budynek moglibyśmy związać w ten sposób takie dokumenty jak np. zdjęcia, czy opisy. W tym ostatnim przypadku aplikacją własną w systemie Windows mógłby być np. MS Word. Specyficznym rodzajem dokumentów są tzw. dokumenty graficzne stereotypowane jako **<<dok_graficzny>>**. Służą one do wiązania z obiektami dokumentów, których zawartość może być za pośrednictwem definiowanej przez nie funkcji rysujących, odrysowywana we własnych oknach graficznych SIP. Przykładem może tu być dokument zawierający dane służące do rysowania na mapie ewentualnych stref skażenia obszaru terenu wokół budynku, z założenia przeznaczonego do magazynowania np. materiałów toksycznych. Dokument taki mógłby powstawać w rezultacie działania specjalnie dla tego celu przewidzianej w klasie Budynek metody reprezentowanej przez stereotypowany jako **<<metoda>>** komponent WyznaczObszarSkażenia. Metoda ta podczas swojego działania powinna zapewne uwzględniać nie tylko dane opisujące sam materiał toksyczny przechowywany w budynku (ilość, rodzaj), ale również ukształtowanie terenu (dane typu DTED) jak również bieżące warunki meteorologiczne (kierunek wiatru, opady itp.). Ostatnim komponentem zdefiniowanym w przykładowej klasie Budynek jest komponent relacji „jest własnością" stereotypowany jako **<<przynależność>>**. Służy on do wiązania z budynkiem jego współwłaścicieli, reprezentowanych zazwyczaj przez obiekty z dwóch klas: OsobaPrawna i OsobaFizyczna. Z założenia relacje wiążą zawsze dwa obiekty, z których jeden jest obiektem nadrzędnym relacji, a drugi podrzędnym. Podrzędnym jest ten, który w swojej strukturze posiada komponent relacji. Relacje dzielimy na „silne" i „słabe". Relacja silna charakteryzuje się tym, że usunięcie jej obiektu nadrzędnego powoduje również automatyczne usunięcie obiektu podrzędnego. Ponieważ usunięcie obiektu, reprezentującego w SIP właściciela budynku, nie oznacza jednoczesnego usunięcia budynku będącego jego własnością (współwłasnością), więc w celu modelowania takiego związku wykorzystana została relacja <<przynależność>>, która z założenia jest relacją słabą. Relacją silną jest natomiast relacja stereotypowana jako **<<podległość>>**. Na rysunku 3 przedstawiony został przykład wykorzystania tego typu relacji do modelowania schematów. Ponieważ wszystkie zaznaczone na diagramie relacje są relacjami podległości, więc usunięcie obiektu z klasy Budy-



Rys. 3. Modelowanie schematów obiektów za pośrednictwem relacji podległości

nek powinno spowodować kaskadowe usunięcie wszystkich powiązanych z nim obiektów z klas SchematBudynku, Kondygnacja, SchematKondygnacji i Lokal.

Szczególnym przypadkiem relacji podległości jest relacja połączenia reprezentowaną przez specjalizowane komponenty stereotypowane jako **<<połączenie>>**. Wykorzystywana do modelowania różnorakich sieci (rys. 4) wiąże zawsze obiekt liniowy (posiadający komponent liniowy) z obiektem punktowym (posiadającym komponent punktowy). Jest relacją silną, a zatem usunięcie obiektu punktowego (Skrzyżowanie) powoduje usunięcie wszystkich powiązanych z nim krawędzi. Z założenia również, wszelkie inne operacje edycyjne realizowane na obiektach pozostających ze sobą w relacji połączenia powinny zachowywać ich zależności topologiczne, a więc np. w konsekwencji przesunięcia obiektu punktowego (węzła sieci) odpowiednio modyfikowane powinny być wszystkie powiązane z nim obiekty liniowe (krawędzie sieci).



Rys. 4. Relacja połączenia

Stany obiektów

Przedstawiona powyżej struktura Uogólnionej Klasy Obiektów pozwala również nieco inaczej postrzegać stany obiektów. Oprócz definiowania stanów i tzw. ścieżek przejść ze stanu do stanu, możliwe jest dodatkowo związanie z każdym specjalizowanym komponentem obiektowym listy stanów, w których ten komponent występuje. Oznacza to, że zmiana stanu obiektu może skutkować znaczącą zmianą jego struktury, np. w zakresie reprezentacji graficznej, zestawu dostępnych metod, czy powiązań międzyobiektowych. Prosty przykład przedstawiony jest na rysunku 5.

Dla klasy Budynek przewidziane zostały dwa stany: Projektowany i Eksploatacja, dla których określona została odrębna symbolika graficzna (kolorystyka). Dodatkowo dla stanu Eksploatacja zdefiniowane zostały specjalizowane komponenty <<metoda>> i <<przynależność>> zapewniające możliwość określania relacji budynków z ich właścicielami. Podczas przejścia ze stanu Projektowany do stanu Eksploatacja, komponenty NapPr i ObszPr zastępowane są przez komponenty NapEkspl i ObszEkspl, a dodatkowo tworzony jest automatycznie komponent relacji „jest własnością", a wraz z nim komponent metody „Ustal właściciela", specyfikujący funkcję umożliwiającą wprowadzanie zmian w zakresie władania budynkiem.

Komponenty Uogólnionej Klasy Obiektów

Niezależnie od stereotypu wszystkie specjalizowane komponenty obiektowe Uogólnionej Klasy Obiektów posiadają wewnętrzną strukturę, której diagram został przedstawiony na rysunku 6. Stereotyp <<xxx>> oznacza tu jeden ze stereotypów komponentów specjalizowanych widocznych na diagramie Uogólnionej Klasy Obiektów, z wyjątkiem <<metoda>> i <<fun_ob_zd>>. Nazwa tych stereotypów pochodzi od nazwy abstrakcyjnej klasy oznaczonej jako *Xxx*, zastępującej na diagramie jedną z nazw: *Własny, Opisowy, Dokument, DokGraficzny, Liniowy, Obszarowy, Punktowy, Napisowy, Podległość, Połączenie* i *Przynależność.* Tak więc, np. specjalizowany komponent <<liniowy>> jest specjalizacją abstrakcyjnej klasy *Liniowy*, natomiast specjalizowany komponent <<własny>> jest bezpośrednim potomkiem abstrakcyjnej klasy *Własny*. Takie abstrakcyjne klasy (*Xxx*), których specjalizacjami są wcześniej omówione komponenty specjalizowane nazywamy **standardowymi komponentami obiektowymi**. Każdy z takich standardowych komponentów posiada ściśle określony, predefiniowany zestaw atrybutów i metod. I tak np. dla komponentu *Liniowy* przewidziane są atrybuty: punkty, kolorLinii, stylLinii i grubośćLinii, natomiast jego zestaw metod obejmuje, m.in.: wstawPunkt, usuńPunkt, ustalKolor, ustalStyl, przesuń, usuń, itp. Atrybuty standardowych komponentów obiektowych nazywamy **atrybutami standardowymi** i analogicznie ich metody – **metodami standardowymi**.

Widoczne na diagramie klasy <<metoda>> i Atrybut łącznie stanowią dekompozycję definicji dowolnej klasy: Atrybut reprezentuje atrybuty klasy, natomiast <<metoda>> jej metody, przy czym parametrom i rezultatowi metody odpowiadają klasy Parametr i Wynik, natomiast wykonywalny kod metody identyfikowany jest przez abstrakcyjną klasę *Funkcja*, będącą agregacją całkowitą klas: Nazwa (nazwa metody), Moduł (identyfikator modułu zawierającego kod metody), TypModułu (COM, DLL, ...) i TrybAktywacji (synchroniczny, asynchroniczny). Klasy <<metoda>> i <<fun_ob_zd>>, zgodnie z przyjętym wcześniej nazewnictwem, są również komponentami specjalizowanymi, posiadającymi wspólnego bezpośredniego przodka, którym jest standardowy komponent *Funkcja.* Poszczególne elementy składowe takich definicji identyfikowane są przez wartości atrybutów abstrakcyjnej klasy *Element.*

Definując w Modelu Systemu za pośrednictwem Edytora Modelu Systemu nową klasę obiektów tworzymy w istocie wystąpienie obiektu z klasy Uogólniona Klasa Obiektów. Wskazujemy które rodzaje komponentów specjalizowanych i z jaką krotnością mają wchodzić w jej skład, określamy nazwy, zarówno samej klasy jak i jej specjalizowanych komponentów, a następnie dla każdego z nich definiujemy atrybuty (klasa Atrybut) i zestawy związanych z nimi specjalizowanych metod (komponenty <<metoda>>) i funkcji obsługi zdarzeń (specjalizowane komponenty <<fun_ob_zd>>). Te definiowane przez użytkownika atrybuty i metody specjalizowanych komponentów nazywamy **specjalizowanymi atrybutami/metodami**. Dla atrybutów specjalizowanych, podobnie jak w przypadku atrybutów standardowych, możemy również określić ich wartości domyślne.

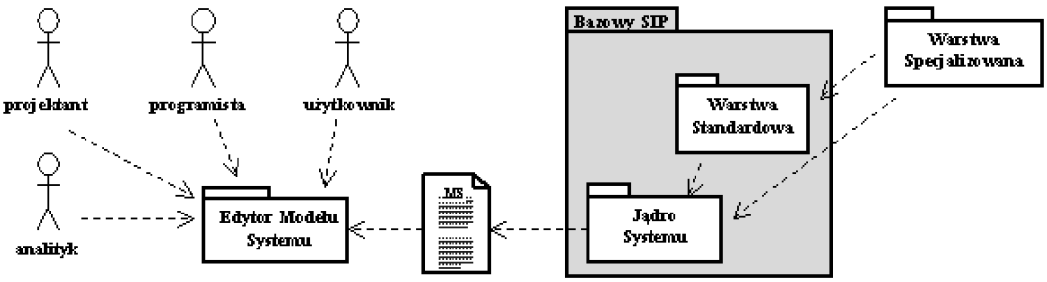
Funkcje obsługi zdarzeń

Z każdą klasą obiektów związane są przynajmniej dwie funkcje: konstruktor i destruktor. Zwykle jednak określane są dodatkowe funkcje (metody), które w stosunku do, czy na rzecz obiektu będą realizowały inne określone operacje. Jednak po wytworzeniu wynikowego oprogramowania funkcje takie w stosunku do obiektów danej klasy zawsze będą działały tak samo. W związku z powyższym modyfikacja ich działania możliwa jest dopiero po dokonaniu modyfikacji kodu źródłowego i ponownym zbudowaniu kodu wykonywalnego. Takie podejście wykorzystywane jest przez narzędzia CASE, które wspomagają automatyzację tworzenia kodu na podstawie przechowywanych w swoich repozytoriach definicji klas obiektów. W przypadku systemów działających w oparciu o koncepcję Uogólnionej Klasy Obiektów możliwe jest zastosowania całkowicie innego rozwiązania.

Mianowicie, niezależnie od klasy obiektu, zawsze wykorzystywane są te same uogólnione funkcje: tworzenia i destrukcji obiektów/komponentów, zmiany stanu obiektów, zmiany wartości atrybutu itp. Oznacza to, że w tych funkcjach mogą zostać zidentyfikowane istotne punkty, po osiągnięciu których mogą być uaktywniane inne funkcje, np. zaimplementowane w zewnętrznych komponentach programowych. W skrajnym przypadku te funkcje, służące do dostosowania funkcji standardowych, dostarczonych przez producenta SIP, do specyficznych potrzeb użytkownika, mogą być przez tegoż użytkownika projektowane i implementowane. W terminologii Uogólnionej Klasy Obiektów osiągnięcie przez metodę obiektu lub komponentu punktu, w którym będzie ewentualnie wywoływana funkcja zewnętrzna określane jest jako „zajście zdarzenia w życiu obiektu", natomiast samą tą zewnętrzną funkcję określa się mianem funkcji obsługi zdarzenia obiektowego. Takimi zdarzeniami w życiu obiektu, które potencjalnie mogą wymagać dodatkowej obsługi programowej, m.in. są: utworzenie obiektu, usunięcie obiektu, zmiana stanu obiektu, utworzenie komponentu, usunięcie komponentu, modyfikacja komponentu, czy modyfikacja wartości atrybutu. Na rysunku 7 przedstawiony jest diagram przypadku użycia „Utwórz nowy obiekt BUDYNEK". Jest to w istocie diagram uogólnionego przypadku użycia „Utwórz nowy obiekt" dotyczący dowolnej klasy, rozszerzony jedynie o przykładową funkcję obsługi zdarzenia CREATE_OBJECT dla klasy Budynek. W rezultacie jej wykonania okno graficzne zostanie wycentrowane na obszarze działki określonej podczas realizacji przypadku użycia „Edycja danych opisowych", w obrębie której użytkownik następnie umieści reprezentację graficzną nowo tworzonego obiektu. Jeśli taka funkcja obsługi zdarzenia CREATE_OBJECT nie będzie zdefiniowana dla klasy Budynek, to użytkownik będzie musiał samodzielnie ustalić właściwy obszar terenu zobrazowywany w oknie graficznym.

Uogólniona architektura SIP

Uogólniona Klasa Obiektów determinuje również w dużym stopniu ogólną architekturę SIP przedstawioną na rysunku 8.



Rys. 8. Ogólna architektura SIP zbudowanego na bazie Uogólnionej Klasy Obiektów.

W architekturze tej możemy wyróżnić cztery zasadnicze elementy: Jądro systemu, Warstwa Standardowa, Warstwa Specjalizowana i Edytor Modelu Systemu.

Edytor Modelu Systemu jest narzędziem typu CASE wspomagającym projektowanie i konfigurowanie dedykowanych wersji SIP. Rezultatem jego działania jest, przedstawiony w następnym rozdziale Model Systemu, który ze względu na swoje znaczenie może być w zasadzie traktowany jako równorzędny element składowy architektury SIP. Działanie wszystkich modułów oprogramowania systemu jest całkowicie uzależnione od definicji Modelu Systemu.

Jądro systemu zawiera komponenty zapewniające dostęp do wszystkich zasobów systemu. Zatem są to komponenty udostępniające m.in.: definicje Modelu Systemu, obiektowe dane przechowywane w bazach danych, dokumenty elektroniczne, czy rastrowe i wektorowe pliki tła.

Warstwa Standardowa implementuje większość najważniejszych funkcji zaawansowanego SIP wraz z odpowiednim dla ich realizacji graficznym interfejsem użytkownika. Zatem są to takie funkcje jak: pozyskiwanie danych, aktualizacja danych, wyszukiwanie danych, zobrazowanie danych, analizy danych czy udostępnianie danych w sieci WWW. Warstwa Standardowa udostępnia użytkownikowi wszystkie standardowe metody obiektowe, a więc metody zdefiniowane w standardowych komponentach obiektowych (*Własny, Opisowy, Dokument, Liniowy, ...*).

Warstwa Specjalizowana obejmuje wszystkie dodatkowe komponenty, zapewniające realizację wszystkich specyficznych funkcji, które są niezbędne dla zabezpieczenia tych wszystkich oczekiwań docelowego użytkownika, które nie są udostępniane przez Warstwę Standardową. W wymagających tego przypadkach komponenty Warstwy Specjalizowanej dostarczają również własny specjalizowany interfejs graficzny. Warstwa Specjalizowana udostępnia użytkownikowi wszystkie specjalizowane metody, a więc metody zdefiniowane w specjalizowanych komponentach obiektowych (*<<własny>>, <<opisowy>>, <<dokument>>, <<liniowy>>,...*). W skład Warstwy Specjalizowanej wchodzą również wszystkie aplikacje własne elektronicznych dokumentów jak również wszystkie funkcje obsługi zdarzeń.

Model Systemu

Model Systemu jest zbiorem obiektów należących do tej samej Uogólnionej Klasy Obiektów, a zatem programowy dostęp w SIP do wyznaczanych przez te obiekty definicji może być realizowany w ujednolicony sposób. Każdy z tych obiektów definiuje: albo jedną właściwą klasę obiektów, a więc klasę obiektów będących właściwym przedmiotem działania SIP, albo jedną z klas konfiguracyjnych określających np.: wykorzystywane układy współrzędnych, przetwarzane sieci, podziały systemowego obszaru terenu, wykorzystywane mapy podkładowe, standardowy interfejs graficzny (główne menu, kontekstowe menu, paski narzędziowe, drzewo rozwarstwienia – legenda, itp.). Na rysunku 10 przedstawiony jest fragment przykładowego Modelu Systemu obejmujący zarówno klasy właściwe jak i konfiguracyjne.

W celu ułatwienia zarządzania Modelem Systemu, zbiór wszystkich zdefiniowanych w nim klas obiektów jest podzielony na rozłączne podzbiory nazywane **kategoriami,** które są reprezentowane przez specjalnie dla tego celu definiowane klasy obiektów (na rys. 9 są one stereotypowane jako <<kategoria>>). Zawartość każdego z tych podzbiorów jest arbitralnie określana przez projektanta, na ogół zgodnie z życzeniami przyszłego użytkownika. Kategorie obejmujące znaczną liczbę klas obiektów mogą być podzielone na tzw. **grupy klas obiektów,** obejmujące podzbiory całego zbioru klas obiektów danej kategorii. Podobnie jak kategorie tak i grupy klas obiektów reprezentowane są przez specjalnie dla tego celu definiowane klasy obiektów (na rys. 10 posiadają stereotyp <<grupa>>). Zakłada się przy tym, że grupy są rozłączne, a więc klasa obiektu należąca do danej grupy, nie może należeć do żadnej innej grupy. W danej kategorii mogą również występować klasy obiektów nie należące do żadnej grupy, natomiast każda z klas obiektów występujących w Modelu Systemu musi należeć do jednej z kategorii w nim zdefiniowanych.

Wprowadzony za pośrednictwem kategorii i grup podział klas obiektów, rozszerzony o stany obiektów i wewnętrzny podział klas na komponenty, jest bezpośrednio odwzorowywany w graficznym interfejsie użytkownika w formie tzw. rozwijalnych drzew, z których jedno, tzw. drzewo rozwarstwienia (rys. 11), służy do ustalania wyświetlanych aktualnie warstw graficznych.

Przykładem konfiguracyjnej klasy, która zazwyczaj definiowana jest w Modelu Systemu może być klasa Mapy. Wykorzystywana jest dla określania przedziałów (tzw. map) wartości skali zobrazowania. Każdy taki przedział zdefiniowany jest przez odpowiadający mu komponent opisowy (rys. 12), z którym może być związanych wiele różnych komponentów określonych w klasach właściwych. Związki te determinują, które komponenty graficzne wyświetlane są dla bieżącej skali zobrazowania graficznego.

Podsumowanie

Zarysowane w referacie koncepcje Uogólnionej Klasy Obiektów i Modelu Systemu pozwalają na postrzeganie procesu wytwarzania SIP jako sekwencji czynności dopasowujących (konfigurujących) istniejące rozwiązanie do potrzeb informatyzowanej organizacji. Spełnienie wymagań docelowego użytkownika odbywa się przede wszystkim poprzez mo-

dyfikacje Modelu Systemu, a nie na drodze programowania, a tym samym „środek ciężkości" procesu wytwarzania SIP przeniesiony jest z programowania na konfigurowanie, a więc tym samym na ponowne wykorzystanie istniejących już komponentów programowych. Zminimalizowany zostaje wpływ typowo programistycznych czynników (ograniczenia języków programowania, błędy, testowanie itp.) na szybkość powstawania systemu. Możliwe jest jednoczesne projektowanie i użytkowanie systemu, gdyż modyfikacje Modelu Systemu automatycznie uwzględniane są przez oprogramowanie eksploatowanego systemu, a przy tym aktywne uczestnictwo użytkownika w procesie wytwarzania systemów informatycznych jest warunkiem sprzyjającym jego pomyślnego zakończenia. Ponadto, projektowanie jak również implementacja funkcjonalności przestaje być domeną „wtajemniczonych" i stają się dostępne w znacznym zakresie również dla przyszłego użytkownika. Wytwarzany SIP opisywany jest poprzez obiekty i relacje między nimi. Definicje te są jednak na tyle intuicyjne, że są zrozumiałe nie tylko dla specjalistów informatyków, ale także dla użytkowników systemów informatycznych, którzy będąc profesjonalistami w swoich dziedzinach, również postrzegają świat jako zbiór obiektów i są w stanie jednoznacznie określić, które z nich powinny zostać odzwierciedlone w systemie. Z kolei programista nie musi dostrzegać wszelkich zawiłości związanych z logiką działania systemu, lecz może koncentrować się na właściwej implementacji funkcji specjalizowanych, przy pełnej świadomości możliwości wpływu na Model Systemu. Jednocześnie otrzymuje na bieżąco informację zwrotną od użytkownika o jakości zaimplementowanych przez niego funkcji, dając mu możliwość szybkiego zidentyfikowania i usunięcia ewentualnych błędów.

Koncepcja wytwarzania SIP bazująca na Uogólnionej Klasie Obiektów została praktycznie zweryfikowana w formie tzw. Programowego Środowiska Geoba. W oparciu o to środowisko wytworzonych zostało do tej pory szereg SIP dedykowanych m.in. dla następujących zastosowań:

- ewidencjonowanie i zarządzanie sieci infrastruktury terenu: gazowej (POLGAZ), ciepłowniczej (SCSYSTEM), wodno-kanalizacyjnej (WOD-KAN), infrastruktury teleinformatycznej Wojsk Lądowych (EiZZT MON),
- wspomaganie pracy centrów antykryzysowych: wojewódzkich/powiatowych (PREVENT) i Ministerstwa Infrastuktury (PREVENT-POTT),
- ewidencjonowanie infrastruktury i zarządzanie zasobami uczelni.

Literatura

Booch G., Rumbaugh J., Jacobson I., 2002*:* Inżynieria oprogramowania. UML przewodnik użytkownika, WNT.

Kołodziński E., Betliński G., 2004: Modelowanie zależności topologicznych za pośrednictwem międzyobiektowych związków w programowym środowisku GEOBA**,** „Roczniki Geomatyki". Warszawa ISSN 1731-5522, T1I (Z2), ss. 121-128.

Kołodziński E., Betliński G., 2004: Resource Multi-Usage During Gis System Development Using GEOBA Programming Environment As Example Of EiZZT MON**,** 6th NATO Regional Conference On Military Communication And Information Systems 2004. Zegrze, 10-12 October, ISBN 83-920120-0-3, ss. 135–141.

Kołodziński E., Betliński G., 2004: Środowisko Programowe GEOBA do tworzenia systemów informacji o terenie – własności i możliwości. Opracowanie wewnętrzne*,* PPW INFOKART S.A. http://www.infocorp.com.pl/html/pub_sip_5.htm_.

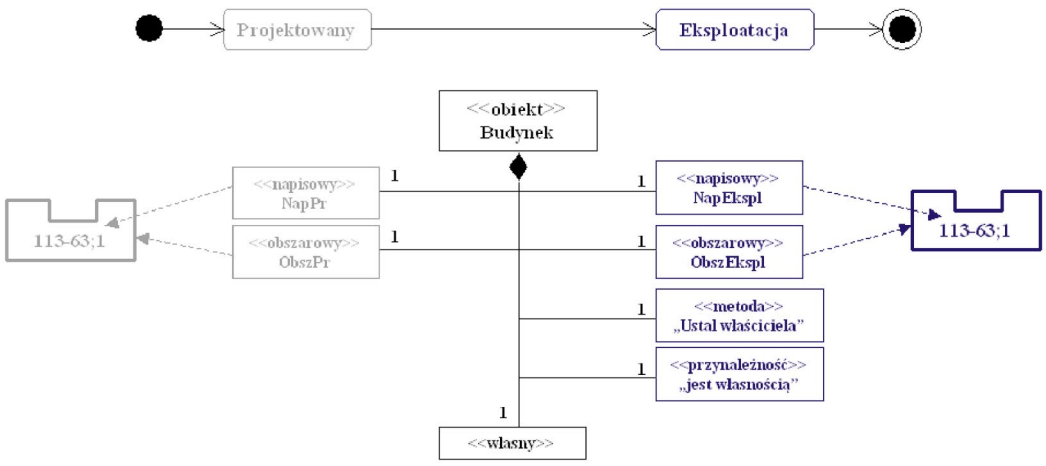
Sommerville I., 2003: Inżynieria oprogramowania, WNT.

Summary

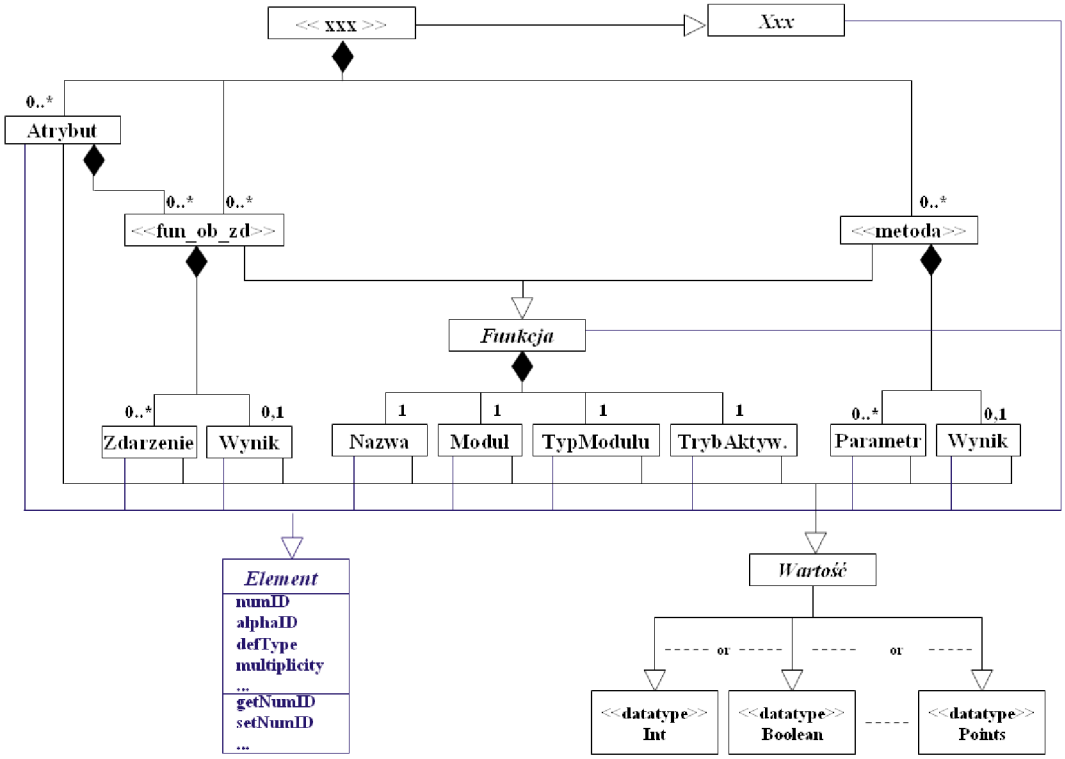
This paper describes a general concept of Generalized Class of Objects. Based on this concept, it is possible to construct models of object-oriented spatial information systems, which in fact deal with processing objects belonging to one class only. The Generalized Class is a composite of elements, so-called specialized object components, which are divided into five main groups: descriptive, documents, graphic, relations, and functions. Descriptive components contain non-spatial class attributes. However, graphic components determine graphic representation of objects. Documents make it possible to connect various standard electronic documents (texts, graphic files, sheets, etc) with individual object as well as specialized documents which contain results of specialized analyses (spatial, vector, network). Relation components are implementing all associations between objects including topological relationships such as connectivity or contiguity. Function components are designed for defining specialized static and non-static object methods as well as event service functions, which are called during life of several points of an object, such as object/component creation/deletion and so on. Each of specialized object components is a specialization of suitable standard object component with suitable sets of standard attributes and methods. These standard methods are responsible for realization of a majority of typical functions which are available in most of current spatial information systems (e.g. spatial or non-spatial data displaying, data updating, data selection or data highlighting). Through such a Generalized Class we can define not only real classes and their relationships but also all system configuration parameters (e.g. coordinate systems, several elements of GUI such as menu bars, toolbars, popup menus, etc). The set of such definitions (set of objects belonging to the Generalized Class of Objects) is a designed system model. It is produced by the CASE tool, so-called System Model Editor. The software of spatial information system is divided into three main layers: System Nucleus, Operating Layer and Specialized Layer. The operation of all binary components belonging to these layers completely depends on system model definitions. The System Nucleus modules make available all low-level system operations to the rest of the system modules, e.g. system resources access. The Operating Layer modules make available appropriate graphic user interface for realization of all standard methods defined in the Generalized Class of Objects. So, when we are ready to use binary components of these two layers, building a new System Model is equivalent to a new version of GIS, completely ready to implement by an end user. During such an implementation, it is simultaneously possible both to fill the system database, as well as to extend the Specialized Layer with new specialized methods in accordance with the end user expectations.

dr hab. inż. Edward Kołodziński, prof. UWM
ekolodzinski@poczta.wp.pl

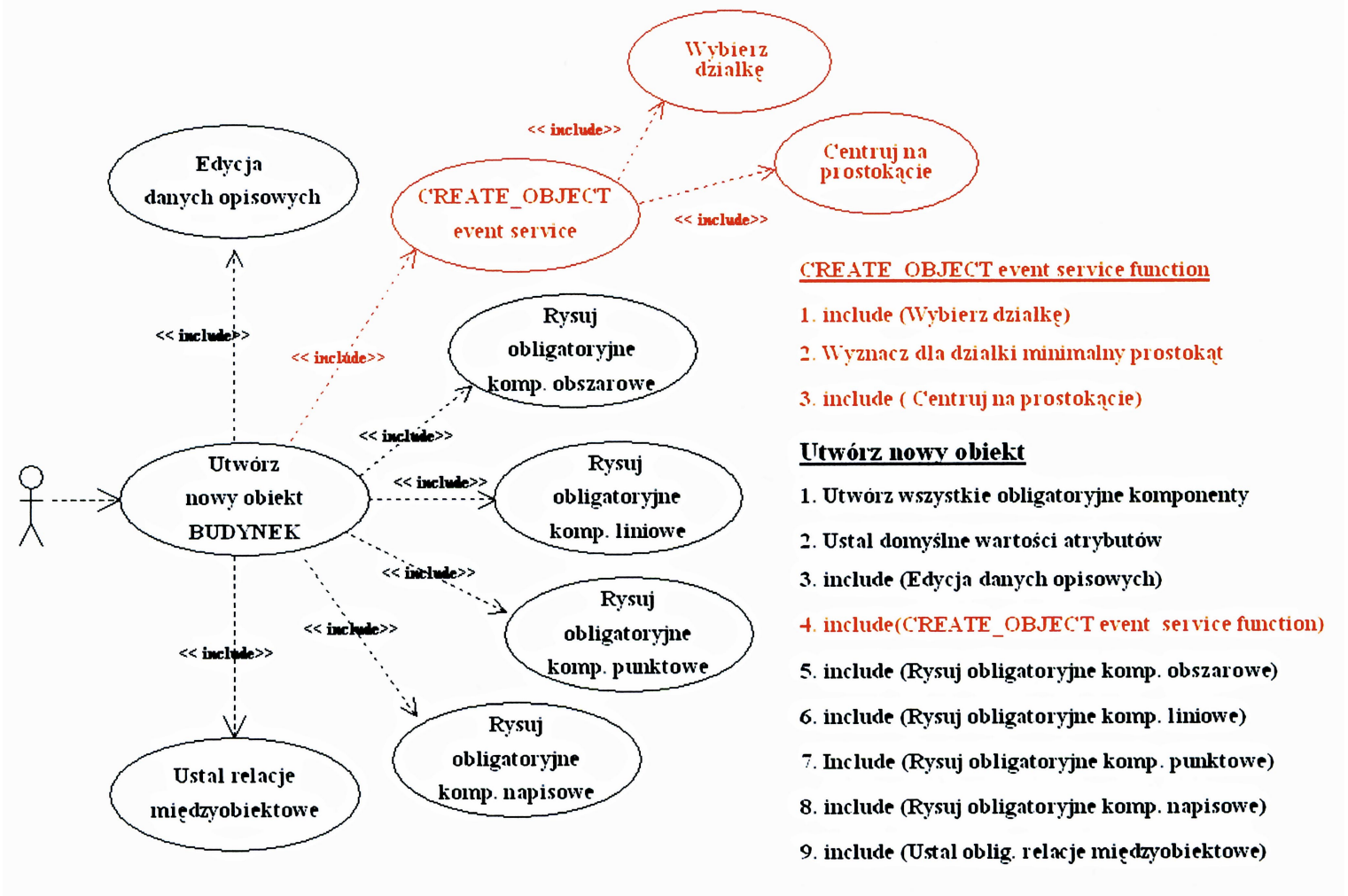
mgr Grzegorz Betliński
gbetl@wp.pl



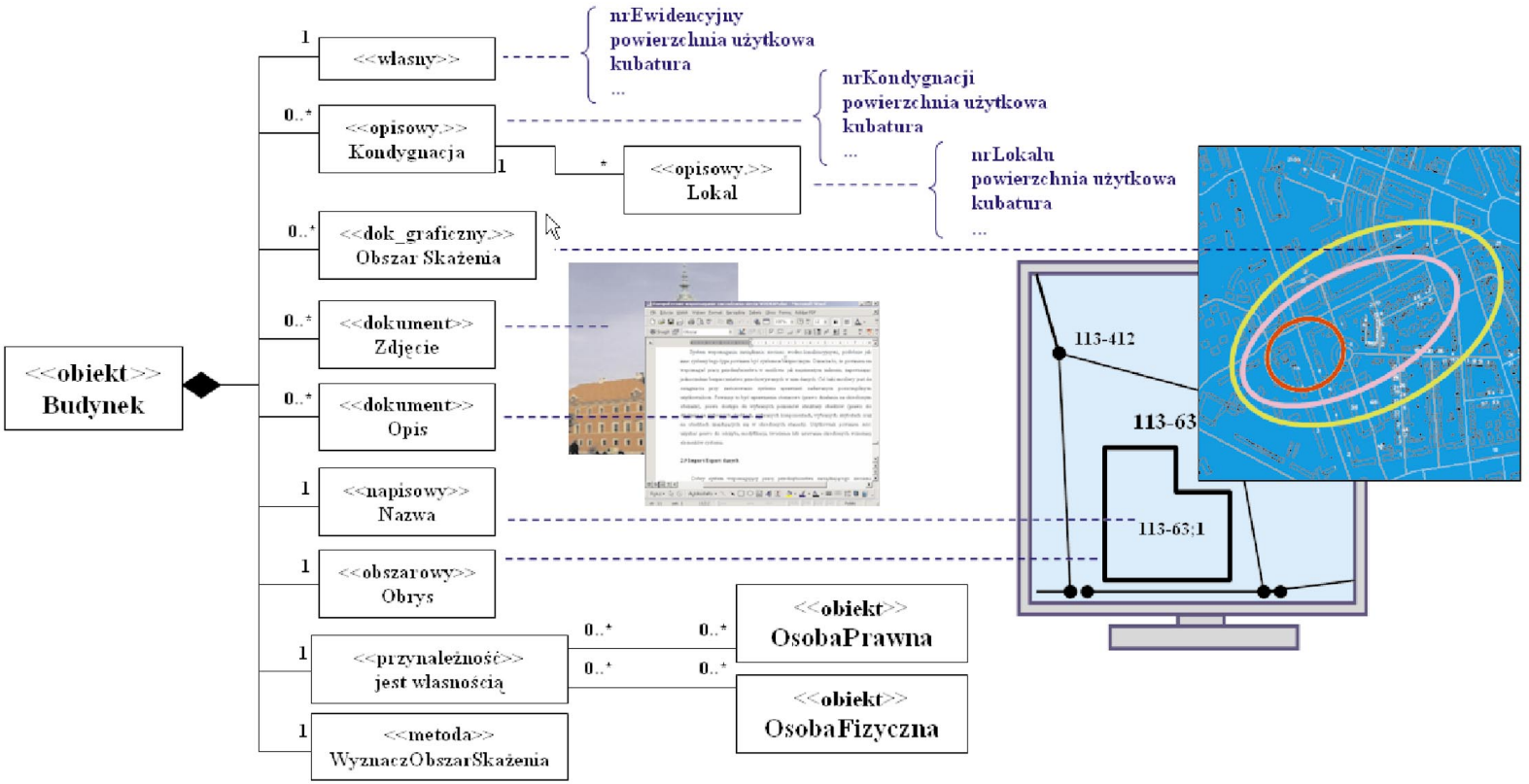
Rys. 5. Struktura obiektu uzależniona od stanu



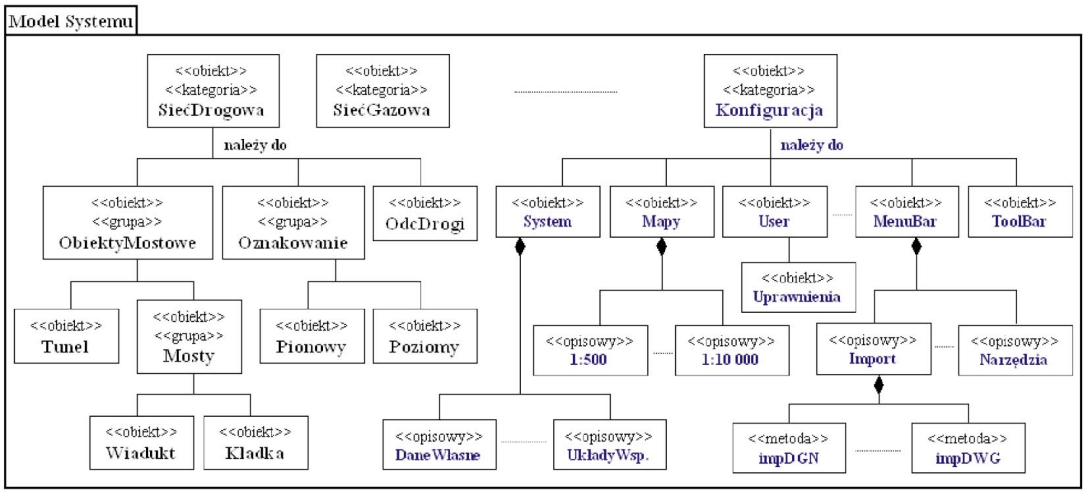
Rys. 6. Struktura komponentu Uogólnionej Klasy Obiektów



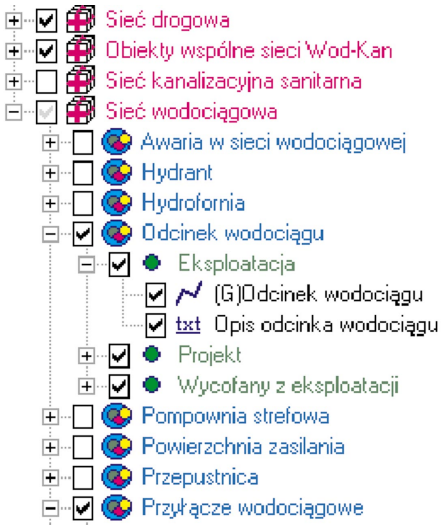
Rys. 7. Funkcja obsługi zdarzenia CREATE_OBJECT w przypadku użycia „Utwórz nowy BUDYNEK”



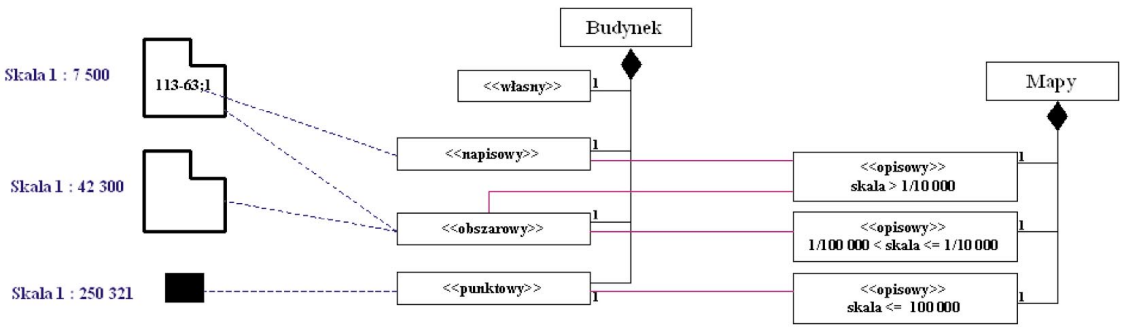
Rys. 9. Przykład modelowania klasy Budynek za pośrednictwem Uogólnionej Klasy Obiektów



Rys. 10. Fragment przykładowego modelu SIP



Rys. 11. Fragment drzewa rozwarstwienia generowanego automatycznie na podstawie Modelu Systemu



Rys. 12. Reprezentacja graficzna obiektu uzależniona od aktualnej skali zobrazowania