Wprowadzenie do modelowania w języku UML

Introduction to UML modeling

Konrad Szynalski[1], Dawid Różański[2]

STRESZCZENIE: UML, czyli Zunifikowany Język Modelowania, służy do zapisywania projektu systemu i może być również stosowany w celu graficznego opracowania lub tworzenia oprogramowania. Umożliwia on konstruowanie diagramów, które przedstawiają różne punkty widzenia systemu. W obecnych czasach UML stosuje się również w innych branżach, ponieważ przy jego pomocy można w komfortowy sposób analizować oraz modelować różne działania. Jedną z największych zalet tego języka jest fakt, że pozwala na jednakową interpretację wszystkich modeli przez osoby, które się nim zajmują. Co ważne, może być on zapisywany w formie graficznej przystępnej dla większości osób oraz w formie kodu ukierunkowanego na programistów.

ABSTRACT: UML or Unified Modeling language is used to save the system design and can also be used for graphic development or software development. It allows to create diagrams that present different points of view of the system. Nowadays, UML is also used in other industries because it allows to conveniently analyze and model various activities. One of the greatest advantages of this language is the fact that it makes it possible to equally interpret all the models by the people who deal with it. Importantly, it can be saved in a graphical form that is accessible to most people and in the form of code that is aimed at programmers.

SŁOWA KLUCZOWE: UML, modelowanie, zunifikowany język modelowania

KEYWORDS: UML, modeling, unifed modeling language

1. Wprowadzenie

UML, czyli Zunifikowany Język Modelowania, to graficzny język służący do wizualizowania, specyfikowania, konstrukcji i dokumentacji elementów związanych z tworzeniem systemów informatycznych[3]. Umożliwia standaryzację sposobu opracowywania przekrojów systemu obejmujących różnorakie obiekty takie jak funkcje systemowe czy schematy baz danych.

Sam język łączy w sobie najlepsze cechy z obszarów:
- modelowania danych (diagram związków encji),
- modelowania przepływów (diagram przepływu danych),
- modelowania obiektowego (analiza obiektowa),
- zarządzania złożonością (komponenty).

Początkowo głównym przeznaczeniem Zunifikowanego Języka Modelowania była budowa systemów informatycznych. Okazało się jednak, że UML jest bardzo dobrą formą przekazu informacji, więc zaczęto go również wykorzystywać w innych dziedzinach takich jak transport, telekomunikacja, przemysł obronny i lotniczy czy usługi bankowe. Zazwyczaj UML jest stosowany wraz ze swoją reprezentacją graficzną – jego elementom przypisane są symbole, które wiąże się na diagramach. Dzięki temu opracowane schematy pozwalają na dokładne śledzenie procesów. W aktualnej wersji UML oznaczonej numerem 2.5.1 można wyróżnić 13 rodzajów diagramów głównych oraz 3 typy interakcji[4]:
- diagramy struktury: diagramy klas, obiektów, komponentów, struktur złożonych, pakietów, wdrożenia;
- diagram zachowania: diagramy przypadków użycia, maszyny stanowej, czynności;
- diagramy interakcji: diagramy przeglądu interakcji, sekwencji, komunikacji, czasowe.

UML nie jest językiem programowania graficznego, ale modele w nim zapisane mogą być powiązane z wieloma językami programowania. Warto zaznaczyć, że model utworzony w języku UML można przekształcić, np. w tabele relacyjnej bazy danych. Oczywiście działa to również w drugą stronę i na podstawie implementacji da się stworzyć model graficzny. Co ważne, szczegółowość diagramów zależy od fazy tworzenia oprogramowania. Do analizy problemu można stworzyć diagram ogólny, a przy dokumentacji technicznej należy skorzystać z diagramu z większą liczbą szczegółów.

1. Wrocławska Wyższa Szkoła Informatyki Stosowanej, e-mail: konrad.szynalski@gmail.com, ORCID 0000-0001-9367-8088.
2. Wrocławska Wyższa Szkoła Informatyki Stosowanej, e-mail: drozanski@outlook.com, ORCID 0000-0002-9394-120X.
3. *Unified Modeling Language,* https://it.pwn.pl/Artykuly/Zarzadzanie-projektami/Unified-Modeling-Language-UML, dostępny w internecie [dostęp 4.12.2021]
4. *UML*, https://mfiles.pl/pl/index.php/UML, dostępny w internecie [dostęp 4.12.2021].

2. Standardy języka UML

W każdym języku istnieją pewne standardy i tak też jest w przypadku UML. Każdy obiekt reprezentowany jest przez jeden prostokąt, w którym zawierają się jego składniki. Prostokąt służy do umieszczania w nim nazw elementów, nazw wraz z polem lub nazw z polem i metodą. Do składników elementów można dołączyć również informacje o typie, typie zwracanym lub argumentach. Warto dodać, że składniki klasy mogą posiadać różne modyfikatory dostępu[5]:

+ jest składnikiem publicznym (public)
# jest składnikiem chronionym (protected)
- jest składnikiem prywatnym (private)
~ jest składnikiem dostępnym w obrębie projektu (package)

2.1. Reprezentacja klas abstrakcyjnych i interfejsów UML

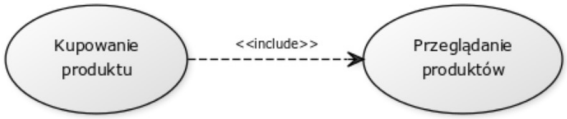
Do oznaczenia metod abstrakcyjnych, które znajdują się w klasie abstrakcyjnej, używa się kursywy tudzież podkreślenia. Sam interfejs przedstawia się tak jak klasy, lecz jego nazwa musi zostać poprzedzona słowem kluczowym <<interface>>. Co więcej, implementację interfejsu do danej klasy należy oznaczyć pustym białym grotem strzałki, który umieszcza się na końcu przerywanej linii. Należy pamiętać, że klasa implementująca interfejs musi wdrożyć jego metody. Jeśli chodzi o klasy abstrakcyjne, w tym przypadku klasa dziedzicząca musi implementować metody abstrakcyjne[6]. Jedną z największych zalet diagramów klas UML jest możliwość opisywania związków, które między nimi występują. Jeśli chodzi o relacje pomiędzy klasami, mogą w nich występować cechy krotności[7]:

1 – oznacza jeden obiekt,
0-3 – od zera do trzech obiektów,
„*”– dowolną ilość obiektów,
3 - * – od trzech do dowolnej ilości obiektów.

Co ważne, krotności należy umieszczać po obu stronach zależności. W przypadku gdy nie zostanie podana krotność, trzeba przyjąć, że jej wartość wynosi 1. Związki między klasami możemy podzielić na pięć typów takich jak zależność, asocjacja, agregacja częściowa, agregacja całkowita i dziedziczenie. Zależność między klasami informuje, że jedna z nich musi mieć informacje o drugiej, aby korzystać z jej obiektów. Występuje w momencie, w którym zmiana specyfikacji jednej klasy powoduje konieczność wprowadzenia zmian w innej klasie. Wynika z tego, że obie klasy są od siebie zależne. Jeśli chodzi o oznaczenia, używa się trzech typów:

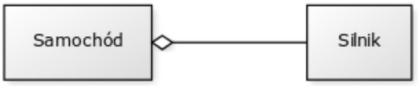
<<call>> – operacje w klasie A wywołują operacje w klasie B,
<<create>> – klasa A tworzy instancje w klasie B,
<<instantitate>> – obiekt A jest instancją klasy B,
<<use>> – do zaimplementowania klasy A wymagana jest klasa B.

Pierwszą i jednocześnie najsłabszą relacją jest zależność[8]. Oznacza ona, że jedna z klas chwilowo wykorzystuje inną klasę tudzież wie o jej istnieniu. Jeśli zajdzie jakaś zmiana w jednej z klas, może to spowodować (ale nie musi) konieczność zmian w drugiej klasie. Ważne jest, żeby zaprojektowany diagram zawierał jak najmniejszą liczbę zależności, ponieważ utrudniają one rozbudowę projektu.



Rys. 1. Zastosowanie diagramu zależności

Asocjacja przedstawia czasowe powiązania pomiędzy obiektami dwóch różnych klas[9]. W tym przypadku wszystkie obiekty są od siebie niezależne, a sama asocjacja jest też używana jako alternatywny sposób zapisu cech klasy. Asocjacja jest silniejszą relacją niż opisywana wcześniej zależność. Wskazuje, że dany obiekt jest powiązany z innym przez pewien czas (czas istnienia obu obiektów nie jest jednak od siebie zależny). Jeśli w jednym z nich zajdą modyfikacje w zmiennych lub zostanie on całkowicie usunięty, drugi będzie istniał w niezmienionej formie. Co ważne, w przypadku asocjacji żaden obiekt nie jest właścicielem drugiego, a zatem, dla przykładu, nie tworzy go ani nie zarządza nim. Należy też pamiętać, że obiekty, które zostały powiązane ze sobą asocjacją, mogą mieć nazwy, które zazwyczaj występują w postaci czasownika. Asocjacja może być jednokierunkowa oraz dwukierunkowa. Nazwy asocjacji wskazują bezpośrednio na czynności, które zachodzą pomiędzy klasami.

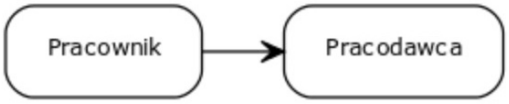


Rys. 2. Zastosowanie diagramu asocjacji

Kolejnym typem związków pomiędzy klasami w UML jest agregacja, która stanowi silniejszą odmianę zwykłej asocjacji. Agregacja występuje w dwóch odmianach: częściowej oraz całkowitej, inaczej nazywanej kompozycją. Agregacja częściowa jest związkiem dwóch klas w formie relacji całość-część. Warto dodać, że usunięcie

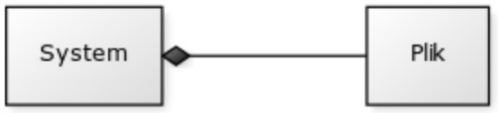
5. *Diagramy klas UML*, https://www.p-programowanie.pl/uml/diagramy-klas-uml, dostępny w internecie [dostęp 4.12.2021].
6. *UML Class Diagram Tutorial*: *Abstract Class with Examples*, https://www.guru99.com/uml-class-diagram.html, dostępny w internecie [dostęp 4.12.2021].
7. *Diagramy klas UML*, https://www.p-programowanie.pl/uml/diagramy-klas-uml, dostępny w internecie [dostęp 4.12.2021]
8. *UML Class Diagram Tutorial*, https://www.lucidchart.com/pages/uml-class-diagram, dostępny w internecie [dostęp 4.12.2021].
9. *UML Class Diagram Tutorial*: *Abstract Class with Examples*, https://www.guru99.com/uml-class-diagram.html, dostępny w internecie [dostęp 4.12.2021].

klasy „wszystko" nie wpłynie na klasę „fragment". W agregacji częściowej element częściowy może należeć do elementu głównego, lecz nie jest od niego zależny. Co za tym idzie, usunięcie elementu głównego nie wpływa w żadnym stopniu na element częściowy. Dodatkowo ten ostatni nie jest przywiązany tylko do jednego elementu głównego, ponieważ może należeć do kilku z nich.



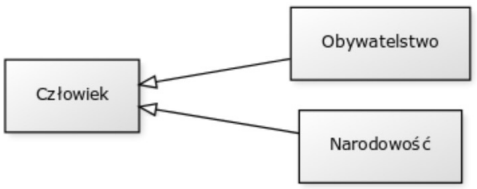
Rys. 3. Zastosowanie diagramu agregacji częściowej

Agregacja całkowita zwaną kompozycją działa podobnie do agregacji częściowej[10]. Największą różnicą jest fakt, że kontroluje cykl życia danej części. Oznacza to, że jeśli zostanie usunięta klasa główna, razem z nią zniknie powiązana z nią klasa częściowa. Co więcej, ani całość, ani część nie mogą bez siebie istnieć, dlatego czas ich działania jest ściśle ze sobą powiązany i się pokrywa.



Rys. 4. Zastosowanie diagramu agregacji całkowitej

Uogólnienie lub inaczej dziedziczenie tworzy hierarchię klas od tych ogólnych do najbardziej szczegółowych. Jest to główny filar paradygmatu programowania obiektowego. Umożliwia on wyodrębnienie cech wspólnych dla kilku klas i zamknięcie ich w klasie ogólnej, która ma wyższy poziom abstrakcji. Co ważne, klasy dziedziczące po klasie bazowej przyjmują jej cechy. Pozwala to na znaczne skrócenie kodu i zorganizowanie go od strony logicznej. Dziedziczenie ma kilka trybów: *overlapping, disjoint, complete* oraz *incomplete.*



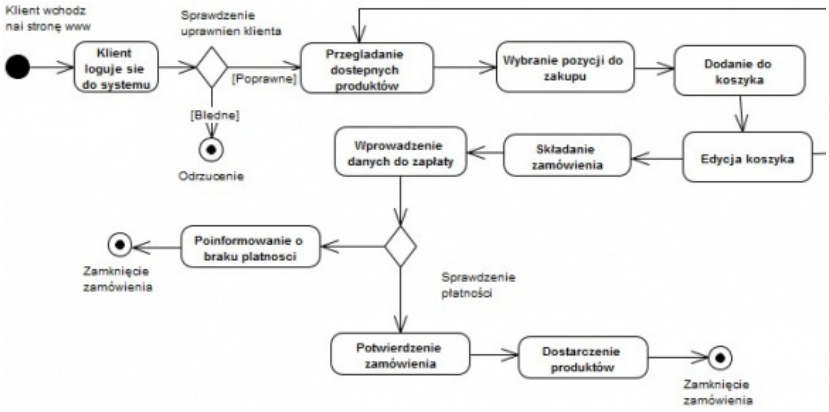
Rys. 5. Zastosowanie diagramu dziedziczenia

2.2. UML w perspektywie graficznej

Za pomocą języka UML w perspektywie graficznej da się przedstawić zasady działania pewnych procesów w biznesie. Dzięki temu w prosty i przejrzysty sposób na spotkaniu z klientem można zaprezentować mu sposób działania tworzonej aplikacji[11]. Modele, które są zapisane w języku UML, ułatwiają też rozszyfrowanie sposobu działania systemów osobom, które ten system mają tworzyć. Grupy projektowe omawiają modele i dzięki temu łatwiej jest im zrozumieć sposób, w jaki ma działać przyszła aplikacja. Graficzne modele są ustandaryzowane, więc modele tworzone w firmie A będą też zrozumiane, gdy zmienimy pracę i zobaczymy inny model w firmie B. Język, którym się posługujemy, by zobrazować rzeczywistość, byłby zbyt trudny, żeby opisać działanie całej aplikacji, dlatego też stosowanie języka UML we wczesnej fazie projektu jest bardzo ważne. Na rynku funkcjonuje wiele aplikacji oferujących tworzenie modeli w języku UML. Każdy architekt oprogramowania lub programista zaczyna tworzenie oprogramowania od rozrysowania wszystkiego na kartce – nawet taki projekt na kartce można nazwać diagramem UML, jeśli jest on prawidłowo wykonany. Osoby, które znają się na języku UML, patrząc na sam model, są w stanie tylko na jego podstawie stworzyć całą aplikację. Pokazuje to, jak bardzo taki model jest w stanie ułatwić prace nad nimi.

W perspektywie graficznej języka UML można wydzielić kilka rodzajów diagramów:

– Diagram przypadków użycia,
– Diagram pakietów,
– Diagram klas,
– Diagram aktywności,
– Diagram sekwencji,
– Diagram komponentów.



Rys. 6. Przykładowy diagram UML pokazujący działanie strony www do obsługi zamówień

2.3. Diagram przypadków użycia (use case diagram).

Ten rodzaj diagramu służy do przedstawienia funkcjonalności systemu wraz z jego otoczeniem, związków zachodzących między użytkownikami a systemem i zobrazowania usług, które są widoczne z zewnątrz systemu.

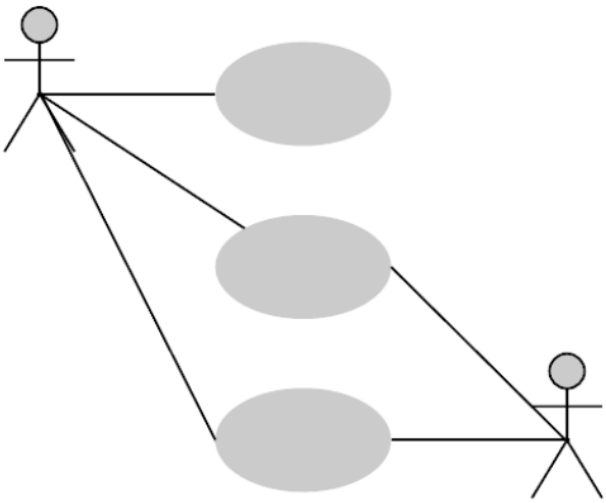
Diagramy przypadków użycia zbudowane są z kilku elementów, ale odgrywają najważniejszą rolę w procesie projektowania systemu. Opisują one wymagania funkcjonalne, jakie system musi spełnić, oraz otoczenie, w którym się znajduje. W tym samym czasie wspo-

10. *Diagramy klas UML*, https://www.p-programowanie.pl/uml/diagramy-klas-uml, dostępny w internecie [dostęp 4.12.2021].
11. *UML – Diagramy behawioralne*, https://it.pwn.pl/Artykuly/Zarzadzanie-projektami/UML-Diagramy-behawioralne, dostępny w internecie [dostęp 4.12.2021].

magają komunikację między uczestnikami projektu[12]. Przypadek użycia to scenariusze powiązane ze sobą wspólnym celem użytkownika. Pozwalają one na zdefiniowanie przyszłego, spodziewanego zachowania systemu, są stosowane w całej analizie systemu i mają za zadanie dostarczyć wyniki, z których użytkownik będzie mógł skorzystać. Przypadek użycia musi być powiązany z interakcją chociaż jednym „aktorem" (rolą, którą pełni użytkownik w stosunku do systemu oraz przypadków użycia) Każdy z nich możemy opisać za pomocą następujących cech:

- nazwa
- opis
- przepływ zdarzeń
- zależności i relacje
- diagramy aktywności
- wymagania specjalne
- warunki wstępne
- warunki końcowe

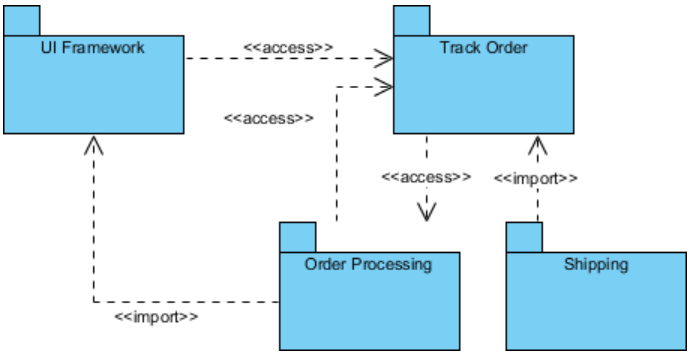
Najważniejszym aspektem opisującym przypadek użycia jest przepływ zdarzeń – scenariusze, które wskazują zestaw czy sekwencję kolejno wykonywanych czynności służących do zrealizowania funkcjonalności zobrazowanej przez dany przypadek użycia.



Rys. 7. Przykładowy diagram przypadków użycia

2.4. Diagram pakietów

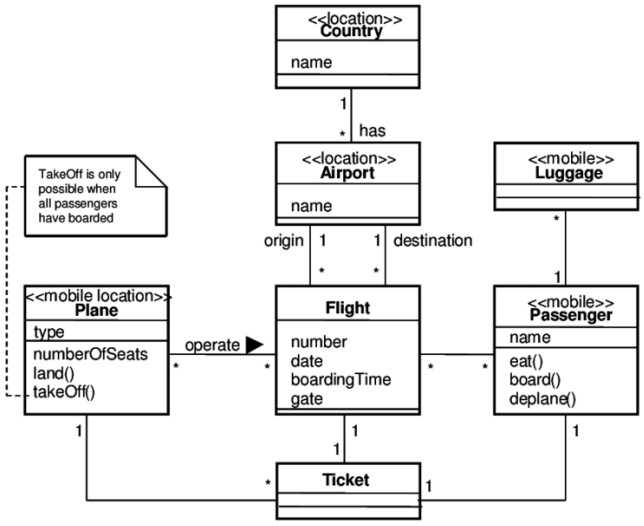
Jest to strukturalny diagram prezentujący pakiety i relacje, które między nimi zachodzą. Służy on do modelowania agregatów bytów, jakimi są pakiety. Pozwala on na modelowanie systemu na wysokim stopniu abstrakcji, gdyż pakiety reprezentują ogromną liczbę klas, interfejsów diagramów i innych bytów – pozwala on na przedstawienie tylko podstawowych cech systemu. Pakiet to uniwersalny mechanizm służący do organizowania elementów w grupy. Z jego pomocą grupuje się elementy modelu i diagramy. Pakiety to jeden z najczęściej stosowanych elementów, dzięki którym utrzymuje się porządek w repozytorium projektu[13].



Rys. 8. Zastosowanie diagramu pakietów

2.5. Diagram klas

Ten rodzaj diagramu obejmuje pewien zbiór klas, interfejsów i kooperacji oraz związki między nimi[14]. Stanowi on opis siatki systemu, który uwypukla związki między klasami, pomijając pozostałe charakterystyki. Podczas modelowania złożonych systemów nie trzeba przedstawiać ich struktury na jednym diagramie. Złożenie wszystkich diagramów (ich elementów i relacji) stanowi kompletny model. Podzbiory zbiorów klas użyte na diagramach klas są wybierane celowo i stanowią wynik decyzji analitycznych i projektowych.



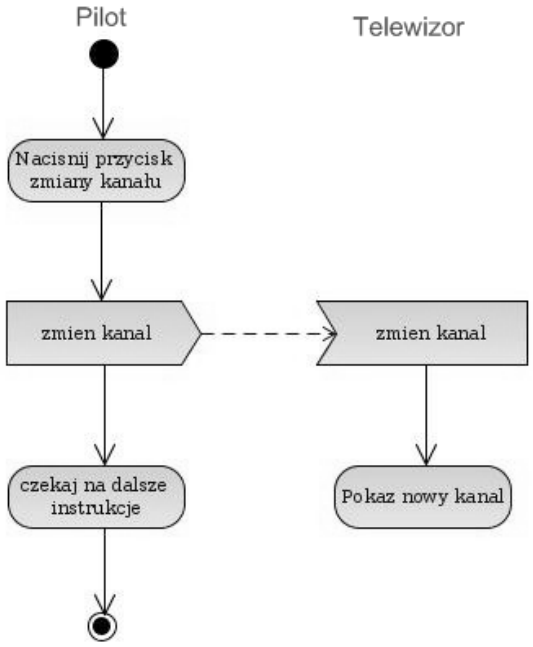
Rys. 9. Zastosowanie diagramu klas

2.6. Diagram aktywności

Diagram ten, zwany również diagramem interakcji, służy do modelowania dynamicznych aspektów systemu[15]. Jego główną funkcją jest przedstawienie sekwencji kroków, które są wykonywane przez modelowany fragment

12. *UML Use Case Diagram Tutorial*, https://www.lucidchart.com/pages/uml-use-case-diagram, dostępny w internecie [dostęp 4.12.2021]
13. *What is Package Diagram*?, https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/, dostępny w internecie [dostęp 4.12.2021].
15. *Diagram aktywności*, https://wolski.pro/diagramy-uml/diagram-aktywnosci/, dostępny w internecie [dostęp 4.12.2021].

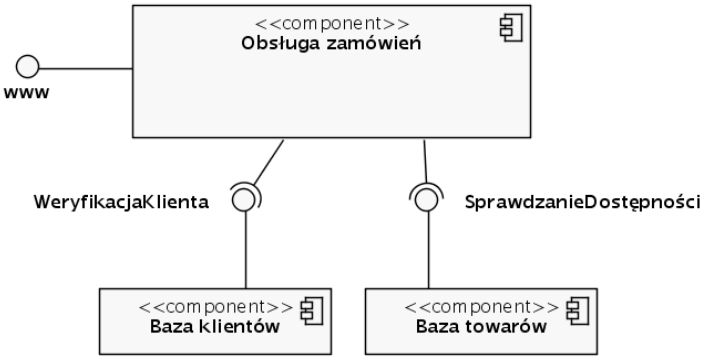
systemu.



Rys. 10. Zastosowanie diagramu aktywności

2.7. Diagram sekwencji

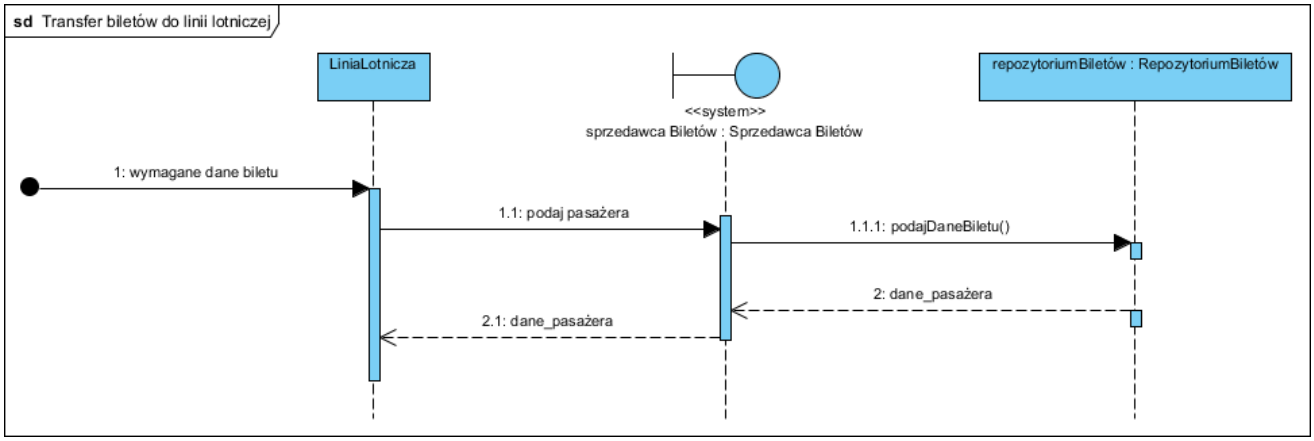
Ów diagram służy do prezentowania interakcji pomiędzy obiektami, ale z uwzględnieniem w czasie komunikatów, jakie są między nimi przesyłane[16]. Tego rodzaju diagramy pozwalają uzyskać odpowiedź na pytanie, jak przebiega komunikacja pomiędzy obiektami w czasie. Stanowią również jedną z podstawowych technik zachowania systemu składających się na realizację przypadku użycia.



Rys. 12. Zastosowanie diagramu komponentów

2.9. Język UML w praktyce

Na rynku jest wiele aplikacji oferujących tworzenie modeli w języku UML[18]. Każdy architekt oprogramowania lub programista tworzenie aplikacji zaczyna od rozrysowania wszystkiego na kartce – nawet taki projekt na kartce można nazwać diagramem UML, jeśli jest on prawidłowo wykonany. Można znaleźć wiele darmowych programów, które oferują tworzenie diagramów UML graficznie, oprócz tego znajdziemy również w nich gotowe rozwiązania. Wybierając odgórnie przygotowany szablon możemy być pewni, że taki model jest dobrze wykonany, wystarczy tylko dostosować go pod tworzoną aplikację. Według założeń projektowanie aplikacji rozpoczyna się od rozmowy podmiotu zamawiającego program z analitykiem, który próbuje zrozumieć specyfikę i potrzeby klienta. Analityk przekazuje informacje architektowi aplikacji i to właśnie zazwyczaj on odpowiedzialny jest za zaprojektowanie całego modelu. Gdy architekt stworzy taki model, trafia



Rys. 11. Zastosowanie diagramu sekwencji

2.8. Diagram komponentów

Diagram ten ilustruje organizację i zależności między komponentami[17]. Prezentuje on system na wyższym poziomie abstrakcji niż diagram klas. Służy do określania szczegółów niezbędnych do budowy systemów.

on do programistów, których zadaniem jest przełożyć model graficzny na gotową aplikację. Architekt aplikacji to zazwyczaj osoba z ogromnym doświadczeniem. Potrafi on zrozumieć aplikację jako całość. Aplikacja czy projektowany system to nie tylko zbiór kodu, to też połączenia z bazami danych, integracje różnych systemów. Architekt oprogramowania cały czas bierze udział w pracach zespołu, programiści mogą coś zmieniać w projekcie, pojawiają się nowe pomysły i korekty. Oprócz tego czuwa

16. *Diagram sekwencji,* https://wolski.pro/diagramy-uml/diagram-sekwencji/, dostępny w internecie [dostęp 4.12.2021].
17. *Diagram komponentów (component diagram),* http://zasoby.open.agh.edu.pl/~09sbfraczek/diagram-komponentow%2C1%2C17.html, dostępny w internecie [dostęp 4.12.2021].
18. *Best UML Tools*, https://www.guru99.com/best-uml-tools.html, dostępny w internecie [dostęp 4.12.2021].

on nad tym czy to, co zaprojektował, jest zgodne z tym, co piszą programiści[19].

3. Podsumowanie

Modele UML tworzy się po to, by zobrazować, jak ma działać program. Pozwala to lepiej zrozumieć potrzeby klienta i przedstawić działanie programu. Dzięki ujednoliconemu językowi da się postrzegać system z różnych punktów widzenia. Najważniejszym elementem podczas tworzenia oprogramowania jest komunikacja. Przełożenie myśli klienta na język informatyki to bardzo trudne działanie. Język, którym posługujemy się na co dzień, okazał się zbyt skomplikowany, by opisywać nim cały system. Powstanie UML umożliwiło w łatwy sposób opisywanie zasad funkcjonowania tworzonej aplikacji, pokazanie systemu z różnych punktów widzenia. Modele zapisane w tym języku prezentują system w sposób graficzny w postaci diagramów. Pokazują aplikację od ogółu do szczegółu, umożliwiając oglądanie modelu systemu z wybraną w danym momencie szczegółowością.
W dzisiejszych czasach aplikacje muszą być zaprojektowane tak, by w przyszłości można było je rozszerzać o nowe funkcjonalności. Model UML pozwala zaprojektować je już w pierwszych fazach tworzenia projektu. Co więcej, programiści nie muszą implementować od razu wszystkich dodatków. Mogą je dodawać wtedy, gdy zajdzie taka potrzeba. Modelowanie złożonych systemów jest trudnym zadaniem i angażuje się w to wiele osób o różnym sposobie postrzegania systemu. Modele w języku UML przedstawiają zazwyczaj „4+1" perspektywy. Cztery pierwsze to obraz wewnętrznej struktury programu na różnych stopniach abstrakcji i szczegółowości. Ostatnia perspektywa prezentuje funkcjonalność systemu widzianą przez jednego z użytkowników.

1. Perspektywa przypadków użycia – opisuje funkcjonalność aplikacji
2. Perspektywa logiczna – opisuje sposób realizacji funkcjonalności
3. Perspektywa implementacyjna – opisuje poszczególne moduły
4. Perspektywa procesowa – opisuje właściwości pozafunkcjonalne aplikacji
5. Perspektywa wdrożenia – służy integratorom i instalatorom systemu.

Modele w języku UML ułatwiają pracę programistom, osobom zaangażowanym w stworzenie danej aplikacji. Dzięki nim można też zaprezentować klientowi sposób działania systemu. W łatwy sposób można wytłumaczyć, jak aplikacja będzie na przykład odbierać zamówienia. Model w języku UML jest niezbędny przy większych projektach – dzięki niemu programiści mogą się dowiedzieć, w jaki sposób mają coś zaprogramować i jak coś ma działać. UML nie jest językiem programowania graficznego, jednak modele w nim zapisane mogą być wprost powiązane z wieloma językami programowania. Model utworzony w języku UML można przekształcić, np. w tabele relacyjnej bazy danych.

Bibliografia

1. *Best UML Tools*, https://www.guru99.com/best-uml-tools.html, dostępny w internecie [dostęp 4.12.2021]

2. *Diagram aktywności*, https://wolski.pro/diagramy-uml/diagram-aktywnosci/, dostępny w internecie [dostęp 4.12.2021]

3.*Diagram sekwencji*, https://wolski.pro/diagramy-uml/diagram-sekwencji/, dostępny w internecie [dostęp 4.12.2021].

4.*Diagram komponentów (component diagram)*, http://zasoby.open.agh.edu.pl/~09sbfraczek/diagram-komponentow%2C1%2C17.html, dostępny w internecie [dostęp 4.12.2021]

5. *Diagramy klas UML*, https://www.p-programowanie.pl/uml/diagramy-klas-uml, dostępny w internecie [dostęp 4.12.2021]

6. *Poradnik tworzenia diagramów i modelowania bazy danych w UML*, https://www.microsoft.com/pl-pl/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling, dostępny w internecie [dostęp 4.12.2021].

7. *UML*, https://mfiles.pl/pl/index.php/UML, dostępny w internecie [dostęp 4.12.2021]

8. *UML Class Diagram Tutorial*, https://www.lucidchart.com/pages/uml-class-diagram, dostępny w internecie [dostęp 4.12.2021]

9. *UML Class Diagram Tutorial*, https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/, dostępny w internecie [dostęp 4.12.2021]

10. *UML Class Diagram Tutorial: Abstract Class with Examples*, https://www.guru99.com/uml-class-diagram.html, dostępny w internecie [dostęp 4.12.2021]

11. *UML – Diagramy behawioralne,* https://it.pwn.pl/Artykuly/Zarzadzanie-projektami/UML-Diagramy-behawioralne, dostępny w internecie [dostęp 4.12.2021]

19. *Poradnik tworzenia diagramów i modelowania bazy danych w UML*, https://www.microsoft.com/pl-pl/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling, dostępny w internecie [dostęp 4.12.2021]

12. *UML Use Case Diagram Tutorial,* https://www.lucidchart.com/pages/uml-use-case-diagram, dostępny w internecie [dostęp 4.12.2021]

13. *Unified Modeling Language*, https://it.pwn.pl/Artykuly/Zarzadzanie-projektami/Unified-Modeling-Language-UML, dostępny w internecie [dostęp 4.12.2021]

14. *What is Package Diagram*?, https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/, dostępny w internecie [dostęp 4.12.2021]