

Inżynieria oprogramowania – wiedza techniczna, dotycząca faz cyklu życia oprogramowania, której celem jest uzyskanie wysokiej jakości produktu – oprogramowania. Jej głównymi zadaniami są:

- określanie cech dobrego programu
- poprawa produktywności
- dostarczenie narzędzi do tworzenia dobrych programów
- prawidłowa organizacja pracy projektantów
- wypracowanie standardów
- dostarczenie formalnych i pół-formalnych metod specyfikacji projektu
- umożliwienie modyfikowania oprogramowania – istniejącego i nowego
- ułatwienie pracy zespołowej (groupware)
- zapewnienie wysokiej jakości oprogramowania

Oprogramowanie wysokiej jakości jest zgodne z wymaganiami klienta, niezawodne, efektywne, łatwe w konserwacji i ergonomiczne.

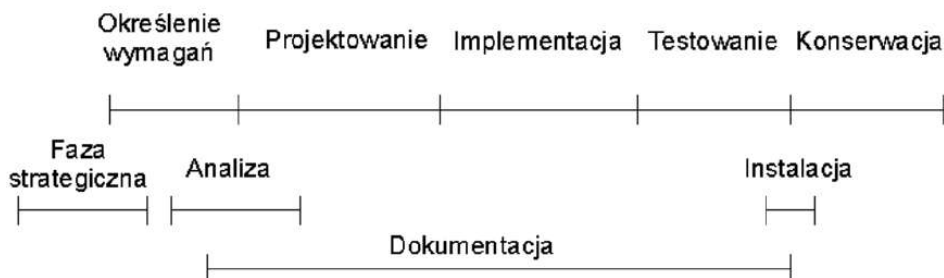
Charakterystyka CASE(Computer Aided Software/System Engineering):

- wspomaganie tworzenia oprogramowania w kilku fazach
- wizualizacja informacji (widoki, diagramy)
- reprezentowanie rozmaitych obiektów w formie symboli i połączeń na diagramach
- wspomaganie rozmaitych metodyk analizy (modelowanie koncepcyjne i logiczne)
- strukturalizacja informacji (danych i procesów)
- badanie poprawności i spójności informacji wyrażonej na diagramach
- utrzymywanie bazy danych o tworzonej projekcie (słownik danych, repozytorium, encyklopedia)
- stosowanie konstrukcji programowania strukturalnego i obiektowego na poziomie diagramów
- automatyczna generacja znacznej części procedur aplikacji
- współdzielenie informacji o tworzonej projekcie między autorami systemu; półautomatyczne dokumentowanie
- eliminowanie powielania danych i procesów
- możliwość wyboru systemu realizacyjnego oraz konwersji formatów danych

CASE dzieli się na 3 grupy:

1. upper (planowanie, analiza)
2. middle (analiza, projektowanie)
3. lower (projektowanie, budowa)

Etapy kaskadowego modelu życia oprogramowania (waterfall)

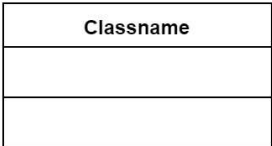


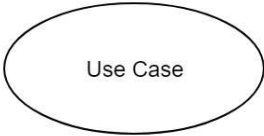
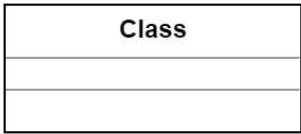
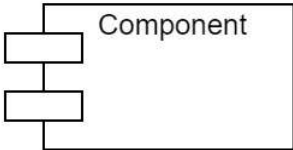
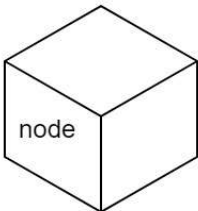


Faza analizy – jej celem jest udzielenie odpowiedzi na pytanie: „Jak system ma działać?”. Wynikiem jest logiczny model systemu, opisujący sposób realizacji przez system postawionych wymagań, unikając szczegółów implementacji.

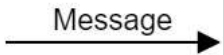
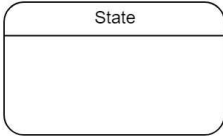
Faza projektowania – jej celem jest udzielenie odpowiedzi na pytanie: „Jak system ma zostać zaimplementowany?”. Jej wynikiem jest projekt oprogramowania, czyli opis sposobu jej implementacji. Sam projekt staje się później dokumentacją techniczną produktu.

UML


Elementy strukturalne – statyczne części modelu.

| | | |
|-----------------------------------|---|--|
| Class Klasa |  | Opis zbioru obiektów, które mają takie same atrybuty, operacje, związki i znaczenie. |
| Interface Interfejs |  | Zestaw operacji, które wyznaczają usługi oferowane przez klasę lub komponent. |
| Collaboration Kooperacja |  | Interakcja, zestaw ról i bytów, współdziałających w celu wywołania pewnego zespołowego zachowania niemożliwego do osiągnięcia w pojedynkę. |
| Use case Przypadek użycia |  | Opis zbioru ciągów akcji wykonywanych przez system w celu dostarczenia danemu aktorowi wyniku. |
| Active class (?) Klasa aktywna |  | Zawiera obiekty, w skład których wchodzi co najmniej jeden proces lub wątek. |
| Component Komponent |  | Fizycznie wymienna część systemu, która wykorzystuje i realizuje pewien zbiór interfejsów. |
| Node Węzeł |  | Fizyczny składnik działającego systemu, reprezentujący zasobu obliczeniowe. |

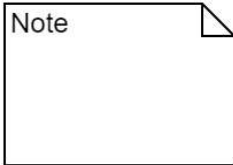
Elementy czynnościowe – dynamiczne części modelu

| | | |
|-----------------|---|---|
| Interakcja |  | Wymiana komunikatów między obiektami w pewnym otoczeniu w pewnym celu. |
| Maszyna stanowa |  | Ciąg stanów, jakie obiekt lub interakcja przyjmuje w odpowiedzi na zdarzenia zachodzące w czasie ich życia, określa ponadto ich odpowiedzi na te zdarzenia. |





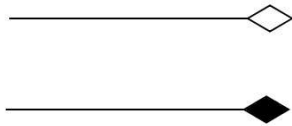
Elementy grupujące – elementy organizacyjne modelu

| | | |
|-------------------|---|-----|
| Package Pakiet |  | ??? |
|-------------------|---|-----|

Elementy komentujące – elementy objaśniające model

| | | |
|-----------------|--|---|
| Note Notatka |  | Symbol graficzny umożliwiający skojarzenie dodatkowych ograniczeń i objaśnień z pojedynczym bytem lub grupą bytów, nie ma wpływu na znaczenie modelu. |
|-----------------|--|---|

Związki

| | | |
|--|---|---|
| Zależność |  | Związek znaczeniowy między dwoma elementami, zmiany dokonywane w definicji jednego mogą mieć wpływ na znaczenie drugiego. |
| Powiązanie Asocjacja |  | Związek strukturalny, który określa zbiór powiązań, między obiektami. |
| Uogólnienie Dziedziczenie |  | Związek między dwoma bytami, ogólnym (przodek) i szczególnym (potomek). |
| Realizacja |  | Związek znaczeniowy między klasyfikatorami, z których jeden określa kontrakt, a drugi zapewnia wywiązanie się z niego. |
| Agregacja częściowa Agregacja całkowita |  | Role, np. pracownik – pracodawca. Liczebność: 1 = jeden, 0..1 = zero lub jeden, 0..* = dowolnie wiele, 1..* = co najmniej jeden, może też być dowolna liczba. Agregacja – związek całość – część Agregacja całkowita – relacja całkowita |

Diagramy – graf, którego wierzchołkami są elementy, a krawędziami związki; wyróżniamy następujące diagramy:

Diagramy struktur:

- Diagram klas – pozwala na sformalizowanie specyfikacji danych i metod. Mogą także pełnić rolę graficznego środka pokazującego szczegóły implementacji klas. Mają także na celu przedstawienie struktury systemu bądź jego fragmentu oraz przedstawienie zależności między elementami tej struktury (między klasami).

W diagramie klas występują związki:

- zależność
 - powiązanie
 - uogólnienie
 - realizacja – związek między interfejsem a klasą – interfejs zapewnia realizację usług klasy
- Diagram obiektów
 - Diagram komponentów
 - Diagram wdrożenia
 - Diagram struktur połączonych
 - Diagram pakietów
 - Diagram profili (od UML 2.2)

Diagramy zachowań (dynamiczne):

- Diagram czynności
- Diagram przypadków użycia
- Diagram maszyny stanowej
- Diagram interakcji
- Diagram sekwencji
- Diagram komunikacji
- Diagram harmonogramowania
- Diagram sterowania interakcją

Klasa – uogólnienie zbioru obiektów, które mają takie same atrybuty, operacje, związki i znaczenie. Każda klasa zawiera więc zestaw informacji istotnych z punktu widzenia kontekstu systemu. Zestaw atrybutów, operacji i związków z innymi klasami może być szerszy lub węższy w zależności od wymagań dotyczących przyszłego systemu.

Klasę przedstawia się jako prostokąt złożony z trzech sekcji:

| | |
|------------------|--|
| Classname | Nazwa klasy |
| | Zestaw atrybutów (informacje o własnościach i cechach klasy) |
| | Zestaw operacji (proces, którego sposób wykonania jest znany klasie) |

Dla każdego atrybutu i operacji określa się ich widoczność:

- + publiczne (np. + name: String)
- - prywatne (np. - age: int)
- # chronione (np. # weight: float)

Dla związku powiązania rozróżnia się dalej:

| | |
|--|--|
| Nazwa | <pre> graph LR Osoba -- "Pracuje dla" --> Przedsiębiorstwo </pre> |
| Nawigacja | <pre> graph LR Osoba -- "Pracownik" --> Przedsiębiorstwo </pre> |
| Role | <pre> graph LR Osoba -- "pracownik" --- "pracodawca" --- Przedsiębiorstwo </pre> |
| Liczebność | <pre> graph LR Osoba -- "1..*" --- "*" --- Przedsiębiorstwo </pre> |
| Kwalifikacja – wyszukiwanie obiektów związanych z daną asocjacją | <pre> graph LR Ramka -- "ID: Integer" --- "*" --- "0..1" --- Okno </pre> |
| Agregacja | <pre> graph LR Dział -- "*" --- "1" --- Przedsiębiorstwo </pre> |
| Agregacja całkowita (kompozycja) | <pre> graph LR Ramka -- "*" --- "1" --- Okno </pre> |

Interfejs – zestaw operacji, wyznaczony przez usługi oferowane przez klasę lub komponent. Umożliwia korzystanie z usług poszczególnych komponentów. Pośredniczy pomiędzy komponentami.

Rodzaje Interfejsów:


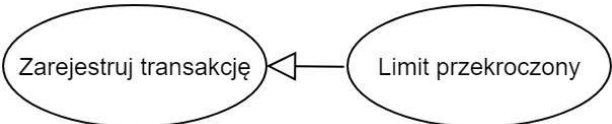
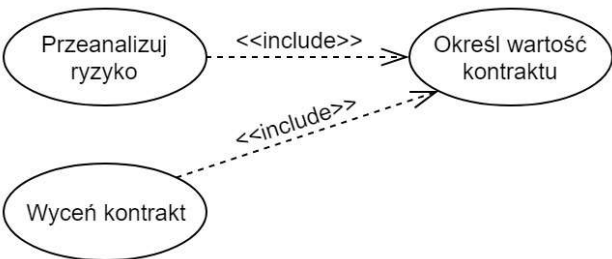

- **Importowany** – interfejs, z którego klasa lub komponent korzysta
- **Eksportowany** – realizowany przez klasę lub komponent (klasa lub komponent udostępnia usługi innym obiektom przez taki interfejs)

Przypadek użycia – opis zbioru ciągów akcji wykonywanych, które system może wykonać przez interakcję z aktorami systemu.

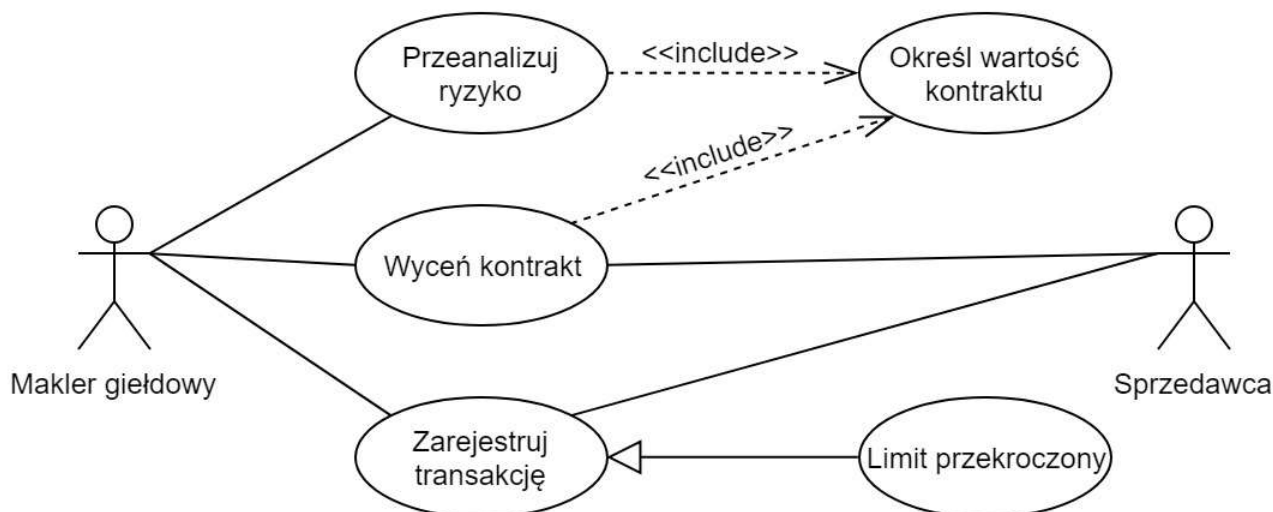
Diagram przypadków użycia – graficzne przedstawienie aktorów, oraz związków między nimi, występujących w danej dziedzinie przedmiotowej. Służy do modelowania funkcjonalności systemu. Przedstawia usługi, które system świadczy aktorom, lecz bez wskazywania konkretnych rozwiązań technicznych. Składa się z:

- Aktorów – nie jest częścią systemu. Wykonuje PU. Dzieli się na:
 - Człowiek/zespół (zwykły ludzik)
 - System zewnętrzny (ludzik z kwadratową główką)
 - Urządzenie (ludzik z główką w kształcie koła zębatego)
 - Czas (ludzik z zegarem zamiast główki)
 Relacje między aktorami to relacje uogólnienia, a między aktorami a PU relacje powiązania.
- Przypadków użycia
- Związków

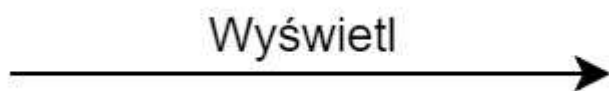
Relacje występujące w diagramie przypadków użycia:

| | |
|--|--|
| Asocjacje – związek pomiędzy dwoma lub więcej klasyfikatorami, opisującymi powiązanie pomiędzy ich instancjami. Między aktorem a przypadkiem użycia. |  |
| Uogólnienie – dziedziczenie cech elementu ogólnego. |  |
| Zawieranie – związku zawierania używa się wówczas, gdy z kilku innych przypadków użycia można wydzielić pewną część wspólną. Stereotyp <<include>>. |  |
| Rozszerzanie – związek ten pozwala na wydzielenie przypadku użycia, który w pewnych sytuacjach może zostać wzbogacony o dodatkowe opcje. Stereotyp <<extend>>. |  |

Przykładowy diagram przypadków użycia:



Interakcja – zachowanie polegające na wymianie komunikatów w grupie obiektów w pewnym celu.



Rodzaje diagramów interakcji:

- **Diagram sekwencji** – opisuje interakcję pomiędzy instancjami klasyfikatorów systemu w postaci sekwencji komunikatów wymienianych między nimi. Składa się z elementów takich jak klasyfikator, komunikat, linia życia, ośrodek sterowania. Można wyróżnić rodzaje:
 - **Konceptualny** – podstawowe kategorie pojęciowe i graficzne. Ogólny zakres i łatwe do zidentyfikowania interakcje.
 - **Implementacyjny** – podstawa opracowania specyfikacji programistycznej. Wszystkie kategorie pojęciowe.
 - **Wystąpieniowy** – wystąpienie diagramu sekwencji w odniesieniu do ustalonego scenariusza. Jednemu implementacyjnemu diagramowi sekwencji może odpowiadać kilka wystąpień.
- **Diagram komunikacji** – specyfikuje strukturalne związki pomiędzy instancjami oraz wymianę komunikatów pomiędzy tymi instancjami. Składa się z elementów takich jak klasyfikator, komunikat, asocjacja.
- **Diagram harmonogramowania** – reprezentuje na osi czasu zmiany dopuszczalnych stanów, jakie może przyjmować instancja klasyfikatora uczestnicząca w interakcji.
- **Diagram sterowania interakcją** – dokumentuje przepływ sterowania pomiędzy logicznie powiązanymi diagramami i fragmentami interakcji z wykorzystaniem kategorii modelowania diagramów czynności.

Diagram czynności – przedstawia przepływ sterowania od czynności do czynności. Opisuje, jak są uszeregowane działania dając możliwość opisu czynności warunkowanych lub współbieżnych.

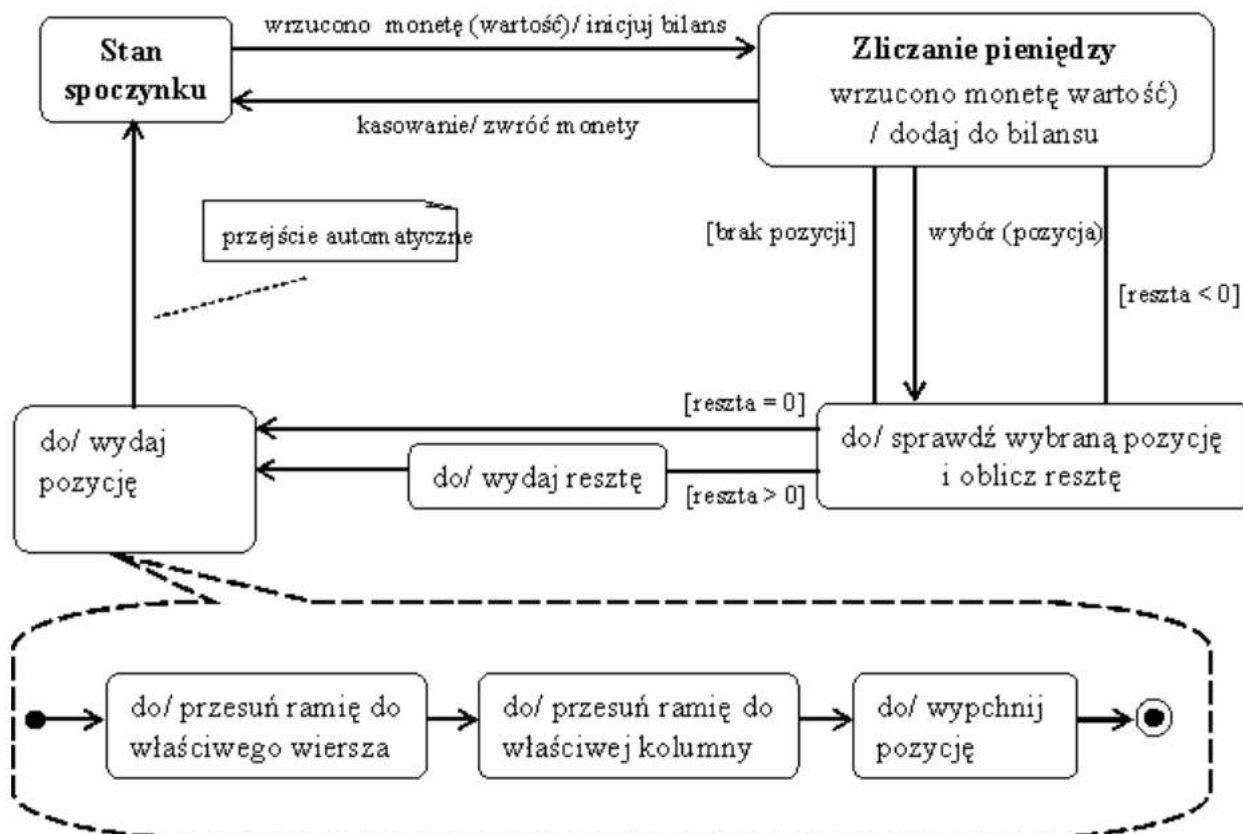
Czynność – podzielna, ogólna, dozwolona dekompozycja, znaczący czas realizacji.

Akcja – niepodzielna, szczegółowy przypadek, niedozwolona dekompozycja, nieznaczący czas realizacji. Jej symbolem, podobnie jak czynności jest zaokrąglony prostokąt, u akcji nieco mniejszy.

Tory – służą do podzieleni stanów czynności na grupy, z których każda reprezentuje jednostkę przedsiębiorstwa odpowiedzialna za przydzielone czynności.

Diagram stanów – pokazuje przede wszystkim możliwe stany obiektu oraz przejścia, które powodują zmianę tego stanu. Tym samym tworzony jest cykl życia obiektu, który może być tym istotniejszy w procesie wytwarzania oprogramowania, im więcej jest możliwych stanów obiektu.

Przykład:



Typy zdarzeń mogące występować w diagramie stanów:

- **Asynchroniczne (zewnętrzne)** – sygnał, upływ czasu, zmiana czasu.
- **Synchroniczne (wewnętrzne)** – wywołanie operacji

Sygnał – reprezentuje nazwany obiekt, który jest asynchronicznie wysłany przez jeden obiekt, a odbierany przez inny. Sygnał może być wysłany:

- W wyniku przejścia w maszynie stanowej
- W wyniku przesłania komunikatu interakcji (wywołania)
- W wyniku wykonania operacji

Upływ czasu – reprezentowane przez zdarzenie czasowe

Wywołanie – zdarzenie synchroniczne, tzn. gdy obiekt wywołuje operację innego sterowanie jest przekazywane od nadawcy do odbiorcy, a po wykonaniu tej operacji sterowanie wraca do nadawcy.

Stan – sytuacja w jakiej znajduje się obiekt, przez skończony okres swojego życia, kiedy spełnia jakiś warunek, wykonuje jakąś czynność lub czeka na jakieś zdarzenie.

Maszyna stanowa – opisuje ciąg stanów przyjmowanych przez obiekt w odpowiedzi na zdarzenia zachodzące w czasie jego życia, a także reakcje obiektu na te zdarzenia. Służy do modelowania zachowania jednego obiektu. Użycie maszyny stanowej to najlepszy sposób opisu zachowania obiektów, które muszą reagować na bodźce asynchroniczne i wpływ czasu.

Diagram komponentów – pokazuje zależności pomiędzy komponentami oprogramowania, włączając komponenty kodu źródłowego, kodu binarnego oraz kodu wykonywalnego. Poszczególne komponenty mogą istnieć w różnym czasie: niektóre z nich w czasie kompilacji, niektóre w czasie konsolidacji (*linking*) zaś niektóre w czasie wykonania. Diagram komponentów jest przedstawiany jako graf, gdzie węzłami są komponenty, zaś strzałki (z przerywaną linią) prowadzą od klienta pewnej informacji do jej dostawcy. Rodzaj zależności jest zależny od typu języka programowania. Diagram może także pokazywać interfejsy poszczególnych komponentów. Strzałki oznaczające zależności mogą prowadzić od komponentu do interfejsu. W skrócie diagram komponentów obrazuje uporządkowanie komponentów i związki między nimi. Zawiera:

- komponenty
- interfejsy
- uogólnienia, powiązania i realizacje
- notatki, ograniczenia, pakiety, podsystemy

Diagram wdrożeń – obrazuje konfigurację węzłów działających w czasie wykonania i zainstalowania na nich komponentów. Zawiera:

- Nodes (węzły)
- Zależności i powiązania
- Ścieżki komunikacji
- Osadzone komponenty i artefakty
- Manifestowanie
- Specyfikację rozlokowania
- Notatki, ograniczenia, pakiety, podsystemy

W UML występują diagramy:

- Klas
- Obiektów
- Przypadków użycia
- Pakietów
- Sekwencji
- Kombinacji
- Maszyny stanowej
- Czynności
- Komponentów
- Wdrożenia
- Struktur połączonych
- Przebiegów czasowych
- Przeglądu interakcji

Wymagania:

- **Funkcjonalne** – opisują funkcje (czynności, operacje) wykonywane przez system
- **Niefunkcjonalne** – opisują ograniczenia przy zachowaniu których system powinien realizować swoje funkcje

Diagram Kooperacji (Komunikacji) – opisuje organizację strukturalną obiektów, wymieniających komunikaty. Przedstawia sposób wymiany informacji pomiędzy obiektami (aktorami, klasami), które wchodzi ze sobą w interakcję. Określa związki strukturalne pomiędzy obiektami biorącymi udział w interakcji zgodnie ze scenariuszem przypadków użycia. Zawiera:

- Obiekty – aktorzy, moduły, klasy biorące udział w interakcji
- Asocjacje – określają strukturę organizacji obiektów, reprezentowane przez linie łączące obiekty
- Komunikaty – realizacja interakcji, opisywanie etykietowanymi strzałkami

Port – wyróżnialny punkt związany z interfejsami, przez które komponent komunikuje się z otoczeniem.

Partycja – mechanizm grupowania elementów diagramu czynności powiązanych przepływami sterowania i przepływami danych, pełniących określoną, wspólną rolę.