Uniwersytet Gdański Wydział Matematyki, Fizyki i Informatyki Instytut Informatyki

Oliver Gruba, Maciej Nasiadka

16 grudnia 2025

Imię i Nazwisko (nr indeksu)	Oliver Gruba (292583) Maciej Nasiadka (292574)
Nazwa uczelni	Uniwersytet Gdański
Kierunek	Informatyka (profil praktyczny)
Prowadzący	dr inż. Stanisław Witkowski
Nazwa ćwiczenia	Diagramy komponentów i wdrożenia w procesie projektowym
Numer sprawozdania	8
Data zajęć	11.12.2025
Data oddania	17.12.2025
Miejsce na ocenę	

Spis treści

1. Wprowadzenie do modelowania architektury systemów informatycznych

Modelowanie architektury systemu odgrywa kluczową rolę w procesie projektowym, ponieważ umożliwia przejrzyste przedstawienie struktury aplikacji, zależności między jej elementami oraz sposobu, w jaki system zostanie umieszczony w środowisku sprzętowo–sieciowym. W tym celu wykorzystuje się dwa komplementarne diagramy: **diagram komponentów** oraz **diagram wdrożenia**.

Diagramy te stanowią istotny element dokumentacji technicznej, wspierają komunikację pomiędzy zespołami projektowymi oraz ułatwiają analizę możliwości rozwoju i skalowalności systemu.

2. Diagram komponentów

2.1. Charakterystyka i przeznaczenie

Diagram komponentów w notacji UML przedstawia logiczną strukturę systemu poprzez identyfikację jego głównych modułów oraz relacji między nimi. Komponent definiowany jest jako niezależna jednostka o określonym interfejsie, która może być rozwijana, testowana i wdrażana autonomicznie.

Celem diagramu komponentów jest:

- wyodrębnienie części systemu odpowiedzialnych za konkretne funkcjonalności,
- zdefiniowanie interfejsów udostępnianych na zewnątrz,
- ukazanie zależności logicznych między modułami,
- wsparcie projektowania architektury warstwowej.

2.2. Poziomy szczegółowości diagramu komponentów

Diagram komponentów może być tworzony na różnych poziomach abstrakcji, w zależności od etapu projektu oraz potrzeb odbiorców dokumentacji.

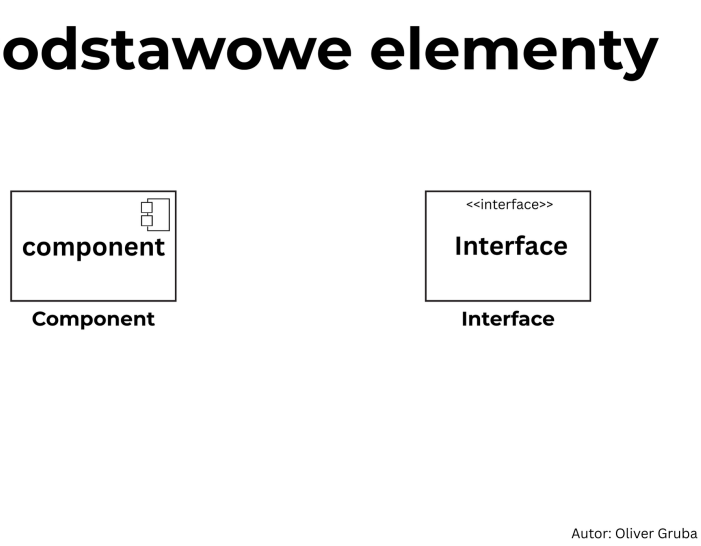
- **Poziom koncepcyjny** – przedstawia główne moduły systemu bez wchodzenia w szczegóły implementacyjne; stosowany na wczesnym etapie analizy.
- **Poziom logiczny** – uwzględnia interfejsy, zależności oraz podział na warstwy architektury.

- **Poziom techniczny** – zawiera konkretne technologie, biblioteki oraz artefakty programistyczne.

Dobór odpowiedniego poziomu szczegółowości wpływa na czytelność diagramu oraz jego użyteczność w procesie decyzyjnym.

2.3. Podstawowe elementy diagramu komponentów

2.3.1 Komponenty



Rysunek 1: Graficzne przedstawienie podstawowych elementów diagramu komponentów

Są to fizyczne lub logiczne moduły systemu, oznaczane prostokątem z symbolem ‘«component»‘. Mogą reprezentować:

- moduły aplikacji (np. moduł raportowania),
- biblioteki,
- usługi (API),
- warstwy logiki biznesowej.

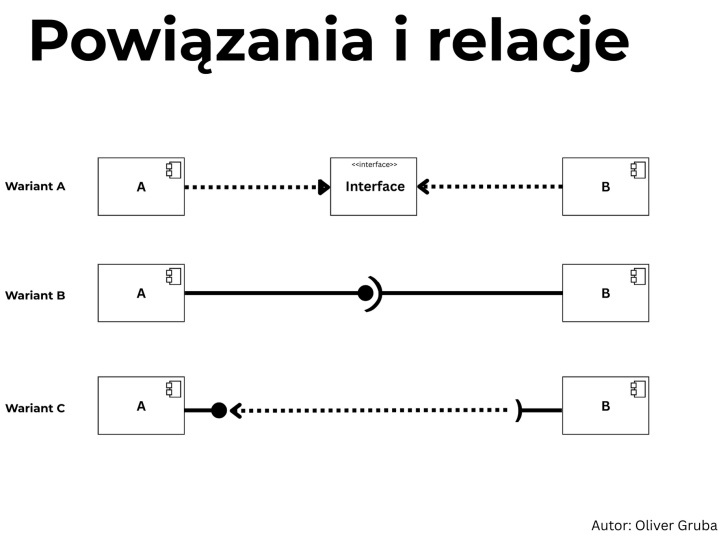
2.3.2 Interfejsy

Interfejsy definiują punkty komunikacji między komponentami. Dzielą się na:

- **interfejsy dostarczane (provided)** – funkcje, które komponent udostępnia,
- **interfejsy wymagane (required)** – funkcje niezbędne komponentowi do działania.

2.3.3 Zależności

Zależności przedstawiają kierunek, w którym komponent korzysta z interfejsu innego komponentu.



Rysunek 2: Graficzne przedstawienie powiązania i relacji diagramu komponentów

2.3.4 Pakiety i grupowanie komponentów

W celu zwiększenia czytelności diagramu komponentów stosuje się pakiety (packages), które pozwalają logicznie grupować powiązane elementy systemu.

Pakiety umożliwiają:

- odzwierciedlenie architektury warstwowej (np. prezentacja, logika biznesowa, dane),
- ograniczenie liczby widocznych zależności,
- lepszą organizację dużych i złożonych systemów.

Grupowanie komponentów sprzyja również podziałowi pracy pomiędzy zespoły projektowe.

2.3.5 Przepływ danych

Rysunek 3: Graficzne przedstawienie przepływu danych w diagramie komponentów

2.4. Diagram komponentów a architektura systemu

Diagram komponentów jest bezpośrednio powiązany z wybraną architekturą systemu, taką jak:

- architektura warstwowa,
- architektura klient–serwer,
- architektura mikroserwisowa,
- architektura oparta na usługach (SOA).

Poprawnie zaprojektowany diagram komponentów pozwala ocenić stopień luźnego powiązania (low coupling) oraz spójności modułów (high cohesion), co ma kluczowe znaczenie dla utrzymania i rozwoju systemu.

2.5. Znaczenie diagramu komponentów w projektowaniu

Diagram komponentów wspiera projektowanie architektury modularnej oraz pozwala na:

- planowanie zespołowej pracy nad modułami,
- ograniczanie ryzyka integracji,
- przygotowanie systemu pod mikroserwisy lub architekturę rozproszoną,
- oszacowanie punktów krytycznych systemu.

3. Diagram wdrożenia

3.1. Rola diagramu wdrożenia

Diagram wdrożenia (Deployment Diagram) ukazuje fizyczne rozmieszczenie elementów systemu w środowisku sprzętowym. Przedstawia sposób uruchamiania komponentów na serwerach, urządzeniach sieciowych lub maszynach wirtualnych. Wskazuje również połączenia komunikacyjne oraz zależności infrastrukturalne.

Diagram ten jest kluczowy w projektach wymagających:

- skalowalności,
- rozproszenia usług,
- integracji systemów,
- specyficznych wymogów wydajnościowych.

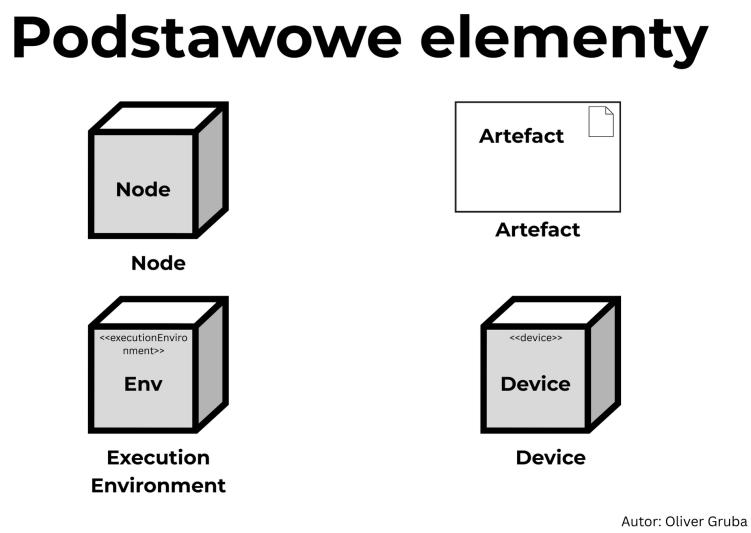
3.2. Diagram wdrożenia w cyklu życia systemu

Diagram wdrożenia znajduje zastosowanie na wielu etapach cyklu życia systemu informatycznego:

- podczas projektowania infrastruktury,
- w fazie wdrożeniowej i testowej,
- przy planowaniu skalowania systemu,
- w procesie utrzymania i monitorowania.

Stanowi on istotne źródło informacji dla administratorów systemów oraz zespołów odpowiedzialnych za utrzymanie środowisk produkcyjnych.

3.3. Elementy diagramu wdrożenia



Rysunek 4: Graficzne przedstawienie podstawowych elementów diagramu wdrożenia

3.3.1 Węzły (Nodes)

Węzły reprezentują fizyczne lub wirtualne zasoby obliczeniowe, np.:

- serwer aplikacyjny,
- urządzenie mobilne,
- baza danych,
- chmura obliczeniowa.

3.3.2 Środowiska wdrożeniowe

Diagram wdrożenia może uwzględniać różne środowiska, w których funkcjonuje system:

- środowisko deweloperskie (DEV),
- środowisko testowe (TEST),
- środowisko produkcyjne (PROD).

Rozróżnienie środowisk pozwala na analizę różnic infrastrukturalnych oraz minimalizację ryzyka błędów podczas wdrażania nowych wersji systemu.

3.3.3 Artefakty

Artefakty oznaczają rzeczywiste pliki wdrożeniowe lub procesy uruchomieniowe, takie jak:

- pliki JAR, WAR,
- kontenery Docker,
- pliki konfiguracyjne.

3.3.4 Połączenia komunikacyjne

Określają sposób komunikacji między węzłami – mogą reprezentować protokoły, porty, szyfrowanie.

3.4. Znaczenie diagramu wdrożenia w analizie architektury

Diagram wdrożenia umożliwia:

- identyfikację potencjalnych wąskich gardeł,
- planowanie pojemności infrastruktury,
- przygotowanie systemu pod dostępność 24/7,
- analizę bezpieczeństwa i redundancji.

4. Powiązanie diagramów komponentów i diagramów wdrożenia

Diagramy te są ze sobą ściśle powiązane:

- diagram komponentów określa **co** tworzy system,
- diagram wdrożenia określa **gdzie** te elementy są uruchamiane.

W procesie projektowym tworzy się najpierw strukturę logiczną, a następnie odwzorowuje ją w środowisku produkcyjnym.

Zależności te wspierają:

- planowanie migracji do chmury,
- analizę wydajności,
- poprawne rozmieszczenie zależności sieciowych,
- rozdzielenie odpowiedzialności pomiędzy zespoły DevOps i programistyczne.

4.1. Spójność logiczna i fizyczna architektury

Zachowanie spójności pomiędzy diagramem komponentów a diagramem wdrożenia jest kluczowe dla poprawnej realizacji projektu.

Każdy komponent logiczny powinien mieć swoje jednoznaczne odwzorowanie w diagramie wdrożenia, co umożliwia:

- kontrolę kompletności architektury,
- identyfikację brakujących elementów infrastruktury,
- analizę wpływu zmian logicznych na środowisko fizyczne.

5. Przykładowe diagramy

5.1. Przykładowy diagram komponentów

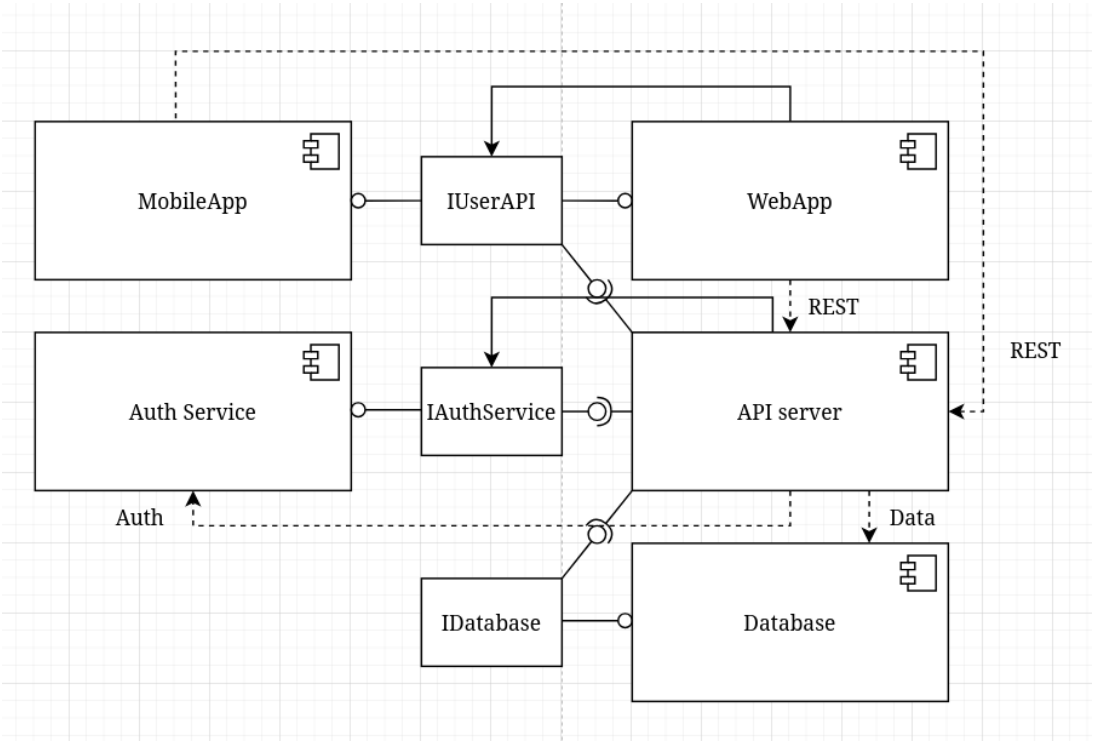


Diagram komponentów przedstawia logiczną strukturę systemu, pokazując główne moduły (komponenty), ich interfejsy oraz zależności pomiędzy nimi. Ułatwia zrozumienie podziału odpowiedzialności oraz komunikacji między częściami systemu.

- **MobileApp** i **WebApp** – dwa różne frontendy korzystające z tych samych interfejsów API.
- **IUserAPI**, **IAuthService**, **IDatabase** – interfejsy definiujące punkty komunikacji pomiędzy komponentami.
- **Auth Service** – odpowiada za uwierzytelnianie, komunikuje się z API server przez interfejs IAuthService.
- **API server** – centralny komponent obsługujący logikę biznesową, komunikuje się z bazą danych (IDatabase) oraz obsługuje żądania REST od aplikacji frontendowych.
- **Database** – przechowuje dane, dostępna przez interfejs IDatabase.
- Połączenia REST i Auth są wyraźnie oznaczone, co ułatwia analizę przepływu danych i bezpieczeństwa.

5.2. Przykładowy diagram wdrożenia

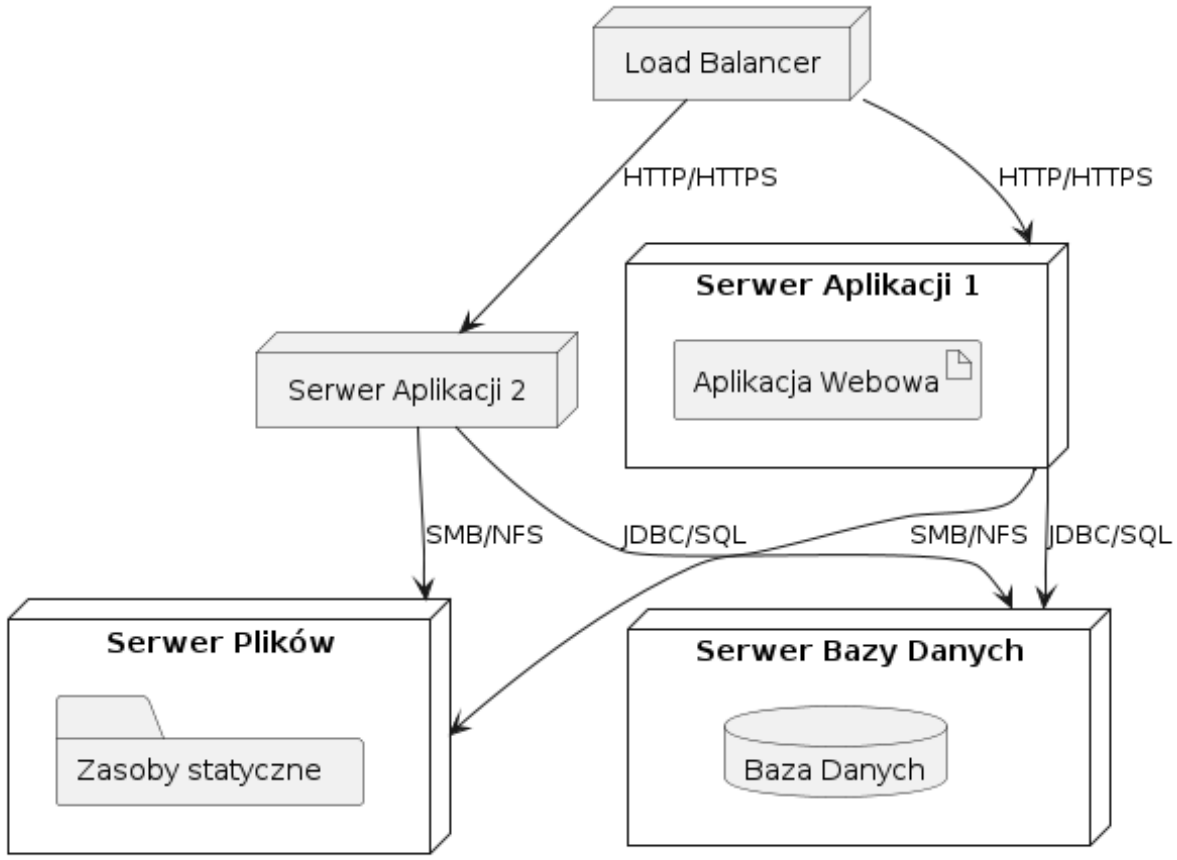


Diagram wdrożenia przedstawia fizyczną architekturę systemu informatycznego, pokazując rozmieszczenie komponentów na serwerach oraz sposób komunikacji między nimi. Umożliwia zrozumienie, jak poszczególne elementy systemu są rozmieszczone w infrastrukturze oraz jakie protokoły i połączenia są wykorzystywane.

- **Load Balancer** – rozdziela ruch HTTP/HTTPS do dwóch serwerów aplikacji, zapewniając równoważenie obciążenia i wysoką dostępność.
- **Serwer Aplikacji 1 i 2** – obsługują aplikację webową, komunikują się z serwerem plików (protokół SMB/NFS) oraz z serwerem bazy danych (JDBC/SQL).
- **Serwer Plików** – przechowuje zasoby statyczne, dostępny dla obu serwerów aplikacji.
- **Serwer Bazy Danych** – przechowuje dane aplikacji, dostępny dla obu serwerów aplikacji.
- Komunikacja między serwerami jest jasno oznaczona (HTTP/HTTPS, SMB/NFS, JDBC/SQL), co ułatwia analizę bezpieczeństwa i wydajności.

6. Znaczenie diagramów w procesie projektowym

Diagramy komponentów i wdrożenia dostarczają architektom, projektantom i zespołom deweloperskim kluczowych informacji o budowie systemu. Ich wykorzystanie prowadzi do:

6.1. Poprawy jakości projektu

- umożliwiają wykrycie błędów strukturalnych na wczesnym etapie,
- pozwalają na spójną organizację warstw systemu,
- sprzyjają modularności i testowalności kodu.

6.2. Wsparcia integracji i wdrożeń

- ułatwiają komunikację między zespołami,
- dostarczają pełnego obrazu zależności logistycznych i sprzętowych,
- wspierają proces przygotowania środowisk (DEV, TEST, PROD).

6.3. Zwiększenia przejrzystości

- umożliwiają analizę skalowalności,
- przedstawiają obszary podatne na awarie,
- pozwalają na planowanie zasobów infrastrukturalnych.

7. Najczęstsze błędy w projektowaniu diagramów

Podczas tworzenia diagramów komponentów i wdrożenia często pojawiają się błędy, takie jak:

- zbyt niski lub zbyt wysoki poziom szczegółowości,
- mieszanie aspektów logicznych i fizycznych w jednym diagramie,
- brak jednoznacznych interfejsów,
- pomijanie połączeń komunikacyjnych i zależności infrastrukturalnych.

Świadomość tych problemów pozwala tworzyć bardziej czytelną i użyteczną dokumentację architektoniczną.

8. Wnioski

Analiza diagramów komponentów oraz diagramów wdrożenia pokazuje, że oba te narzędzia pełnią komplementarne role w procesie projektowania architektury systemów informatycznych. Ich wspólne wykorzystanie umożliwia uzyskanie spójnego i pełnego obrazu struktury systemu – zarówno na poziomie logiki, jak i fizycznego rozmieszczenia elementów. Poniższe subsekcje przedstawiają najważniejsze konkluzje wynikające z przeprowadzonego opracowania.

8.1. Wnioski dotyczące diagramu komponentów

Diagram komponentów stanowi fundament projektowania architektury logicznej systemu. Na podstawie analizy można wskazać następujące kluczowe wnioski:

- Diagram ten umożliwia jasne wyodrębnienie modułów systemu oraz ich odpowiedzialności, co zwiększa spójność projektową.
- Interfejsy (provided/required) stanowią przejrzysty sposób definiowania komunikacji pomiędzy modułami, ograniczając ryzyko niejednoznaczności w implementacji.
- Reprezentacja zależności pozwala na wczesne wykrycie potencjalnych problemów integracyjnych oraz miejsc, które mogą stać się wąskimi gardłami.
- Diagram komponentów wspiera podejście warstwowe, ułatwiając rozdzielanie odpowiedzialności i zwiększając modularność systemu.
- Przedstawiona struktura logiczna jest kluczowa przy planowaniu przyszłego rozwoju systemu oraz przy przechodzeniu w kierunku architektury rozproszonej (w tym mikroserwisów).

8.2. Wnioski dotyczące diagramu wdrożenia

Diagram wdrożenia dostarcza wiedzy o faktycznym rozmieszczeniu systemu w środowisku sprzętowym i sieciowym. Najważniejsze obserwacje to:

- Diagram wdrożenia ułatwia ocenę zapotrzebowania na zasoby infrastrukturalne, takie jak moc obliczeniowa, pamięć czy przepustowość łączy.
- Graficzna reprezentacja węzłów i artefaktów pozwala precyzyjnie określić, gdzie działają poszczególne komponenty systemu.
- Diagram ten jest niezbędny przy analizie skalowalności, bezpieczeństwa, redundancji oraz wysokiej dostępności systemu.

- Jasne przedstawienie połączeń komunikacyjnych umożliwia identyfikację potencjalnych miejsc awarii oraz planowanie mechanizmów ochronnych.
- Dokumentacja wdrożeniowa wspiera zespoły DevOps w konfiguracji środowisk oraz automatyzacji procesów CI/CD.

8.3. Wnioski dotyczące wzajemnych powiązań diagramów

Połączenie diagramu komponentów i diagramu wdrożenia pozwala na stworzenie pełnej, wielopoziomowej dokumentacji architektury systemu. Z tej relacji wynikają następujące wnioski:

- Diagram komponentów określa funkcjonalne elementy systemu, natomiast diagram wdrożenia pokazuje ich fizyczne umiejscowienie – oba są niezbędne do zrozumienia całości architektury.
- Powiązanie tych diagramów ułatwia planowanie migracji systemu do chmury lub do środowiska rozproszonego.
- Wspólna analiza komponentów i wdrożenia pozwala zidentyfikować zależności sieciowe, opóźnienia komunikacyjne oraz wymagania dotyczące integracji.
- Dzięki połączeniu obu modeli możliwe jest lepsze rozdzielenie obowiązków pomiędzy zespoły projektowe i operacyjne.
- Spójna dokumentacja architektury przyczynia się do poprawy jakości całego projektu oraz ogranicza ryzyko błędów na etapie implementacji i wdrożeń.

8.4. Podsumowanie ogólne

Zastosowanie diagramów komponentów i wdrożenia stanowi ważny element profesjonalnego projektowania systemów informatycznych. Pozwala na:

- zrozumienie konstrukcji systemu na poziomie logicznym i fizycznym,
- usprawnienie komunikacji między zespołami technicznymi,
- wczesne wykrycie problemów oraz racjonalne planowanie rozwoju,
- przygotowanie architektury na przyszłą skalowalność i automatyzację wdrożeń.

Wspólne wykorzystanie obu diagramów tworzy solidną podstawę dla dalszych etapów projektowania, implementacji i utrzymania systemu.