UML

Spis treści

Czym jest UML? . . . 1
Po co znać UML? . . . 1
Jak głęboko powinniśmy ten język znać? . . . 2
Z jakich narzędzi korzystać w praktyce? . . . 2
Omówienie poszczególnych elementów . . . 2

Czym jest UML?

UML jest skrótem od **Unified Modeling Language**. Language nie oznacza tutaj, że jest to język tak jak np. Java. Język oznacza pewien rodzaj standaryzacji.

UML jest stosowany do wizualizacji elementów systemów informatycznych w postaci diagramów. Czyli można powiedzieć, że **UML** jest językiem obrazkowym, który służy do narysowania komponentów oprogramowania.

Celem wprowadzenia języka takiego jak **UML** było wprowadzenie standardowej metodologii wizualizacji oprogramowania w postaci diagramów. **UML** jest niezależny od języka programowania.

Po co znać UML?

Oczywiście **UML** nie służy tylko programistom, ludzie "biznesowi" również mogą z niego korzystać. (Z Javy w sumie też). Wspominam o tym dlatego, że jeżeli chcemy wytłumaczyć komuś jakieś koncepcje dotyczące napisanego kodu, to łatwiej jest to zrobić opierając się o taki diagram niż bez niego. Szczególnie jak ktoś jest wzrokowcem.

W realnym projekcie, zanim napiszemy kod, **UML** może służyć jako bardzo przydatne narzędzie do komunikacji między programistami w zespole, aby zwizualizować koncepcję rozwiązania problemu. Często rysując jest o wiele łatwiej porozumieć się i zakomunikować swoją wizję. Rysunki takie mogą przedstawiać górnolotną wizję architektury systemu, zależności pomiędzy klasami, kolejność wywołań metod lub kolejność wywołań systemów. Do przedstawienia takich obrazków często wykorzystywany jest właśnie **UML**.

Ze wspomnianych wcześniej powodów, warto jest znać popularny język obrazkowy służący do narysowania elementów oprogramowania. Głównie wykorzystamy go do komunikacji z innymi. Natomiast sam **UML** jest ogromny, jeżeli ktoś ma ochotę to zapraszam do przeczytania specyfikacji 😄.

UML może pojawiać się albo w dokumentacji (ale jeżeli to Ty będziesz coś takiego rysować to i tak będziesz szperać w internecie żeby upewnić się, że dobrze rysujesz) albo może być używany nieformalnie przy rozmowie z innymi (a wtedy nikt jakoś mocno nie trzyma się tej specyfikacji, bo przecież chodzi tylko o to żeby się zrozumieć). Dlatego ważne jest aby znać podstawy tych oznaczeń i umieć się z ich pomocą dogadać z innymi.

Jak głęboko powinniśmy ten język znać?

W tym materiale bardziej stawiamy na praktyczne zastosowanie tych wykresów niż na faktyczną zgodność ze specyfikacją. Ważniejsze jest to, żebyśmy rozumieli koncepcję niż pamiętali dokładną specyfikację każdego znaczka. W praktyce i tak wychodzi tak, że każda firma tworzy swój odłam tych oznaczeń, albo używa **UML** nie do końca zgodnie ze specyfikacją. Ja osobiście też nigdy nie spotkałem na swojej drodze nikogo, kto znałby ten standard w całości i bezbłędnie go stosował. Ważne jest to aby znać podstawy tego języka, a jak będziemy potrzebować czegoś bardziej skomplikowanego to i tak zostanie nam Googlować i szukać. Ważniejsze jest to aby przy wykorzystaniu **UML** umieć się porozumieć z innymi niż żeby znać dokładnie specyfikację na pamięć.

Pamiętaj też, że materiały, które są przedstawione w tym warsztacie są pokazywane z perspektywy programisty. Oznacza to, że mówiąc o tym jak głęboko powinniśmy ten język znać, cały czas mam w głowie perspektywę programisty, a nie innego członka zespołu IT. Oznacza to, że jeżeli mówilibyśmy o analityku biznesowym (w wielkim skrócie, jest to osoba, która może przygotowywać programistom wymagania, na podstawie których tworzą później oni aplikacje), to taki analityk może potrzebować głębszej znajomości **UML** niż developer. Pytanie *Jak głęboko powinniśmy ten język znać?* Podsumowałbym to stwierdzeniem "to zależy". W ramach tych materiałów chcę natomiast pokazać taki zakres tego języka, który wydaje mi się **must have** w codziennej pracy programisty/programistki.

Z jakich narzędzi korzystać w praktyce?

Ja osobiście jeżeli chcę coś komuś na szybko wytłumaczyć (i to w dodatku w erze pracy zdalnej) korzystam z darmowej wersji draw.io. Oczywiście nie jest to jedyne możliwe narzędzie, jest bardzo dużo innych, pokazuję tylko przykład 😉. Jeżeli natomiast chodzi o **draw.io** to możliwe jest zainstalowanie pluginu do Intellij i korzystanie z tego narzędzia w Intellij.

Drugie narzędzie, które często wykorzystuję w praktyce to **PlantUML**. Narzędzie to pozwala napisać kod (tak jak w Java), na podstawie któego następnie jest generowany diagram **UML** (przy wykorzystaniu tego narzędzia). Możliwe jest również zainstalowanie pluginu do Intellij, który pomoże nam pracować z tym narzędziem i generować diagramy na podstawie plików źródłowych **.puml**. Nie chcę natomiast wchodzić w temat omawiania tego narzędzia, bo takie diagramy można również rysować w **draw.io**. Zakładam, że jeżeli będziesz chciał/chciała skorzystać z **PlantUML** to poszukasz w internecie tutoriali. Nawet w oficjalnej dokumentacji PlantUML jest podanych bardzo dużo przykładów jak z tego narzędzia korzystać.

Omówienie poszczególnych elementów

Z powodu ilości oznaczeń dostępnych w **UML**, w poniższym opracowaniu pokażemy tylko najczęściej stosowane elementy podczas rysowania diagramów. Oznacza to, że nie pokazujemy wszystkich możliwych oznaczeń.

W dalszych materiałach zostaną przedstawione elementy konstrukcyjne oraz diagramy, które najczęściej można spotkać w praktyce. Nie będziemy pokazywać diagramów czysto akademickich.

Elementy jakie możemy znaleźć na diagramach **UML** mogą zostać podzielone na poszczególne grupy. Grupy, które wymienimy to:

- Things
- Relationships
- Diagrams

Każda z tych grup zostanie omówiona w kolejnych materiałach.