TOTYLKOKOD
MATEUSZ ZIELASKIEWICZ

Mateusz Zielaskiewicz

JAK DOGADAĆ SIĘ Z PROGRAMISTĄ?

Podręcznik dla odkrywcy świata IT

Spis treści

To tylko kod

Kod jest podobny do książki. Trzeba trzymać się pewnych reguł, używać odpowiednio znaków interpunkcyjnych i liter, które przekształcają się w słowa, zdania, a następnie całe opowieści pisane specyficznym językiem.

Książki zaczynasz czytać strona po stronie, od lewej do prawej, od góry do dołu. Mogą być one napisane różnymi językami oraz w różny sposób.

Początkujący programista zaczyna poznawać pomału zasady programowania oraz specyfikę danego języka/języków. Ucząc się na różnego rodzaju błędach (zazwyczaj popełnianych osobiście), zaczyna zdobywać cenne doświadczenie. Z czasem pisanie i czytanie idzie mu coraz lepiej, a po kodzie zaczyna się poruszać jak ryba w wodzie.

Żeby zrozumieć świat IT, musimy również zacząć od pewnych podstaw i poznać rzeczy, które dzieją się pomiędzy wystartowaniem projektu a jego realizacją (cykl życia oprogramowania). Zrozumienie ich pozwoli Ci poznać środowisko, w którym porusza się programista, a to na pewno ułatwi komunikację. Będziesz wiedzieć, z jakimi wyzwaniami/problemami mierzy się taka osoba każdego dnia, w jaki sposób możesz jej pomóc, a przede wszystkim jak jej nie przeszkadzać.

Zastanawiasz się pewnie, w jaki sposób można przeszkadzać programiście? Zebrałem całą masę odpowiedzi, co ich konkretnie wkurza i pod większością rzeczy również się podpisuję. Dowiesz się więcej na ten temat w rozdziale „Co wkurza programistów w PM-ach?" na stronie 70.

może podejmować wspólnie z zespołem – jednak to on ma ostatnie słowo w temacie).

Każdy z tych poziomów oznacza jakiś staż pracownika oraz posiadane doświadczenie. W każdym miejscu może to wyglądać inaczej – różnić się może ilość doświadczenia wymaganego na dany stopień. Jak natomiast wygląda to w praktyce? To zależy od:

- liczby problemów, z którymi miał okazję zmierzyć się programista,
- liczby awarii, z którymi miał do czynienia,
- ilości czasu spędzonego przy tworzeniu kodu,
- liczby projektów, przy których pracował,
- technologii, które poznał,
- codziennie wykonywanej pracy,
- miejsca pracy,
- pasji wkładanej w to, co robi,
- sposobu doszkalania się poza czasem pracy.

Można powiedzieć, że junior juniorowi nierówny, podobnie jak mid midowi oraz senior seniorowi. Co to znaczy?

Możemy spotkać się z sytuacją, w której ktoś, kto jest seniorem, pracował wcześniej tylko w jednej firmie i przy jednym projekcie, i robił w sumie ciągle to samo i nie miał żadnych wyzwań / styczności z nowymi technologiami. Taka osoba z racji stażu, przykładowo w korporacji, awansowała po 2 latach na mida, po 3 kolejnych na seniora i – łącznie po 5 latach – postanowiła zmienić pracę.

Wróćmy na chwilę do języków programowania

Można się posłużyć jeszcze inną analogią odnośnie do różnych języków programowania, a mianowicie: wyobrazić je sobie jako książkę. Każda książka składa się z okładki, ma spis treści i zazwyczaj numerację stron.

To są tak jakby elementy stałe – zazwyczaj się powtarzają.

Kolejne elementy, które się różnią, to zazwyczaj typ książki, np.:

- beletrystyka,
- science-fiction,
- reportaż,
- psychologia.

Może również występować okładka twarda, miękka lub nawet e-book.

Język, w którym jest napisana, to również coś, co może się różnić w zależności od kraju i języka w nim stosowanego.

W zależności od tego, co chcemy osiągnąć daną książką:

- dowiedzieć się czegoś nowego,
- rozwiązać jakiś problem,
- po prostu się zrelaksować,

wybierzemy dla siebie odpowiedni typ książki.

Podobnie jest z językami programowania, które są dobierane według potrzeby – jaki problem chcemy rozwiązać / jaki cel osiągnąć.

Brak dokumentacji / opisania najistotniejszych funkcji projektu

Nie zawsze kod jest pisany w ładny sposób. Możesz to porównać do bardzo starej księgi, w której używane jest słownictwo z danej epoki i żeby zrozumieć, co autor chciał nam przekazać, musisz czytać to bardzo wolno, strona po stronie i w ogromnym skupieniu oraz z pomocą internetu – w poszukiwaniu nieużywanych już pojęć.

Podobne wyzwania spotykają programistów:

- gdy pracują na bieżąco w projekcie, w którym zrozumienie sposobu działania danej funkcjonalności nie jest takie proste; wymaga żmudnego szukania i czytania, co dany fragment robi i główkowania, jak jest zależny od innego fragmentu kodu;
- podczas awarii, gdy trzeba coś szybko naprawić, a znalezienie przyczyny nie jest takie proste; w dość krótkim czasie można problem załatać tzw. obejściem (zrobieniem czegoś w brzydki sposób, na skróty, żeby tylko działało), ale dobrą praktyką jest, w następnym kroku, naprawienie tego „systemowo", czyli tak, jak należy, oraz zadbanie, aby problem nie wrócił w przyszłości;
- gdy trzeba dorobić nową funkcjonalność do programu i zrozumieć zależności, które będą do użycia w kodzie.

problemu możemy dojść, gdy mamy zbyt dużo na głowie i zbyt wieloma rzeczami zajmujemy się równolegle – nie kończąc skutecznie żadnej z nich. Mamy wrażenie, że robimy dużo, ale nie jest to działanie efektywne a tym bardziej produktywne.

Jeżeli to możliwe, starajmy się tak planować rzeczy, żeby mieć jak najmniej zmian kontekstów w ciągu dnia oraz przerywników powodujących narzuty czasowe w powrotach do efektywności i toku myślowego sprzed przerwania.

Dlaczego to nie działa, przecież powinno?

Zdarza się tak, że często programiści zadają sobie pytanie:

- dlaczego to nie działa, przecież powinno?

Rzadziej zdarza się:

- jakim cudem to działa!?

Oba przypadki mogą bardzo często wystąpić, gdy:

- jesteśmy już tak bardzo zmęczeni materiałem, że nie widzimy miejsca, w którym popełniliśmy podstawowy błąd; bardzo często spojrzenie kolegi na nasz kod wystarczy, by szybko dojść do problematycznego miejsca;
- nie wiemy, w jaki sposób to działa i dlaczego jest to tak skomplikowane, przecież można było prościej;
- przyjdzie nam się zastanawiać, co autor wiedział, a czego ja nie wiem, że to tak powstało – czy była to niewiedza, czy może jakieś celowe działanie?

Jak może wyglądać typowy dzień/tydzień/miesiąc programisty?

Większość firm IT stara się być mocno Agile i wprowadzać Scruma. Jednym to się udaje lepiej, innym gorzej – jednak w praktyce część spotkań jest adaptowana i stosowana na co dzień – według tego jakie są potrzeby.

Nie będę tutaj wchodzić mocno w szczegóły, tematowi Agile i Scrum poświęcony jest rozdział Scrum.

Jak może wyglądać typowy dzień programisty?

Poniedziałek:

- sprawdzenie poczty i odpisanie na maile,
- sprawdzenie komunikatora i odpisanie na wiadomości,
- sprawdzenie narzędzia do zarządzania zadaniami – czy nie pojawiły się jakieś nowe zadania/awarie/komentarze,
- przygotowanie się do daily i zastanowienie, co robiło się w piątek i do czego dzisiaj wracamy,
- spotkanie daily,
- bieżąca praca nad zadaniami / konsultacje z zespołem / odpowiadanie na przerywniki osób spoza projektu lub PM-owi,

> Forward tasiemcy mail'owych oczekując wykonania jakiegoś zadania bez słowa wstępu i jasnego określenia co jest tym zadaniem.

RE: Czy można z takiej komunikacji wyciągnąć tylko właściwą część, która jest do zrobienia, i wrzucić to koledze do wykonania? Czy jest sens, aby programista przedzierał się przez te wszystkie wątki, zamiast tylko zapoznać się z podsumowaniem? W tego typu przypadkach jako PM-i zawsze możemy zrobić jeden dodatkowy krok, podsumować dotychczasowe ustalenie i poprosić o potwierdzenie klienta. Jeżeli uda nam się to zrobić, mamy klienta, który wie, co otrzyma oraz szczęśliwego programistę, który wie, co jest konkretnie do zrobienia.

Jeżeli również jest to możliwe, trzymajmy zadania w jednym systemie do zarządzania nimi, a nie na mailach (w zadaniu zawsze można zaznaczyć, którego maila z którego dnia to dotyczy).

> Zabiera Ciebie na spotkanie z Klientem mówiąc „spoko ja ogarnę, Ty mi w razie czego pomożesz z pytaniami", po czym rozpoczyna od przywitania i informacji „To jest Marcin, przedstawi Państwu założenia rozwiązania".

RE: Jeżeli czeka Was ważniejsza rozmowa z klientem, a zespół nie jest przyzwyczajony do takich spotkań, zorganizujcie sobie wcześniej zebranie wewnętrzne i porozmawiajcie, jak to będzie wyglądać. To jest właśnie miejsce, gdzie

PS To światełko w tunelu rozpędza się do 320 km/h, więc nie ma co oszukiwać siebie i innych.

> -o właśnie malowanie trawy na zielono nawet przed samym sobą

RE: Nazywajmy rzeczy po imieniu, mówmy, jak jest i starajmy się szukać sposobów na wprowadzenie usprawnień. Wszystko to małymi krokami, ale skutecznie.

> -Widze, ze zostalo ci roboty na jakis tydzien a deadline jest za dwa dni, wiec zabiore cie 80% z tych dwoch dni na bezsensowne konferencje na ktorych bedziesz sie musiala tlumaczyc dlaczego kiepsko zarzadzasz czasem.

RE: Umiejętność rozmowy z ludźmi, świadomość tego, co dzieje się w projekcie oraz umiejętność mówienia NIE – to takie minimum, aby rzeczy nie wybuchały. Nie bójmy się powiedzieć komuś, że zespół nie pojawi się na konferencji. Gdy jednak ktoś (zazwyczaj „góra") będzie nalegać, upewnijmy się, że jest świadomy konsekwencji oraz czy je akceptuje. Zazwyczaj okazuje się, że projekt jest jednak ważniejszy, a „góra" nie miała oglądu na całą sytuację – dlatego warto rozmawiać. Zawsze.

stwierdzić, że udało mi się „zaoszczędzić" godzinę, którą mogę lepiej spożytkować. Polecam zapoznać się lepiej z prawem Parkinsona i wypróbować je na własnym przykładzie.

2. WSZYSCY MAMY TAKĄ SAMĄ DOBĘ

Doba oferuje 24 godziny każdemu z nas. Czasami kiedy włączy mi się marudzenie pt. *nie mam czasu...*, mój telefon przypomina mi, że znowu gadam głupoty. Nie wiem, czy wam również, ale mój smartfon co tydzień przygotowuje podsumowanie tygodnia, w którym czarno na białym widać, ile czasu spędziłem na „czasopożeraczach". Oczywiście czasami trzeba pozwolić sobie i na taką rozrywkę, jednak może warto przemyśleć, czy chcemy dać sobie na nią 30 minut, czy 3 godziny?

Podobno jeśli czegoś się mocno chce, to szuka się sposobów, a jeśli się nie chce, to powodów. To chyba właśnie to podejście sprawia, że jedni sprawiają wrażenie bardziej produktywnych niż inni. Jeżeli nic nie stoi na przeszkodzie, warto poszukać takich kół napędowych, które pomogą nam realizować cele, zamiast szukać wymówek.

3. NADGODZINY NIE SĄ ROZWIĄZANIEM

Nadgodziny mogą się zdarzyć. Sytuacja awaryjna, wszystkie ręce na pokład – raz na jakiś czas ratują projekt. Jeśli jednak zaczynają być regularne, to znaczy, że dzieje się coś niedobrego i mamy problem do zidentyfikowania i rozwiązania.

Nadgodziny nie przekładają się na produktywność. Każdy z nas działa efektywnie w określonym czasie, nasz organizm ma swoje limity i należy ich przestrzegać. Dobrze jest obserwować samego siebie i wiedzieć, kiedy jest moment

Rozpisywanie zadań/funkcjonalności

Niektórzy lubią pracować, gdy mają wcześniej przygotowane zadania wraz z opisami, co jest do zrobienia. Takie podejście ułatwia również ściągnięcie do projektu dodatkowych osób, które nie muszą znać całej logiki biznesowej – mają do wykonania tylko mały fragment kodu zgodnie z wcześniej przygotowanym opisem. Nic więcej i mniej niż to, co się tam znajduje.

Sprawdza się to również w zespołach, w których lead projektu lub architekt rozbijają projekt na mniejsze części i poprzez zadania/funkcjonalności rozpisują cały zakres do dowiezienia.

Review/Demo

Spotkania z klientem, na których zespół pokazuje progres prac i przygotowane nowe funkcjonalności. Klient ma możliwość zadawania pytań oraz omawiania zarówno obecnych, jak i przyszłościowych funkcjonalności.

i dobrym rozmówcą. Tematy mogą być związane z pracą, ale nie jest to konieczność. Celem tego spotkania jest budowanie relacji, a taką najlepiej się buduje, kiedy jesteśmy dobrym słuchaczem.

Zrozum i zajmij się problemami, które ma Twój podopieczny

W tygodniu może zadziać się bardzo dużo rzeczy. Zarówno Twój rozmówca może do Ciebie z czymś konkretnym przyjść, jak i Ty możesz mieć sprawę do słuchacza. Pamiętaj, że problemy, które Tobie mogą wydać się błahe, dla Twojego podopiecznego mogą być czymś poważnym. Nigdy ich nie ignoruj i wracaj z odpowiedzią.

Jakie pytania możesz zadawać?

- Co u Ciebie słychać?
- Jak tam na projekcie?
- Czy od ostatniego spotkania wydarzyło się coś niepokojącego na projekcie lub u Ciebie?
- Czy jesteś zadowolony z bycia na tym projekcie?
- Jak Ci się pracuje?
- Czego nowego nauczyłeś się w tym tygodniu?

Jeżeli będziesz dobrym słuchaczem, możesz również wracać do tematów, które były poruszane poprzednio:

- czy udało się rozwiązać problem z tymi testami jednostkowymi?
- jak poszło spotkanie, którym tak się stresowaliście?