Hipsterskie metody wyboru technologii

Czy zdarzyło Ci się kiedyś pracować pod wpływem fascynacji jakąś technologią? Wracając z konferencji, czytając artykuł, oglądając wideo, biorąc udział w szkoleniu, słuchając rozmów znajomych z pracy – czy czułeś ekscytację nową technologią? Mnie zawsze przechodzą ciarki po plecach, kiedy siadam do nowego projektu lub nowego fragmentu kodu i chcę wykorzystać wszystko to, czego się dowiedziałem przez ostanie pół roku. Wiem, że jeżeli tak zrobię, jest spora szansa, że nie wszystko pójdzie tak dobrze jak to było obiecywane, a ja będę zastanawiał się, dlaczego tak się stało.

ZNAJ PRZESZŁOŚĆ, PATRZ W PRZYSZŁOŚĆ

Jaki był początek twojej kariery zawodowej? Jaki był pierwszy projekt, za który otrzymałeś wynagrodzenie? Jakie technologie były w nim użyte?

Może zaczynałeś swoją pracę od budowania stron w PHP? Może zaczynałeś od programowania w C? A może pamiętasz, jakby to było wczoraj, kiedy wyszła pierwsza wersja Springa? Spójrz na swoje życie zawodowe z innej perspektywy. Czy dostrzegasz całą historię swojego życia zawodowego? Czy widzisz, że świat wokół Ciebie się zmienił?

Swoją karierę zawodową zaczynałem w czasach, w których SOAP oraz SOA były na pierwszych stronach gazet. WSDLe pisało się z palca, a schematy XSD to był chleb powszedni. Wtedy byłem święcie przekonany, że ten sposób tworzenia usług sieciowych będzie wieczny. Że tak będziemy robili przez przynajmniej kilka dekad. Dzisiaj świat jest inny i nie stosuję już żadnej z wymienionych powyżej technologii, z których korzystałem kilka lat temu. Technologie i narzędzia przychodzą i odchodzą. Jedne są z nami krócej, inne są z nami dłużej. Pamiętam, jakby to było wczoraj, kiedy metodyki Agile były wynoszone na piedestał, jakby miały rozwiązać wszystkie problemy tego świata. Dziś otwarcie mówi się o tym, żeby programiści porzucili Agile[1].

Każda technologia, z której korzystamy, prędzej czy później zostanie zastąpiona przez nową. Tak powinno być. Kiedy się nad tym zastanawiam, jest to najskuteczniejsza metoda rozwoju. W czasach pradawnych czerpaliśmy światło ze spalanego drewna, następnie wynaleziono świece, elektryczność i żarówkę żarnikową, potem świetlówki, a teraz diody led. Tak samo jest z naszą branżą IT. Pytanie brzmi: co zrobić, żeby być zawsze na czasie i nie pozostać przy świecach, kiedy na około wszyscy korzystają z oświetlenia ledowego?

Jedną z metod bycia na czasie, z jaką się spotkałem, jest branie wszystkiego do projektu, co jest najnowsze, bez względu na konsekwencje. Niektórzy wybierają nawet wersje alpha nowych bibliotek i zależności. Inni starannie wybierają swój stos technologiczny i są zamknięci na dyskusję. Czasami jedziemy na konferencję i pod wpływem oczarowania nowymi rzeczami zmieniamy architekturę projektu. Jeżeli pracujemy w doświadczonym zespole, możemy dyskutować na temat zmian i tego, co one nam dadzą. Na podstawie wymiany doświadczeń możemy podjąć decyzję, co dalej.

„NEVER ASK A BARBER IF YOU NEED A HAIRCUT"

Ponieważ mój już prawie pełnoletni samochód był objęty jakąś akcją serwisową, odbyłem wizytę w salonie samochodowym. Salon ten sprzedawał zarówno nowe samochody, jak i używane. Nie ukrywałem przed sprzedawcą, że przymierzam się do zakupu samochodu, nowszego i większego, ze względu na powiększającą się rodzinę. Niestety moja 10-minutowa rozmowa ze sprzedawcą nie zakończyła się dobrze, ponieważ czułem się manipulowany. Przedstawiłem mu swoje wymagania, a on usilnie starał się mi sprzedać nowy i najdroższy samochód. Ja takiego nie chciałem. Chciałem zakupić 2-, 3-letnie auto, które nie stoi w salonie, tylko przed salonem, gdzie klient oddał go w rozliczeniu. Nasłuchałem się, jak to 3-letni samochód jest przestarzały, nie na czasie i tym podobnych, kiedy tak naprawdę chciałem coś zupełnie innego. To zdarzenie nauczyło mnie jednego: nieważne, z kim rozmawiasz, pamiętaj o tym, kto siedzi po drugiej stronie i jakie ma motywy. Sprzedawca chciał zarobić, nie interesowały go zupełnie moje potrzeby.

Jak to się ma do branży IT? Każdy z nas słucha prezentacji nowych rozwiązań, czyta blogi tematyczne, rozmawia z ludźmi przy okazji konferencji lub spotkania, albo dostaje maile informujące o nowych rozwiązaniach. Czy zdaję sobie sprawę, kto jest po drugiej stronie? Jakie ma motywy do tego, co robi? Na ile treści, które dostaję, są przypudrowane i ubrane w ładne słowa, a na ile to, co słyszę, rozwiązuje prawdziwe problemy, które mam lub mogę mieć w niedalekiej przyszłości.

My, programiści, software developerzy, możemy być całkowicie obiektywni, jeżeli tylko chcemy. Nie można zaprzeczyć, że mamy wrodzoną skłonność do inżynierii i wymyślania koła na nowo. Bardzo łatwo ulegamy trendom, modzie, sezonowemu nastawieniu na nowe rozwiązanie, tylko po to, żeby... za rok przepisać system od nowa, ponieważ będzie coś nowego. Sam też często zastanawiam się na kodem, który popełnię. Czy nie dałoby się prościej? Czy czasami nie przesadzam? A może jest biblioteka, która rozwiązuje ten problem?

1. https://ronjeffries.com/articles/018-01ff/abandon-1/

ENCJA NA TWARZ I PCHASZ

Mamy dużo szczęścia, że pracujemy w IT. Ja doceniam ten zawód i nie wiem, co innego mógłbym robić w życiu. Kiedy rozmawiam z innymi znajomymi z czasów szkolnych, większość z nich wykonuje codziennie te same, powtarzalne zadania, codziennie, dzień za dniem, przez cały rok.

Tymczasem my, inżynierowie oprogramowania, jak tylko pojawi się coś nieciekawego do zrobienia – narzekamy. Że trzeba rozwiązywać bugi, że jest kolejny JSON do wysłania, lub robimy systemy w architekturze, która nam się nie podoba – często nie podejmując żadnych działań, żeby to zmienić.

Doceńmy, że nasza branża jest otwarta na ciągłe zmiany i ciągły rozwój. Nie wszędzie tak jest, nie w każdym sektorze jest taka kultura pracy.

Niemniej każde zadanie, każdy system, nad którym pracowałem, miał interesujące problemy do rozwiązania. Zawsze jest miejsce, żeby nauczyć się czegoś wartościowego. Podam prosty przykład. Od kilku lat przeżywamy „boom" na NoSQL. Każdy, dosłownie każdy programista chciał robić Cassandrę, mimo że nie było ku temu żadnych racjonalnych powodów. Większość z nas robiła systemy ze zwykłymi bazami relacyjnymi oraz jakimś ORMem – u mnie to był Hibernate.

Co można zrobić? Można narzekać, że Cassandra rozwiązałaby wszystkie nasze problemy (a zapewne tak nie będzie). Można też wykorzystać ten czas, żeby zgłębić, jak działa to, co mamy, żeby móc z pełną świadomością powiedzieć, że panujemy nad tym. Ja zainwestowałem w wiedzę z Hibernate. Czy to była dobra inwestycja? Zdecydowanie tak! Ogromna ilość systemów stosowała, stosuje i będzie stosować ORMy. Wiedza, którą zdobyłem, okazała się później niezbędna i przydatna, mimo że wszyscy zapowiadali koniec relacyjnych baz danych.

Dla pozostawienia czystości obrazu dobrze jest wiedzieć, kiedy i w jakich warunkach wykorzystać nierelacyjne bazy danych. Warto też zaznajomić się z nimi, żeby sprawnie ich używać. Nie należy ich stosować wszędzie, gdzie popadnie.

ZAKŁAD PRACY

Kiedy wychodzimy z zespołem na lunch, w jaki sposób dokonujemy wyboru tego, co zjemy? Bierzemy to, co braliśmy wczoraj, lub to, co wybrał kolega lub koleżanka. Nie przywiązujemy do tej decyzji dużej uwagi.

Spójrzmy prawdzie w oczy. Czy poświęcamy dużo uwagi temu, w jakich technologiach pracujemy? Jak wybieramy narzędzia, w których spędzimy następne miesiące naszego życia? Tak samo! Na przykład przejdźmy do wyboru języka programowania. Jakie są kryteria wyboru języka programowania do nowej usługi? Wybieramy to, co jest znane w naszym „zakładzie pracy", lub wybieramy to, co robiliśmy w ostatnim projekcie. Tak samo jak wybór potrawy na lunch. Wszystko jest w porządku, jeżeli wiemy, że wybór, który podejmujemy, jest dopasowany do potrzeb projektu. Gorzej, jeżeli znamy tylko jedno narzędzie i cały czas używamy tylko tego jednego narzędzia do wszystkich rodzajów problemów.

CV DRIVEN DEVELOPMENT

Otwieram teraz swoje CV sprzed kilku lat – kiedy zaczynałem swoją karierę. Co tam znajduję? Dużą ilość technologii. W zasadzie same wypunktowania i listy technologii, z którymi miałem do czynienia. Otwieram teraz swoje CV. Zostawiłem w nim kilka kluczowych technologii i narzędzi, z którymi pracuję i które uznaję za dobre. Te technologie, które wykreśliłem, zastąpiłem znajomością podstaw inżynierii oprogramowania. W sekcji z doświadczeniem pojawia się więcej wpisów na temat tego, jak oprogramowanie, które tworzyłem, miało wpływ na upraszczanie procesów i biznesów w firmie.

Dlaczego tak? Ponieważ nie chcę być oceniany tylko i wyłącznie na podstawie znajomości frameworków. Narzędzia przychodzą i odchodzą. Zdaję sobie z tego sprawę. W zdecydowanej większości przypadków biblioteka, która ukazała się miesiąc temu, nie pozwoli nam zbudować lepszego projektu. Ważne jest to, żeby się rozwijać i iść z duchem czasu, właśnie dlatego, żeby wiedzieć, które narzędzia i rozwiązania stosować w których sytuacjach.

Kiedy patrzysz na swoje CV, czy widzisz tam zbiór luźno wrzuconych technologii, czy dostrzegasz to, co zrobiłeś? Czy jesteś z tego dumny? Warto – i opłaca się – wybierać takie projekty i takie technologie, żeby być dumnym z tego, czego się dokonało.

HYPE DRIVEN DEVELOPMENT

Według mnie konferencje techniczne są bardziej niebezpieczne dla naszych projektów, niż dostęp do bazy produkcyjnej z prawami kasowania danych. Przed każdą konferencją uczestnik powinien dostać ostrzeżenie, że po powrocie do pracy z konferencji nie należy wywracać projektu do góry nogami. Hype Driven Development to najbardziej beztroskie, a zarazem najbardziej nieodpowiedzialne podejście do wytwarzania oprogramowania.

Programiści dzielą się na tych, którzy tworzą projekty na fali mody, oraz tych, którzy potem muszą to utrzymywać. Oczywiście, ci, co wybrali pracę „na fali", szybko muszą zmieniać projekty. Po pół roku pracy szukają dla siebie nowego projektu, ponieważ ten, nad którym pracują obecnie, jest już przestarzały i nieatrakcyjny.

Jakie jest największe zagrożenie? Nie czytamy dokumentacji, nie sprawdzamy technologii przed jej produkcyjnym użyciem, nie zagłębiamy się dokładnie w „nowość", którą trzymamy w ręku przed jej zastosowaniem. Często kończy się to robieniem projektu bez przemyślenia, ponieważ rzeczy, które są w projekcie, nie do końca do siebie pasują.

Wcale nie trzeba być uczestnikiem konferencji, żeby zachłysnąć się jakąś technologią i położyć przez to cały projekt. Na przykład pracując w iteracyjnym podejściu, ze Sprintu na Sprint, bardzo łatwo jest stracić wizję, którą chcemy zrealizować. Każdego dnia w pracy podejmujemy dziesiątki decyzji architektonicznych, które w przyszłości mogą mieć negatywny wpływ na cały projekt, ale bardzo pozytywny i mile widziany wpływ na ten Sprint. Nie chodzi o to, żeby wygrać bitwę. Chodzi o to, żeby wygrać wojnę.

MALOWANIE

Robiłem remont, zatrudniłem sprawdzoną przeze mnie ekipę fachowców. Pracę wykonywali szybko i sprawnie. Wiedzieli dokładnie, co i jak będą robili. Prace były już mocno zaawansowane, zostało tylko malowanie ścian. Tego dnia dostaję telefon, że zjawią się godzinę później niż zwykle. Jako wytłumaczenie podali, że w miejscu, w którym zazwyczaj robią zakupy, nie było farby podkładowej i muszą jechać do innego sklepu. Zadałem pytanie: „Nie było żadnej farby podkładowej?". Fachowiec przyznał, że inne były. Dodał też, że największe zaufanie ma do tej, która chwilowo była niedostępna, że nie będzie na mnie eksperymentował z inną farbą, że pojedzie do innego sklepu.

Byłem w szoku. Naszła mnie też wtedy refleksja, ile razy to my, programiści, eksperymentujemy na naszych projektach, zostawiamy funkcjonalności nieprzetestowane, które przecież trafiają na produkcję. Mówimy wtedy sobie, że taka branża, że jak nie spróbujemy, to nie będziemy wiedzieli, robimy to przecież, żeby się rozwijać itp. Bardzo często jest to prawdą. Próbowanie czegoś nowego jest skuteczną metodą poznawania innych rozwiązań. Również bardzo często – za często – jest to tylko naszą wymówką, ponieważ czegoś nie chce nam się sprawdzić, napisać dobrych testów lub dowiedzieć się, jak powinniśmy to zrobić poprawnie. Jedni twierdzą, że w zawodzie programisty występuje niska odpowiedzialność za popełnione błędy. Inni próbują zwalić winę na wymagania, które były nieprecyzyjne. Jak temu zaradzić? Należy utrzymywać czystą architekturę naszego systemu. Powinniśmy mieć na uwadze to, że jak system będzie się rozrastał, architektura będzie stawała się coraz bardziej zagmatwana. Jedyne wyjście, jakie widzę, to ciągły refaktoring, tak żeby tempo pracy nad systemem było stałe.

BLOG DRIVEN DEVELOPMENT

Na czym polega Blog Driven Development? Załóżmy, że masz jakieś wyzwanie lub decyzję projektową do podjęcia. Możesz zrobić to, co zwykle, czyli podjąć decyzję na podstawie intuicji. Albo możesz zacząć tworzyć wpis na bloga, w którym opiszesz:

- Problem lub wyzwanie, z którym się mierzycie – Ty lub zespół,
- Analizę możliwych rozwiązań,
- Wybrane rozwiązanie – wraz z argumentami za oraz przeciw,
- Skutki podjętej decyzji.

Nie musi to być poemat na kilka stron, może to być drobna notatka złożona z kilku punktów. Nie jest istotne to, czy kiedykolwiek opublikujesz ten wpis na blogu, ale załóż, że tak może się zdarzyć.

Takie ćwiczenie jest formą futurospekcji. Formą zreflektowania się nad swoimi najbliższymi działaniami, czy doprowadzą one do tego, co chcesz osiągnąć.

Czy będziesz się czuł dumny z tego, co napisałeś? Czy może kiedy patrzysz na to, co napisałeś, to widzisz, że to nie będzie profesjonalne podejście?

Taka metoda pozwoli spojrzeć nam z boku na to, co robimy. Pozwala nam uspokoić emocje, wziąć głęboki oddech oraz jeszcze raz na spokojnie wszystko przemyśleć.

TRADE OFF

Pytanie jest proste. Czy powinienem na każdym kroku inwestować w najnowsze rozwiązania, czy powinienem skupić się na szlifowaniu swoich umiejętności z tego, co jest tu i teraz?

Odpowiedź nie jest łatwa, na pewno nie jest zero-jedynkowa. Branża IT zmienia się szybko. Jeśli prześpisz jakiś moment, możesz się obudzić w zupełnie innym świecie i być przestarzałym. Z drugiej strony, nie znając dobrze swoich narzędzi i technologii, możemy nigdy nie być docenieni za bycie prawdziwym ekspertem. To każdy z nas powinien sam sobie odpowiedzieć na pytanie, jak chce pokierować swoją ścieżką kariery.

ZAKOŃCZENIE

I would advise students to pay more attention to the fundamental ideas rather than the latest technology. The technology will be out-of--date before they graduate. Fundamental ideas never get out of date.

David Parnas

Czym w takim razie powinniśmy się kierować, tworząc nasze systemy? Największą uwagę powinniśmy zwrócić na design i architekturę naszego systemu, a nie na stosowaniu w nim wszystkiego, co najnowsze. Każdy system z biegiem czasu robi się coraz bardziej skomplikowany oraz zawiera coraz więcej rozwiązań „na sznurki". Twoim zadaniem jako programisty jest dbanie o wysoką jakość oprogramowania, przeciwdziałanie temu zjawisku poprzez ciągły refaktoring. Dodawanie nowych narzędzi niewiele nam tu pomoże, a jest duże zagrożenie, że przyspieszy proces erozji naszego oprogramowania.

Skąd wiem, że mój warsztat jako programisty jest dobry? Na to pytanie odpowiem pytaniem: skąd wiesz, że fachowiec, który przychodzi do domu wykonać u Ciebie jakieś prace, jest dobry? Jeżeli jego skrzynka z narzędziami jest duża, jest poukładana, a fachowiec wie, jakiego narzędzia do czego użyć, to wiesz, że trafiłeś na osobę, która poważnie podchodzi do swojej pracy. Naprawa może zająć 10 minut, a fachowiec może użyć jednego narzędzia ze swojej skrzynki z narzędziami – tego jedynego i odpowiedniego do tego zadania. Właśnie na tym polega profesjonalizm – znam swoje narzędzia, które umiem dobrze wykorzystać, dzięki temu jestem skuteczny w tym, co robię.

Nie spoczywam na laurach. Wiem, że świat się zmienia, a ja razem z nim. Swój rozwój planuję, robię to w sposób świadomy. Dzięki temu rozwijam się szybko i w odpowiednich dla siebie kierunkach.

MICHAŁ LEWANDOWSKI

Software developer. Przez ostatnie kilka lat zajmował się różnymi rzeczami związanymi z wytwarzaniem oprogramowania, począwszy od analizy wymagań, przez prowadzenie zespołu, samo kodowanie, skończywszy na utrzymaniu i odpalaniu systemu. Jest przekonany, że każdy profesjonalista, oprócz twardych umiejętności związanych z kodowaniem, powinien mieć rozwinięte umiejętności miękkie.