Ideologia Polskiego Programisty wer. 3

Odnajdywanie Sensu Na Przekór Propagandzie

autor: Jacek Marcin Jaworski

pseudonim: Energo Koder Atlant

utworzono: 2023-01-27

wersja: 416 z dnia: 2023-04-07

Spis treści

Wstęp

Tezą Ideologii Polskiego Programisty wer. 3 jest:

Ideologia nadaje sens życiu w obliczu wszechogarniającego nonsensu zmasowanej propagandy.

Jak wiadomo propaganda dzieli się na krajową i zagraniczną. W praktyce nie ma ten podział znaczenia, bo cel propagandy jest jeden: ogłupianie. Tak! Twój umysł jest wrogiem władzy w kraju i tym bardziej władzy za granicą.

Choć nie naucza się filozofii w polskich szkołach, to jednak wiadomo, że od wieków szukano "filozofii wszystkiego" która przyświecała twórcy wszechświata i która definiuje sens istnienia w nim istot inteligentnych (sens istnienia pozostałych istot jest znany bo wiadomo, że pozostałe istoty mają służyć istotom inteligentnym). Podobnie pasjonaci historii wertują książki na temat historii by poznać źródła sukcesów i powody porażek imperiów i ich wodzów.

My programiści też musimy się mierzyć z problemami filozoficznymi i analizować źródła sukcesów i powody porażek. I to w wielu aspektach od analizy poszczególnych algorytmów, przez architektury programów, dzieje systemów operacyjnych, małych firemek i wielkich, światowych korporacji. Jest nad czym się głowić i to całymi latami.

Na przestrzeni lat 1993-2022 byłem w Polsce piratem, samoukiem, uczniem biednego technikum, studentem zwyczajnej, prowincjonalnej politechniki, programistom w kilku małych firmach. Wiem ile kłamstw i głupot wmawiano na łamach książek, w klasie przy tablicy i w pracy. Wiem też ile upokorzeń trzeba znosić będąc entuzjastą informatyki i programistą. To wszystko było w brew wrogiej propagandzie która trąbiła jakie to sielskie i anielskie jest życie polskiego informatyka (bo słowo programista jest na cenzurowanym). Dlatego jestem zdania, że trzeba omówić jak sobie z tym wszystkim radzić by nowym pokoleniom otworzyć drogę do życia sensowniejszego od mojego.

Wykaz Skrótów

komp.	komputer
sys.	system
sys. komp.	system komputerowy
sys. op.	system operacyjny
prog.	program
arch.	architektura
j. prog.	język programowania
kś.	książka
s.	strona
p.	punkt

1 Co ma znaczyć dumping tekstów kultury?

Wiedza informatyczna jest dumpingowa: poradnikowe filmy na youtube.com są całkowicie darmowe. Ktoś płaci ogromne pieniądze by przestrzeń dyskowa na youtube.com rosła szybciej niż wszyscy cywilizowani ludzie na Ziemi są w stanie nagrywać filmiki. Jest to całkowicie nienormalne i wymierzone przeciwko wszystkim państwom na świecie, bo ten dumping uniemożliwia rozwój alternatywnych rozwiązań. Akceptowanie youtube.com przez rządy na całym świecie jest dowodem zmowy, bo dumping normalnie powinien być tępiony z całą premedytacją - były takie awantury jak np. tania stal z Chin.

Podobnie ktoś finansuje Abi Szeki z Indii by pisali oni poradniki dla programistów i administratorów na s. WWW. Co prawda hosting s. WWW jest dużo tańszy niż hosting filmów, ale też jest to dumping tekstów kultury. Też nie wiadomo kto ani dlaczego za to płaci.

Podobnie sprawa ma się z darmowymi serwisami informacyjnymi. Też nie wiadomo kto ani dlaczego za to płaci.

Nie jest przy tym prawdą, że hosting tego wszystkiego i redaktorów można utrzymać z reklam. Zadaj sobie pyt.: kiedy ostatnio kliknąłeś w reklamę i coś kupiłeś? Ja jeszcze nigdy! A internet mam od końca 2001r.

Z drugiej strony wiadomo, że budżet USA to ok. 17 bilionów dolarów (polskich bilionów), natomiast ich długi przekraczają 35 bilionów dolarów. Tak więc widać, że biorą ogromne kredyty i w coś inwestują. Pytanie tylko kto ma za to zapłacić? Czy jeśli my, programiści korzystamy z tych darmowych poradników na youtube.com i na s. WWW to czy jakiś bankier nabija nam kwity i kiedyś je wyciągnie i będzie gadał, że mamy długi? Jeśli tak to co? Sprzedadzą nas za te długi? Na roboty do jakichś obozów koncentracyjnych?

Jest na to dość prosty sposób: W razie pytań należy mówić, że nikt u nich nie zamawiał youtube.com ani filmików poradnikowych, ani nikt nie żądał wynajmowania Abi Szeków z Indii. Jeden jest tego warunek by móc tak odp.:

Nasze głowy muszą być przynajmniej na tyle świadome co dzisiaj. Niestety po śmierci ludziom w standardzie kasuje się pamięć i osobowość - jest to zabijanie umysłu. Wtedy nic już nie można zrobić w swojej obronie, bo wtedy myślenie jest na poziomie niemowlęcia (przez to łyka się każdą głupotę jak małpa kit).

1.1 Co ma znaczyć zalew tanich, grubych książek na banalne tematy programistyczne?

Kś. są b. tanie. Jak się temu przyjrzeć, to widać, że na jeden i ten sam temat może być kilka grubych kś. - mimo, że powinny być tylko 2: podstawy (cieniutka ok. 150s.) i zaawansowane (do 300s.).

Obecnie (2023r.) nie potrafię wskazać żadnej kś. inf. na tematy zaawansowane.

Mimo, że informatyka to algorytmy i architektury programów, to kompletnie brak kś. na te tematy - bo przecież kś. pt. "Wprowadzenie do alg. i struktur danych", to powinna być szkolna lektura a nie pomoc programisty! W ogóle nic się nie mówi, nie pisze ani nie uczy o współczesnym poziomie badań nad algorytmami i architekturami programów komputerowych.

Dla nas to znaczy jedno: przez naukową blokadę informacyjną polska informatyka stanęła w miejscu - gdzieś na poziomie lat 80. Bo przecież nowe j. prog. z Ameryki nie oznaczają żadnego postępu w dziedzinie naszego intelektu - bo ich nauka i opanowanie to czyste ogłupianie i zawracanie głowy. Jest nawet gorzej, bo ja osobiście nie znam żadnego normalnego j. prog. - każdy ma defekty albo w składni albo w bibliotekach jakie są dostępne. Dlatego jeśli chcemy być normalni i nowocześni każdy z nas musi sam badać sprawy nowoczesnych algorytmów i nowoczesnych architektur na własną rękę.

1.2 Co ma znaczyć zalew darmowych sys. op. rodz. Linuks?

To znowu dumping cenowy zabijający na świecie wszelką konkurencje w dziedzinie sys. op. Ja muszę korzystać z darmowego sys. Linuks, bo nie widzę sensownej, płatnej alternatywy. Bezskutecznie próbowałem się dowiedzieć jakich miesięcznych kwot oczekuje Canonical albo Trisqueel.

To, że Canonical i Trisquel ukrywają oczekiwane kwoty miesięcznych wpłat użytkowników oznacza, że te organizacje mają ukryte cele jakie są sprzeczne z interesem użytkowników. Przez to zatajanie oczekiwań finansowych dają do zrozumienia, że kasę w wystarczającej ilości dostają od bankierów i nam tylko nabijają kwity.

Na to jest rada: należy wpłacać co miesiąc "sensowne kwoty" na konto dostawcy distro jakiego używamy. Aby nie okazało się, że płacisz za mało, to każdy użytkownik darmowej dystrybucji sys. Linuks powinien sam sobie odp. na pyt. "Jak ważny jest dla mnie ten system?". Wtedy kwota powinna wyłonić się sama.

"Sensowne kwoty" to nie to co chce Mkorsoft ani nawet nie to co chce Aple, bo tylko z jednego powodu mogą szpiegować wszystkich swoich użytkowników:

Jedyną podkładkę jaką mają wielkie korpo to gadka, że opłaty użytkowników nie pokrywały kosztów - i dlatego szpiegowali i handlowali z bankierami i z rządem danymi swoich użytkowników.

Jeśli chodzi o sprytny tel. to należy wybierać wyłącznie te prog. jakie umożliwiają wykupienie wersji "pro" - zwykle kosztuje to grosze, ale teoretycznie daje święty spokój.

1.3 Co ma znaczyć zalew darmowych języków i bibliotek programistycznych?

Dziesiątki języków programowania to też ogłupianie programistów. Co chwilę ktoś wyskakuje z takim zakodowanym w C++ "wynalazkiem" i twierdzi, że od teraz programy same będą się tworzyły. Jednak jest to możliwe tylko dla SI i jej nie robiło by to różnicy jaki był by to j. prog. Jednak mimo, że jest to do zrozumienia dla każdego programisty, to kier. proj. karzą kodować w coraz to nowych j. prog. i w nowych bibl.

Jest prosty dowód na to, że te dziesiątki j. prog. mają tylko ogłupiać programistów: one są po prostu darmowe, czyli bankier lub rząd płaci i wymaga by wszystkie one były nienormalne. Robią to bo boją się geniuszy.

Natomiast jak Ty byś płacił uczciwie za soft, to dla Ciebie by musieli go poprawiać i rozwijać. Dzięki temu byś pracował coraz wydajniej i coraz lepiej - wkrótce tworzyłbyś cuda i to z łatwością, zamiast obecnych męczarni i szarpanin z najprostszymi sprawami.

Trudno coś na to poradzić. Bo jak będziesz oportunistą i będziesz się upierał przy programowaniu w tym co już dobrze opanowałeś, to możesz stracić pracę i mieć trudności by znaleźć nową (znam to z własnego doświadczenia). Z drugiej strony ciągłe poznawanie nowych j. prog. powoduje że się idiocieje, co widać choćby po pomyłkach do jakich dochodzi w trakcie programowania.

2 Jakie kupować książki do nauki programowania?

Odnośnie kupowania kś. dotyczących programowania to wypracowałem taką strategię:

1. Kupuję najcieńszą kś. na interesujący temat jak jestem w nim początkujący. Grubszą kś. kupuję gdy się upewnię, że temat "jest przepotężny i ma potencjał". Robię tak, bo nie jest sztuką wywalić kasę na najgrubszą kś. natomiast sztuką jest zrozumienie nowego tematu;

2. Kupuję kś. polskich autorów, bo mam takie odczucie, że treść polskich kś. lepiej zapamiętuję. Nawet tł. polskiego autora który napisał kś. w j. ang. i która potem została przetłumaczona na j. pol. wyraźnie gorzej się zapamiętuje;
3. Należy kupować kś. które zwiększają wiedzę pod względem jakości a nie jej ilości. Tak więc kolejna kś. o nowym j. prog. to jedynie zwiększenie wiedzy pod wzg. ilościowym. Natomiast poznawanie nowych arch. lub nowych alg. jest zwiększeniem swojej wiedzy pod wzg. jakościowym - wynika to z prostego faktu, że te sprawy są w informatyce najważniejsze;
4. Po przeczytaniu każdej kś. należy napisać jej recenzję - ja stosuję 4 punktową skalę ocen: rewelacje, zalety, wady i partactwa. W celu napisania recenzji należy zaznaczać w kś. istotne fragmenty.

Wykonywanie recenzji i sprawozdań z podejmowanych działań jest nauczane w szkole, jednak tam nie mówią, że tak należy też postępować w życiu i zamykać jego etapy w formie pisemnych podsumowań.

3 Jaki kupować sprzęt do programowania?

Kiedyś wierzyłem w potężne PC. Teraz wystarcza mi z powodzeniem laptop. Wiem, że laptopy są droższe, ale: w razie potrzeby są przenośne, procki w nich też są wielordzeniowe jak w PC, brak łącz rozwiązuje HUB USB, daje się podłączyć duży monitor i normalną klaw. i mysz. Dzięki temu w pracy na funkcjonalności nic się nie traci, a zyskuje możliwość szybkiego zabrania komputera na wyprawę.

Jeśli chodzi o firmy czy modele, to zauważyłem, że w "zwykłych" laptopach stosuje się stare podzespoły: stare matryce, stare kamery i stare głośniki. Nowe są tylko bebechy. Skutek jest taki, że ten sprzęt jest 3 ligą w porównaniu do laptopów Aple. Dlatego zalecałbym kupowanie mocnych laptopów Aple.

Zalecam kupowanie do programowania najmocniejszych laptopów na jakie cię stać, z zastrzeżeniem by nie ucierpiały na tym inne dziedziny twojego życia (żeby zakup laptopa do programowania nie był żadnym poświęceniem).

Oczywiście kupując laptop do programowania nie trzeba kupować drogiej karty graficznej dla graczy, bo ona niepotrzebnie podwaja cenę zestawu.

4 Jaki kupować sys. op. do programowania?

Tak. Zalecam kupowanie oprogramowania, a szczególnie sys. op. Wyjaśniłem to w Co ma znaczyć zalew darmowych sys. op. rodz. Linuks? Jednak można zadać pyt.: Czemu by nie kupować sys. MacOS. Odp. jest prosta: Ten system byłby dopuszczalny tylko w przypadku gdyby okazało się, że możliwe jest zapewnienie sobie prywatności. Bo od czasu afery z ukrytym plikiem z współrzędnymi GPS jaki był w AiFonie co jakiś czas wybucha nowa afera związana ze szpiegowaniem użytkowników przez Aple.

Oczywiście nie jest tak, że używając sys. Linuks mamy 100% prywatności. Na szpiegostwo sys. Linuks jest wiele dowodów:

1. Prawie wszystkie prog. przechowują "historię otwieranych plików";
2. Wiele prog. ma pliki "cache" jakich usuwanie "nic widocznego nie powoduje";
3. Domyślnie wszyscy użytkownicy w sys. Linuks mogą czytać cudze pliki;
4. Domyślnie w sys. Linuks nie ma żadnej kontroli nad poł. wychodzącymi (każdy programik może wysyłać "co chce, kiedy chce i gdzie chce").

Jednak ja osobiście wypracowałem sobie sposoby pracy z sys. Linuks jaki pozwala mi mieć nadzieję, że ograniczam to szpiegostwo. Te sposoby, to:

1. Praca bez poł. z Internetem: po instalacji sys. skryptem instaluję wszystkie potrzebne mi do pracy pakiety. Z Internetem łączę się tylko gdy jest to niezbędne.
2. UFW i jego konfigurowanie skryptem w którym mam profile takie jak: google, praca, zakupy, poczta itd.;
3. Firejail i jego strojenie w celu wyłączania programom dostępu do plików jakich nie potrzebują oraz do wyłączania dostępu do sieci tym programom którym nie chcę na to pozwalać;
4. czysc.sh: napisałem sobie ten skrypt w celu usuwania podejrzanych plików i kat. z systemu oraz z kat. użytkowników.

Wiem, że UFW i Firejail mogą się wydawać amatorskie, bo przykrywają inne narzędzia (IPTables i AppArmor lub SELinuks), jednak ja byłem w stanie je zrozumieć i byłem w stanie ich skutecznie użyć.

5 Jaki wybrać język do programowania?

Tak na prawdę program to skompilowany do kodu maszynowego kod napisany w j. kompilowanym. Czyli skrypty nie są programami, bo nie dają wyniku w postaci kodu maszynowego. Czyli do programowania przydatne są: Asembler, C, C++, D, Paskal, Delfi, Rust, Fortran. Natomiast skrypty, to: Jawa, Jawa Skrypt, C#, PHP, Pyton, Perl, Lua, Go oraz j. powłoki z Baszem i PowerSzel na czele.

Niewątpliwie aby mieć jakiekolwiek pojęcie o programowaniu trzeba nauczyć się Asemblera. Wtedy będziesz wiedział co to są rejestry procesora i jak on przyjmuje kolejne polecenia. Kolejnym zalecanym j. jest C. W nim są zaprogramowane wszystkie współczesne sys. op. Drugą ważną sprawą jest to, że j. C ma od początku ustandaryzowane ABI dzięki któremu może on służyć do rozszerzania innych j. (np. można robić w nim wtyczki do motorów SQL, albo do j. Pyton w celu uzyskania pełnej prędkości przetwarzania). Trzecią sprawą jest to, że składnia C była inspiracją dla wielu innych twórców j. prog. (z C++ i D włącznie). J. C++ jest składniowo i binarnie zgodny z j. C. J. D jest tylko binarnie zgodny z j. C. Moim zdaniem to dobre argumenty by poznać j. C. Jak to się opanuje, to warto poznać jakiś j. obiektowy, czyli C++, D lub Delfi. Z czego najpopularniejszy i najpotężniejszy jest C++.

6 Jakie kupować narzędzia do programowania?

Osobiście nie wiem nic na temat płatnych narzędzi programowania dla sys. Linuks. Dlatego można przyjąć strategię żeby używać tylko to co jest dostępne w repo distro i płacić za to distro i niczego więcej nie instalować. Wtedy mamy podkładkę, że płaciliśmy dostawcy sys. Linuks i nie interesują nas jego rozliczenia z poddostawcami.

Gdyby pojawiło się jakieś fajne, komercyjne oprogramowanie na sys. Linuks i było by możliwe jego normalne, komercyjne nabycie, to myślę że warto było by je stosować.

Jednak wcale bym nie stosował darmowych koni trojańskich wielkich korpo w stylu: Chrome/Chromium, Android Studio, Teams, Visual Code, Edge itd. Jak ktoś to na mnie wymusza ich użycie - robię co trzeba i odinstalowuję to z systemu z opcją **--purge** . Innym sprytnym sposobem na te wymuszenia jest chwilowa instalacja tego softu na sprytnym tel.

W ogóle nie należy też używać programów w paczkach Snap ani Flatpak z uwagi na niejawny charakter ich manifestów i kompletny brak możliwości kontroli nad tym do czego one mają dostęp i co wysyłają w świat. Słowem: paczki Snap i Flatpak nie nadają się do niczego bo nie można ich uruchamiać w piaskownicy Firejail (argumentuje się to tym, że maja one własną piaskownicę. Jednak co z tego gdy nie ma nad nią żadnej kontroli?).

Należy używać jedynie paczek podatnych na działanie w Firejail. Dla sys. rodź. Debian/Ubuntu, na chwilę obecną są to jedynie paczki *.deb i *.appimage .

7 Czy programować po polsku a może po ang.?

Oczywiście może to być w proj. narzucone. Ale załóżmy, że mamy wybór (np. w domowych proj.), to co wtedy? Oczywiście wygodniej jest programować po polsku - bo znamy ten język. Ale czy to wszystko? Nie. Okazuje się, że stosowanie w programowaniu nazw z obcego języka jest czymś w stylu emulacji. Tak samo jak emuluje się procesor innego komputera, tak samo człowiek może emulować inny język ludzki. W przypadku procesorów ta emulacja oznacza jedynie drastyczny spadek wydajności. Natomiast u ludzi oprócz tego dochodzą konflikty nazewnicze i gramatyczne jakie przenikają do mowy potocznej każdego kto emuluje język obcy. Te konflikty powodują degenerację umysłową i językową u tych ludzi którzy próbują posługiwać się kilkoma językami. Jest to dalszy etap ogłupiania w starym, szkolnym stylu gdzie wymuszano na nas naukę kretyńskich zasad polskiej ortografii (w zakresie ż-rz, u-ó i h-ch). Dlatego ja osobiście mam poliglotów za umysłowo upośledzonych.

Programowanie w języku ojczystym jest najefektywniejsze i daje największy pożytek dla umysłu.

Programowanie w j. obcym jest sprzeczne z rozwojem własnej potęgi umysłowej.

8 Jakie wykształcenie jest konieczne do programowania?

Jak się okazuje, po latach, wcale nie jest dobre uczenie się i studiowanie w jednej dziedzinie. Moim zdaniem jest to nieoczywisty temat i warto go omówić.

Zakładamy, że chcę zrobić zawrotną karierę programisty, czyli chcę być ekspertem najwyższej klasy oraz chcę zrobić karierę kierowniczą.

8.1 Technik

Naukę o komputerach należy zacząć w technikum. Ale czy na kier. technik informatyk? NIE!!! Trzeba zacząć od podstaw, czyli od elektroniki. Elektroniki można uczyć się w technikach elektronicznych oraz w technikach łączności. Nauka w technikum ma wiele zalet: tam podają jakie są elementy elektroniczne w komputerach, jak zbudowane są bramki, czyli procesory i pamięci. Uczą tam też asemblera (mikro-kontrolery). W technikum poznajesz procedurę nauki w pracowaniach jaka jest analogiczna do procedur w laboratoriach na polibudzie. To daje świetne podstawy do przyszłej kariery programisty.

Dla dalszej kariery kluczowe znaczenie mają projekty semestralne (nawet gdy ich nikt nie wymaga należy je sobie samemu wyznaczać), praca dyplomowa i matura.

8.2 Inżynier

Prawdziwym inżynierem można zostać jedynie po polibudzie. Koniec kropka.

Oczywiście by zostać programistą należy studiować informę. Tu też kluczowe są prace semestralne i dyplomowe.

Studiując nie należy realizować projektów hobbystycznych. Wynika to z faktu, że to są pożeracze czasu. Wyjątkiem od tej zasady są proj. zaliczeniowe (ale one są w ramach studiów).

Po studiach inż. można śmiało iść do pracy, a dalsze studia zrealizować zaocznie.

8.3 Mgr Inż.

Kiedyś kolega powiedział mi, że inż. może samodzielnie konstruować urządzenia, natomiast mgr może kierować ludźmi. Dla mnie ma to sens. Dlatego jak chcesz zrobić karierę kierowniczą, to koniecznie trzeba mieć mgr w tytule.

Jednak same studia nie wystarczą do kierowania. Potrzebne jest doświadczenie techniczne:

Aby być samodzielnym konstruktorem potrzebny jest dyplom inż. i 5 lat komercyjnego doświadczenia.

By być kierownikiem proj. potrzebny jest dyplom mgr i 10 lat komercyjnego doświadczenia w programowaniu.

8.4 Studia podyplomowe

Studia podyplomowe są interesujące w zakresie metod prowadzenia projektów oraz zarządzania i automatyzacji produkcji.

Te studia należy podjąć po ok. 5-7 latach pracy na stanowisku programisty. Wynika to z faktu, że o f. kierownika proj. należy się starać po 10 latach pracy na stanowisku prog. Moim zdaniem jest to uczciwe i daje szansę na kolejne sukcesy.

Na stanowisku kierowniczym zalecam dalej kodować. Może to być część kierowniczego etatu, może to być projekt hobbystyczny po godzinach. Chodzi tu o to by nie dać się odizolować od prawdziwych problemów z jakimi zmagają się programiści - w tym twoi ludzie. Gdy się dopuści do takiej izolacji wtedy nie będziesz wiedział co o czym mówią twoi ludzie.

8.5 Własne projekty hobbystyczne i projekty komercyjne

Aby efektywnie prowadzić projekty trzeba sobie uświadomić, że:

Dobry projekt inżynierski jest sprzeczny z prostotą i wymaga dużo więcej pracy niż projekt w stylu "byle jak". Oznacza to, że "w garażu" nie można osiągnąć tego co jest możliwe w komercyjnym zespole programistycznym.

Wynika to z faktu, że w profesjonalnym projekcie konieczne są (m. in.):

1. Przedefiniowanie wszystkich używanych typów bibliotecznych i przedefiniowanie używanych typów wbudowanych: w celu ich łatwego podmienienia własną klasą: gdyby zaszła taka konieczność;
2. Uzupełnienie typów bibliotecznych: dodanie tego co brakuje i zastąpienia tego co nienormalne;
3. Kodowanie wszystkiego ręcznie w celu pełnej kontroli (nie można używać Qt Designer, ani innych czarodziei);
4. Profilowanie programu: wycieki pamięci i analiza pracy;
5. Zabezpieczenia programu przed typowymi atakami;
6. Użycie serwera repo (w stylu GitLab) z automatycznym budowaniem paczek i automatycznie uruchamiającym testy automatyczne.

Dlatego projekty hobbystyczne i zaliczeniowe (w technikum i na polibudzie) z definicji muszą być amatorskie z powodu braku środków na ich profesjonalne wykonanie. W przeciwnym wypadku można ich wcale nie ukończyć.

8.6 Własne opracowania teoretyczne

Niestety obecna propaganda odmawia nam kś. na zaawansowane tematy programistyczne. Dlatego należy sobie zadawać pyt. "Czego mi brakuje w programowaniu?" albo "Co jeszcze jest interesujące w tym programowaniu?" albo "O co się codziennie potykam programując?" Wtedy należy takie tematy zgłębiać z użyciem wyszukiwarki internetowej, dostępnych książek i gazet i na ich podstawie i na podstawie własnych przemyśleń tworzyć dokumentację podobną do tej którą właśnie czytasz.

8.7 Większa wydajność przez metodologię i automatyzację a nie przez pośpiech

Niektórzy twierdzą, że dobra jest zdolność do pracy pod presją. Możliwe, że to wymusza największe zaangażowanie. Pełne zaangażowanie jest dobre. Jednak generalnie w pracy pracy programisty pośpiech nie jest dobry. Praca w pośpiechu powoduje degenerację projektu z tego powodu, że w pośpiechu myślenie zastępuje szalone kodowanie. Ja natomiast widzę dwie drogi do większej wydajności:

Metodologia pracy: wprowadzenie systematyczności w celu eliminacji powtarzalnych błędów. Dotyczy to np. organizacji dnia, albo procedury wykonania nowego wymagania, naprawy błędu lub dodania zgłoszonej poprawki.

Automatyzacja pracy: jak coś wklepujesz tak samo setki razy, albo coś musisz zawsze tak samo wyklikać, to znak, że należy napisać skrypt który to będzie robił automatycznie. Porobienie sobie tych skryptów sprawia, że pracuje się zupełnie inaczej, bo eliminuje frustrację i powoduje zadowolenie z pracy głową a nie wyłącznie rękami.

9 Jak pracować na stanowisku programisty?

Niezależnie od szczebla w hierarchii należy próbować nowych rozwiązań i metodologii i się nimi bawić w celu ustalenia co się sprawdza a co jest słabe.

Nie programuj funkcji typu omnibus ani programów typu omnibus, bo one nie będą działać. Zamiast tego dziel problemy i zamykaj je w "izolowanych paczkach". Paczki te powinny być łatwe do testowania. Te "paczki" powinny łatwo współpracować ze sobą.

Izoluj projekt od świata zewnętrznego. Nie jest tak, że trzeba reagować na każde nowe żądanie. Bo sabotażystów chcących ubić twój projekt jest cała masa. Dlatego po rozpoczęciu kodowania należy się bronić rękami i nogami przed zmianami.

Pieniądze to zapis w umowie. Nie zależnie od niego na stanowisku należy dawać z siebie wszystko co najlepsze.

Godziny pracy to zapis w umowie. Nie zależnie od niego należy z pełnym zaangażowaniem realizować cele firmy. Np. dotrzymywać terminów wysyłki urządzeń.

Gdy pracujesz od 6:00 to wiesz, że z dnia bierzesz tyle ile się da: w pełni wykorzystujesz swoje możliwości oraz dostępny czas.

W pracy, tak jak w rodzinie, musisz dawać ludziom fory mimo, że denerwują, a niektórzy nawet sabotują twój projekt.

Fajnie jest pracować na najnowszym sprzęcie. Jednak nawet jeśli jest on najstarszy, to należy pracować najlepiej jak to jest możliwe.

Należy być aktywnym i reagować na problemy opracowując ich rozwiązanie. Rozwiązanie należy zgłaszać ustnie. Przy sobie można jedynie mieć karteczkę z podpunktami. Wynika to z faktu powszechnego szpiegostwa systemów komputerowych. A przecież nikt nie chce by odp. na jego wniosek racjonalizatorski przyszła z lokalnej delegatury.

Każdy projekt powinien się zaczynać i kończyć analizą, która wskazuje na silne i słabe strony organizacji, oraz na okazje dla firmy i zagrożenia dla projektu. Analiza powinna wskazywać na sposób idealnej realizacji projektu. Dodatkowo po zakończeniu projektu należy podjąć próbę jego oceny wg. czteropunktowej skali: rewelacje, zalety, wady, partactwa.

Jak długo będziesz kodował, tak długo będziesz miał pojęcie jak wygląda praca programisty.