KPF - Czy w pracy inżyniera jest miejsce na intuicję i twórczość?

Bartosz Latosek

Kwiecień 2024

Spis treści

1 Wstęp

Rozprawkę tę poświęcono bardzo ogólnym rozważaniom na temat roli intuicji i twórczości w pracy inżyniera. Termin inżynier jest dość szerokim określeniem, w którym mieszczą się specjaliści z szeregu dziedzin, w których kontekście każda odpowiedź na postawione pytanie jest dostatecznym materiałem na osobną rozprawkę (o ile nie całą książkę). W niniejszym tekście skupiono się na dziedzinie bliskiej autorowi - informatyce, a konkretniej inżynierii oprogramowania.

Rozprawka podzielona została na dwie zasadnicze części, w których rozważane są role poszczególnych składowych pytania w pracy inżyniera-programisty. Wiedza w nich zawarta pochodzi częściowo z osobistych doświadczeń autora zebranych z przestrzeni kilku lat pracy w zawodzie oraz innych źródeł internetowych powiązanych z poruszanymi tematami.

2 Intuicja

„Intuicja to jedynie suma drobnych, ale całkiem konkretnych rzeczy, którym mózg nie zdążył jeszcze nadać nazwy"

Jo Nesbø

Zgodnie z definicją, którą dostarcza nam *Wikipedia* [3] - **Intuicja** (z łac. *intuitio* – wejrzenie) to sądy oraz przekonania pojawiające się bezpośrednio, niebędące świadomym operowaniem przesłankami, rozumowanie bez uświadamiania procesu dochodzenia do rozwiązania problemu. Intuicja nie pojawia się u ludzi samoistnie. Jest ona mechanizmem, który wykształca się przez ciągłe powtarzanie podobnych zadań i ma na celu swojego rodzaju automatyzację i optymalizację czasową kolejnych (poprzez np. wczesne przewidywanie efektów danego działania, przez co z góry można zdeterminować jego skuteczność). Z tego też powodu intuicja dotycząca tego samego zagadnienia będzie bardzie lub mniej rozbudowana w zależności od doświadczenia jej posiadacza.

Zgodnie z powyższym nie można powiedzieć, że w życiu zawodowym programisty nie istnieje intuicja. Z reguły tam, gdzie rozwiązywane są podobne w naturze zadania - z biegiem czasu kształtuje się mechanizm pomagający w przyszłym szacowaniu ich rozwiązań. Im bardziej doświadczony programista, tym z szerszym spektrum zadań miał styczność podczas swojej kariery zawodowej i szanse na to, że spotka się z problemem całkowicie dla niego obcym są bardzo niskie. Jednakże nawet w takiej sytuacji intuicja doświadczonego inżyniera oprogramowania pozwoli mu na dobranie najefektywniejszej metodyki pracy w nowym kontekście i zaoszczędzi mu sporo czasu, który mniej doświadczony pracownik prawdopodobnie przeznaczyłby na opanowanie szoku związanego z napotkaniem egzotycznego problemu oraz na próby jego zrozumienia. W ten sposób intuicja staje się nieodłącznym elementem profesjonalnego rozwoju programisty, umożliwiając mu szybsze i skuteczniejsze radzenie sobie z wyzwaniami zawodowymi.

Zastanówmy się teraz nad obecnością intuicji w przypadku programisty, który dopiero zaczyna swoją karierę. W optymistycznym scenariuszu, *junior* [1] raczej jest świadomy konsekwencji płynących z usunięcia bazy danych na serwerze produkcyjnym aplikacji, której wytwarzaniem zajmuje się jego przełożony. Pomimo tego, że wcześniej nie miał doświadczeń związanych z wystąpieniem podobnych sytuacji, młody programista jest w stanie przewidzieć potencjalne konsekwencje i powstrzymać się przed działaniami mogącymi wywołać podobnie dotkliwe problemy. Mając na uwagę przytoczoną wcześniej definicję intuicji - należy stwierdzić, że nie występuje ona w przytoczonym przykładzie. Skąd więc opór młodego programisty przez popełnianiem podobnych błędów? W powyższej sytuacji mamy do czynienia raczej z czymś bliższym instynktom samozachowawczym. Są one głęboko zakorzenione w *jaszczurzym mózgu* [2] człowieka, a co za tym idzie współdzielone również przez przedstawicieli innych gatunków. Adept sztuki programowania wie, że narażając się na konsekwencje swoich działań może pożegnać się z pracą - co w obecnych realiach rynku pracy może być dewastujące na tak wczesnym etapie kariery, a co za tym idzie stracić źródło utrzymania i w naprawdę skrajnym przypadku skończyć w roli testera oprogramowania aplikacji mobilnych. Podsumowując: intuicję należy raczej rozumieć jako mechanizm, który do pełnego wykształcenia potrzebuje czasu i ekspozycji na szereg doświadczeń i nie należy jej mylić z podstawowymi instynktami samozachowawczymi zaszytymi w mózgu pierwotnym.

1. Fachowa nazwa programisty, który dopiero zaczyna przygodę w tej profesji
2. Jaszczurzy mózg, nazywany także mózgiem pierwotnym, jest częścią mózgowia ssaków odpowiedzialną głównie za instynkty, podstawowe funkcje życiowe oraz reakcje walki, ucieczki lub zamrożenia w sytuacjach zagrożenia. Jest to starsza ewolucyjnie część mózgu, która kontroluje fundamentalne funkcje fizjologiczne i emocjonalne, a także wpływa na zachowania związane z przetrwaniem.

Na pytanie *"Czy w pracy inżyniera jest miejsce na intuicję..."* można by udzielić bardziej dosłownej odpowiedzi. W wielu stosowanych algorytmach (szczególnie tych związanych z szeroko pojętym uczeniem maszynowym i sztuczną inteligencją) istnieje zagadnienie *funkcji heurystycznych*. Są to narzędzia odpowiedzialne za podejmowanie szybkich i często przybliżonych decyzji w sytuacjach, gdzie pełne analizy są trudne lub niemożliwe. Stosują one właśnie coś w rodzaju intuicji do ogólnej oceny sytuacji, a następnie na podstawie uzyskanych wniosków pomagają bardziej precyzyjnym algorytmom na szybsze i skuteczniejsze rozwiązywanie zdefiniowanego problemu. Mając na uwadze powyższą definicję możemy jednoznacznie powiedzieć, że w pracy inżyniera nie tylko jest miejsce na intuicję, ale ta intuicja pod postacią licznych heurystyk występuje i jest bardzo mile widziana z perspektywy zadań na tyle skomplikowanych, że podejście ścisłe i logiczne jest suboptymalne.

W ostatniej części niniejszego paragrafu należałoby się zastanowić nad korzyściami jakie niesie ze sobą używanie intuicji podczas pracy w życiu codziennym przeciętnego programisty. Niewątpliwym pretendentem w kategorii zalet jest oszczędność czasu. Przez możliwość skojarzenia części nowo postawionego zadania z napotkanymi do tej pory problemami niekiedy dobry inżynier jest w stanie od razu zaproponować działające rozwiązanie lub potencjalne źródło występujących komplikacji. Z biegiem czasu wykształca się u niego również optymalny szablon działania, dzięki któremu od pierwszej chwili styczności z problemem ma jasno nakreślone etapy do realizacji, a co za tym idzie oszczędza czas, który w przeciwnym wypadku musiałby poświęcić na wstępny rekonesans i dobór właściwej metodyki do postawionego zadania. Kolejnym ważnym elementem jest zwiększona skuteczność działania a także dostęp do niekonwencjonalnych i bardzo często znacznie bardziej optymalnych metod działania. Intuicja może podszepnąć programiście wykorzystanie elementów znanych mu z pokrewnych problemów, których wykorzystanie w obecnej sytuacji na pierwszy rzut oka mogły by wydawać się abstrakcyjne a nawet pozbawione sensu.

3 Twórczość

„Żeby napisać jedno własne zdanie, trzeba przeczytać tysiące cudzych"

Ryszard Kapuściński

Na samym początku warto spojrzeć na zagadnienie twórczości pod innym kątem niż otwarte dyskusje o charakterze filozoficznym. W polskim prawie obowiązuje ulga podatkowa dla inżynierów oprogramowania wykonujących pracę w swoim fachu. Profesjonalnie, zjawisko to nosi nazwę *autorskich kosztów uzyskania przychodu* i w praktyce pozwala na zmniejszenie odprowadzanych podatków nawet o **50%**.[1]

W tym momencie można by się sprzeczać, czy programista etatowy faktycznie *tworzy* oprogramowanie w sposób kreatywny, niemniej jednak w świetle prawa jego praca nosi miano kreatywnej. Przeciwnicy nadmiernego wzbogacania się programistów mogliby rzec, że w **95%** przypadków, **95%** programistów [3] po prostu bezczelnie kopiuje własność intelektualną innych programistów (którzy prawdopodobnie zrobili dokładnie to samo) a następnie lekko modyfikuje je wedle własnego uznania. Czy nie jest to jednak zjawisko analogiczne, które obarczone jest znacznie mniejszą pogardą w kontekście klasycznie rozumianych artystów? Nie rozglądając się daleko, możemy natknąć się na przykład na nowoczesne interpretacje znanych klasyków (Takie jak np. postać *Chewbacci* w formie *Człowieka Witruwiańskiego* autorstwa Leonardo Da Vinci[2]). Autorom podobnych dzieł ciężko jest odmówić pierwiastka kreatywności, pomimo że ich praca ograniczona została do połączenia dwóch istniejących już ikon kulturowych.

Wierne odwzorowanie arcydzieł jest równie (a może nawet i bardziej) wymagające od wyżej wymienionych form czerpania inspiracji z pracy kolegów po fachu. Przeniesienie słynnej *Mona Lisy* z ekranu telefonu na płótno, zachowując przy tym autentyczne cechy i tajemniczość oryginału nie jest zadaniem trywialnym. Wbrew pozorom - przeniesienie elementów oprogramowania z najpopularniejszej aplikacji typu *open source* (kod udostępniany jest publicznie) i poprawne zintegrowanie jej z obecnie wytwarzanym systemem w mniejszym lub większym stopniu również wymaga pracy. Tak jak kiepski artysta jest w stanie odrysować najistotniejsze kontury kopiowanego obrazu - tak i kiepski programista jest w stanie analogicznie skopiować i wkleić w praktyce cały znaleziony kod do swojego edytora. Szanse na to, że operacja zakończy się sukcesem i rozwiązywane problemy faktycznie znikną jest minimalna, a gdy taka sytuacja faktycznie wystąpi - stanowi dla wykończonego programisty dostateczny pretekst do wynagrodzenia w postaci czwartej filiżanki kawy tego dnia. Przed zaczerpnięciem inspiracji, zarówno w przypadku obrazu oraz fragmentów kodu - trzeba zrozumieć wybrany wzorzec. Należy poświęcić faktyczne nakłady pracy i czas by obiekt przeanalizować, pojąć pierwotne intencje autora, a co najważniejsze

3. Dane nie są poparte żadnymi badaniami, służą jedynie zwróceniu uwagi czytelników

spojrzeć na dzieło w kontekście, do którego uporczywie staramy się je przenieść. W całym tym procesie niewątpliwie wykorzystywana jest prawa półkula mózgowa, a wykonaną pracę możemy z czystym sumieniem nazwać kreatywną.

W tym momencie należy stwierdzić fakt - nadmierna twórczość w pracy programisty - jak każda skrajność, nie jest zjawiskiem mile widzianym. Wyobraźmy sobie sytuację, w której pewien programista z zespołu postanawia przeciwdziałać swojemu wypaleniu zawodowemu i puszcza wodze fantazji podczas implementacji jednej z funkcjonalności. *Reviewer* [4], któremu przypisane zostało zadanie weryfikacji jakości pracy wyżej wymienionego szaleńca może doznać szoku widząc egzotycznie sformatowane linijki kodu źródłowego przeplatane z jeszcze bardziej ekscentrycznymi funkcjami. Przykład ten pomimo oczywistego wyolbrzymienia problemu dobrze obrazuje sytuacje, w której łamanie przyjętych konwencji i reguł jest zjawiskiem niepożądanym i spowalniającym pracę nad projektem dla całego zespołu.

W kontraście do powyższego przykładu - drobne, niespotykane dotychczas rozwiązania, które w dodatku okazują się działać lepiej od konwencjonalnych metod są postrzegane przez programistów jako osobiste trofea i nierzadko nagradzane uznaniem ze strony współpracowników. Zwykle wynikają ze znalezienia niecodziennego zastosowania dla powszechnie znanych wzorców, które w danym kontekście na pierwszy rzut oka wydają się źle umiejscowione. Dopiero odpowiednio opakowane i skonfigurowane pod potrzeby rozwiązywanego zadania ukazują swoje unikalne właściwości i powalają swoje konwencjonalne odpowiedniki. Cechą kluczową programisty, która umożliwia mu wykorzystywanie pospolitych elementów języka do budowy niestandardowych rozwiązań jest właśnie szeroko pojęta twórczość.

Podsumowując: praca programisty jest z natury kreatywna i choć często niedostrzegana, stanowi istotny element jego procesu twórczego. Jest cechą odróżniającą programistów wybitnych od tych przeciętnych. Oczywiście, jej znaczna część przychodzi z czasem (tak jak intuicja) a płynne posługiwanie się nią w codziennej pracy niesie za sobą szereg korzyści.

4 Podsumowanie

Rozważania ujęte w niniejszej pracy przybliżają nieco problem roli, jaką odgrywają twórczość i intuicja w życiu codziennym inżyniera oprogramowania. Na podstawie przytoczonych argumentów można stwierdzić, że nie tylko są one obecne w jego życiu ale stanowią jego integralną część. Obydwie te cechy pozwalają programiście działać bardziej optymalnie, znajdować ścieżki i rozwiązania w miejscach, w których z pozoru ich nie ma. Intuicja narzuca kompozycję i służy jako dobra wróżka przy podejmowaniu mniejszych i większych decyzji w procesie wytwarzania oprogramowania. Twórczość z kolei pozwala spojrzeć na przedstawione problemy w sposób niekonwencjonalny - przez pryzmat artysty, który chciałby przyozdobić bezbarwne płótno tak, aby było odbiciem lustrzanym duszy i intencji jego autora. Obydwie te cechy współgrają ze sobą i dopełniają się, napędzając tym samym ciągły rozwój targanego nimi programisty.

Bibliografia

[1] Michał Nosowski. *Autorskie koszty uzyskania przychodu i praca programisty.* `https://www.wsroddanych.pl/post/autorskie-koszty-uzyskania-przychodu-i-praca-programisty`. 2022.

[2] SA Rogers. *Art Remix: 26 Modern Takes on Famous Historical Paintings.* `https://weburbanist.com/2011/11/07/art-remix-26-modern-takes-on-famous-paintings/`.

[3] *Wikipedia - Intuicja.* `https://pl.wikipedia.org/wiki/Intuicja`.

4. W środowisku informatycznym często przed wypuszczeniem nowego kodu w świat przypisuje mu się kilka osób - *reviewerów*, których zadaniem jest wyłapanie błędów i kontrolę jakości otrzymanego tekstu