Michał Bartyzel

one press

Getting Things Programmed

Droga do efektywności

Helion

Redaktor prowadzący: Magdalena Dragon-Philipczyk
Projekt okładki: Jan Paluch

Ilustracje w książce oraz na okładce: Maria Pankowska

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *onepress@onepress.pl*
WWW: *http://onepress.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://onepress.pl/user/opinie/droppp
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-8889-0

Printed in Poland.

Spis treści

Rozdział 3.

Burze mózgu

Przyjdzie taki moment, gdy z wielu tematów znajdujących się w Twoim rejestrze wybierzesz jeden do realizacji. W tym rozdziale dowiesz się, w jaki sposób zebrać różne pomysły na zrealizowanie danego przedsięwzięcia i przyjrzeć się im. Zapoznasz się z technikami służącymi do przeprowadzania indywidualnej oraz grupowej burzy mózgów.

Ćwiczenia zamieszczone na końcu rozdziału umożliwią Ci przećwiczenie zaprezentowanych tu treści. Mówi się, że najlepiej kuć żelazo, póki gorące. Pamiętaj zatem, że najkorzystniejsze dla Twojego procesu nauki będzie wykonanie ćwiczeń zaraz po przeczytaniu tego rozdziału. Zanim rozpoczniesz kolejny.

Nie wiem, od czego zacząć

Być może często powtarzasz to zdanie, gdy masz do czynienia z jakimś złożonym tematem. Bardzo często ta autodiagnoza, że „nie wiesz, od czego zacząć", oznacza, że właśnie wpadłeś w pewien specyficzny nastrój, w którym nie sposób podjąć jakiejkolwiek decyzji związanej z tematem, nad którym pracujesz. Niektórzy programiści mawiają: „zablokowałem się". To takie dziwne uczucie niemocy.

Zanim zaproponuję Ci, co robić w takiej sytuacji, najpierw chciałbym napisać, czego nie robić.

Nie szukaj lepszego narzędzia wspomagającego Cię w pracy. Narzędzia i formalizmy ograniczają kreatywność (przynajmniej na początku pracy). Zaprzęgając do pracy nowe narzędzie, będziesz poszukiwał rozwiązania problemu nie tam, gdzie on powstał. Brak narzędzia rzadko kiedy jest przyczyną braku pomysłów na wykonanie zadania.

Nie próbuj porządkować pomysłów. Na przykład przygotowując od razu plan działań. Gdy zastanawiasz się nad nowym zagadnieniem, struktury myśli powstających w Twoim mózgu są bardzo nietrwałe (Csíkszentmihályi, 2005). Pomysły przychodzą i odchodzą, przekształcają się. Gdy na tym etapie zaczniesz zajmować się przygotowywaniem planu lub listy działań, możliwe, że stracisz wartościowe idee.

Najprostsza rada

Zrób cokolwiek. Piszę to całkowicie poważnie. Podejmij jakiekolwiek działanie zmierzające w kierunku realizacji danego przedsięwzięcia. I nie ma żadnego znaczenia, co konkretnie zrobisz. Zrób cokolwiek.

Im dłużej się zastanawiasz nad niedoprecyzowanym pomysłem, tym bardziej w procesie myślenia komplikujesz to zagadnienie. Jest tak dlatego, że niedoprecyzowane pomysły zawierają w sobie wiele potencjalnych możliwości rozwiązania i wiele ścieżek, którymi można podążyć. Im dłużej się zastanawiasz, tym więcej pojawia się pytań i dylematów. W końcu zrezygnowany stwierdzasz, że po kilku godzinach spędzonych nad tematem wiesz o wiele mniej niż na początku.

Podjęcie jakiegokolwiek działania redukuje liczbę możliwości, które jednocześnie rozważasz. Skupią Twoją uwagę na jednym konkretnym aspekcie zagadnienia.

Przypuśćmy, że masz do napisania transformacje kopiujące dane pomiędzy różnymi schematami danych i pomimo zapoznania się ze skryptami i dokumentacją nie masz żadnego wyobrażenia, jak należy to zrobić. Napisz małą procedurę kopiującą jedną kolumnę. To proste zadanie spowoduje, że zadasz sobie następujące pytania:

- *Jakie są adresy baz?*
- *Jakie są nazwy schematów i tabel?*
- *Jaką metodą połączyć się z bazą źródłową?*
- *Jaki ma być nowy typ danych z tej kolumny?*
- *Jakie są zasady konwersji typów pomiędzy danymi?*
- ...

Znacznie łatwiej czyta się dokumentację, jeśli chcesz znaleźć w niej odpowiedzi na konkretne pytania niż w celu ogólnego „zapoznania się" z nią.

Dopiero z tymi pytaniami możesz zacząć przyglądać się dokumentacji. Znacznie łatwiej się ją czyta, gdy chcesz znaleźć w niej odpowiedzi na konkretne pytania niż w celu ogólnego „zapoznania się". Po kilku takich krótkich zadaniach będziesz miał już odpowiednią wiedzę na dany temat, aby zająć się nim w całej złożoności bez poczucia, że nie wiesz, od czego zacząć.

Najpierw zwizualizuj swoje pomysły

Wychodząc z przytoczonego wcześniej spostrzeżenia, że struktury myśli mózgu są bardzo nietrwałe (Csíkszentmihályi, 2005), można sformułować praktyczną zasadę postępowania. Brzmi ona następująco: najpierw zwizualizuj swoje pomysły (spisując je albo rysując ich wyobrażenie), oceń je, a dopiero w drugiej kolejności pracuj nad szczegółami i planem działania.

Najpierw zwizualizuj swoje pomysły, oceń je, a dopiero w drugiej kolejności pracuj nad szczegółami i planem działania.

Brzmi to jak oczywistość. Echa tej oczywistości możesz odnaleźć w zaleceniach na temat tworzenia czystego kodu (McConnell, 2004; Beck, 2008), gdzie jest mowa o napisaniu kroków algorytmu na kartce, zanim zaczniesz kodować, albo w metodzie programowania sterowanego testami (ang. *test-driven development*), gdzie spodziewany rezultat kodu zapisujemy w postaci testu jeszcze przed rozpoczęciem implementacji[1].

Możliwość zobaczenia swoich pomysłów da Ci szansę na zdystansowanie się od nich i przyjrzenie się im w szerszej perspektywie.

Zapisz 30 pomysłów

Bardzo prostą techniką indywidualnej burzy mózgu jest zapisanie większej ilości pomysłów na kartce. Kroki są następujące:

1. Na górze kartki zapisz temat burzy mózgu w postaci pytania, na które starasz się znaleźć odpowiedź.
2. Zacznij wypisywać pod spodem wszystkie odpowiedzi, które będą Ci przychodzić do głowy.
3. Nie oceniaj pomysłów. Jedynie zapisuj po kolei jeden pod drugim.
4. Pisz w równym tempie w sposób ciągły, to znaczy bez zastanawiania się, co masz napisać. Pisanie powinno być płynne.

[1] W zasadzie zapisujemy pewne hipotezy co do oczekiwanej implementacji, a następnie weryfikujemy je za pomocą testu. Dlatego jest to programowanie STEROWANE testami. W tym momencie to rozróżnienie nie jest istotne.

Rozdział 6.

Planowanie pracy

Czy zdarza Ci się czasem myśleć, że mijający dzień „przeciekł przez palce"? Czy miewasz poczucie, że ostatnie osiem godzin intensywnie pracowałeś, ale trudno Ci określić efekty swojej pracy? Jeśli na co najmniej jedno z tych pytań odpowiedziałeś twierdząco, to ten rozdział jest właśnie dla Ciebie.

Poznasz tu metody na wyznaczanie priorytetów, organizowanie i planowanie swojej pracy. Natomiast dzięki zamieszczonym na końcu ćwiczeniom od razu sprawdzisz działanie opisanych technik w praktyce.

Czym zajmujesz się w ciągu dnia?

W trakcie jednego z badań ankietowych przeprowadzonych na próbie 246 programistów[1] zapytaliśmy, ile (w ich subiektywnym odczuciu) czasu poświęcają oni na poszczególne czynności w ciągu dnia pracy. Wybrane odpowiedzi były następujące:

[1] 202 osoby z 24 dużych firm, 18 osób z 6 średnich firm, 24 osoby z 8 małych firm z branż: finanse, telekomunikacja, media, dostawcy. Więcej informacji: *http://www.slideshare.net/BNSIT/strategie-sukcesu-zespow-it-spotkanie-strefy-liderw-it*.

- Kodowanie — 31,2% czasu.
- Testowanie — 12,6% czasu.
- Debugowanie — 7,3% czasu.
- Poszukiwanie błędów — 6,5% czasu.
- Czytanie e-maili i odpowiadanie na nie — 6,5% czasu.
- Tworzenie dokumentacji — 5,2% czasu.
- Czytanie dokumentacji — 4,5% czasu.
- Przeglądanie ciekawych stron internetowych — 2,6% czasu.

Oznacza to, że przyjmując idealistyczne założenie ośmiogodzinnego dnia pracy, uśredniony ankietowany programista na programowanie poświęca dwie i pół godziny[2].

Celem technik przedstawionych w tym rozdziale **nie jest** zwiększenie czasu przeznaczonego na programowanie, gdyż byłoby to manipulowanie przy produktywności, a nam chodzi przede wszystkim o efektywność. Celem technik jest takie ułożenie Twojego dnia, aby praca wykonywana była maksymalnie efektywnie.

O pilności, ważności i macierzy Eisenhowera

„Nie wszystko, co jest pilne, jest ważne, i nie wszystko, co ważne, jest pilne" — miał powiedzieć Dwight Eisenhower, generał i późniejszy prezydent Stanów Zjednoczonych. Stephen Covey na tej podstawie opracował metodę analizy priorytetów (Covey, 2005). Obrazuje ją tabela 6.1.

[2] Pamiętaj, że jest to subiektywne poczucie ankietowanych programistów.

TABELA 6.1. Analiza priorytetów wg Coveya

Ważne	Planuj • Zarządzaj • Zajmij się, zanim staną się pilne	Zajmij się natychmiast • Najlepiej osobiście • Jest to zadanie typu „pożar”
Nieważne	Unikaj • Ponieważ mają niewielką wartość	Deleguj • Rutynowe zadania
	Niepilne	Pilne

Pierwsze pytanie, które nasuwa się na myśl, brzmi: „Co jest pilne, a co ważne?”. Pilność jest związana z terminem wykonania zadania — im bliższy, tym większa jest pilność. Ważność mówi o potencjalnych konsekwencjach wykonania lub zaniechania wykonywania zadania. Im większe konsekwencje (uzyskane korzyści i rozwiązane problemy wskutek wykonania zadania lub utracone korzyści i powstałe problemy wskutek zaniechania wykonywania), tym ważność jest większa.

> Pilność jest związana z terminem wykonania, ważność z konsekwencjami wykonania lub zaniechania wykonywania zadania.

Kluczowym odkryciem Coveya było to, że sprawy pilne „wypychają” z naszego dnia pracy sprawy ważne. Weźmy na przykład większą refaktoryzację typu wprowadzenie warstwy lub korekta architektury. Długoterminowymi (wewnątrzzespołowymi) konsekwencjami zaniedbywania refaktoryzacji są:

- Niska jakość kodu.
- Trudne testowanie.
- Utrudnione poszukiwanie błędów.
- Słaba wymiana wiedzy.
- Wydłużający się czas wprowadzania zmiany lub dodawania funkcjonalności.

- Zwiększona ilość poważnych błędów na produkcji.
- Pogorszenie wydajności.

Z drugiej strony refaktoryzacja nie jest pilna. Zazwyczaj można zastosować rozwiązanie „na szybko”[3], które sprawi, że oprogramowanie będzie działać. Do pewnego momentu to podejście będzie się sprawdzać, ponieważ problemy wynikające z braku refaktoryzacji będą objawiały się przede wszystkim na poziomie technologicznym i nie będą wykraczały poza zespół.

Jednak w pewnym momencie kumulatywne zaniedbania będą objawiać się już w sposób zauważalny przez użytkowników i klienta. Sprawi to, że nagle pojawi się informacja dla zespołu: „zróbcie coś z tym”. W ten sposób „refaktoryzacja” z zadania ważnego stanie się jednocześnie zadaniem pilnym.

Niestety niezwykle trudno jest poradzić sobie z zadaniami jednocześnie bardzo pilnymi i bardzo ważnymi. W przypadku refaktoryzacji zdarzają się sytuacje, gdy taniej jest zainwestować w nowy system, niż porządkować istniejący.

Jaką podpowiedź co do refaktoryzacji daje tabela 6.1? Mówi: zarządzaj, planuj, to znaczy: zajmij się zadaniami ważnymi, zanim staną się pilne. Może to na przykład oznaczać:

- Spraw, aby drobne refaktoryzacje były częścią pracy programisty[4].
- Rozwijaj i wdrażaj standardy tworzenia czystego i czytelnego kodu.
- W każdej iteracji zarezerwuj czas na istotne poważniejsze refaktoryzacje.

[3] W postaci *if*-a i //FIXME.

[4] Możesz o tym przeczytać w artykule *http://www.infoq.com/articles/natural-course-refactoring*.

Zacznij od zadania, którego unikasz

Bywają zadania, do których jakoś trudno Ci się zabrać. Przesuwasz je wtedy na koniec swojej listy, na następny dzień, na przyszły tydzień. Wtedy zaczyna kształtować się bardzo niebezpieczny nawyk — „odkładanie na później”[4].

Pierwsze, co warto w tej sytuacji zrobić, to przeanalizować zadania na zgodność z kryteriami zadania możliwego do wykonania (patrz rozdział 4. „Wyodrębnianie zadań”). Gdy jednak to nie pomaga, doskonałą radę daje Brian Tracy (Tracy, 2007) — zacznij dzień od tego właśnie zadania.

Trudno o bardziej demotywującą strategię działania niż czekanie na odpowiedni moment, odpowiednią porę dnia, odpowiedni nastrój albo wenę. Wtedy przez cały dzień nie możesz pozbyć się myśli, że nie zacząłeś jeszcze zasadniczego zadania i że cały czas się do niego przymierzasz. Przejmij kontrolę nad tymi myślami i zacznij wykonywać to zadanie od razu. Odpowiedni nastrój i wena pojawią się już za chwilę.

10-minutowe wyzwanie

Być może wydaje Ci się, że aby wykonać jakieś zadanie, potrzebna jest odpowiednia motywacja. Jest to połowa prawdy. Motywacja jest przyczyną Twojego działania, lecz jest również jego skutkiem. Oznacza to, że jeśli czujesz się zmotywowany do wykonania jakiegoś zadania, to zaczynasz chętnie nad nim pracować. Jeśli pozostaniemy tylko przy takim modelu motywacji, to logicznym wnioskiem jest, że skoro nie

[4] Nie piszę „prokrastynacja”, która jest utożsamiana z odkładaniem spraw na później. Prokrastynacja to skłonność patologiczna do odsuwania w czasie momentu podjęcia działania i jest to zaburzenie psychiczne. Tutaj mówimy o łagodniejszym, ale także uciążliwym nawyku przekładania spraw na później.

czujesz się zmotywowany, to nie ma sensu zabierać się do zadania. Lepiej poczekać, aż motywacja spłynie na Ciebie w jakiś szczególny sposób.

Na szczęście jest jeszcze druga strona medalu. Gdy w daną czynność zaczynasz inwestować zogniskowany wysiłek, gdy zaczynasz dostrzegać małe rezultaty swojej pracy, pojawia się motywacja albo, precyzyjniej, uczucie zmotywowania (rysunek 7.2).

RYSUNEK 7.2. Motywacja to zarówno przyczyna, jak i skutek działania

Ten efekt jest esencją techniki 10-minutowego wyzwania. Jeśli masz trudność z rozpoczęciem zadania, postanów, że przez najbliższe dziesięć minut będziesz zajmować się tylko tym zadaniem z największym zaangażowaniem, na jakie Cię stać. Tylko dziesięć minut, potem możesz znów odłożyć to zadanie na bok, jeśli tylko będziesz miał na to ochotę.

Trik polega na tym, że dziesięć minut „udawanego" zaangażowania to wystarczający czas, aby pojawiła się „prawdziwa" motywacja.

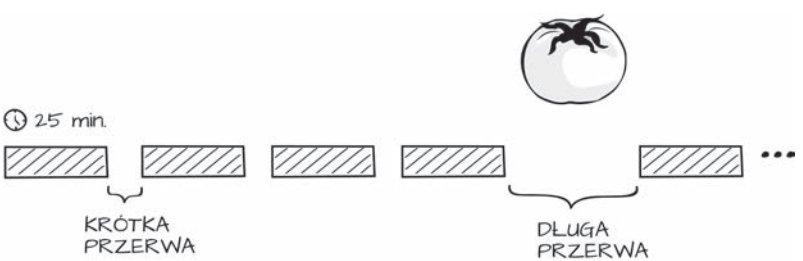
> Motywacja może pojawiać się dopiero PO rozpoczęciu prac nad zadaniem. Jeśli brak Ci motywacji do pracy, zacznij pracować.

Pomodoro Technique®

Pomodoro (Nöteberg, 2009) to gra, dzięki której uczysz się kontrolować wewnętrzne i zewnętrzne rozpraszacze. Celem tej gry jest zdobycie jak najwięcej interwałów czasu, w których pracujesz nad wybranym zadaniem w pełnym skupieniu.

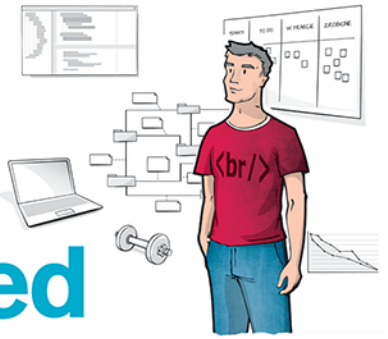
Podstawowe zasady (rysunek 7.3) gry są następujące:

- Pracujesz w skupieniu nad jednym zadaniem przez 25 minut.
- Po 25 minutach następuje 5 minut przerwy.
- Po każdych czterech 25-minutowych cyklach następuje 15 minut przerwy.
- W ciągu 25-minutowego czasu pracy:
  - Dozwolona jest tylko praca.
  - Jeśli pojawia się rozpraszacz wewnętrzny lub zewnętrzny, należy zapisać to w rejestrze i wrócić do pracy.
  - Jeśli po pojawieniu się rozpraszacza zrobisz coś więcej niż wyłącznie zapisanie w rejestrze, to ten interwał uważa się za niezaliczony.
  - Oznaczaj pojawiające się rozpraszacze wewnętrzne i zewnętrzne.
- Celem gry jest zebranie jak największej ilości 25-minutowych interwałów w ciągu dnia.
- Jeśli zakończysz zadanie przed upływem interwału, nie wolno rozpoczynać nowego zadania. Musisz kontynuować pracę nad bieżącym zadaniem i na przykład dopracowywać szczegóły. Celem jest tu nabycie umiejętności dzielenia pracy na 25-minutowe odcinki czasu.



RYSUNEK 7.3. Pomodoro — zasady gry

Getting Things Programmed



Na pracę programisty składa się wiele zadań. Nawet jeżeli 90% czasu spędzamy na „klepaniu w klawiaturę", to w trakcie tych działań wykonujemy całą gamę najróżniejszych czynności.

Sprawne sterowanie swoją uwagą, planowanie zadań czy ich oszacowanie to umiejętności, których brakuje większości programistów. Często nie zdajemy sobie nawet sprawy z istnienia problemów spowalniających naszą pracę.

W niniejszej książce autor przedstawia liczne wyzwania stojące przed świadomym programistą. Nie poprzestaje na tym: wysuwa propozycje samodoskonalenia. Opisuje sprawdzone recepty pomagające zrozumieć codzienne problemy, rozbija je na czynniki pierwsze i przygotowuje do walki o lepszą organizację czasu.

Maciej Aniserowicz

programista i szkoleniowiec, twórca bloga devstyle.pl oraz podcasta devtalk.pl

Michał Bartyzel — coach i trener w firmie szkoleniowo-doradczej BNS IT. Zajmuje się doskonaleniem programistów i zespołów programistycznych, wdrażaniem metodyk pracy oraz rozwijaniem kompetencji pracowników branży IT. Prowadzi szkolenia oraz konsultacje z zakresu zwiększania efektywności zespołów programistycznych, usprawniania współpracy z klientami oraz tworzenia użytecznego oprogramowania. Autor książki *Oprogramowanie szyte na miarę. Jak rozmawiać z klientem, który nie wie, czego chce.* (Helion, 2012, 2015).

one press

Sprawdź najnowsze promocje:
http://onepress.pl/promocje
Książki najchętniej czytane:
http://onepress.pl/bestsellery
Zamów informacje o nowościach:
http://onepress.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: onepress@onepress.pl
http://onepress.pl

Cena 34,90 zł

ISBN 978-83-246-8889-0

9 788324 688890