Praktyka czyni mistrza

Wzorce, inspiracje i praktyki rzemieślników programowania

Dave H. Hoover & Adewale Oshineye

Przedmowa: Ward Cunningham

O'REILLY®

Helion

Tytuł oryginału: Apprenticeship Patterns: Guidance for the Aspiring Software Craftsman
Tłumaczenie: Lech Lachowski
ISBN: 978-83-283-3521-9

© 2017 Helion S.A.

Authorized Polish translation of the English edition of Apprenticeship Patterns, ISBN 9780596518387 © 2010 David H. Hoover and Adewale Oshineye

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje yły kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

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

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

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

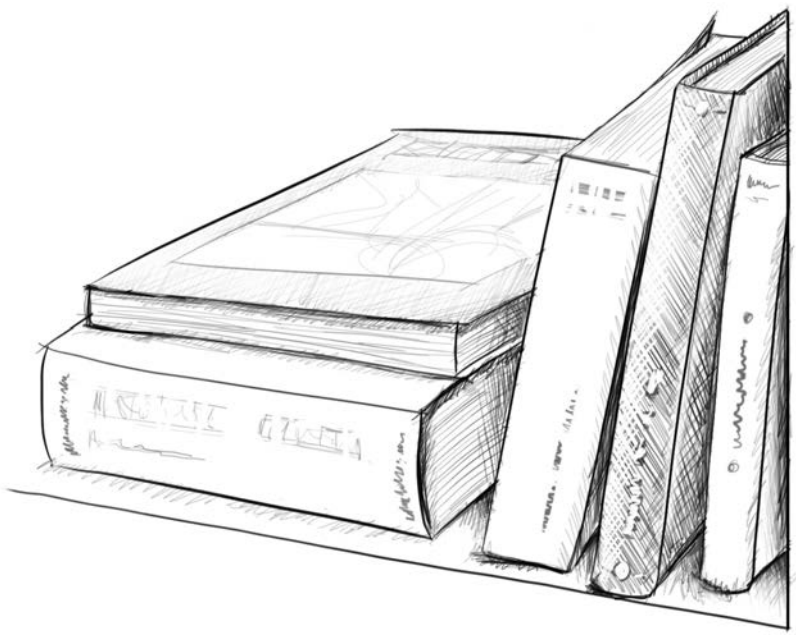
Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę



ROZDZIAŁ 6.

Skonstruuj swój program uczenia się

Nie będzie już osobą motywowaną ocenami. Będzie osobą motywowaną wiedzą. Nie będzie potrzebował żadnego zewnętrznego bodźca do nauki. Jego bodźce będą wypływać z wnętrza… Gdy tego rodzaju motywacja raz zostanie zaszczepiona, jest okrutną siłą.

— Robert Pirsig, *Zen i sztuka obsługi motocykla*

Kup książkę Poleć książkę

Żyjemy w erze obfitości informacji. Wynalezienie prasy drukarskiej zapoczątkowało epokę, która umożliwia nawet najuboższym członkom społeczeństwa zdobywanie wiedzy, a tym samym mocy do zmiany własnej sytuacji. Stale rozszerzająca się sieć WWW i niekończąca się seria innowacji technicznych dalej obniżają bariery dostępu do praktycznie każdy informacji, jakiej moglibyśmy kiedykolwiek potrzebować. Ponieważ zwiększa się przepustowość łączy internetowych, a urządzenia przenośne zdają się przechowywać nieograniczone ilości danych, możemy w dowolnym miejscu i czasie uzyskać dostęp do multimediów wysokiej rozdzielczości w formatach tekstowym, dźwiękowym i wideo. Jak każdy dobry uczeń, prawdopodobnie będziesz korzystać z najnowszych i najlepszych urządzeń i platform medialnych, ale niektóre informacje można znaleźć głównie w zwykłych starych książkach. Choć blogi mogą zapewnić doskonały strumień materiałów do czytania, rozległych mądrości zawartych w księgach doświadczonych praktyków, takich jak Jerry Weinberg, Fred Brooks, Steve McConnell i Kent Beck, nie można zastąpić, nawet wyższą przepustowością dostarczania informacji. Nawet jeśli nie jesteś molem książkowym, udana nauka wymaga przeczytania kilku książek i poświęcenia czasu na studiowanie. Nie jesteś jednak w szkole. Nie ma lektur obowiązkowych — to do Ciebie należy znalezienie rekomendacji i stworzenie własnego programu uczenia się.

Lista Lektur

Nikt nie może nauczyć się wszystkiego naraz, ale żadna zasada lub reguła nie zabrania uczniowi uczyć się jednego dnia trochę tego, drugiego trochę tamtego, wybierać zagadnienia w kolejności, o jakiej nikt wcześniej nie pomyślał, lub uczyć się do momentu, kiedy będzie chciał się zatrzymać i przejść do czegoś innego. Gdy będzie chciał nauczyć się określonej procedury, uczeń nie musi czekać, aż nadejdzie odpowiedni czas we wcześniej ustalonym harmonogramie. Nie musi też uczyć się czegoś, na co nie jest gotowy, co uważa za nieciekawe, przerażające lub niepotrzebne. Uczeń tworzy własny program.

— Howard S. Becker, *A School Is a Lousy Place to Learn Anything In*

Kontekst

Gdy rozwiniesz wystarczające kompetencje i umiejętności, aby stać się biegłym w Twoim pierwszym języku, zaczynasz rozglądać się dookoła i dostrzegać ogromną ilość informacji, które nadal musisz przyswajać.

Problem

Liczba książek, które musisz przeczytać, rośnie szybciej, niż jesteś w stanie je czytać.

Kup książkę Poleć książkę

Rozwiązanie

Prowadź listę lektur, żeby śledzić książki, które planujesz przeczytać, i pamiętać o książkach już przeczytanych.

Zgodnie z duchem wzorca Dziel Się Tym, Czego Się Uczysz, rozważ zamieszczenie Twojej listy w przestrzeni publicznej. Pozwoli to innym ludziom korzystać z rzeczy, których się uczysz. My używamy wiki dostępnej pod adresem *http://bookshelved.org* (uruchomionej przez Laurenta Bossavita w 2002 r.), ale równie dobrze sprawdzi się każda lista publiczna. Najlepiej byłoby, żeby Twoja lista pozwalała Ci układać książki w odpowiedniej kolejności z zaznaczaniem, które książki przeczytałeś i kiedy.

W tym wzorcu nie chodzi tylko o zarządzanie książkami, które planujesz przeczytać. Jest to również mechanizm refleksji nad przeszłymi nawykami czytelniczymi. Posiadając dane obejmujące kilka lat, możesz zacząć dostrzegać wzorce, trendy i luki w tym, czego decydujesz się uczyć. Może Ci to pomóc podejmować lepsze decyzje w kwestii wyboru następnej książki. Jeśli upublicznisz te informacje, prawdopodobnie inne osoby będą dopisywać własne sugestie dotyczące przyszłych lektur. Możliwe, że dzięki temu odkryjesz ukryte powiązania i mało znane perełki.

Jedną z najcenniejszych rzeczy, jakie można wynieść z każdej książki, jest lista innych książek, które są warte przeczytania. Z biegiem czasu przekonasz się, że niektóre książki wciąż pojawiają się w bibliografiach, i powinieneś przesunąć te książki na samą górę listy lektur. Inne książki spadną w dół. Ponieważ lista lektur jest w rzeczywistości kolejką priorytetów, w końcu uświadomisz sobie, że pewne książki spadły w rankingu tak bardzo, iż prawdopodobnie nigdy ich nie przeczytasz. Nie ma problemu. Celem tego wzorca jest zapewnić Ci sposób priorytetyzowania i filtrowania zalewu potencjalnej wiedzy.

Główną trudnością w implementacji tego wzorca jest to, że potrzebujesz dobrego zrozumienia tematu, aby zdecydować, które książki przeczytać i w jakiej kolejności. Jednym ze sposobów uniknięcia tego paradoksu jest wybranie najpierw tych książek, które dają szeroką wiedzę na dany temat, a następnie wybranie książek, które drążą konkretne interesujące Cię aspekty. Innym sposobem na uniknięcie tego paradoksu jest poleganie na bratnich duszach i swoich mentorach. Twoi mentorzy będą w stanie polecić Ci książki warte przeczytania, a dyskusja z kolegami uczniami może pomóc Ci wypracować kolejność, w jakiej należy je czytać. Możesz również wykorzystać publiczne listy lektur dostarczone przez innych ludzi, którzy wdrażają ten wzorzec.

Inna trudność polega na zdecydowaniu, od czego zacząć. Doskonałą listę książek, która zapełni Twoją listę lektur, możesz znaleźć w rozdziale 35. książki *Kod doskonały. Jak tworzyć oprogramowanie pozbawione błędów* i w bibliografii książki *Pragmatyczny programista. Od czeladnika do mistrza*. Możesz również zapoznać się z bibliografią do tej książki, aby zobaczyć, co nas zainspirowało.

Ten wzorzec wiele zawdzięcza idei wzorców Łańcuch Książek[1] (ang. *Book Chain*) Raviego Mohana i Studiowanie Sekwencyjne (ang. *Sequential Study*) z pracy *Pattern Language for Study Groups* Joshuy Kerievsky'ego[2]. Podczas gdy Łańcuch Książek dotyczy proszenia ludzi o polecenie zestawu

[1] *http://ravimohan.blogspot.com/2005/08/apprenticeship-pattern.html.*

[2] *http://www.industriallogic.com/papers/khdraft.pdf.*

Kup książkę Poleć książkę

książek, które wprowadzą Cię do nowego tematu, w tym wzorcu chodzi raczej o zarządzanie ciągłym strumieniem książek, które wydają Ci się interesujące. Ten wzorzec różni się także od wzorca Studiowanie Sekwencyjne, ponieważ nie koncentruje się na czytaniu książek w kolejności chronologicznej w celu zrozumienia, w jaki sposób wpływają one na siebie. W tym wzorcu książką, którą powinieneś przeczytać w następnej kolejności, jest książka, która zabiera Cię krok dalej w Twojej podróży.

Powinieneś pamiętać, że jest to *Twoja* lista lektur. Wygodnie jest kierować się sugestiami innych osób, ale tylko Ty naprawdę znasz swój aktualny kontekst. Dlatego sam powinieneś dokonywać wyborów kolejnych lektur do studiowania. Ważne jest jednak również czytanie właściwej książki w odpowiednim czasie. Takie postępowanie jest o wiele wydajniejsze niż przebijanie się przez różne książki, których nie możesz do końca zrozumieć z uwagi na brak doświadczenia lub pogłębionej wiedzy. Zbyt wiele osób czyta książkę *Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku* na zbyt wczesnym etapie swoich studiów, podczas gdy znacznie łagodniejszym wprowadzeniem do wzorców byłaby książka *Refaktoryzacja. Ulepszanie struktury istniejącego kodu*. Znajdź Mentorów i poproś ich o poradę, którą książkę powinieneś przeczytać w następnej kolejności. Właściwe wyczucie czasu ma potężny wpływ na doświadczenie z książką.

Działanie

Utwórz plik tekstowy i jeśli chcesz, zastosuj dla niego system kontroli wersji. Zapisz w nim wszystkie książki, które aktualnie czytasz. To jest Twoja Lista Lektur i najprostsza możliwa implementacja tego wzorca. Teraz musisz jedynie na bieżąco aktualizować ten plik.

Zobacz również

Znajdź Mentorów (rozdział 4.), Bratnie Dusze (rozdział 4.), Dziel Się Tym, Czego Się Uczysz (rozdział 5.) i Twój Pierwszy Język (rozdział 2.).

Nieustannie Czytaj

Jeśli będziesz czytał chociaż jedną dobrą książkę o programowaniu co dwa miesiące, około 35 stron tygodniowo, wkrótce będziesz miał całkiem dobre pojęcie o tej branży i wyróżnisz się z prawie wszystkich otaczających Cię osób.

— Steve McConnell, *Kod doskonały. Jak tworzyć oprogramowanie pozbawione błędów*

Kontekst

Uwolniłeś swój entuzjazm, aby otworzyć mnóstwo drzwi.

Kup książkę Poleć książkę

Problem

Wydaje się, że omija Cię nieskończony strumień głębszych i bardziej fundamentalnych koncepcji mimo Twojej biegłości w pierwszym języku.

Rozwiązanie

Skup swoje pragnienie uczenia się na pochłanianiu możliwie najwięcej słowa pisanego. Przy tworzeniu listy lektur przedkładaj książki nad blogi.

Powinny istnieć takie okresy długiej drogi, kiedy będziesz miał (lub wykorzystasz) okazję do czytania znacznej liczby książek. Dla Dave'a były to lata 2002 – 2003, kilka lat po tym, jak zaczął programować, i w momencie, kiedy właśnie zaczął osiągać stały poziom w swoim pierwszym języku, czyli w Perlu. Ten okres intensywnego czytania był możliwy dzięki transportowi publicznemu: Dave miał około 90 minut dziennie podczas jazdy pociągiem, żeby czytać, co chciał. Był tak zdeterminowany, że nie przestawał czytać po wyjściu z pociągu, kiedy szedł ponad kilometr do pracy. Zanurzenie się w klasyce i pierwotnych źródłach danej dziedziny zapewnia niezrównaną edukację w połączeniu ze znalezieniem mentorów i częstymi interakcjami z bratnimi duszami.

Część tego zanurzania się w klasyce i źródłach powinno obejmować odkrywanie ogromnego magazynu wiedzy, jakim jest społeczność akademicka. Czytanie okolicznościowych opracowań będzie rozciągnąć Twój umysł i pozwoli Ci pozostać w kontakcie z awangardą informatyki, a także zapewni Ci źródło nowych ambitnych pomysłów. Próby implementowania tych pomysłów poszerzą Twój zestaw narzędzi o nowe algorytmy, struktury danych i wzorce projektowe na wiele lat przed tym, zanim dotrą one do głównego nurtu.

Działanie

Czytając tę książkę, już zacząłeś stosować ten wzorzec. Sztuka polega na tym, aby utrzymywać pęd po skończeniu tej książki. Zdecyduj już teraz, jaka będzie Twoja następna książka. Kup ją lub wypożycz, żebyś po zakończeniu tej lektury mógł od razu przejść do następnej.

Powinieneś także próbować cały czas nosić ze sobą jakąś niegrubą książkę. Pozwoli Ci to wykorzystać na naukę krótkie chwile jałowego czasu w ciągu każdego dnia (takie jak przejazdy kolejowe lub czekanie w kolejkach).

Zobacz również

Znajdź Mentorów (rozdział 4.), Bratnie Dusze (rozdział 4.), Lista Lektur (rozdział 6.), Długa Droga (rozdział 3.), Uwolnij Swój Entuzjazm (rozdział 2.) i Twój Pierwszy Język (rozdział 2.).

Kup książkę Poleć książkę

Studiuj Klasykę

Odkryj wielką literaturę związaną z Twoim zawodem lub obszarem zainteresowań — najwspanialsze książki, artykuły i przemówienia, jakie kiedykolwiek zostały napisane — a potem zacznij usilnie studiować te prace.

— Joshua Kerievsky w *Knowledge Hydrant: A Pattern Language for Study Groups*[3]

Kontekst

Jesteś samoukiem albo odebrałeś bardzo praktyczne wykształcenie, oparte bardziej na szkoleniu umiejętności niż teorii.

Problem

Doświadczone osoby, z którymi współpracujesz, stale odwołują się do pojęć takich jak prawo Brooksa z książek, które według nich przeczytałeś (tak jak każdy szanujący się programista).

Rozwiązanie

Przyznaj Się Do Niewiedzy i zapytaj o tę nieznaną Ci koncepcję oraz o książkę, z której pochodzi. Dodaj tę książkę do Twojej Listy Lektur.

Joshua Kerievsky zapytał kiedyś Jerry'ego Weinberga, jak nadąża za tymi wszystkimi książkami, które się ukazują. Jerry odpowiedział: „Z łatwością — czytam tylko te dobre" [Kerievsky, s. 33]. Poprzez nieustanne czytanie i bycie refleksyjnym podczas pracy w końcu tak jak Jerry będziesz mógł „czytać tylko te dobre". Kiedy wybierasz jakąś książkę i pierwszą rzeczą, nad którą zaczynasz się zastanawiać, jest to, jak bardzo jest ona nieaktualna, to znaczy, że czytasz niewłaściwy rodzaj książek. Odnoszący sukcesy uczniowie z reguły skupiają się na „książkach o długiej żywotności" i korzystają z internetu lub eksperymentów, aby dowiedzieć się, w jaki sposób dana informacja ewoluowała. Dave dokładnie pamięta doświadczenie z czytania pierwszego klasyka w tej dziedzinie, czyli książki *The Psychology of Computer Programming*, oraz swoje zdziwienie, jak bardzo istotna wydała mu się ta książka pomimo opowieści o kartach perforowanych i komputerach wielkości pokoju. Mądrości ujęte w takich klasykach stanowią niezbędne informacje, które pozwalają Ci zachować właściwy kierunek, krocząc długą drogą.

Jednym z niebezpieczeństw skupiania się na klasyce jest pójście za daleko i porzucenie bardziej pragmatycznej wiedzy i informacji, które pozwalają Ci poprawiać Twoje codzienne rzemiosło. Upewnij się, że na Twojej liście lektur klasyka przeplata się z nowoczesnymi, pragmatycznymi książkami i (lub) artykułami.

[3] *http://www.industriallogic.com/papers/khdraft.pdf.*

Kup książkę Poleć książkę

Działanie

Jaka jest najstarsza książka w Twoim zbiorze? Przeczytaj ją najpierw. Następnym razem, gdy będziesz przeglądać księgozbiór innego programisty, zwróć uwagę na najstarsze książki i spytaj go, dlaczego wciąż je trzyma.

Zobacz również

Przyznaj Się Do Niewiedzy (rozdział 2.), Nieustannie Czytaj (rozdział 6.), Lista Lektur (rozdział 6.), Bądź Refleksyjny Podczas Pracy (rozdział 5.) i Długa Droga (rozdział 3.).

Kop Głębiej

W praktyce problemy z algorytmami nie powstają na początku dużego projektu. Zazwyczaj powstają raczej jako podproblemy, kiedy nagle staje się jasne, że programista nie wie, jak posunąć się do przodu, lub że obecny program jest nieodpowiedni.

— Steven S. Skiena, *The Algorithm Design Manual*

Kontekst

Żyjesz w świecie napiętych terminów i skomplikowanych projektów informatycznych, które wykorzystują wiele narzędzi. Twoi pracodawcy nie mogą sobie pozwolić na luksus zatrudniania wystarczającej liczby specjalistów do wypełnienia każdej roli. O każdym narzędziu uczysz się tylko tyle, żeby wykonać dzisiejsze zadanie. Wybierasz kilka tutoriali na temat języka lub biblioteki, z którymi pracujesz dzisiaj. Podejmujesz decyzje bez poświęcenia czasu na zrozumienie problemów i kopiujesz przykłady zabawek dostarczonych z narzędziem. Sprawdza się to do tego stopnia, że możesz podjąć się czegokolwiek. Nabywasz zdolność bardzo szybkiego poznawania nowej technologii i wymyślania rozwiązania. Zawsze zapoznajesz się tylko z tymi elementami technologii, które są potrzebne, aby uruchomić Twoją część systemu, i polegasz na innych członkach zespołu, którzy uczą się pozostałych części. Możesz być na przykład programistą Javy zajmującym się programowaniem po stronie serwera, a co za tym idzie, mieć niewielką lub żadną wiedzę o tym, w jaki sposób został zbudowany interfejs użytkownika.

Problem

Ciągle napotykasz trudności związane z utrzymywaniem kodu, który napisałeś, ponieważ okazuje się, że wybrane przez Cienie tutoriale idą na łatwiznę i upraszczają złożone zagadnienia. Przekonujesz się, że powierzchowna znajomość tysiąca narzędzi oznacza, że zawsze musisz przedzierać się z trudem, gdy pojawia się jakiś subtelny błąd, lub musisz zrobić coś, co wymaga głębokiej wiedzy. Ludzie często oskarżają Cię o posiadanie wprowadzającego w błąd CV, ponieważ nie odróżniasz kilku tygodniami rozszerzenia istniejącej usługi internetowej od głębokiej wiedzy na temat zagadnień

Kup książkę Poleć książkę

związanych z utrzymaniem interoperacyjnego i wysoce skalowalnego systemu korporacyjnego. Co gorsza, ponieważ Twoja wiedza jest tak powierzchowna, nie jesteś nawet świadomy, jak mało wiesz, dopóki ktoś lub coś nie wystawi Cię na próbę.

Rozwiązanie

Naucz się kopać głębiej w narzędziach, technologiach i technikach. Zdobądź głębię wiedzy do tego stopnia, że zrozumiesz, dlaczego rzeczy są takie, jakie są. Głębokość oznacza zrozumienie sił, które doprowadziły do powstania projektu, a nie tylko szczegółów tego projektu. Oznacza to na przykład zrozumienie teorii typów (lub przynajmniej uproszczenia oferowanego przez kwadrant typowania *http://c2.com/cgi/wiki?TypingQuadrant*) zamiast po prostu powtarzania rzeczy zasłyszanych od innych.

Jeden z naszych byłych kolegów (Ravi Mohan) stwierdził:

> Znajomość różnych form współbieżności (i ich ograniczeń) jest bardziej przydatną wiedzą niż „podklasa `Thread` lub implementacja interfejsu `Runnable`".

Istnieją obszary, w których masz głęboką wiedzę karmiącą Twoją pewność siebie, i poprowadzą Cię one, gdy zdecydujesz, jak zastosować wzorzec Zamiataj Podłogę, ponieważ wskazują miejsca, w których od razu możesz dostarczyć pewną wartość w nowym zespole. Co ważniejsze, pogłębiona wiedza jest czymś, do czego możesz się wycofać, aby nabrać sił do angażowania się w nowe obszary. Możesz zawsze powiedzieć sobie: „Skoro opanowałem ziarna EJB, to poradzę sobie z metaklasami".

Kolejną zaletą kopania głęboko w technologii jest to, że rzeczywiście możesz wyjaśnić, co dzieje się pod powierzchnią systemów, nad którymi pracujesz. W rozmowach kwalifikacyjnych to zrozumienie będzie odróżniać Cię od innych kandydatów, którzy nie potrafią w znaczący sposób opisać oprogramowania, które pomogli zbudować, ponieważ rozumieją tylko jego jedną małą część. Gdy jesteś częścią zespołu, to zastosowanie tego wzorca odróżnia tych, którzy tworzą losowe sterty gruzu (pragmatyczni programiści nazywają to „programowaniem przez przypadek", a Steve McConnell nazywa to „inżynierią oprogramowania spod znaku kultu cargo"), od tych, którzy budują katedry.

Jak rozpoznać budowniczych katedr? Są to ci członkowie zespołu, którzy zajmują się debugowaniem, dekompilacją i inżynierią odwrotną oraz czytają specyfikację, RFC lub normy dotyczące używanych technologii. Ludzie, którzy to robią, dokonali zmiany perspektywy i opanowali wyćwiczone zrozumienie narzędzi, które im pomaga.

Ta zmiana perspektywy polega na tym, żeby chcieć podążać za problemem poprzez warstwy systemu i być gotowym poświęcić czas na zdobywanie wiedzy, która pozwoli to wszystko zrozumieć. Przykładowo przerzucenie się z laptopa jednordzeniowego na wielordzeniowy może zmienić zachowanie Twoich testów współbieżności Javy. Niektórzy ludzie po prostu wzruszą ramionami i zaakceptują to, że ich testy będą teraz zachowywać się nieprzewidywalnie. Inni będą śledzić ten problem aż do poziomu procesora poprzez biblioteki współbieżności, model pamięci Javy i specyfikacje sprzętową.

Kup książkę Poleć książkę

Narzędzia, z którymi powinieneś się zapoznać, obejmują debugery (takie jak GDB, PDB i RDB), które umożliwiają wgląd w działający program, debugery poziomu okablowania (takie jak Wireshark), pozwalające zobaczyć ruch w sieci. Obudź w sobie chęć czytania specyfikacji. Jeśli będziesz w stanie czytać specyfikacje tak samo jak kod, nic nie będzie przed Tobą ukryte. Daje Ci to możliwość zadawania twardych pytań o używane biblioteki, a jeśli nie spodoba Ci się otrzymana odpowiedź, będziesz w stanie zaimplementować je ponownie samodzielnie lub przejść do bardziej zgodnych ze standardami implementacji.

Jednym ze sposobów wykorzystania tego wzorca jest uzyskiwanie informacji ze źródeł pierwotnych. Oznacza to, że następnym razem, gdy ktoś będzie mówił Ci o stylu REST (ang. *Representational State Transfer*), potraktuj to jako pretekst do przeczytania doktoratu Roya Fieldinga, w którym jest zdefiniowana ta koncepcja. Rozważ napisanie posta na blogu, aby objaśnić to, czego się nauczyłeś, lub podzielić się tym i zachęcić innych do zapoznania się również z oryginalnym dokumentem.

Nie wystarczy przyjmować za pewnik słowa kogoś, kto cytuje książkę parafrazującą artykuł, który wspomina stronę Wikipedii zawierającą link do oryginalnego dokumentu RFC stowarzyszenia IETF. Aby naprawdę zrozumieć jakąś ideę, trzeba zrekonstruować kontekst, w którym po raz pierwszy została ona wyrażona. Dzięki temu można sprawdzić, czy istota idei przetrwała, przechodząc przez wszystkich tych pośredników.

Dowiedz się, kto pierwszy wymyślił dane idee, i postaraj się zrozumieć, jakie problemy próbowano rozwiązać. Ten rodzaj kontekstu zwykle ginie w tłumaczeniu, gdy idea jest przekazywana dalej. Czasami odkryjesz, że z pozoru nowe pomysły zostały porzucone dawno temu, często z jakiegoś ważnego powodu, ale wszyscy już dawno o tym zapomnieli, ponieważ oryginalny kontekst został utracony. Wielokrotnie przekonasz się, że oryginalne źródło idei jest o wiele lepszym nauczycielem niż łańcuch ludzi, którzy selektywnie cytują siebie nawzajem od lat. Cokolwiek się zdarzy, śledzenie rodowodu idei, którą uznałeś za pomocną, jest ważnym ćwiczeniem i nawykiem, który będzie dobrze służyć Ci przez lata, gdy będziesz próbował uczyć się nowych rzeczy.

Kiedy czytasz tutorial, nie powinieneś szukać kodu do skopiowania, ale struktury psychicznej, w której możesz umieścić swoją nową wiedzę. Twoim celem powinno być zrozumienie historycznego kontekstu koncepcji oraz tego, czy jest to jakiś specjalny przypadek czegoś innego. Zadaj sobie pytanie, czy za tym, czego się uczysz, jest jakiś bazowy koncept informatyczny i jakie kompromisy zostały poczynione w implementacji, której używasz. Uzbrojony w tę głębszą wiedzę, powinieneś być w stanie wyjść poza początkowy tutorial, gdy napotkasz problemy.

Ludzie często mają na przykład kłopoty z wyrażeniami regularnymi, ponieważ posiadają tylko powierzchowne zrozumienie tego zagadnienia. Możesz dobrze sobie radzić przez wiele lat, a może nawet dziesiątki lat, bez prawdziwego zrozumienia różnicy między deterministycznym automatem skończonym i niedeterministycznym automatem skończonym. I wtedy pewnego dnia Twoja wiki przestaje działać. Okazuje się, że jeśli Twój silnik wyrażeń regularnych jest zaimplementowany rekurencyjnie, w przypadku niektórych danych wejściowych, które wymagają nawracania, będzie działał przez bardzo długi czas i w końcu rzuci wyjątek przepełnienia stosu (`StackOverflowException`). Ade przekonał się o tym na własnej skórze, ale na szczęście stało się to w jego zabawkowej implementacji wiki, a nie w środowisku produkcyjnym.

Kup książkę Poleć książkę

Przy tym całym skoncentrowaniu się na dogłębnym zrozumieniu technologii i narzędzi trzeba uważać, żeby przypadkowo nie stać się wąskim specjalistą. Chodzi o to, żeby być w stanie uzyskać jak najwięcej specjalistycznej wiedzy, która jest niezbędna do rozwiązania każdego problemu, nie tracąc przy tym perspektywy dotyczącej względnego znaczenia różnych aspektów rozwoju oprogramowania.

Próbując zastosować ten wzorzec, nauczysz się z pewnością jednego — że zdobywanie głębokiej wiedzy jest trudne. To dlatego wiedza informatyczna większości ludzi dotycząca rozwoju oprogramowania ma kilometr szerokości i kilka centymetrów grubości. Łatwiej i często korzystniej jest polegać na cudzej wiedzy na temat podstaw, niż dołożyć dodatkowych starań, aby nabyć ją samemu. Możesz powiedzieć sobie, że zawsze będziesz mógł nauczyć się tego w razie potrzeby. Gdy przyjdzie jednak ten dzień, będziesz musiał wiedzieć wszystko do końca tygodnia, ale samo nauczenie się wszystkich warunków wstępnych zajmie miesiąc.

Inną możliwą konsekwencją posiadania tylko powierzchownej wiedzy jest to, że możesz nigdy nie zdać sobie sprawy, że problem, który próbujesz rozwiązać, ma dobrze znane rozwiązanie lub jest w rzeczywistości niemożliwy do rozwiązania (w tym przypadku może istnieć wiele prac naukowych o tym, dlaczego jest to niemożliwe i jak przedefiniować problem, aby stał się rozwiązywalny). Jeśli tylko prześlizgniesz się po powierzchni, nie będziesz wiedział, czego nie wiesz, a bez zrozumienia granic własnej wiedzy nie można odkrywać nowych rzeczy. Często proces przekopywania się przez wszystkie warstwy problemu ujawnia bazową koncepcję informatyczną. O ile praca informatyków może wydawać się niepraktyczna, ci, którzy mogą zastosować najbardziej zaawansowane teorie do rzeczywistych problemów, uzyskują zdolność robienia rzeczy, które dla innych mogą wydawać się niemal magiczne. Proste zmiany wyboru algorytmu lub struktury danych mogą sprawić, że trwające miesiące przetwarzanie wsadowe będzie się kończyć, zanim jeszcze użytkownik zwolni nawet przycisk myszy. Ktoś, kto zna tylko listy, zbiory i tablice mieszające, raczej nie będzie zdawał sobie sprawy, że do rozwiązania danego problemu potrzebuje drzewa trie. Zamiast tego po prostu przyjmie, że ten problem, tak jak dopasowywanie najdłuższego prefiksu, jest niewiarygodnie trudny, i zrezygnuje lub zapyta, czy ta funkcja może mieć niższy priorytet.

Jeśli będziesz stosować ten wzorzec regularnie, staniesz się jednym z tych, którzy naprawdę rozumieją, jak działają ich narzędzia. Nie będziesz już tylko sklejał fragmentów kodu i w trudniejszych kwestiach polegał na cudzej magii. Musisz mieć świadomość, że to zrozumienie odróżni Cię od większości programistów, z którymi pracujesz, i sprawi, że staniesz się logicznym wyborem w przypadku najtrudniejszych zadań. W konsekwencji najprawdopodobniej poniesiesz całkowitą porażkę lub spektakularny sukces. Ponadto nie pozwól na to, żeby ta wiedza zrobiła z Ciebie aroganta. Zamiast tego w dalszym ciągu poszukuj możliwości, żeby być najgorszym. Zmierz się z budowaniem przydatnych narzędzi z tych podstawowych cegiełek, zamiast jedynie delektować się własną zdolnością do rozkładania rzeczy na czynniki pierwsze.

Działanie

Znajdź i przeczytaj dokument RFC 2616, który opisuje HTTP 1.1, oraz dokument RFC 707 opisujący stan wiedzy o technologii zdalnego wywołania procedury (ang. *Remote Procedure Call* — RPC) na styczeń 1976 r. Uzbrojony w tę głębszą znajomość HTTP spróbuj zaimplementować klienta i serwer

Kup książkę Poleć książkę

dla RFC 707. Kiedy poczujesz, że już dobrze zrozumiesz kompromisy poczynione przez redaktorów RFC 707, zbadaj nowoczesną implementację *open source* tych samych pomysłów, taką jak framework Apache Thrift, który zasila Facebook. Następnie napisz na blogu o ewolucji naszej wiedzy dotyczącej zdalnych wywołań procedur i systemów rozproszonych na przestrzeni ostatnich trzech dekad.

Teraz poczytaj artykuły Steve'a Vinoskiego o RPC. Czy nabrałeś wątpliwości co do głębokości Twojego rozumienia? Napisz na blogu o swoich wątpliwościach i aktualnym poziomie zrozumienia.

Zobacz również

Bądź Najgorszy (rozdział 4.) i Zamiataj Podłogę (rozdział 4.).

Znajome Narzędzia

Rutyna oznacza, że kręcisz kołami i stoisz w miejscu. Jedynym postępem jest zakopywanie się jeszcze głębiej w rutynę. Biegłość oznacza coś innego: koła kręcą się, a Ty bez trudu posuwasz się do przodu… Rutyna jest konsekwencją trzymania się sprawdzonych i przetestowanych metod, które nie uwzględniają, jak zmieniłeś się Ty i jak zmienił się świat.

— Twyla Tharp, *The Creative Habit: Learn It and Use It for Life*

Kontekst

Każdy projekt jest pełen nowych rzeczy do nauki. Są nowi członkowie zespołu, nowe role w zespole, dziedziny biznesu, techniki i technologie.

Problem

W tym całym strumieniu zmian coś musi pozostawać takie samo albo równie dobrze możesz zaangażować się w badania. Jak można zapewnić klientowi jakiekolwiek gwarancje dotyczące czegokolwiek? Kiedy mówisz, że przygotowanie pewnej funkcji zajmie określoną ilość czasu, klienci potrzebują jakichś podstaw, żeby zaufać Twojej zdolności do dostarczenia produktu.

Rozwiązanie

Określ zestaw znanych Ci narzędzi i skup się na nim. Najlepiej, żeby były to narzędzia, dla których nie potrzebujesz już dokumentacji — powinieneś znać na pamięć wszystkie najlepsze praktyki, potencjalne problemy i najczęściej zadawane pytania lub mieć je opisane na swoim blogu, w wiki lub tam, gdzie zapisujesz to, czego się uczysz. Uzbrojony w tę wiedzę będziesz mógł dostarczyć wiarygodne szacunki dotyczące niektórych części Twojej pracy, ograniczając ryzyko do nowych i niezbadanych obszarów.

Kup książkę Poleć książkę

To, że te narzędzia są Ci znajome, nie znaczy, że zawsze powinieneś polecać je innym. Czasami najlepsze narzędzie do wykonania jakiejś pracy i narzędzie, które znasz najlepiej, to dwie różne rzeczy. W takich przypadkach musisz zdecydować, czy Twoja wydajność jest ważniejsza niż produktywność zespołu. To, że znasz Struts jak własną kieszeń, nie znaczy, że łatwo będzie Ci go użyć.

Mimo to są to narzędzia, które zabierasz ze sobą z jednego projektu do drugiego. Składają się one na to, co czyni Cię bardziej wydajnym niż następny przesłuchiwany kandydat. Jeśli napotkasz problemy, od razu będziesz wiedział, gdzie szukać odpowiedzi. Znasz problemy, które te narzędzia rozwiązują, i wiesz, jakie problemy one powodują. W związku z tym wiesz, gdzie ich *nie* używać, co jest tak samo ważne, jak wiedza, gdzie najlepiej je stosować.

Z biegiem czasu będziesz czuł się coraz bardziej komfortowo z tym małym zestawem narzędzi. Przynosi to korzyści w postaci zwiększonej wydajności, ale wiąże się też z pewnymi niebezpieczeństwami. Jeśli nie będziesz ostrożny, możesz zacząć postrzegać swoje znajome narzędzia jako „złote młotki", które są w stanie rozwiązać każdy problem. Istnieje również niebezpieczeństwo, że staniesz się tak świetnym ekspertem od tych narzędzi, że nie będziesz w stanie z nich zrezygnować lub nawet zorientować się, gdy pojawią się lepsze.

Prawdziwe wyzwanie pojawia się wtedy, gdy musisz wyrzucić dużą część Twojego zestawu narzędzi. Czasami narzędzia staną się przestarzałe. Innym razem odkryjesz, że dostępne są lepsze. W rzadkich przypadkach Twoja znajomość aktualnego stanu wiedzy doprowadzi Cię do wynalezienia czegoś, co uczyni nieprzydatnymi narzędzia, które już znasz.

> W czasie drastycznej zmiany to uczniowie dziedziczą przyszłość. Uczeni zwykle okazują się odpowiednio wyposażeni, aby żyć w świecie, który już nie istnieje.
>
> — Eric Hoffer, *Reflections on the Human Condition*

Ade bardzo wcześnie zaadaptował scentralizowany system kontroli wersji o nazwie Subversion. Kiedy stał się on popularniejszy, klienci poszukiwali Ade'a do swoich projektów ze względu na jego wiedzę o Subversion. Pomimo tego Ade od początku śledził pojawienie się nowej generacji *rozproszonych* systemów kontroli wersji. Zanim Subversion stanie się przestarzały, jego miejsce w zestawie narzędzi Ade'a będzie już zajęte przez Git lub Mercurial. Porzucanie znanych i cennych narzędzi to bolesny proces, ale jest to umiejętność, którą musisz nabyć.

Możemy zagwarantować, że narzędzia, których używasz jako uczeń, staną się przestarzałe, zanim zostaniesz czeladnikiem. *Z czasem wszystkie Twoje ulubione narzędzia staną się śmieciami.* Aby Twoja kariera się rozwijała, musisz nauczyć się z łatwością porzucać znane narzędzia i pozyskiwać nowe. Konstruowanie programu uczenia, który wspiera ten cel, jest jednym z wyzwań, z jakimi muszą zmierzyć się wszyscy uczniowie w procesie stawania się czeladnikiem.

Kup książkę Poleć książkę

Działanie

Spisz listę Twoich znanych narzędzi. Jeśli ta lista ma mniej niż pięć pozycji, rozpocznij polowanie na narzędzia, które wypełnią luki w Twoim przyborniku. Może to być po prostu kwestia identyfikacji narzędzia, którego już używasz, ale niezbyt dobrze je znasz, lub może się to wiązać ze znalezieniem całkowicie nowych narzędzi. Tak czy inaczej ułóż plan uczenia się tych narzędzi i rozpocznij jego wdrażanie już dziś.

Jeśli masz już pięć znanych narzędzi, dokładnie je zbadaj. Czy dostępne są lepsze i wydajniejsze? Czy jesteś przywiązany do narzędzi, które już są przestarzałe? Czy zaczynają pojawiać się narzędzia, które sprawiają, że niektóre elementy Twojego zestawu narzędzi stają się nieaktualne? Jeśli odpowiesz twierdząco na którekolwiek z tych pytań, już dziś rozpocznij proces zastępowania tych narzędzi. Jeśli potrzebujesz bezpiecznego środowiska, żeby poeksperymentować z nowymi narzędziami, wykorzystaj wzorzec Zabawki Do Zepsucia.

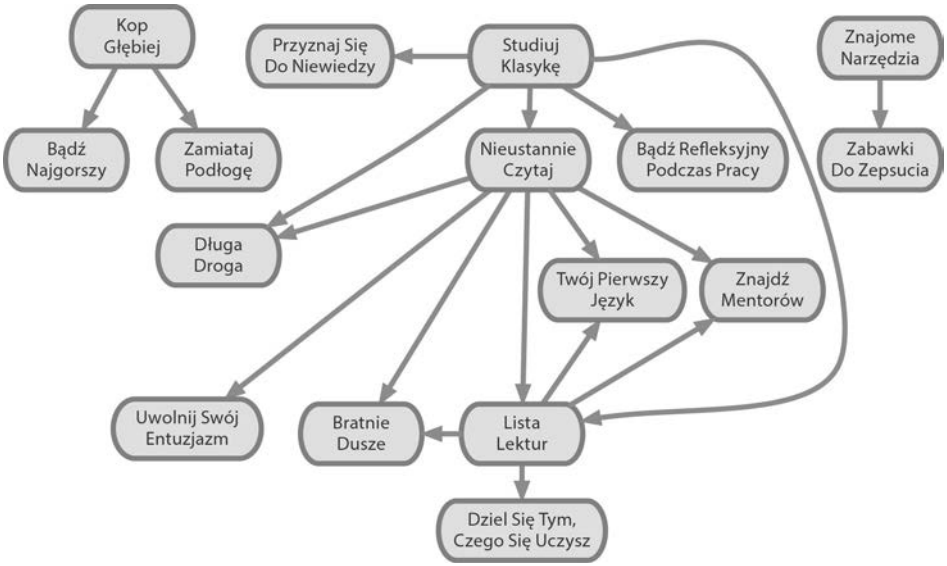
Zobacz również

Zabawki Do Zepsucia (rozdział 5.).

Podsumowanie

W swojej formalnej edukacji mogłeś przyzwyczaić się, że ktoś inny przedstawia Ci program nauczania i przedzierasz się przez niego, prawie zupełnie nie zastanawiając się, czy są to najlepsze książki do przeczytania i czy dana kolejność lektur jest optymalna. Teraz musisz stać się aktywnym uczestnikiem Twojego trwającego samokształcenia, żeby stać się programistą, który potrafi posługiwać się wszechstronnymi narzędziami do organizowania i zbierania informacji. Możesz nie wiedzieć wszystkiego, co trzeba wiedzieć, aby skonstruować swój program uczenia się, ale masz prawo do syntetyzowania mądrości wielu osób, które będą oferować różne sugestie. Jeśli nauczysz się cieszyć samym procesem uczenia się, przysłuży Ci się to dobrze w ciągle zmieniającym się krajobrazie technologicznym, który wymaga od nas zachowania nieustającej czujności.

Kup książkę Poleć książkę



Kup książkę Poleć książkę

Skorowidz

Kup książkę Poleć książkę

Kup książkę Poleć książkę

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl

GRUPA WYDAWNICZA
Helion SA

Rzemieślnik — mistrz, czyli najlepszy fachowiec!

By być świetnym programistą, nie wystarczy Ci sama wiedza techniczna, opanowanie narzędzi deweloperskich i kilku języków programowania. Przyjdzie moment, że staniesz przed zagadnieniami, które Cię przerosną. By ułatwić życie programistom, opracowuje się wzorce — łączą one rozwiązania z wielu dziedzin. Autor tej książki wyszedł z założenia, że wszyscy rzemieślnicy mają ze sobą coś wspólnego. No dobrze, ale w jaki sposób rzemieślnik programowania może się uczyć od, dajmy na to, mistrza tańca?

Temat uczenia się rzemiosła potraktowano tu interdyscyplinarnie. Jej podstawą są dziesiątki wywiadów z praktykami oraz literatura na temat uczenia się i wydajności. Pokazano tu, jak wybitni chirurdzy, choreografowie, filozofowie oraz architekci aplikacji dochodzili do mistrzostwa. Książka stanowi źródło inspiracji: poszczególne wzorce rozwiązywania problemów są osadzone w określonych kontekstach, a sugerowane rozwiązania wzbogacono o odniesienia do literatury i historii mistrzów z różnych dziedzin.

Dzięki tej książce dowiesz się:

- Czy jesteś uczniem, czeladnikiem, czy może już mistrzem rzemiosła programistycznego
- Czym są wzorce i jakie mają znaczenie dla nauki rzemiosła
- Jak nieustannie się uczyć i dążyć do mistrzostwa
- Jak oceniać swoje umiejętności
- Jak skutecznie rozwiązywać różne problemy, począwszy od współpracy z klientem, a skończywszy na pokonaniu własnego wypalenia zawodowego

Dave H. Hoover — prowadzi dział rozwoju oprogramowania oraz program praktyk zawodowych w firmie Optiva. Programuje od 2000 roku, jego pasją jest rozwijanie idei rzemieślniczego podejścia do pisania aplikacji.

Adewale Oshineye — jest inżynierem w fimie Google. Programuje od wczesnego dzieciństwa. Pracował nad wieloma projektami o różnej skali, dzięki czemu mógł się uczyć od najlepszych praktyków Europy Zachodniej.

Helion

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

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

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

Informatyka w najlepszym wydaniu

O'REILLY®

ISBN 978-83-283-3521-9

9 788328 335219

cena: 39,90 zł

sięgnij po WIĘCEJ

KOD KORZYŚCI