Максим Сохацький

Топовий програміст т

Поклоніння простору
Теоретичні основи
Мислення
Простір професійного розвитку
Таємні настанови
Практика
Посвята

УДК 13
УДК 821.161.2

Топовий програміст т

Автор: Максим Сохацький (1980)

Про Автора

Намдак Тонпа (Максим Сохацький) — доктор філософії КПІ (02070921), буддистський піп-капелан лінії передачі Лонгчен Нінгтік тибетського буддизму школи Нінгма (38778275), провідний інженер-програміст ДП «ІНФОТЕХ» (34239034) у підпорядкуванні МВС України (00032684). Автор систем «Депозити ПриватБанк» та «MIA: Документообіг» побудованих на авторських творах ERP.UNO та N2O.DEV.

Постійне посилання твору: https://axiosis.top/top/
Видавець: Інститут формальної математики «Групоїд Інфініті»
Сайт інституту: https://groupoid.space/institute/

ISBN: 978-617-8027-23-0

Якщо мої передплатники і просять про якусь масштабну контрибуцію, то це монографію на тему «як стати топовим програмістом». Хоча таке формулювання інфантильне, воно досить добре відображає сутність запитуваного: детальний розгляд професії програміста, стратегію вивчення предмета виходячи з особистого досвіду, розбавлений автентичною філософією.

Анотація

Збірка відкритих листів на тему ультимативного розвитку в професійному та емоційному аскпектах. Програмування як спорт і як спосіб досягнення досконалості. У вигляді гонзо-журналістики, розбавлено східною філософією.

Тема професійного розвитку була атакована багато разів некваліфікованими педагогами, що намагались денонсувати педагогіку не тільки вищої школи, але і молодших класів. Пізніше у корпоративному середовищі, плани розвитку софт та хард навичок подають вже у більш науковому пакунку, але до кінця не відкривають двері, які ведуть за межі професії, після її повної та остаточної реалізації.

Як я чув колись: секрет успіху тільки трохи залежить від таланту і багато від методики навчання і головним чином від мотивації досягти успіху, джерелом якого зазвичай є наші глибокі травми. Східна філософія каже, що неперервне вдосконалення майстра, може привести його на вершину реалізації. Поєднання діамантової мотивації і правильної спортивної техніки розвитку мислення при гарній долі може відкрити реалізацію професії програміста не тільки у розрізі повного циклу виробництва по ISO 9001, але і відкриє осягнення дотичних професійних реалізацій у сфері дизайну (продуктовий та промисловий дизайн, машино-будівництво, промислова та цивільна архітектура), фізики та техніки, математики та філософії, або машинної лінгвістики.

Оскільки математики, як і програмісти, пенсійним віком для топ-перформерів вважають вік приблизно 25-30 років, отримати правильне знання як стати топ-програмістом або топ-математиком особливо важливо у молодому віці. Ця книга —це скарб двох ключів: перший ключ —це діамантова мотивація і другий ключ —це бажання всевідання, нестримний порив пізнання феноменів який проявляється на ранніх етапах розвитку дитини. Такі діти здатні неперервно тривалий час сфокусовуватися та мають належний перелік якостей мислення необхідних для курсу навчання, які теж можна і потрібно розвивати.

Ця книга —це і мотиваційний спіч, і ретроспектива власного досвіду, і відоповіді на запитання по техніці навчання, і звернення до наступних поколінь, і навчальна методичка для програми «Інституту формальної математики». Бібліографічне забезпечення потрібно шукати в проекті УДК 51, там же ви знайдете курс Дани Скота який показує історію теорії обчислень та сучасної теоретичної інформатики.

Подяка

Дякую всім, хто формував своїми питаннями цей твір.

Зміст

1 Поклоніння простору

1.1 Топовий програміст

Якщо мої передплатники і просять про якусь масштабну контрибуцію, то це монографію на тему «як стати топовим програмістом». Хоча таке формулювання інфантильне, воно досить добре відображає сутність запитуваного: детальний розгляд професії програміста, стратегію вивчення предмета виходячи з особистого досвіду, розбавлений автентичною філософією.

1.2 Висловлювання нескіченної поваги

Перед тим як розпочати розповідь про професію програміста насамперед хочеться висловити пошану предмету вивчення та практики програмування, а саме формальним математичним обчислювальним побудовам, у яких можливе програмування в принципі. В останні роки стало зрозуміло, що простір цих побудов настільки глибокий, що може поглинути не тільки всі дискретні програми всіх формальних граматик, а й континуальну математику, у якій робота з простором йде на іншому, більш фундаментальному рівні. Тому без зайвого перебільшення можна сказати, що саме простір народжує мовну групу мов, які є первісною матрицю всіх без винятку мов програмування.

Принцип глибокої поваги до предмету, який майстер повинен реалізувати є одним із секретних ключів східної філософії. Позаяк програмування народжується з простору феноменологічних побудов, що ведуть до абстрактної класифікації просторів та логік з ними пов'язаних, то утримання у фокусі мети вивчення простору та програмування як практичного людського процесу з цим пов'язаним є головним завданням на шляху вивчення. Тому, без недооцінки та зайвого перебільшення можна сказати, що простягання або поклоніння, як прояв поваги до самого простору, як об'єкту вивчення, виглядає для мене логічним. Я простягаюся перед простором.

1.3 Перевірка мотивації

Важливою характеристикою, яка, хочете вірте, а хочете ні, впливає на процес вивчення мистецтва програмування, є чистота мотивації. Якщо розглянути граничный популярний споживацький приклад, то він буде виглядати так: ваша мотивація полягає у збільшенні своїх навичок програміста для досягнення матеріальних благ і підвищення конкурентоспроможності на ринку праці. Повна нісенітниця, така мотивація впливає на критерії вибору об'єктів вивчення і це може завести вас у ситуацію, коли вам 50 років і ви пишете на Core Java для якось швейцарского банку. Очевидно, що люди, які просили у мене цей текст, не очікують чогось подібного.

Свої ілюзії про легкість цього шляху можна відразу відкинути. Цей шлях по-самурайськи скдадний і на нім сходили з розуму не тільки випускники прикладної математики, немало людей перегоріло на підприємствах від перенавантаження та неконтрольованості інформації. Тому 10 років ув'язнення з постійним виділеним каналом в інтернет на повному зовнішньому забезпеченні — ідеальний ресурс, який я рекомендував би виділити для успішної підготовки на майстра програмування.

Чому такий великий термін буде пояснено в наступних частинах. 10 років цілком адекватний інтервал навчання для лікаря, то чому для програміста повинно бути менше. Кількість мов якими кваліфікований програміст володіє на практиці може сягати тисяч, за кожною з них стоїть теорія, своя логіка і своя математика нею породжена. Це не просто латина, есперанто та пару мов романо-германської групи. Половину цього часу можна проводити в реальних проектах, типу інтернатури, але мов і матеріалів так багато, що для топового програміста 10 років можна виділити тільки на теорію.

Так, можна і в 50 років влаштуватися на галєру «цифровим сантехніком», але це теж ніяк не попадає під курс топового програміста, який повинен покривати широкий діапазон дисциплін: від створення процесорів, асемблерів, компіляторів, операційних систем, систем управління базами даних, мережевих протоколів, сервісів, шин та додатків до теоретико-типових ве-

рифікаторів математичних моделей та теорем, сертифікованих компіляторів, систем доведення теорем.

Мотивація настільки важлива, що без правильної мотивації висувати будь-які претензії про марно втрачені 10 років життя абсолютно безрезультатно, сертифікат відкликається. Як перевірити чистоту мотивації і наскільки точні можуть бути рекомендації? Можу лиш сказати, що видо повинні бути достатньо чесним перед самим собою, адже програмування — це складний виснажливий процес, а мислення — найвища форма управління організмом, тому вади в його роботі можуть призвезти до непоправних наслідків.

Якщо крім програмування ви нічого не вмієте, то непогано було би розвинутив в собі первні стратегії відступу: мінімальні техніки управління диханням та дієтою, легкий спорт без фанатизму, трохи йоги, можливо активні види спорту. Якщо ви вважаєте, що у цілому ви психічно стабільна людина, то пригутуйтеся до сюрпризів на шляху осягнення загадок простору без внутрішної чистоти намірів.

Моєю особистою мантрою, з якою я вивчаю програмування — це посвята резульатів своє роботи людям та всім істотам, не нашкодивши нікому без виключення. Взагалі вивчення програмування мало кому може зашкодити та може мати форму глибокого відлюдництва святого монаха. Хоча є виключення, програмісти, а особливо гарні програмісти, в своїй більшості не жорстокі істоти, і їх надмірна агресія і сердитість, направлена в позитивному ключі інстроспекції є двигуном аутичного осягнення потаємностей професії програміста.

Взагалі, якшо мотивація алзмазної візниці привести усі істот до абсолютного просвітлення знається вам занадто езотеричної, то хочи би стара етично норма інженерів минулого «не нашколь, а краще допоможи людям» є тим мінімомум, який необхідно перевіряти перед кожною сесією програмування. Уявіть собі, що ви з рвінням лева кладете на вівтар просвітлення 10 років самоосвіти в області програмування зі скрині свого життя для того аби принести користь людям та суспільству. Без подібної мотивації вам просто не бути звідки черпати енергію для щоденних вправ в програмуванні та мисленні.

1.4 Всевідання як джерело натхнення

Головна риса характеру, яка необхідна в людині, щоб стати топовим програмістом — це схильність до вивчення та дослідження феноменів, їх аналізу, синтезу та абстракції. Це бажання розібрати і досліджувати іграшку має так глибоко перебувати у свідомості, що здається, ніби дитина вже народжується з цим даром і швидко розбиратися у феноменах за належного інтенсивного навантаження на нейросіточку. Іншими словами — це хакерство, якщо ви любите досліджувати системи, розбиратися в програмному коді, розумієте, як працюють процесори, знаєте, як працює логіка та математика, то ви вже можете стати топовим програмістом. Бажання побудувати максимально точну модель феномена має бути гіпертрофованим, воно має бути незакритим гештальтом, який не дає вам спати ночами, поки ви його не закриєте. Саме ця фанатична одержимість конвертується в те, що буде дровами у нашому вогнищі просвітлення на шляху до всезнавства у світі програмування. Звідки взялося всезнання? Це друга сторона медалі головного джерела натхнення хакера. Якщо при локальному розгляді феноменів головною думкою має бути побудувати максимально точну модель феномену, то при фокусуванні в нескінченність до країв горизонту, це бажання проявляється у вигляді максимально швидкого пізнання всіх феноменів і їх універсальні принципи пристрою. Такий мета-хакерський трансцедентальний напів-фрічний майндсет необхідний для розуміння того, наскільки абстрактними і широкими можуть бути виклики на шляху пізнання глибинних мов, якими написано наш всесвіт.

Так, як мови програмування використовуються у всіх сферах людської діяльності, то топовий програміст абсолютно точно повинен розбиратися у всіх доменних моделях, усіх типах та всіх математиках, які виникають у різних мовах програмування. Зазвичай, університетські 5 років я б рекомендував провести якраз у охопленні всіх математик та всіх видів мов програмування, перед тим як поринути у фундаментальну математику та системне програмування. Взагалі хороша сучасна освіта рівня PhD автоматично має на увазі вільне володіння мовним

та математичним забезпеченням у дослідженні всесвіту, так що нічого такого, що не вимагають топові університети, курс топового програміста в цій частині всезнавства тут не вимагає. Потрібна повна автономність на рівні полетів у космос і відновив усі знання та навички за потреби у найкоротші терміни шляхом легкого спогаду.

Щоразу поглинаючи якийсь пласт інформації ви вивільняєте величезний простір свободи, який або заповнюється новими недослідженими пластами, або звільняється абсолютно, якщо вже всі пласти поглинули. Але коли ви повністю вичерпаєте всю карму, тоді буддахуд прийде автоматично, тож це вже програма максимум. Адже після того, як ви вивчили якийсь предмет і дали кілька майстер класів по ньому, ви просто гортаєте всі книги по ньому, за якими навчалися і це все для вас навіть не буквар, тому що буквар ви вже самі написали, це для вас просто шум дерев у лісі. Ви повністю вичерпали цей предмет, стали майстром у ньому, ви вже бачите всі перерізи глобулярних фазових просторів, маєте на руках кілька моделей і прототипів. Це стан всезнавства. Бажання цього стану — необхідний компонент топового програміста.

Якщо ви побудували якийсь простір феноменів, наповнивши їх змістом і залишаючись там у комфортному середовищі обмеженого знання, ви вже втрачаєте топову мотивацію як компонент всезнання. Не спрямувавши своє мислення в нескінченність, ви не зможете побачити весь ландшафт і правильно розставити пріоритети в поглинанні наукових дисциплін, щоб здійснити «стрибки Тигра» між цими пріоритетними реперними точками.

Тільки дослідники, які сповнені вродженого бажання будувати нові теорії та мовні простори, наділені насінням творчості, що веде до топової реалізації. Міждисциплінарний підхід може виникнути лише за умов широкого профілю. Ніхто ніколи не ставив завдання скласти курс для підготовки людей, яких можна було б назвати культовими хакерами, тому й вимоги до підготовки мають бути позамежними. Тільки олімпійське бажання всезнання може реально наблизити вас до нього.

2 Теоретичні основи

2.1 Метафілософія

Дуже коротко про сучасні філософії. Як визначити філософію предикативно, то це науа, що вивчає наступний перелік питань: 1) як жити добре в достатку та гедонізмі максимально довго всім і не померти від воєн та метеоритів; 2) реальний світ та інші питання гнесеології; 3) свобода волі; 4) етика; 5) математика; 6) музика; 6) література, — це всі питання, або мовні набори та форми, що цікавлять сучасних філософів.

У цій нотатці ми спробуємо побудувати формальну систему і на її прикладі показати інтерпретацію трьох ліній передачі сучасної філософії: європейської чи континентальної філософії — школи, яка задала початок глобальній інтерпретації світу та реконструкції мов, у тому числі й математики; східної філософії як приклад особливої школи, на прикладі якої ми будуватимемо модель; та аналітичної чи англосаксонської філософії, що формалізується сучасною математикою.

2.2 Європейска філософія

На наш погляд, головне питання європейської філософії — це Good Life. Як жити, як жити добре самому, у соціумі, які цілі можуть стояти перед індивідом та видом, баланс етики та етика балансу. Європейська філософія народила геометорію, психоаналіз, навчила людей не боятися свободи, трансформувати агресію, бути більш зрілою істотою, і під вінець свого розвитку поставила питання про мову та мовну гру, як основний інструмент рефлексуючої свідомості.

Мова перестала мати ґрунт, вона стала просто візерунками, семантика яких втрачена, філософія стала формою літературного мистецтва.

Представники континентальної філософії: Арістотель, Платон, Кант, Декарт, Ніцше, Фрейд, Юнг, Юм, Хайдегер, Адорно, Хабермас, Делез.

2.3 Тибетська філософія

У східній філософії центральним питанням є визволення себе і інших, в першу чергу від різних форм страждання. Ця філософія має чітку систему, яка нерозривно пов'язана з тілесними та розумовими практиками, і вижила протягом тисячоліть у законсерованому гірському плато. Тут також порушуються питання етики та свободи волі, але основний наголос робиться на інтелектуальні та неконцептуальні вправи, що ведуть до безпосереднього переживання простору.

Деякі формулювання східної філософії, такі як недвійність всіх феноменів піддаються формалізації в гомотопічній теорії типів (використовуючи методи аналітичної філософії), що спонукало до подальших досліджень у галузі формалізації езотеричних теорій.

Представники східної (тибетської) філософії: Атіша, Нагарджуна, Бхававівека, Камалашила, Шантаракшита, Арьядева, Буддхапаліта, Чандракірті, Цонкапа, Міпам, Лонгченпа.

2.4 Аналітична філософія

Аналітична філософія народжена в математиці, рання аналітична філософія починається напевно з Лейбніца, Ньютона та Ейлера. Пізня аналітична філософія починається З Фреге і за списком: Рассел, Уайтхед, Дедекінд, Пеано, Гільберт, Фон-Нейман, Каррі, Акерманн, Карнап, Сколем, Пост, Гедель, Черч, Берньє, Тюрінг, Кліні, Россер, Мак-Лейн Ловір, Гротендік, Скотт, Джояль, Терньє, Мартін-Леф, Мілнер, Жирар, Плоткін, Рейнольдс, Бакус, Барр, Барендрехт, Лер'є, Силі, Кокан, Х'юет, Ламбек, Воєводський, Еводі, Шульман, Шрайбер.

Якщо описати двома словами головне питання аналітичної філософії — це мова простору. Побудова мови, яка дасть формальний фундамент не лише математики та роздумів, а й самої філософії.

2.4.1 Мова простору

Формальні підстави мови роздумів, математики (усієї) та фізики (всесвіту).

2.4.2 Мовні фреймворки

Мовні фреймворки для менш формальних (з парадоксами) та нечітких (стохастичних) систем.

2.4.3 Конкретні соціальні дифузійні моделі

Прикладна філософія Використання мовних фреймворків для опису конкретних феноменів.

3 Мислення

3.1 Що таке мислення

Перед тим як розпочинати процес навчання непогано було б кілька слів сказати про основний інструмент у процесі вивчення — людське мислення. Минаючи фізичні сторони мислення відразу хочеться поговорити про його когнітивні властивості.

3.2 Характеристики чистого мислення

Перша і головна властивість мислення — це істотність — визначальна характеристика істоти. Інтегральна вища форма, яка керує всіма підсистемами та сприймається істотою, майндстрімом або аватаром. В одному тілі може жити кілька майндстрімів, і деякі з них можуть бути програмістами! Якщо ви думаєте — ви є істотою.

Друга когнітивна характеристика мислення, яку можна відчути у медитаціях — це абсолютна сферична відкритість у всіх напрямках та її безмежність. Така характеристика мислення навіває думки про ізоморфізм мислення та простору. З фізичної точки зору, мислення — це складна система квантових полів, які нашаровуються на квантовий, молекулярний рівень, нервову систему, тому довго доводити не потрібно, що мислення як квантово-механічна система поширюється на весь простір.

Умовно існує два розділи вищої медитації, перший із яких називається розділом мислення, а другий розділом простору. Перший розділ присвячений технікам роботи з феноменами, аналітичної медитації, роботі з мисленням з погляду майндстріму, очним вправам, розвитку відчуття перспективи, роботи з уявою, візуалізаціям. Другий розділ присвячений технікам роботи з мисленням з погляду простору, де мислення асоціюється з простором, в якому воно перебуває, неаналітичної медитації, прагнення до нескінченності, медитації відпочинку.

Третя когнітивна характеристика мислення, яку можна сприйняти на досвіді, — це його необумовленість. Чим вищий рівень розвитку мислення, тим вища його воля до свободи і необу-

мовленість, до перевірки, критичного мислення і переоцінювання. У своїй повній свободі мислення вільно обирає спрямованість і інтенсивність потоку, без різких перепадів і гормональних фонів, рухаючись оптимальною траєкторією дорослішання плоду мислення на шляху до всезнавства.

Четверта характеристика мислення – це безперервність. Будь-які спроби зупинити мислення приводять у місце самоусвідомлення як несучу частоту відчуття присутності себе в цьому світі, в медитації. Навіть у процесі сну, мислення не спить, а перетворюється на інший агрегатний стан, більш розріджене, часом безформне, нечітке, мерехтливе. Повний контроль над безперервністю мислення, від якої не можна відмовитись і яку не можна припинити — завдання топового програміста на шляху до звільнення ресурсів для вивчення програмування. Чим більша точність дискретизації цього контролю – тим краще. Контроль за безперервністю мислення називається точністю мислення.

П'ята характеристика мислення –взаємозалежність. Ви як мислення — це продукт абсорбції інших фрагментів мислень чи просто феноменів, тому обумовлені цією спадщиною. Вирватися за межі цієї традиції та розкопати інсайти на шляху еволюції свого мислення –справжня коштовність як нагорода за працю навчання. Коли ви стаєте майстром, обумовленість зникає, ви реструктуруєте себе наново виходячи вже з особистого досвіду, побудованого на низці інсайтів, за якими ви стрибаєте на шляху до майстерності. І навіть їх ви потім зможете видалити і забути зі свого мислення залишивши тільки пам'ять про те, як потрібно одразу робити правильно, можливо і не згадайте навіть, коли вас спитають, як це ви так швидко помудріли, а навіщо.

Ці п'ять характеристик послужать вам підказками у якому ключі потрібно думати про своє мислення (перша похідна) як інструмент пізнання, можливо для істот з високими здібностями це одразу прояснить деякі моменти. Будь-яка нездатність спостерігати ці характеристики в практичних медитаціях або роздумах про своє мислення, говорить про те, що їх потрібно розвивати, або зайнятися йогою, піти до психолога, розвіятися

з друзями, піти в бар, сісти на таблетки, склянку, все за бажанням — головне щоб спрацювало! Чек лист пройшли переходимо до рекомендацій та індикатора

3.3 Коштовне намисто мислення

У традиції Тибету існує шість типів мислення або програм, які вважаються, позитивно можуть вплинути в цілому на процес вивчення, роздуми і медитації.

Щедрість у контексті мислення означає не скупитися в процесі вивчення, не хапатися за все одразу, мати методологію, з повагою ставиться до будь-якої обраної теми, раз вона вже спливла в медитації як комлекс, який все одно доведеться закрити (пізнати). Здатність до реплікації, викладання, зворотної контрибуції по дорозі поглинання інформації — це щедрість мислення.

Дисципліна означає, що мислення має дотримуватися якогось спортивного, бажано олімпійського режиму, надто хаотичні режими мислення не сприятимуть навчанню, тому приступати до еволюції свого мислення потрібно, коли гормональне тло може залишатися рівним значний час, це необхідно для глибоких медитацій, без яких неможливий прогрес.

Терпіння –це здатність переносити проблеми у процесі навчання. Є матеріал, який може не закриватися роками, але до нього все одно доведеться повертатися, адже назад дороги немає, обрано шлях топового програміста. На шляху може бути занадто багато інсайтів і надто багато наснаги, яке може створювати гормональне тло, яке не завжди можна контролювати, пересиджувати на бенчі такі періоди — це терпіння.

Старанність — означає з непідробним інтересом вивчати предмети, тому правильно їх розмістити дуже важливо. Можливо, саме для вас існує своя послідовність предметів, кожен з яких в окремий момент часу ви вивчатимете з максимальною старанністю. Із цим доведеться працювати, кожному індивідуально.

Фокусування — фокусування, чи концентрація, чи медитація, чи шаматха — це основний режим роботи програміста. Ось

ви сіли за комп'ютер, поставили чашку з кавою, протерли дисплей, всмоктались в пікселі, запустили шелл — ви сфокусовані на роботі, це медитація.

Мудрість — це система накопичених інсайтів, що формує нові структури мислення, нову його топологію. Ця система може переписувати старі неефективні та невалідні структури, з яких ми сміємося подорослішавши. Мислення мудрості — це мислення, засноване виключно на таких перевірених рафінованих структурах, які покладені в фундамент нашої істоти.

3.4 Отрути мислення

Три найбільш несприятливі форми мислення з моєї особистої класифікації.

Інертність мислення –це колесо медитації. Будучи вкотре запущена деяка звичка, йде у автоматичний режим на підсвідомість — це інертність. Якби не було інертності мислення, ми б не змогли вчитися. Хоча це корисна властивість мислення, іноді буває погано, коли погано — потрібно відловлювати. Зрозуміло, що бешкетувати своє мислення, яке заховано в підсвідомості не вчать у школах, доведеться працювати самому.

Лінощі. Занадто інтенсивне мислення може перерости в затяжну рекреаційну прокрастинацію, яка зміниться лінощами. Спостереження за видимим прогресом необхідно, яким би був охуенный відпочинок треба повертатися за програмування, оновлювати мотивацію, якщо потрібно щодня.

Байдужість. Корінь усіх отрут, жадібності та іншого. Якщо вам все раптом стало байдуже, це дуже погано, але не смертельно. Іноді може перерости в екзистенційну кризу, але ж ми з вами вже домовилися, що тіло, йогу і таблетки і своє самопочуття ви берете на себе, з мене тільки рекомендації щодо процесу навчання. Занедбана байдужість — це тупість.

Нічого хорошого успішного студента, який виходить за перерахування чеснот, тут немає. Як і першому випадку постійно застосовуємо техніку роздуми, вивчення та медитації до цих видів мислення, як і до основних характеристик мислення. Постійно валідуємо своє мислення відповідно до індикаторів.

3.5 Ядро логіки

3.5.1 Квантор узагальнення

3.5.2 Квантор існування

3.5.3 Багатовимірний ізоморфізм

4 Простір професійного розвитку

4.1 Структура курсу

Не те, щоб це була якась новина, впевнені багато хто дотримується такої карти топового програміста, але я візьму на собі сміливість відкрити це таємне знання. Почну опис курсу з відомої мемної картинки.

4.2 Дракон

Юнікорнами називають тих програмістів, які однаково добре володіють CSS скажімо, а також можуть повністю побудувати будь-якої складності тонкий чи товстий клієнт не обмежуючись HTML5, а й переходячи у SVG чи WPF, чи DirectX чи OpenGL.

Фулстек програмістами називають фахівців із побудови інформаційних систем на кордоні з єдинорогами (які зазвичай не займаються процесиногом, інфраструктурою, мережами та захистом).

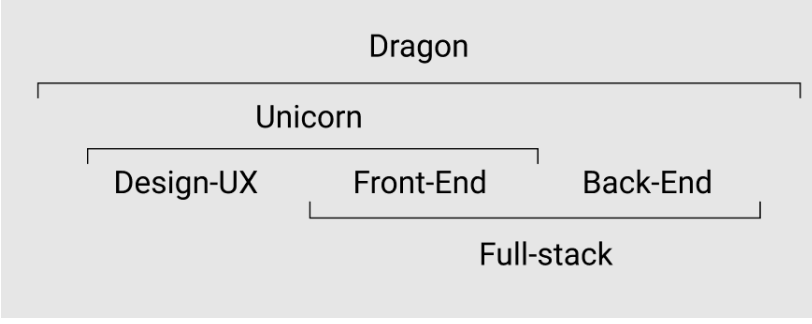

Рис. 1: Дракон

Знайти програміста який може дати гарантію контролю якості на всьому спектри цих спеціалацій побідно до того, як відшукати дракона у східній міфології, істота надзвичайно рідкіста та потужна.

4.3 Лямбдагарбха

Наступний рівень – це платформоутворюючий рівень, який включає мову програмування, рантайм та апаратуру. Зазвичай дорослі академічні мови створюються відразу з рантаймом, тому назвемо цю секцію рівень університетського професора, а секцію рантайму (ОС) та апаратура назвемо підприємницької, оскільки ОС зазвичай продають разом із залізом і всі, хто це намагався просувати на ринок, можна прирівняти до бодхісатств. Останні відомі лямбдагарбхи — це давні автори перших Ліспмашин та XEROX PARC.

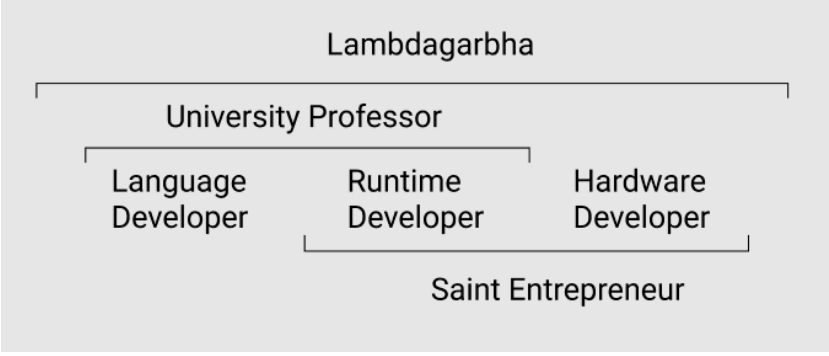

Рис. 2: Лямбдагарбха

Однак дракони, як майсти кінцевих продуктів, покладаються на роботу інших потужніх божеств — які можуть побудувати самостійно і довести до впровадження віртуальні машини та мови програмування, таких істот ще менше ніж драконів на платені Земля.

4.4 Гротендік

На абсолютному рівні програмісти (у тому числі і топові) є математиками, тому тут можна відзначити ядро, яке було відкрито Квілен —модельні категорії, в яких працювали не тільки медалісти Філдса - Воєводський і сам Квіллен, але які є також основним інструментом сучасних теоретико-типових математиків як Шульман. Предмет вивчає модельні категорії Квіллен назвав гомотопічною алгеброю, за допомогою якої була побудована не тільки модель топології алгебри самим Квілленом, але і A^1-теорія гомотопій Воєводського. Усе це кришується Гротендиком, як мультидисциплінарним програмістом абсолютного рівня (топ-математиком).

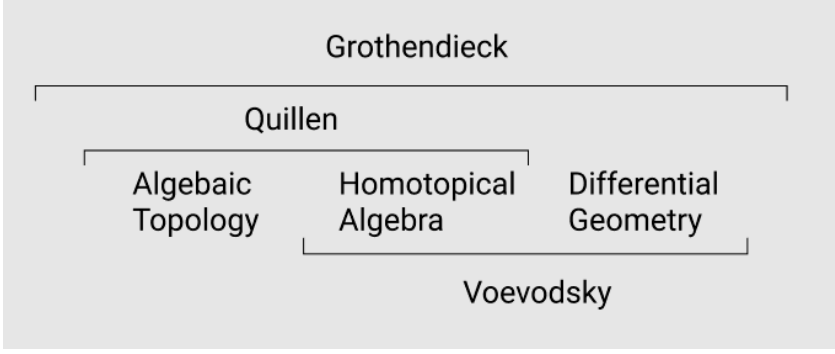

Рис. 3: Гротендік

4.5 Будда

Без зайвої скромності, будь-який програміст, який зміг не тільки уявити, а й встигнути попрацювати за життя на всіх рівнях, може вважати себе Буддою програмування, або як ми скромно називаємо таких пацанів —хуй з гори.

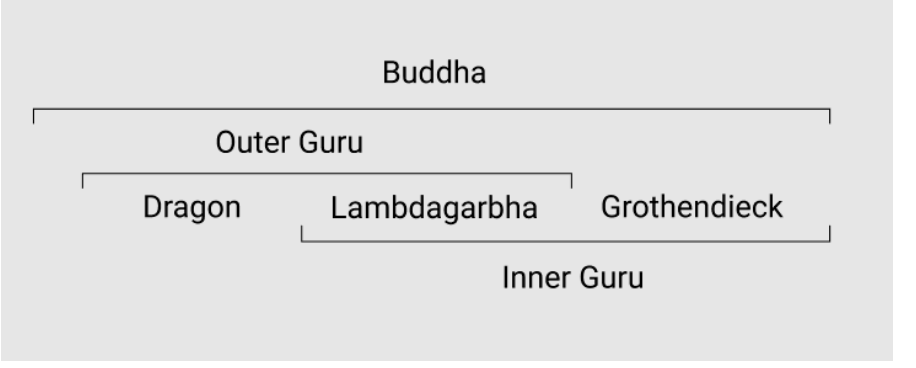

Рис. 4: Будда

4.6 Класифікація мов програмування

Повна класифікація об'єктів дослідження була злоблена в рамках проекту Енциклопедія Мов Програмування, де була зроблена спроба надати формальну БНФ-нотацію для всіх мов.

Домен	Мови програмування
HW	VHDL, Verilog, Clash, Chisel, SystemC, Lava, BSV
ASM	PDP-11, VAX, S/360, M68K, PowerPC, MIPS, SPARC, Super-H Intel, ARM, RISC-V
ALG	C, BCPL, ALGOL, SNOBOL, Simula, Pascal, Oberon, COBOL, PL/1
ML	SML, Alice ML, OCaml, UrWeb, Flow, F#
PURE	HOPE, Miranda, Clean, Charity, Joy, Mercury, Elm, PureScript
F_ω	Scala, Haskell, 1ML, Plutus
MACR	LISP, Scheme, Clojure, Racket, Dylan, LFE, CL Nemerle, Nim, Haxe, Perl, Elixir
OOI	Simula, Smalltalk, Self, REBOL, Io JS, Lua, Ruby, Python, PHP, TS, Java, Kotlin
CMP	C++, Rust, D, Swift, Fortran
SHELL	PowerShell, TCL, SH, CLIPS, BASIC, FORTH
SVC	IDL, SOAP, ASN.1, GRPC
MARK	TeX, PS, XML, SVG, CSS, ROFF, OWL, SGML, RDF, SysML
LOGIC	AUT-68, ACL2, LEGO, ALF, Prolog CPL, Mizar, Dedukti, HOL, Isabelle, Z
$\Pi\Sigma$	Coq, F*, Lean, NuPRL, ATS, Epigram, Cayenne, Idris, Dhall, Cedile, Kind
HoTT	Menkar, Cubical, yacctt, redtt, RedPRL, Arend, Agda
CHKR	TLA+, Twelf, Promela, CSPM
PAR	Ling, Pony, Erlang, BPMN, Ada, E, Go, Occam, Oz
ARR	Julia, Wolfram, MATHLAB, Octave, Futhark, APL SQL, cg, Clarion, Clipper, QCL, K, MUMPS, Q, R, S, J, O

Таблиця 1. Класифікація мов порграмування.

5 Таємні настанови

5.1 Аспекти курсу

Після того, як у перших двох розділах ми розповіли про топову мотивацію та топове мислення, перейдемо до топового курсу топового програміста. Структура цього курсу програмування буде дуже відрізнятися від інших підходів до навчання, але цей курс є курсом, який я намагався "побачити"крізь ретроспективу свого професійного досвіду, що стосується безпосередньо програмування, того, як я бачу топовість цього процесу.

Перед початком зі структури курсу я хочу показати основні індикатори (аспекти курсу), на які варто орієнтуватися при виборі топових напрямків у програмуванні. Тому що саме ці індикатори визначають ті предмети та їх послідовність в якій ви повинні поглащати інформацію, щоб наблизитися до максимальної топовості.

5.2 Гранична точність

Гранична точність означає абсолютні обчислення. У цю вигадану категорію я включаю всі галузі програмування та математичного моделювання, які вимагають абсолютної формальної точності: системи доказу теорем, спеціалізовані формальні верифікатори моделей, системи символьної алгебри, системи обчислювальної гомологічної алгебри, тобто. ті системи моделювання, які не тільки розкладатимуть саму систему до атомів, але ще й найближчу метамодель на рівень вище, яку теж потрібно формалізувати, щоб верифікувати самі моделі. Сюди входять такі галузі як процесоробудування (модель чекерів друкованих плат, процесорів, спеціалізовані мови програмування типу VHDL), мовобудування (системи типів, формальні мови програмування, мови загального призначення). Сюди також входять: будь-які формальні математичні теорії, формальні логіки, мова кванторів Пі та Сігма.

5.3 Гранична оптимальність

Гранична оптимальність означає мінімальну кількість зусиль, доданих до досягнення мети. Сюди входять дискретні завдання міні-максу, лінійне програмування, симплекс-методи, поліедральне багатовимірне симплектичне програмування, методи оптимізації. У фізиці це основний принцип варіаційного числення, мінімальні геодезичні лінії. Ваша система не тільки повинна бути максимально точною, в апогеї абсолютно точною, але і повинна бути закодована оптимальним чином, не містити частин, що повторюються, займати мінімальний футпринт по обмеженій кількості пам'яті і обчислювальних потужностей. Краще, щоб була теорема яка доводить цю мінімальність, як наприклад, чорч-кодування індуктивних типів як природне кодування будь-яких структур у лямбда численні, оптимальні лямбда евалуатори і т.д. Гранична оптимальність означає також вищий пілотаж у прототипуванні та створенні MVP ескізів.

5.4 Гранична складність

Гранична складність означає дослівно максимальну складність системи, з якою потрібно працювати. Якщо раптом виявиться, що вам складності декартово-замкнутих категорій мало, завжди можна перейти до симетричних модальних категорій, в яких живуть такі мови програмування: квантові мови програмування, конкруретні паралельні системи типу Erlang, системи лінійних типів, мови для обробки тензорів. Також гранична складність має на увазі наявність вже в системі граничної точності та граничної оптимальності, інакше без першого у вас буде просто непрацююче гівно, у відсутності другого у вас буде мільйони рядків дублюючого коду, які не впливають на реальну складність проекту. Якщо ви хочете побачити реальну складність подивіться кубічні докази K-теорії.

Це стосується більш-менш чистих тем з логіки, дискретної математики та програмування. У міждисциплінарному підході, якщо ви хочете стати не топовим програмістом, а, наприклад, топовим біофізиком, то список тем буде іншим. Гранич-

на точність там замінюється на математичну статистику та стохастичну фізику, гранична оптимальність пов'язана з топологією просторів, а гранична складність виражається в об'єднанні полів і, як приклад, стандартної моделі. Ну а в біофізиці складність збільшується топологічними різноманіттями нейромереж як ієрархічних процесів, що працюють з тензорними потоками. Гомотопічна теорія типів як мова програмування інфітіні топосу є у фізиці аналогом теорії струн. Біофізика ж є більш дисипативними структурами, які втрачають межі точності, таким чином завдання створення AI не повною мірою відповідає абсолютній точності. Я би сказав що AI це більше системна інженерія та прикладна область, в той час, як абсолютна точність це більше сфера математичного програмування. І навіть машинне навчання при детальному розгляді зводиться до статистичної точності, прогнозованої оптимальності та системної складності.

Я як прихильник абсолютної складності вважаю, що будь-яка сучасна PhD повинна містити міждисциплінарний підхід. Як необхідний мінімум у міжнародній науковій практиці пропоную розглянути міждисциплінарний підхід, який базується всього на двох особливих дисциплінах: УДК 51 (математика) та УДК 004 (програмування). В якості особливих вони обрані тому, що будь-яка інша дисципліна базується на чистій математиці, а для проведення будь-якого моделювання (перевірки перевірки теорії), потрібен фундаментальний курс програмування.

Цікаво, що третій із цих індикаторів можна інвертувати і при цьому оптимізаційний вектор (максимальна точність, максимальна оптимальність та мінімальна складність) теж виявиться цікавим об'єктом розгляду. В суті це вимоги, що висуваються до бібліотек програмного забезпечення: вони повинні бути максимально формальними з доказами властивостей їх моделей, повинні бути оптимально змодельовані або володіти оптимальним екстрактом в інші моделі, але при цьому також повинні бути максимально простими або мінімально складними у загальній картині компонентів. Філософія N2O точно відповідає такому альтернативному оптимізаційному вектору, але вирішуючи завдання технологічних стеків не можна стати топовим програмі-

стом, потрібно збільшувати складність, наприклад у бік складних соціо-інформаційних систем, як ERP системи управління підприємством, де десятки тисяч таблиць кожна по сотні полів звична складність для бізнес-аналітиків, що працюють з такими продуктами як SAP S/3.

Забавно, що в езотеричному буддизмі Тибету, ці три критерії застосовуються в настановах з візуалізації у вищих тантрах: ви повинні максимально детально (до волосся і фактури тканин одягу — принцип максимальної точності) представити дуже складний об'єкт (18-руке божество в союзі з дружиною, свитою та аттрибутами — принцип максимальної складності) максимально швидко (бажано миттєво — принцип оптимальності). Учні, які можуть це робити не "на словах а "на ділі вбудовуючи картинку силою думки і уяви навіть не на січень очі, а прямо на кору головного мозку, випалюючи зображення у своєму мисленні на простирадлі уявного всесвіту — вважаються топовими медитаторами . Для наочного прикладу хочу показати приблизну якість візуалізації, що утримується в просторі свого мислення, якого може досягти середній за здібностями європеоїд-медитатор за 3 роки наполегливої практики:

Напевно, тому досі не існує жодного просвітленого європейця! Ця картинка також наочна демонстрації того, що ви можете досягти за 3 роки навчання (думаю, що у відділу PhD хлопців із NASA, підготовка до рендеру цієї картинки зайняла три роки).

5.5 Гранична унікальність

5.6 Гранична застосовуваність

6 Практика

6.1 Спочатку йога розуму, потім вже йога тіла

Рано чи пізно всі, хто займається йогою розуму при належному успіху, так чи інакше переходять до освоєння йоги тіла, яке є продовженням розуму. Секрет успішної практики в тому, що як і йога розуму, йога тіла потребує ще більшої усвідомленості. Під йогою тіла ми розумітимемо тут такого роду енергії, які ви можете пережити тільки в режимі екстремального спорту, там де реварда дуже високий. У спорті це X-спорт, у сексі це BSDM, підходить усі дисципліни, де є стоп-слово, за кордоном якого одразу настає термінація істоти.

Чому йога тіла повинна йти обов'язково після укрупнення на практиці йоги розуму? На це є кілька причин. Вважається, що основа сталого і дорослого мислення це правильне світогляд, яке має формуватися істотою під час вивчення філософських дисциплін, невирішених питань трансгуманізму та інших базових принципів. Дотримуючись позасектарного іміджу, самі базові принципи світогляду топового програміста були зацементовані в першому випуску.

Поклавши неправильний майндсет у основу Вивчення, Роздуми та Медитації ви створите пролом, через який у критичний момент вашого життя при зустрічі з Буддою ваше бачення світу зруйнується як дитячий замок із піску після припливу. Для тренування розуму алмазної міцності та гостроти і призначені практики формування правильно погляду на об'єкт дослідження свого власного розуму, за зрадами якого не існують ніякі феномени.

Як і йога розуму, йога тіла передбачає два режими дослідження: пандіта стайл (вивчення теорії) та йога стайл (практика). Засвоївшись і зміцнившись у своїй свідомості, істота, що рухається головним принципом будда-такості, прагненням до всезнання і повної реалізації, зерном якого є пізнання феноменів, починає виходити за рамки ментальних феноменів і починає усвідомлювати себе і своє тіло як частину мислення, і природним чином починає експериментувати з тілом, розши-

рюючи свій фронтир сприйняття.

Головний критерій, який показує, чи можна вам переходити до спорту, це повний контроль над дофаміновою та епініфриновою системою. Деякі занадто вразливі спортсмени використовують ТНС для супресії дофаміну та м'якшої йога-сесії. Зазвичай рекомендується входити в спорт у 40 років, тому що в 25 і кілька років після цього потрібно присвятити математиці та філософії, адже кращого часу вже не буде і повернути його буде непросто! А у 40 уже дофаміновий фон сам по собі зникне і залишиться лише чистий розум та террейн. Це друга безжальна причина через яку спорт краще відкласти до adult віку. Є й інший бік медалі: рани після 40 гояться гірше, тому і ставки і гострота і ревард у такому разі вищі. Ідеально це маючи гострий розум не робити взагалі серйозних помилок на шляху спорту. Неідеальні випадки вирішуються імплантацією титанових пластин.

Непрямий критерій це коли ви досягли рівня безпосереднього переживання ототожнення мови простору (випуск X), карти місцевості (випуск 4) і свого мислення (випуск 2), у такому разі вихід з локальної самсари у вигляді темниці розуму знаменуватиме вихід у реальний світ на планету Земля. Алегорія яка мені бачиться тут така: мати відправляють свого сина до університету, передавши йому всі необхідні знання, які допоможуть йому жити далі автономно

6.2 Програмування та спорт

Гуру у спорті знайти так само важко, як і гуру в програмуванні. У світському житті гуру спорту працюють олімпійськими чемпіонами або чемпіонами з дисциплін, які до цього прирівнюються. Саме їхнє існування вже є вченням. Розбираючи до найдрібніших деталей покадрове трейси топ-спорсменів на youtube ви отримуєте алмазні знання віртуоза-ньюскулера методично відточуючи техніку, маючи зразок для верифікації. Гуру спорту меншого калібру працюватимуть інструкторами на найближчому спортивному курорті, рекламуватимуть газировку та еквіпмент або пропонуватимуть вам тури національними заповід-

никами.

Не всі це говорять прямо, але у спорті важливими є картинки, які проходять через вашу сіточку та всі органи почуттів, тому цінуються картинки природного ландшафту Землі, щоб звільнити своє мислення вже за межами тіла, охопивши своїм мисленням усю планету та її феномени, головний з яких гравітація, таким чином ставши воістину космічною дитиною планетарного масштабу.

Однак у розмовах з локальними спортивними гуру ви отримаєте максимум репресивні монотонні лекції про шкоду куріння на Джомолунгмі, жахливі команди інструкторів-обивателів, юрби туристів на своєму шляху. Доросла людина, що оволоділа йогою розуму, сама повинна стати собі гуру і планувати кожну вилазку на зустріч з гравітацією як проект з багатьма параметрами-змінними, від опрацювання якого залежить ваше життя.

Всі ці гуру будуть говорити вам, що тільки вони розуміють суть речей, пізнали природу в немисленні, а у вас великий тягар йоги-розуму, який заважає вам досягати результатів, заморочки, зайва концептуалізація, надмірна начитаність та інші смертні гріхи. Тут діє таке ж правило як і в дитячому садку, школі або університеті "розумних не люблять тому майте це на увазі поговорити по душах за багаттям з черговим спортивним гуру.

Розумна і раціональна людина завжди вибере більш рідкісну та філігранну йогу розуму замість йоги тіла, якою володіє значно більша кількість істот. Адже отримати алмаз розуму набагато складніше, ніж алмаз тіла, тому партнери зі спорту і навіть локальні гуру, будуть готові в прямому сенсі підсвідомо вас убити на схилі, тут теж потрібне око та око.

У буддизмі Тибету аналогом спорту є таємне посвята каналів, вітрів і сфер структури вашої алмазної мережі. Всі ці йоги виконуються в парі з опорою на партнера і дістатися цього рівня буде важко обивателю. Тим більше, що таких гуру ще менше, ніж спортивних.

6.3 Баланс

Головний наркотик для йоги-розуму та йогі-тіла це цукор. Усі спорсмени як мінімум висять на кока-колі, ред-булі, гелях, стимуляторах. Баланс цих речовин і правильне харчування –ключ до швидкого відновлення після спортивних сесій, які так чи інакше потрібно буде заліковувати перед повноцінними сесіями йоги розуму. Ігнорувати допоміжні речовини на шляху спорту марнославно, але й зловживати не варто. Головний критерій, безперервність, ви повинні планувати вашу подорож таким чином, щоб ваша свідомість не похитнулася від раптової зміни гормонального фону, дефіциту того чи іншого паливного елемента.

7 Посвята всім святим програмістам

Складаю список святих програмістів. З кого почати? Почну а Ади Гольдберг та Алана Кея[1]. Потім Грінблат та Госпер[2]. Далі йде Кей Айверсон[3] і Джон МакКарті[4]. Далі йдуть Лінус Торвальдс[5], Дейв Катлер[6], Аветіс Теванян[7]. Не забуваємо про Джона Бакуса[8] та Робіна Мілнера[9]. Головним чином пам'ятаємо Ксав'є Лероя[10], Джо Армстронга[11], Саймона Пейтона-Джонса[12] і мало-відомого Ральфа Юнга[13]. У часи нюдьги молимося Джону Кармаку[14], Казунорі Ямаучі[15], Амерікан МакГі[16].

[1] автори Smalltalk
[2] автори TX-0
[3] автор APL
[4] автор LISP
[5] автор Linux
[6] автор Windows NT
[7] автор NeXT
[8] автор BNF
[9] автор ML
[10] автор OCaml
[11] автор Erlang
[12] автор Haskell
[13] автор формальної моделі Rust
[14] автор Quake
[15] автор Gran Turismo
[16] автор MacGee's Alice

Виділяємо окремо майстів нідерландської школи: Ерік Мейер[17], Хенк Барендрегт[18], Ніколас де Брейн[19]. Тері Кокан[20], Жерар Юе[21] топові математики-програмісти. Ми дякуємо Вінту Сьорфу[22], Леслі Лампорту[23], Алану Коксу[24], Тео де Раадту[25], Кліву Моулеру[26], Джефу Діну[27]. Не забуваємо про Фабріса Беллара, Тревіса Гейсельбрехта[28], Александра ван дер Грінтена[29]. Шануємо та поважаємо Ульфа Норела[30], Леонардо де Мура[31]. Відмічаємо Бена Фрая[32], Стефана Вольфрама[33] і Йоахіма Нойбузера[34]. Йохен Лидтке[35], Ада Лавлас[36], Девід Люкхем[37].

[17] автор LINQ
[18] автор лямбда-кубу
[19] автор фібраційної верифікації
[20] автор числення конструкцій
[21] математик логік
[22] автор TCP/IP
[23] автор LaTeX та TLA+
[24] автор Linux SMP
[25] атвор OpenBSD
[26] автор LAPACK
[27] автор leveldb, TensorFlow
[28] автор newos, BeOS, BeFS
[29] автор Managarm
[30] автор Agda
[31] автор Lean
[32] автор Processing
[33] автор Mathematica
[34] автор GAP
[35] автор L4
[36] піонер програміст
[37] піонер формальної верифікації