

(MoH)

MET[Ä]CADEMY OF HUMANITY
· (A) · {Ä} · [Ä] · {Ä} · (Ä) ·

DOCUMENT 7 · Research Essay · v1.0RC · 2026-08-17

ПІСЛЯ VIBE CODING

Як навчити машину не втрачати, навіщо ми це робимо

ПІСЛЯ VIBE CODING

Уяви, що ти нарешті вирішив побудувати дім. Не архітектурний куб із журналу, де на кухні ніхто ніколи не смажив цибулю, а на білому дивані, судячи з фотографій, сидіти заборонено якоюсь Женевською конвенцією. Нормальний дім. Той, у якому будуть ранкові чашки, друзі, секс, сварки через ті самі чашки, робота, книжки, діти, музика о другій ночі, старість і викрутка, яку через одинадцять років знайдуть у коробці з ялинковими прикрасами. Ти даєш майбутній системі бюджет, план ділянки, будівельні норми, кількість кімнат, фотографії інтер'єрів, які тобі подобаються. Вона рахує сонце, вітер, тепловтрати, каналізацію, навантаження, утеплення й видає прекрасний проєкт. Пожежна безпека настільки переконлива, що пожежа, побачивши документацію, тихо переосмислює своє життя й іде горіти в інший район.

І все одно це може бути не твій дім. Система знає, скільки квадратних метрів належить спальні, але не знає, що для однієї людини спальня у сімдесят сім років — це телевизор, книжка, зручне світло й тиша, а для іншої вона є одним із головних просторів близькості, і прокидатися між шафою та зарядкою для пилососа він якось не планував. Вона знає оптимальне співвідношення будинку й ділянки, але не знає, що ти без жалю віддаси двадцять квадратних метрів житла за великий сад. Вона прекрасно розмістить кабінет за всіма правилами, але не здогадається, що ти працюєш ночами й хочеш бачити, як над горизонтом з'являється перше світло. Вона навіть може знати, що ти любиш гостей, але не розуміти, чи це двоє близьких людей біля каміна, чи п'ятнадцять друзів із гітарами, вином і п'ятнадцятьма несумісними версіями того, хто з них уміє співати.

Усі параметри можуть бути виконані бездоганно. Просто ніхто не поставив головного питання: **яке життя має відбутися всередині цих параметрів?**

Приблизно в таку ситуацію ми зараз дуже швидко входимо з програмуванням. Колись програміст пояснював машині майже кожен рух: візьми це значення, поклади сюди, порівняй, повтори, переходи далі. Потім ми навчилися описувати структури, об'єкти, відношення, цілі шматки системи. Тепер можна сказати AI: «Зроби мені сервіс бронювання для маленького готелю, простий, красивий, із календарем, оплатою й мобільною версією, яка не виглядає як кара за гріхи», піти варити каву й повернутися до коду, бази даних, API і кнопки з border-radius 12 px, про який ніхто не просив, але машина, очевидно, вважає його необхідною умовою існування цивілізації.

Це не просто черговий productivity trick. На наших очах радикально скорочується відстань між «**я хочу**» і «**ось воно працює**». Те, що ще недавно вимагало невеликої команди й кількох місяців, іноді може народитися в однієї людини за вечір. Але казки попереджали нас про цю проблему задовго до комп'ютерів. Магія найнебезпечніша не тоді, коли не виконує бажання. Найцікавіше починається, коли вона виконує його **ідеально, але зовсім не так, як ти мав на увазі**.

Ти попросив стати найбагатшою людиною міста, а наступного ранку всі інші мешканці загадково зникли. Формально контракт виконано. У software проблема зазвичай виглядає менш театральна, але механіка знайома. Деся між твоїм наміром, тим, як AI його зрозумів, технічною архітектурою і реальною поведінкою системи може непомітно оселитися PAYMENT = AUTHORITY, UNKNOWN = FALSE, SIMILARITY = IDENTITY або CLIENT STATE = SERVER AUTHORITY. Інженер у цей момент уже чує запах диму. Для всіх інших переклад простий: система могла вирішити, що той, хто

платить, автоматично має право вирішувати; що «ми не знаємо» означає «ні»; що дві схожі людини чи ситуації можна вважати одним і тим самим; що версія реальності, яку приніс один учасник або пристрій, автоматично стає реальністю для всіх.

Код може бути красивим. Типи правильні. Тести зелені. Сервер задоволений. Ніхто нічого технічно не «зламав». Ми просто **зразково, акуратно й із чудовою інженерною дисципліною реалізували маленький майбутній апокаліпсис**.

Саме звідси в IMAGO виросло питання, яке ми зараз досліджуємо під назвою Meta.Semantic Programming. Не ще одна мова з новим різновидом дужок, людство вже пережило достатньо дужок. Питання значно старіше й людяніше: чи можемо ми переносити разом із командою ще й **причину, заради якої команда взагалі існує?** Що тут головне? Що можна змінити? Що змінювати небезпечно? Де ми знаємо, а де лише припускаємо? Що повинно залишитися істинним, навіть якщо сама форма задуму зміниться?

Бо ідея майже ніколи не народжується готовим технічним завданням. Спочатку вона може взагалі не мати точних слів. «Хочу дім». Потім починає розгортатися: хочу бачити захід сонця; хочу сад; хочу, щоб друзі могли залишитися на ніч; хочу працювати так, щоб робота не з'їдала спальню; не хочу рубати старе дерево заради ще одного паркомісця. Потім це стає вимогами, кресленням, матеріалами, будівництвом. І тільки коли ти нарешті заходиш усередину, виникає питання, яке не влезить у жоден Excel: **це воно?**

Software тепер проходить майже ту саму дорогу, тільки вся подорож може зайняти десять хвилин. Людський задум стає prompt, prompt стає внутрішньою інтерпретацією, інтерпретація стає архітектурою, архітектура — кодом, код — системою, а система починає щось робити зі світом. На кожному переході сенс може пошуховтись всього на кілька градусів. Один градус тут, два там, ще три на наступному переході — і в кінці корабель прибуває не на той континент, хоча кожен окремий поворот був цілком розумним. Якщо ми прискорили всі ці переходи в сотні разів, питання **що саме подорожує між ними** перестає бути філософським хобі й стає інженерною проблемою.

Ось тут і входить MSL. Найменш неправильно уявляти її поки що не як нову programming language, а як спробу дати задуму **паспорт для подорожі**. У ньому записано не тільки, що треба зробити, а й що не можна загубити, які перетворення допустимі, де лишається невизначеність, чому виникло певне рішення, для кого взагалі існує результат і що сталося із сенсом після переходу в інше тіло. Якщо ідея пройшла через п'ять аеропортів, нас цікавить уже не тільки зелена лампочка «багаж прибув». Хочеться ще перевірити, чи приїхали книжки, фотографії й бабусин лист, чи всередині тепер три шкарпетки та інструкція від мікрохвильовки.

Але тут наша поточна робота з M4M підсовує наступне, ще дивніше питання. Ми звикли думати, що задум має пережити переклад у **код**. А що, коли іноді має змінитися й саме обчислювальне тіло? Одній задачі потрібна майже годинникова детермінованість, іншій корисна контрольована випадковість, третя може народжувати форму поступово, через локальні правила й взаємодії, а не отримувати її готовою зверху. Тоді паспорт сенсу не повинен визначати, чи мандрівник летить літаком, їде потягом або пливе кораблем. Його задача інша: допомогти перевірити, що після всіх пересадок приїхав усе ще той самий мандрівник, а не дуже ввічливий незнайомець із його валізою.

І, можливо, не кожна ідея взагалі є кресленням. Деякі більше схожі на **насіння**. У кресленні ти намагаєшся заздалегідь визначити форму. У насінні важливіше зберегти походження, внутрішні обмеження, середовище, відносини й закони росту. Ти не знаєш точного положення кожного майбутнього листка, але все одно можеш відрізнити дуб від пластикової пальми. Для MSL це радикальна зміна перспективи: іноді треба берегти не готову форму задуму, а його здатність **залишатися собою, поки він росте**.

Тому RUN SUCCESSFUL більше не може бути останнім словом. Воно повідомляє лише одне: **щось виконалося**. А нас дедалі більше цікавить інше: **чи виконалося саме те, що ми мали на увазі?** Літак теж може бездоганно пройти маршрут. Залишається одна маленька, іноді дуже емоційна подробиця: це був той аеропорт?

І після прибуття хочеться отримати не тільки результат, а ще й **чек різниці**. Не бухгалтерський роман на сімсот сторінок, а зрозумілий гесеірт: ось що збереглося; ось що ми додали; ось тут зробили припущення; ось тут щось втратили; ось тут два значення випадково злилися в одне; а тут ми вже не впевнені, чи результат усе ще є дитиною початкової ідеї, чи його тихо підмінили в пологовому. Втрата не завжди є помилкою. Небезпека починається, коли втрата поводить так, ніби нічого не сталося.

І тут програмування несподівано торкається речей, які людство робило завжди. Ми переносили сенс через міфи, закони, пісні, математику, картини, театр, молитви, жарти, родинні історії й фразу «не забудь купити хліб», семантична вага якої різко залежить від того, хто її сказав і скільки разів ти вже повертався додому без хліба. Сенс ніколи не

жив тільки в словах. Він живе у відношеннях, контексті, очікуванні, пам'яті, інтонації, у тому, що сказано, і часом ще сильніше в тому, що залишилося неказаним.

Саме тому MSL поступово рухається від вузької ідеї «перекласти інформацію» до ширшої задачі **міграції сенсу між різними тілами**. Текст може стати схемою. Схема — кодом. Код — поведінкою. Поведінка — досвідом. Іноді нове тіло навіть відкриває в задумі те, чого стара форма не могла показати. Але кожна така зміна повертає нас до одного питання: **що ця річ повинна зберегти, щоб усе ще залишатися собою?**

Є й інша проблема. Можна зберегти абсолютно всі факти й усе одно втратити сенс. Ті самі події, розказані в іншому порядку, можуть стати іншою історією. Той самий набір фактів, поданий іншими порціями, може привести людину до іншого висновку. Тому доводиться дивитися не тільки на інформацію, а й на **траєкторію**, якою вона проходить через систему й через людину. Недостатньо знати, які цеглини привезли. Іноді важливо ще знати, яку стіну з них побудували й з якого боку в ній залишили двері.

Уяви тепер машину, яка не просто відповідає: «Готово». Вона каже: «Ось як я тебе зрозуміла. Ось що я зберегла. Ось тут зробила припущення. Ось тут частина сенсу могла загубитися. Я бачу три способи реалізації, але третій змінює річ, яку ти раніше називав важливою. Можливо, тут краще повернутися на крок назад».

Це вже не просто швидший спосіб створювати software. Це інша якість відношення між наміром і машиною.

Можливо, частину людського сенсу взагалі неможливо повністю формалізувати. І це не буде поразкою. Якщо ми знайдемо межу, за якою система повинна сказати: «Я розумію це недостатньо добре, тут має залишитися людина», це може виявитися одним із найцінніших результатів усього дослідження. Інтелект дорослішає не тоді, коли отримує відповідь на все. Він дорослішає тоді, коли починає розрізняти знання, припущення й таємницю.

Десятиліттями ми ставили комп'ютеру одне велике питання: **як навчити тебе робити більше?** AI радикально прискорив відповідь. Meta.Semantic Programming починається з іншого питання. Воно трохи слабше виглядає на слайді для інвесторів, але, можливо, значно важливіше для всіх інших: **як навчити тебе не втрачати, навіщо ми взагалі це робимо?**