Про кар'єрне зростання, рівні компетенції та методи співбесіди

Кар'єрне зростання: як це працює

У ринковій економіці кар’єрне просування базується на результатах, а не на формальних званнях чи дипломах. Якщо ви хочете вищу посаду — спершу покажіть, що здатні виконувати її обов’язки. Не потрібно чекати дозволу чи сертифікатів — потрібні реальні результати, які підтверджують вашу компетентність.

Головне — наявність прозорих і об’єктивних критеріїв оцінки. Без них — суб’єктивна думка керівництва може стати єдиним бар’єром або пропуском угору.

Принцип Пітера: пастка ієрархії

"У будь-якій ієрархії працівник піднімається до рівня власної некомпетентності."

Це означає, що людей підвищують, поки вони справляються. Але рано чи пізно їх переводять на рівень, де вони вже неефективні — і залишають там. Результат: система наповнюється некомпетентними керівниками, що знижує загальний рівень ефективності.

Щоб цього уникнути, грамотні організації спочатку дають складніші задачі в межах поточної посади. І лише якщо людина справляється — підвищують офіційно. Так перевіряється не лише потенціал, а й реальна готовність.

Навчання потрібне, але не є остаточним доказом компетентності.

Чи потрібне навчання для кар’єрного зростання?

Так, навчання важливе, але не достатнє саме по собі. Знання, сертифікати і дипломи — це лише потенціал. Реальну готовність до посади показують **результати на практиці.

Інакше кажучи:

Навчання — це підготовка.
Результати — це доказ.

У ринковій системі перш за все цінується практична компетентність. Ви можете не мати формальної освіти, але якщо показуєте результат — це важливіше за “корочку”.

Водночас без навчання результату часто не буде. Тому ефективна стратегія — це:

1. Навчаєшся →
2. Пробуєш на практиці →
3. Демонструєш результат →
4. Отримуєш підвищення.

—---

Модель компетенції SWECOM
Модель компетенції SWECOM, крім загальних компетенції вводить компетенції по конкретним навичкам та вмінням.
Модель компетенцій у галузі програмної інженерії (SWECOM, IEEE) організована за областями навичок (наприклад, вимоги до програмного забезпечення), навичками в межах цих областей (наприклад, збір вимог до програмного забезпечення) та діяльністю в межах навичок (наприклад, створення прототипу для збору вимог).

Діяльність класифікується за п’ятьма рівнями компетенцій:

• Технік (Technician)

• Практик початкового рівня (Entry Level Practitioner)

• Практик (Practitioner)

• Технічний керівник (Technical Leader)

• Старший інженер-програміст (Senior Software Engineer)

Загалом, Технік виконує інструкції, Практик початкового рівня допомагає у виконанні діяльності або виконує її під наглядом; Практик виконує діяльність з мінімальним або без нагляду; Технічний керівник керує окремими людьми та командами під час виконання діяльності; Старший інженер-програміст змінює існуючі методи та інструменти, а також створює нові. Деякі організації можуть об’єднувати рівні Техніка і Практика початкового рівня. Старший інженер-програміст може виконувати роль "головного інженера" (CTO) для організації з розробки програмного забезпечення, і деякі з них можуть бути визнані експертами галузі, які роблять внесок у формування та розвиток професії програмної інженерії.
Крім діяльності, визначеної на різних рівнях компетенції, додатковою компетенцією всіх інженерів-програмістів є навчання та наставництво інших, у відповідних випадках, у методах, інструментах і техніках, які використовуються для виконання цих діяльностей. Наприклад, Технік або Практик початкового рівня можуть навчати або наставляти інших у використанні інструментів керування конфігурацією для виконання своїх завдань, або Технічний керівник може навчати чи наставляти Практика в тому, як проводити інспекції та огляди.

—--

Bottom-up та top-down підходи до питань на співбесіді. Якщо людина розуміє загальну архітектуру та методики, це скоріше за все Сеньйор. Якщо ж вона не знайома з цими аспектами, то, ймовірно, це Мідл або Джуніор. Варто зазначити, що навіть Сеньйори можуть забувати, здавалося б, очевидні, але дуже специфічні речі. Проте архітектуру вони повинні знати, адже їхня основна задача — забезпечити надійність, цілісність і масштабованість додатку. Мідли та Джуни можуть детально розуміти конкретні аспекти, але не завжди мають глибоке розуміння найкращих практик та архітектурних рішень.

Bottom-up підхід передбачає, що розробник починає з найменших деталей і поступово рухається до більш загальних. Кандидат, який відповідає на питання в цьому стилі, може зосереджуватися на технічних аспектах, деталях реалізації та конкретних технологіях. Він добре знає, як працюють окремі компоненти системи, але може не зовсім розуміти, як ці

частини взаємодіють в межах всієї архітектури. Такий підхід часто притаманний Мідлам або Джуніорам, які мають хороші технічні навички, але не завжди орієнтуються в глобальному баченні системи чи не мають досвіду в розробці архітектурних рішень.

Top-down підхід передбачає початок з загальної картини: розробник спочатку вивчає архітектуру, принципи взаємодії компонентів, а потім заглиблюється в конкретні технічні деталі. Такий підхід більше властивий Сеньйорам, адже вони мають досвід побудови масштабних додатків і добре розуміються на архітектурних рішеннях. Сеньйор може не зосереджуватись на деталях реалізації кожної функціональності, але добре орієнтується в загальній структурі системи, може передбачити наслідки тих чи інших рішень для масштабу та підтримки додатку.

Сеньйор має розуміти, як спроектувати систему, забезпечити її масштабованість, надійність і продуктивність. Він часто приймає рішення на рівні архітектури і поважає "найкращі практики". Тому питання на співбесіді будуть спрямовані на його здатність бачити загальну картину і приймати обґрунтовані рішення.

Мідл може бути хорошим спеціалістом, що володіє конкретними технологіями, але йому може бракувати досвіду в побудові складних архітектурних рішень. Він може бути здатний вирішувати технічні проблеми, але не завжди розуміє, як ці рішення впливають на всю систему.

Джуніор орієнтований на вирішення конкретних завдань, але йому часто бракує досвіду для аналізу архітектури в контексті великого проекту.

Сеньйор зазвичай може описати, чому обирає саме таку архітектуру або технологію, як це допоможе додатку в масштабі, як будуть взаємодіяти компоненти, які можливі проблеми можуть виникнути на великому проєкті.

Мідл може добре розібратися в технології, в деталях реалізації, але не завжди зможе зразу побудувати архітектурну картину великого проєкту. Він може зазвичай працювати в рамках визначених вимог, але не так вільно вносити зміни в структуру додатка.

Джуніор може не бачити взаємозв'язків між компонентами, а лише вирішувати локальні задачі, без глибшого розуміння того, як це вписується в загальний контекст.

Ключова різниця між рівнями полягає в здатності бачити "велику картину". Сеньйор повинен орієнтуватися в загальній архітектурі, щоб будувати і підтримувати цілий додаток, тоді як Мідл і Джуніор більше фокусуються на дрібніших деталях і не завжди мають достатньо досвіду для роботи з великими, складними системами.

—---

Про методи співбесіди

1. Питання про досвід. Про складні задачі.

Плюси: Дають в загальному зрозуміти рівень кандидата.
Мінуси: Не завжди релевантні, тобто не дають змогу точно оцінити.

2. Постановка закритих питань (бінарних питань), які потребують короткої та однозначної відповіді.

Плюси: Легко виявити помилку.
Мінуси: Не показують цілісність знань та взаємозв'язки між ними.

3. Відкриті питання (open-ended question).

Плюси: Можуть виявити цілісність теоретичних знань.
Мінуси: Потребують більше часу та аналізу.

4. Ситуативне інтерв'ю: під час інтерв'ю кандидату пропонується вирішити змодельоване завдання, певну ситуацію (наприклад, вихід за межі спринту, або збір вимог).

Плюси: Показує практичні вміння кандидата.

5. Лайф кодинг.

Плюси: виявляє практичні навички кандидата, та його вміння.
Мінуси: Потребує багато часу і не точно відповідає умовам реальної роботи.

6. Метод STAR—це техніка для відповідей на співбесіді, що включає чотири елементи: Ситуація (опис контексту), Завдання (виклик, який виник), Дія (конкретні кроки, які ви зробили), та Результат (ваші досягнення або вплив дій).

S – situation – ситуація,

T – task – завдання,

A – action – дія,

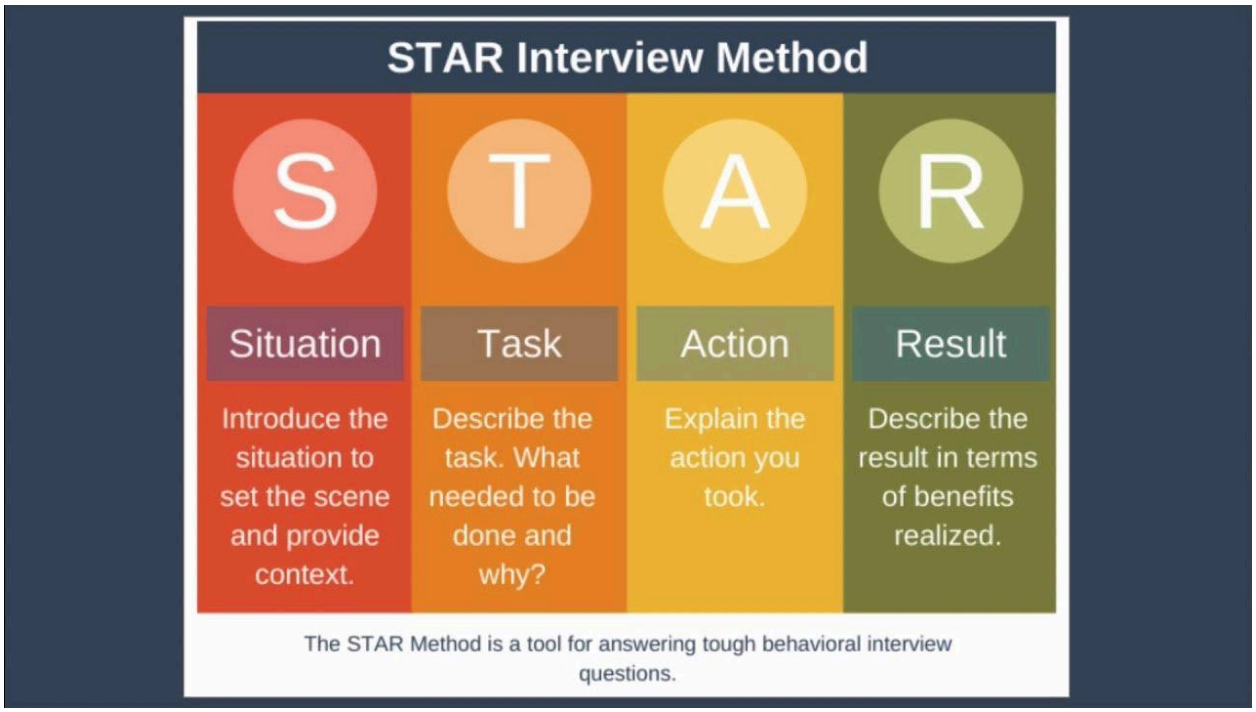
R – result – результат.

Метод STAR найдоречніше застосовувати при відповідях на запитання, які починаються зі слів: "Опишіть ситуацію, коли ви…".

Тривалість співбесіди бажано, щоб була не більше двох годин, краще година, півтори (тобто дві академічні години).

—--

STAR



STAR — це метод відповіді на поведінкові запитання під час співбесіди. Назва походить від абревіатури англійських слів:

S – Situation (ситуація)
T – Task (завдання)
A – Action (дії)
R – Result (результат)

Цей метод допомагає структурувати відповідь так, щоб вона була чіткою, конкретною і демонструвала ваші навички, досвід і підхід до вирішення проблем.

Приклад

Питання: "Розкажіть про ситуацію, коли вам довелося приймати складне рішення."

S (Ситуація): Під час роботи над важливим проєктом у нас виник конфлікт в команді — два члени команди мали різні підходи до виконання одного завдання, що загрожувало затримкою.

T (Завдання): Мені потрібно було знайти оптимальний підхід для вирішення конфлікту та обрати стратегію, яка дозволила б команді рухатися вперед без зупинок.

A (Дії): Я організував зустріч, де кожен міг висловити свою точку зору. Після цього я проаналізував ситуацію і вибрав найбільш збалансований варіант, який задовольняв би обидві сторони, водночас не затримуючи виконання проєкту.

R (Результат): Завдяки цьому рішенню ми швидко досягли консенсусу, і проєкт не зазнав затримок. Команда продовжила працювати згуртовано, а результат був дуже успішним, що допомогло покращити атмосферу в колективі.