

איך עובד Forward Deployed Engineering ב-OpenAI: שיחה עם Colin Jarvis

ראש צוות ה-FDE של OpenAI מסביר איך מביאים ארגונים ל-production: מטמיעים צוות אצל הלקוח, מתחילים מ-evals ובונים guardrails דטרמיניסטיים. מה שחוזר על עצמו אצל לקוחות שונים הופך למוצר

Colin Jarvis ומראיין ששמו לא נאמר · המקור כנראה ערוץ OpenAI (הלוגו מופיע על המסך), הועלה מחדש ב-X על ידי Hanako · תאריך ההקלטה לא צוין, הפרסום מחדש ב-4.8.2026
ראיון ללא שקפים, עם כמה צילומי מסך ודמו · משך: כ-40 דקות · X
סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זה ראיון של כ-40 דקות עם Colin Jarvis, ראש Forward Deployed Engineering (FDE) ב-OpenAI. לדבריו הוא הצטרף לחברה בנובמבר 2022, החודש שבו יצא ChatGPT, כשהיה אחד משניים-שלושה עובדים באירופה. המראיין לא מציג את עצמו. מהשיחה עולה שהתחיל את הקריירה כ-FDE ב-Palantir, ושהוא מזכיר משחק שאוהבים ב-"Ultimater". זה כנראה Altimeter, אבל זו השערה שנשענת על תמלול אוטומטי. בפריימים מופיע הלוגו של OpenAI, וזה מרמז שהמקור הוא ערוץ של OpenAI.

הטענה המרכזית: מה שמביא ארגונים ל-production בצורה עקבית הוא לא תמיכה טכנית מרחוק. צריך צוות קטן שיושב אצל הלקוח, בוחר בעיה עם סיכון אמיתי, בונה סביבה evals ו-guardrails, וזז ללקוח הבא אחרי שהבעיה הקשה נפתרה. לפי Jarvis, מה שמתגלה כחוזר אצל כמה לקוחות הופך למוצר. כך נוצרו Swarm, ה-Agents SDK, ו-AgentKit.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, ומה לא נאמר.
- 2. **Morgan Stanley:** מקרה הבוחן הראשון, ולמה בניית האמון לקחה יותר זמן מהטכנולוגיה.
- 3. **מוליכים למחצה:** eval-driven development וסוכן שמטפל בבאגים של הלילה.
- 4. **מייעוץ למוצר:** המסלול AgentKit → Agents SDK → T-Mobile → Swarm → Klarna.
- 5. **הדמו:** תכנון מחדש של שרשרת אספקה בעקבות מכס, עם בדיקות דטרמיניסטיות.
- 6. **הכלכלה של צוות FDE:** גודל הצוות, סף הערך, והטעות של הכללה מוקדמת מדי.
- 7. **סבב מהיר:** על מה הוא long ועל מה short, כלים לא מוערכים, ותחזית ל-2026.
- 8-9. **מסקנות ומילון:** מה כדאי לעשות עכשיו, שאלות פתוחות ומונחים.

הערת מקור המסמך נוצר אוטומטית מהתמלול של ההקלטה ומהפריימים שחולצו ממנה. שמות ומונחים עלולים להשתבש בתמלול האוטומטי, למשל "codecs" במקום "ChadGPT", "Codex" ו-"Palantir". הציטוטים מועתקים מילה במילה מהתמלול ונבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים. הכיתוב של מי שהעלה את הסרטון ב-X (כולל הטענה על שכר של עד 785 אלף דולר) לא מופיע בתמלול, ולכן לא שימש כמקור.

1. תקציר מנהלים

Jarvis מתאר את FDE ב-OpenAI כצוות זריז ויקר שמטרתו לייצר מוצר ולא הכנסות משירותים. לאורך השיחה הוא חוזר על אותה נוסחה: בוחרים בעיה עם סיכון גבוה, מגדירים evals עוד לפני הקוד, נותנים לחלקים הדטרמיניסטיים לעשות את מה שאסור לטעות בו, ומחליצים את מה שחוזר על עצמו.

- **האמון הוא צוואר הבקבוק, לא הטכנולוגיה:** ב-Morgan Stanley הבעיה הטכנית נפתרה תוך 6–8 שבועות. אחרי זה נדרשו עוד כארבעה חודשים של פיילוטים ו-evals. בסוף כ-98% מהיועצים אימצו את הכלי, והשימוש בדוחות המחקר גדל פי 3 [3:03].

- **קוד שנכתב על ידי LLM לא גמור בלי evals:** זה עקרון ה-eval-driven development. בפרויקט המוליכים למחצה הצוות עשה fork ל-Codex והוסיף טלמטריה כדי לבנות evals מפורטים יחד עם המומחים של הלקוח [7:42, 9:14].

- **יעד של 50% חיסכון בזמן המהנדסים, 20%–30% בפועל:** בחברת מוליכים למחצה אירופית נבנו 10 use cases לאורך שרשרת הערך. בחטיבות הראשונות החיסכון כרגע 20%–30% [6:07].

- **כל מה שאפשר לעשות דטרמיניסטית, עושים דטרמיניסטית:** בדמו של שרשרת האספקה את האילוצים בודק קוד ולא המודל. למשל: לפחות שני ספקים לצמיגים, זמני אספקה, כיסוי של כל החומרים [17:03].

- **מוצר נולד מהכאב של לקוח:** הפרמטריזציה של הוראות וכלים אצל Klarna הגדילה את מספר המדיניות שהמערכת מכסה מ-20 ל-400. היא הפכה ל-Swarm, נבדקה ב-T-Mobile (מורכבת פי 10 לדבריו), ומשם צמחו ה-Agents SDK ו-AgentKit [10:45–12:19].

- **צוות zero-to-one שלא נשאר אצל הלקוח:** FDE פותר את הבעיה הקשה ומעביר אותה למהנדסי הלקוח או לשותף. הפרויקט הראשון כ-20% רב-פעמי, אחרי שניים-שלושה נוספים מגיעים לכ-50%, ואז מעבירים לחלק של החברה שעובד בקנה מידה [13:51, 26:28].

- **סף הכניסה: עשרות מיליונים:** הצוות לוקח בעיות שערכן ללקוח הוא עשרות מיליוני דולרים ולפעמים מיליארדים בודדים. הצוות גדל משני אנשים בתחילת השנה ל-39, ויגיע ל-52 עד סוף השנה [21:51, 23:21].

- **הטעות הגדולה: הכללה מוקדמת מדי:** הם לקחו פיצ'ר מ-ChatGPT וחיפשו לו בעיה, וזה לא עבד. כשצללו לעומק הבעיה של לקוח מסוים, כמעט תמיד יצא בסוף משהו כללי [28:01].

מה לא נאמר

- **שכר:** הטענה מהכיתוב ב-X על שכר FDE של עד 785 אלף דולר לא מופיעה בשיחה בכלל.

- **שמות לקוחות:** חברת המוליכים למחצה ויצרנית הרכב מ-APAC לא נקראות בשמן. הדמו הוא "toy version" ולא המערכת האמיתית.

- **מספרים בלי מתודולוגיה:** 98% אימוץ, פי 3 שימוש ו-20%–30% חיסכון נאמרים בלי להסביר איך נמדדו, מה היה ה-baseline או כמה ערך הפרויקט עלה.

- **הכלכלה של הצוות עצמו:** לא נאמר כמה גובים, אם בכלל, ואיך מודדים הצלחה של product bet ברמת הצוות.

- **תוצאות סוכן ה-debug:** המצב שבו רוב הבאגים מתוקנים עד הבוקר הוא חזון ל"mid-next year". אין נתון על כמה באגים הוא מתקן היום.

2. Morgan Stanley: הלקוח שהוביל להקמת הצוות



NEWSLETTERS • CFO DAILY

MIT report: 95% of generative AI pilots at companies are failing



By Sheryl Estrada

Senior Writer And Author Of CFO Daily
August 18, 2025, 6:54 AM ET

Add us on



הכתבה שהמראיין מזכיר בפתיחה: 'MIT report: 95% of generative AI pilots at companies are failing'. על הרקע הזה הוא מציג את Jarvis כמי שנמצא ב-5% שמצליחים.

Jarvis מתאר את 2022–2023 כתקופה שבה היה קשה להפיק ערך מהמודלים. לקוחות הגיעו ל-production, אבל לא באופן עקבי, ובכל פעם זה לקח הרבה זמן. מה שעבד שוב ושוב היה לשבת אצל הלקוח, להבין את התחום ולעבוד עם המשתמשים בשטח. מודל התמיכה הטכנית שלא מתערב מקרוב לא הביא לקוחות ל-production בצורה אמינה [1:31].

Morgan Stanley היה לדבריו הלקוח הארגוני הראשון שעלה ל-production עם GPT-4, ב-2023. המטרה הייתה לשים את דוחות המחקר של החברה בידי כל יועצי ניהול העושר. Jarvis מדגיש שהם בחרו ליבה עסקית ולא מקרה שולי, ושזה דפוס שהוא רואה אצל מי שמצליח. RAG עוד לא היה מושג מוכר, אז הצוות עשה retrieval tuning, הריץ פיילוטס, והיועצים תייגו דאטה.

Colin Jarvis [3:03] the main technical hurdle was probably solved within like six to eight weeks. — But it took a further like four months of like doing pilots, collecting evals, and iterating to get to the point where the advisors actually trusted it

התוצאה לפי Jarvis: כ-98% מהיועצים אימצו את הכלי, והשימוש בדוחות המחקר גדל פי 3. הוא מסביר את הפער בין שמונה שבועות לשישה חודשים בכך שמדובר בסביבה רגולטורית עם דרישת דיוק גבוהה, בזמן שהטכנולוגיה הסתברותית. לכן צריך מסגרת עבודה מתאימה, וצריך לתת למשתמשים כלים לאמת את התוצאות בעצמם.

3. מוליכים למחצה: eval-driven development וסוכן debug



דף המוצר של 'Codex: One agent for everywhere you code' ופקודת ההתקנה `@openai/codex -g i npm`. זו נקודת ההתחלה של סוכן ה-`debug` אצל הלקוח.

לפי Jarvis זה אחד הפרויקטים הגדולים של הצוות כרגע. חברת מוליכים למחצה אירופית ביקשה למצוא את מוקדי הבזבז הגדולים בכל שרשרת הערך: תכנון, `verification` ומדידת ביצועים אחרי שהשבב נשלח ללקוח. הצוות ישב באתר הלקוח כמה שבועות ובחר ב-`verification`, עבודה שהמהנדסים לא אוהבים ושלפי Jarvis תופסת 70%–80% מהזמן שלהם. בסך הכול נבנו 10 `use cases`. היעד הוא 50% חיסכון בזמן המהנדסים, ובחטיבות הראשונות הם עומדים כרגע על 20%–30%.

הדוגמה המרכזית: בדיקות רצות כל לילה, ובבוקר כל מהנדס מוצא מאות באגים. הפתרון נבנה בשלבים. בהתחלה המודל רק חקר את הבאג וכתב טיקט עם אבחנה משוערת. כשהמהנדסים התחילו לסמוך עליו, הוא התחיל לתקן ולפתוח PR. אז התברר שחסר משהו:

— Colin Jarvis [6:07] we discovered actually we need to give the model an execution environment so it can test its code

כך נוצר מה שהם קוראים לו `debug investigation and triage agent`. החזון ל"mid-next year" הוא שכשהמהנדס מגיע בבוקר רוב הבאגים כבר תוקנו, והקשים מתועדים היטב. היום טיפול בערימת הבאגים לוקח לפעמים 4–6 שבועות במקביל לעבודה השוטפת. הצעד הראשון היה `fork` ל-Codex (בתמלול: "codecs") עם טלמטריה. הם מיפו עם המומחים של הלקוח את המסלול שמהנדס היה עובר, בערך 20 פעולות, ובנו ממנו סט מתויג [7:42].

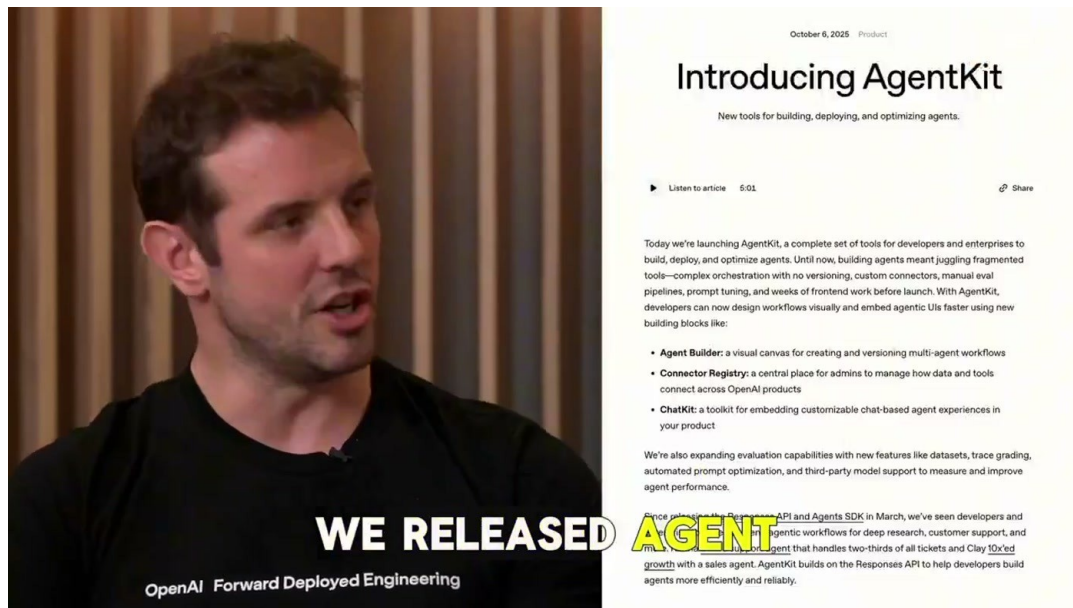
— Colin Jarvis [9:14] We call it eval-driven development where effectively a bit of LLM written code is not done until you have a set of evals that verify the efficacy

ומה קורה כשהמודלים משתפרים? אם המודל יעבור את סט ה-`evals` בלי שינויים, מצוין. בפועל אנשי ה-FDE מוסיפים `scaffolding`: מחליטים אילו חלקים חייבים להיות דטרמיניסטיים, ואת הסימולטור הקיים של הלקוח מפעילים כ-`tool` ולא נותנים למודל לנחש את התוצאה.

4. מייעוץ למוצר: Klarna → Swarm → AgentKit

המראיין שואל את השאלה הצפויה: האם FDE זה לא פשוט ייעוץ? Jarvis עונה עם הדוגמה של Klarna מ-2023. היה קשה להגדיל את מערכת שירות הלקוחות כשכותבים `prompt` ידני לכל מדיניות ויש 400 כאלה. הצוות עשה פרמטריזציה להוראות ול-`tools` ועטף כל `intent` ב-`evals`, וכך עבר מ-20 מדיניות ל-400 ויותר. את השיטה הם הפכו ל-`framework` פנימי בשם Swarm ופרסמו אותו כקוד פתוח.

במקביל התחילה עבודה עם T-Mobile, שלדבריו מורכבת פי 10 מ-Klarna בנפח, במספר המדיניות ובמורכבות שלהן. ה-framework עבד שם עם כמה הרחות. לקוחות ב-production, יחד עם הכוכבים ש-Swarm צבר ב-GitHub, שכנעו את צוות המוצר להקים צוות שבנה את ה-Agents SDK, ומאוחר יותר את AgentKit, ה-visual builder [12:19].



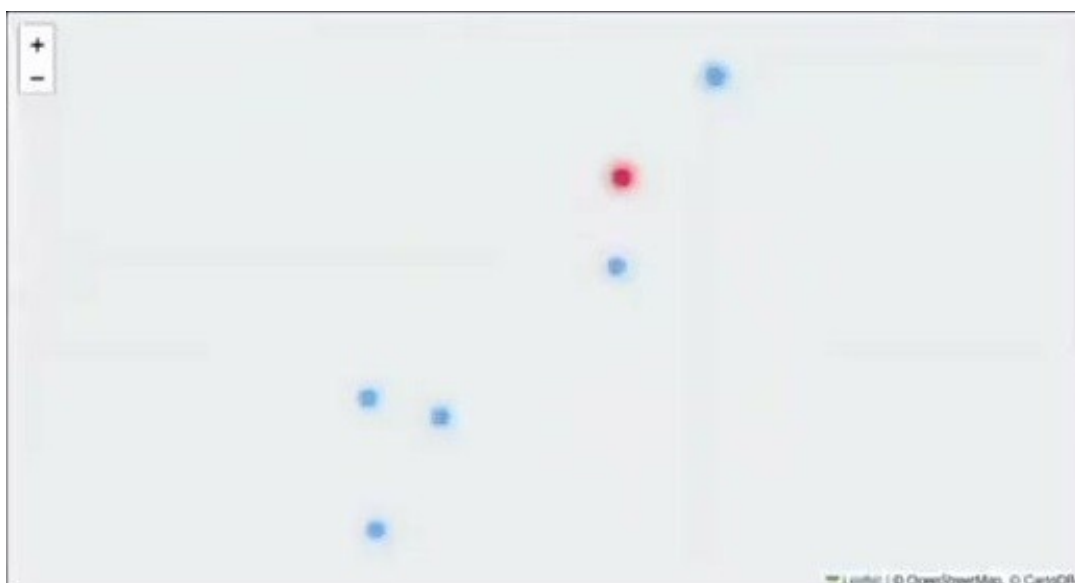
פוסט ההכרזה 'Introducing AgentKit': ערכת כלים לבנייה, פריסה ואופטימיזציה של agents, עם Agent Builder, Connector Registry, ChatKit-י.

— המראיין, מצטט את הבוס לשעבר שלו ב-Palantir (בתמלול: "Shams Shankar", השם לא ודאי) [12:19] the job of a forward-deployed engineer is to eat pain and excrete product.

מה הלקוח מקבל בסוף? Jarvis מונה שני דברים. הראשון הוא תוכנה שמחזירה השקעה שאפשר להראות, ושעוברת למהנדסי הלקוח או לשותף שיתפעל אותה. השני הוא דרך עבודה: מתזמרים את התהליך מקצה לקצה, עוטפים אותו ב-evals, מציבים guardrails לדברים שאי אפשר לשלוט בהם בזמן ריצה, ובודקים עקביות מול סט evals שהולך וגדל.

— Colin Jarvis [13:51] Because we see our FDE team very much as a zero-to-one team

5. הדמו: מכס של 25% ושרשרת אספקה שמתוכננת מחדש

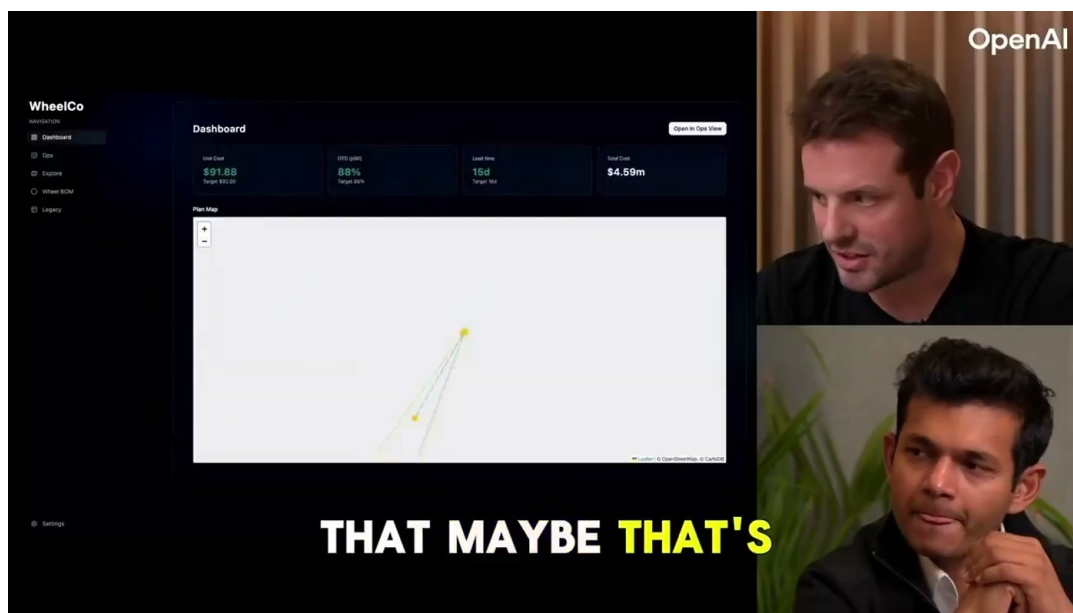


דשבורד הדמו: מפה עם נקודות של ספקים ומפעלים, ובצד ימין פאנל שיחה עם ה-LLM שמשמש כ-orchestrator.

הדמו הוא גרסה מצומצמת ("toy version") של מערכת שנבנתה ליצרנית רכב ב-APAC. לפני המערכת, כל שיבוש באספקה דרש טלפונים בין חטיבת הייצור, הלוגיסטיקה והמפעלים. הצוות לא העביר את הדאטה למקום אחר. הוא בנה מעליה כמה API-ים, ומעליהם LLM שמשמש כ-orchestrator, מריץ שאילתות ומחזיר ניתוח ראשוני שקודם דרש שעות או ימים [15:26].

התרחיש בדמו: מכס של 25% על סחורות מסין לדרום קוראיה. מעל כל תוכנית שהמודל מציע רצות בדיקות דטרמיניסטיות: לפחות שני ספקים לצמיגים, אילוצים על זמני אספקה, כיסוי של כל החומרים. בגלל שהלקוח היה שמרני, הוסיפו גם שלב שבו המודל מסביר את ההיגיון שלו, widgets להצגה, וטבלאות מפורטות שבהן אפשר לבדוק בעצמך שהמכס הוחל על כל החלקים שמקורם בסין [18:41].

Colin Jarvis [17:03] Like whenever you can use determinism, do it. And then use LLMs for what — they're best at, which is like using this probabilistic approach where there's some kind of like nuance required as to what you need to do



מסך ה-Dashboard בדמו: כרטיסי מדדים בשורה העליונה ומתחתם מפה עם מסלולי האספקה אחרי התכנון מחדש.

Jarvis אומר שהערך האמיתי הוא בשלב הבא. במקום לבקש מהמודל לנחש את התמהיל הנכון של מפעלים, נותנים לו גישה לסימולטור עם אותם פרמטרים שמשמש עסקי מנוסה היה משנה. בדמו הוא מריץ חמש ריצות אופטימיזציה, מציג טבלה של עלות מול זמן אספקה וממליץ על אחת מהן. התוכנית שנבחרה עוברת את הבדיקות הדטרמיניסטיות, ועל המפה רואים שחלק מהייצור עבר ממפעל בסין למפעל בטיוואן [21:51]. לדבריו, בעבודה האמיתית לקוחות כבר מריצים בגישה agentic מאות או אלפי סימולציות במצב offline ומקבלים בחזרה המלצה מתועדת.

6. הכלכלה של צוות FDE: גודל, סיכון וטעויות

המראיין מציין ש-Palantir מכניסה כ-5–6 מיליון דולר ACV חציוני ללקוח, ושואל מה סף הכניסה אצל OpenAI. Jarvis אומר שהצוות לא אמור להיות עצום, ושהם בוחרים בעיות שעשויות להפוך לפלטפורמה או לקדם את המחקר. היעד הוא בעיות שערכן ללקוח "tens of millions, sometimes the low billions in terms of value" [21:51]. גודל הצוות: בתחילת השנה הוא ועמיתו Arno בפרז, שהתחילו עם John Deere. היום הם 39, ועד סוף השנה יהיו 52 [23:21].

העצה שלו למייסדים: להחליט מראש מה מטרת צוות ה-FDE. הכנסות משירותים או product bets הם שני מודלים שונים, ומי שמנסה לעשות את שניהם בדרך כלל נגרר לשירותים. ב-OpenAI הקיבולת של הצוות מחולקת לשני צירים. צד אחד הוא השערות מוצר, כמו שירות לקוחות או כתיבת מסמכי ניסויים קליניים, שמחפשות את שותף התכנון המתאים. הצד השני הוא תעשיות עם בעיות שמעניינות את המחקר, כמו מוליכים למחצה ומדעי החיים [26:28].

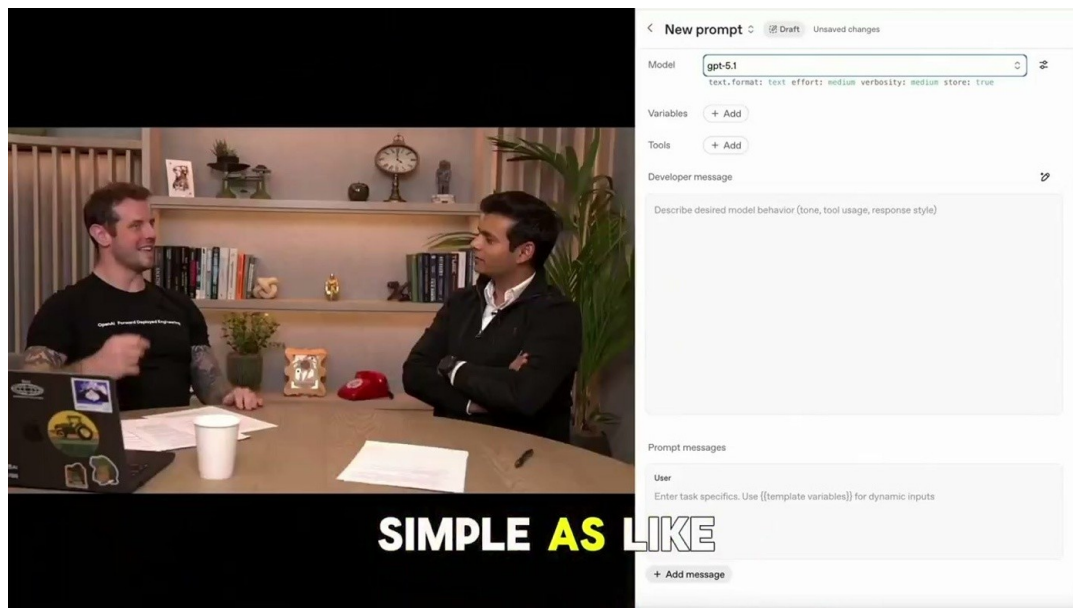
Colin Jarvis — על כשל שראה בחברות ייעוץ [24:57] the short-term lure of services revenue just starts to drag everyone in that direction

הטעות שהם עשו [28:01] "one of the biggest ones is probably generalizing too early", לפי Jarvis, הם לקחו יכולת מ-ChatGPT שנראתה כמו פתרון כללי לארגונים וחיפשו לה בעיה, ודילגו על שלב ה-zero-to-one. כשצללו לבעיה ספציפית של לקוח, כמעט תמיד יצא בסוף משהו שאפשר להכליל. לדבריו "doing what doesn't scale" הוא אחד מכללי היסוד של הצוות [29:34].

7. סבב מהיר: long, short, כלים ותחזית

Long: שכבת התרגום בין הדאטה הגולמי ללוגיקה העסקית. לדבריו זה המקום שבו FDE משקיעים את רוב הזמן כשמביאים מערכת ל-production. חיבורי MCP לבד לא מספיקים, וצריך שכבת לוגיקה קלה שאומרת למודל מתי לשלב, למשל, את ה-data warehouse עם התוכן של SharePoint. הוא מזכיר כלי metadata ותיקים (בתמלול "Calibra", כנראה Collibra), והמראיין משווה ל-ontology של Palantir [30:00–32:38].

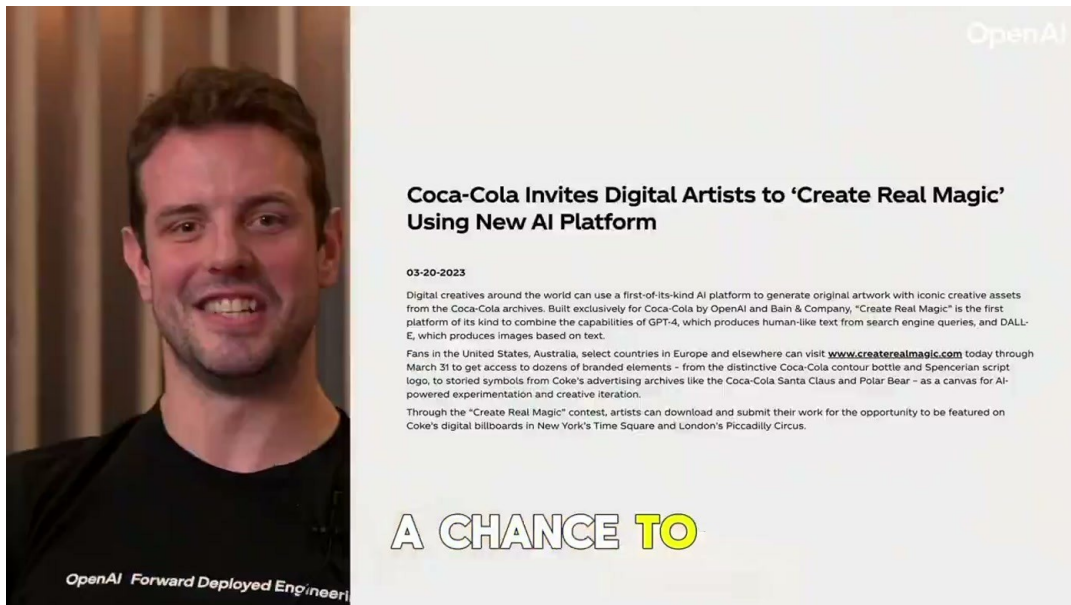
Short: חברות שמנסות רק חוליה אחת ברשרת evals, labeling, tracing, orchestration, ו-guardrails. ארגונים לא רוצים לחבר חמישה כלים, הם רוצים "one throat to choke with this problem" [32:38].



OpenAI Playground: מנך 'New prompt' עם בחירת מודל, Developer message ו-Prompt messages. זה הכלי ש-Jarvis מכנה הכי לא מוערך.

כלים: Codex הוא הכלי שהוא אוהב, אבל לדבריו הוא לא "underrated". הוא מספר שהפעיל אותו, נכנס לארבע שעות של פגישות, וכשחזר העבודה הייתה גמורה. הכלי שלדעתו לא מקבל מספיק הערכה הוא OpenAI Playground ב-platform.openai.com, כולל voice playground ל-Realtime API. השיטה שלו לבדוק היתכנות: צילם עמוד אינטרנט עבור use case של browser automation, הדביק ב-Playground והריץ 10 פעמים. אם זה מצליח 7–8 פעמים, אולי זה יעבוד ב-production [34:09].

רגע השיא: קמפיין "Create Real Magic" עם Coca-Cola, הלקוח הראשון שקיבל גישה ל-DALL·E 3 לפני ההשקה הרשמית. נקודת השפל: התקופה שבין ה-DevDay הראשון לשני, כשהחברה התמקדה ב-B2C. לדבריו זו אחת הסיבות ש-Swarm שוחרר כקוד פתוח. לקראת סוף 2024 המטוטלת חזרה ל-B2B, ואז אושר גם ה-business case של FDE [37:09].



הודעת העיתונות 'Coca-Cola Invites Digital Artists to Create Real Magic Using New AI Platform'.

התחזית ל-2026 (בתמלול "2086"): לדעתו הסוכנים עדיין דורשים הרבה עבודה עד production, וייתכן שנחזור ל-fine-tuning. התשתית ליצירת דאטה, תיוג ובניית סט אימון כבר קיימת, וכך יהיה אפשר לכוון מודל ל-use case agentico בתחומים כמו תכנון שבבים או פיתוח תרופות. הוא קורא לזה "year of optimization" או "one click to production" [38:48]. המראיין מהמר על wearables וחומרה.

8. מסקנות ומה לעשות עכשיו

- להגדיר evals לפני הקוד: לתעד עם מומחה תחום את המסלול שאדם היה עובר, לבנות ממנו סט מתיג, ורק אחרי זה לתת ל-agent לעבוד.
- להתחיל מתפקיד מיעץ ולהתקדם בהדרגה: טיקט עם אבחנה, אחרי זה תיקון ו-PR, ואחרי זה סביבת הרצה. עוברים לשלב הבא רק כשהמשתמשים כבר סומכים על הקודם.
- להעביר אילוצים קשיחים לקוד: כל כלל שחייב להתקיים ב-100% מהמקרים נבדק דטרמיניסטית אחרי המודל. סימולטורים קיימים מפעילים כ-tools.
- לתקצב זמן לבניית אמון: לפי Morgan Stanley, היחס הוא בערך חודשיים של פיתוח מול ארבעה חודשים של פיילוטים. כדאי לתת למשתמשים טבלאות שבהן הם יכולים לבדוק בעצמם.
- להשקיע בשכבת metadata ולוגיקה עסקית: חיבורי MCP לא מספיקים. צריך שכבה שמסבירה למודל איך לשלב בין מקורות.
- אם מקימים צוות FDE: להחליט אם המטרה היא מוצר או שירותים, לסרב לכסף שלא משרת את האסטרטגיה, ולהימנע מהכללה לפני שבעיה אחת נפתרה עד הסוף.

שאלה פתוחה	למה היא חשובה
איך נמדדו 98% אימוץ ו-20-30% חיסכון?	בלי baseline ומתודולוגיה קשה להשוות למקרים אחרים.
כמה מהבאגים סוכן ה-debug מתקן היום?	מה שהוצג הוא חזון ל-mid-next year, לא תוצאה.
האם OpenAI תוציא מוצרים ורטיקליים מעבודת FDE?	Jarvis אמר שזו אפשרות פתוחה, אבל ש-OpenAI פחות צפויה לבנות מוצרים ספציפיים לתעשייה מהעבודה הזו.
מה המודל המסחרי של מעורבות FDE?	לא נאמר אם ואיך הלקוח משלם על הצוות.
האם שכבת ה-metadata מחייבת להעביר את הדאטה?	Jarvis אמר במפורש שאין לו תשובה סופית לזה.

9. מילון מונחים

מונח	משמעות
Forward Deployed Engineering (FDE)	מהנדסים שיושבים אצל הלקוח, פותרים בעיה קשה מקצה לקצה ומחלצים ממנה מוצר

מונח	משמעות
eval-driven development	עבודה שבה קוד שנכתב על ידי LLM לא נחשב גמור עד שיש סט evals שמאמת אותו
evals	סטים מתויגים של מקרים שבודקים את איכות המערכת ואת העקביות שלה
guardrails	בקורות שמגנות מפני התנהגות לא צפויה של המודל בזמן ריצה
scaffolding	קוד שעוטף את המודל: כלים, שלבים דטרמיניסטיים, סימולטורים
telemetry	רישום מפורט של פעולות ה-agent כדי לבנות evals ולנתח מסלולים
zero-to-one	השלב שבו בעיה חדשה נפתרת לראשונה, לפני שמכלילים אותה
orchestrator	מודל שמתאם קריאות לכלים ולמקורות דאטה כדי לענות על שאלה
Swarm	framework פנימי לפרמטריזציה של הוראות וכלים, ששחרר כקוד פתוח
Agents SDK / AgentKit	מוצרי OpenAI לבניית AgentKit. visual agents. הוא visual builder
ACV	הכנסה שנתית ממוצעת מחוזה עם לקוח
ontology / metadata layer	שכבה שמתארת את הדאטה ואת הלוגיקה העסקית כדי שמודל יוכל להשתמש בהם
fine-tuning	אימון נוסף של מודל על דאטה "יעודי" למשימה מסוימת