

למה סוכני קוד ממשיכים להרוס לכם את ה-codebase

Dex Horthy מ-HumanLayer על SlopCodeBench, ההייפ של dark factories, ועל workflow שמוציא קוד עם agents בלי לטבוע ב-slop

Patrick · Beyond Coding · 30.09.2026 בשיחה עם המגיש Dex Horthy (HumanLayer)

משך: כ-45 דקות · YouTube

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זה ראיון של כ-45 דקות בערוץ Beyond Coding. המראיין הוא Dex Horthy, מנכ"ל HumanLayer, שמוצג בפתיחה כמי שטבע את המונח context engineering. המגיש, שבסוף השיחה Dex פונה אליו בשם Patrick, מנהל לדבריו ארגון של כ-1,000 מהנדסים ומנסה להבין איך מכניסים אצלם agentic engineering. זו שיחה על ספה, בלי מצגת. הפורמט הוא שאלות ותשובות לפי סדר הפרקים שבעמוד הווידאו.

הטענה המרכזית של Dex: מודלים עדיין גרועים בתחזוקת codebase לאורך זמן, ויש לזה עכשיו גם benchmark ולא רק תחושה. לכן לא כדאי לרדוף אחרי אחרים, אחר dark factories או אחרי token maxing. מה שעובד בפועל: מטרה של פי 2-4, תכנון מדורג לפני שכותבים קוד, בני אדם שבודקים maintainability, ומעט עבודה במקביל. בסוף השיחה הוא מתאר את חלוקת העבודה בצוות שלו: one-shots, janitor bots, שרצים בלילה, ותהליך research-design-plan.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, ומה לא נאמר בשיחה.
- 2. **SlopCodeBench:** למה benchmarks כמו SWE-bench מטעים, ומה מודד ה-benchmark החדש.
- 3. **ארגונים גדולים:** background agents, תכנון מדורג ו-PRD שמבוסס על המצב האמיתי של ה-repos.
- 4. **הייפ מול מציאות:** dark factories, token maxing ולמה פי 2-4 מספיק.
- 5. **PRs, code review ו-taste:** מה לא ניתן להחליף בכלים דטרמיניסטיים, ואיך לומדים taste היום.
- 6. **ה-workflow של Dex:** חלוקת העבודה באחוזים, janitor bots, RPI, ועבודה במקביל "כמו פוקר".
- 7-8. **מסקנות ומילון:** מה לעשות מחר, שאלות פתוחות ומילון מונחים.

הערות מקור המסמך הופק אוטומטית מהתמליל של ההקלטה ומהפריימים שלה. אין בה שקפים, כולם צילומי ראיון. ייתכן שזיהוי הדיבור (ASR) שיבש שמות ומונחים. בפרט, שמות המודלים כפי שנשמעו (Fable, Astra, Soul) ושמות של אנשים צדדיים לא ודאיים, והם מסומנים ככאלה. הציטוטים מועתקים מילה במילה מהתמליל ונבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים.



המגיש (משמאל) ו-Dex Horthy (מימין) באולפן של Beyond Coding, עם כיתוב השם של Dex על המסך

1. תקציר מנהלים

Dex מגיע לשיחה מהעבודה של HumanLayer עם ארגונים של 100 מהנדסים ומעלה. הזווית שלו פרקטית ובוטה: רוב מה שמוצג ברשת על agents הוא הצגה, והשאלה היא מה עובד בארגון עם עשר שנות היסטוריה ומאות repos. אלה הטענות העיקריות:

- **מודלים מקלקלים codebase לאורך זמן, ויש לזה מדידה:** ב-SlopCodeBench מבקשים מהמודל לבנות מערכת ואז להוסיף לה פיצ'רים ב-3 עד 10 שלבים, בלי לחשוף לו מראש את הבעיה כולה. לפי Dex, GPT 5.5 Codex קיבל כ-15% ו-Astra כ-16.5% [3:05].
- **החלק הקשה הוא 10% האחרונים:** לדבריו המעבדות מתמקדות ב-computer use, ו-maintainability עשויה להיות קשה כמו כל 90% שכבר הושגו [4:38].
- **אי אפשר לעשות one-shot לפיצ'ר גדול, אבל אפשר לצמצם עבודה חוזרת:** תכנון מדורג (מסמך עמוד אחד, ואחריו design) מנפה פתרונות גרועים, ומוריד את העבודה החוזרת מ-25-75% ל-0-10% [9:11].
- **background agents לשינויים קטנים זה כבר פתור:** ticket-to-PR שסוגר 50% מה-PRs אפשר לבנות ב-hack week. האתגר האמיתי הוא שיתוף פעולה על משימות גדולות [6:09].
- **רוב הסיפורים על dark factories הם BS:** הוא מתאר מהנדסי principal ש-PRs של slop קוברים אותם, ובאחד מהם 30,000 שורות, בזמן שמנהלים מתחרים מי מגיע לאחוזון גבוה יותר [16:55].
- **פי 2-4 זו מטרה מספיקה:** מי שרודף אחרי 100x מפספס את השיפור המצטבר [20:00].
- **כלים דטרמיניסטיים לא מבטיחים maintainability:** cyclomatic complexity חוסמים קטגוריות של slop, אבל בלי אדם בלולאה הם לא עוצרים קוד שקשה לתחזק [24:32].
- **design doc הוא המקום החדש ללמוד בו:** מי שמעיר על design doc מלמד את הכותב לכתוב prompt טוב יותר [29:08].
- **החלוקה בצוות שלו:** one-shots 50-60%, כ-10% PRs של janitor bots מהלילה, כ-30% מהקוד דרך RPI. ובאופן כללי, לעבוד על פחות דברים במקביל [33:46], [38:21].

מה לא נאמר

- **אין מתודולוגיה מלאה ל-SlopCodeBench:** הוצגו שני ציונים בלבד, בלי פירוט של משימות, מספר הרצות או השוואה בין מודלים נוספים. Dex עצמו מציין שזה benchmark אחד ממעבדה אחת, ו-HumanLayer ממנת חלק מהמחקר.
- **אין מדידה לטענות על פרודוקטיביות:** "פי 2-4" וחלוקת האחוזים בצוות הן הערכות אישיות. המגיש בעצמו אומר שלא מדד את ההשפעה של עבודה במקביל.
- **אין מספרי עלות:** לשאלה איך מונעים שימוש במודל היקר לכל דבר, התשובה היחידה היא שיש ארגונים שחוסמים אותו. אין נתוני spend ואין מדיניות ניתוב בין מודלים.
- **המוצר של HumanLayer מוצג בכלליות:** אין דמו, אין ארכיטקטורה ואין לקוח שמוזכר בשמו.

2. SlopCodeBench: הוכחה שמודלים מקלקלים codebase

Dex פותח בסיכום של שלוש משפחות קיימות של benchmarks. הראשונה היא SWE-bench: לוקחים PR קטן מ-repo ציבורי, מחזירים את ה-repo למצב שלפני ה-PR ובודקים אם המודל מעביר את הטסטים. לדבריו, ציון של 95% שם לא אומר ש-coding נפתר. השנייה היא benchmarks ארוכים, כמו SWE Marathon עם משימות של 4 עד 400 שעות ("לבנות מחדש את Excel" או את Redis). השלישית, לפי Dex, הגיעה מ-Cognition: משימות גדולות יותר, ואחריהן מודל judge שבדק שמירה על קונבנציות של ה-codebase.

לטענתו, כל השלוש נותנות למודל את כל הבעיה מראש, וככה לא בונים תוכנה אמיתית. SlopCodeBench, שמינח בתמליל למעבדה של Gabe Orlansky מאוניברסיטת ויסקונסין (ייתכן שהשם שובש), מתחיל ממערכת חדשה ואז מוסיף פיצ'ר אחרי פיצ'ר. לכל אתגר יש 3 עד 10 checkpoints. התוצאה היא שמספר הטסטים הנכשלים עולה לאורך הזמן אצל כל המודלים.

Dex Horthy [1:32] It's an entire benchmark designed around proving that like models make code — bases worse over time

המספרים כפי שנאמרו: GPT 5.5 Codex קיבל כ-15% בהרצה ממאי. HumanLayer מממנת חלק מהמחקר, ובהרצה שלהם המודל שנשמע כ-"Astra" קיבל 16.5%. השם כנראה שובש ב-ASR. לדבריו ה-benchmark לא אומץ ב-model cards, כנראה כי ההרצה שלו איטית ויקרה. לשאלה אם המודלים ישתפרו בזה, הוא מזכיר את ההנחה מימי Claude 3.5-3.7 שבעיות ה-slop "היו הבעיה של GPT-7", ואומר שאנחנו כבר ב-GPT-6 ולא הייתה התקדמות של ממש.

— Dex Horthy [4:38] the last 10% of software maintainability may be as hard or harder than the whole first 90% that we spent the last three or four years on

3. ארגונים גדולים: background agents, תכנון מדורג ו-PRD

HumanLayer ממוקדת בארגונים של 100 מהנדסים ומעלה. שם, לדברי Dex, cloud agents (הוא מזכיר את Cursor Cloud agents ואת Devin של Cognition) נתקלים בחיכוך: עשר שנות היסטוריה, מאות repos, סביבות פיתוח מסובכות, ושאלה פתוחה של איך ה-agent בודק את העבודה של עצמו. Slack-to-PR לשינויים קטנים הוא בעיניו לא מעניין. Ramp ו-Brex פרסמו שה-agent background שלהם סוגר 50% מה-PRs, אבל הוא מכיר אנשים שבנו דבר כזה בסוף שבוע. הבעיה הקשה היא שיתוף פעולה על משימות גדולות. כשה-agent כבר כתב קוד שגוי, קשה הרבה יותר לתקן אותו בדיעבד.

הפתרון שלו הוא shift left, כלומר להעביר כמה שיותר החלטות לשלב שלפני הקוד: ליישר קו לפני שכותבים קוד, כמו שצוותים עשו 30-40 שנה. מסמך של עמוד אחד, מבט "מגובה 25,000 רגל", מסלק כ-50% מהדרכים הגרועות לפתור. הזום ל-10,000 רגל מסלק עוד 80%. המטרה היא להגיע ל-10%-0 עבודה חוזרת במקום 25-75%, שזה מה שקורה בדרך כלל כשמנסים one-shot ל-PR גדול. הוא מדגיש שאין תוכנית מושלמת, ו-"the source of truth" [7:40] "should be the code not some specification or plan" ב-HumanLayer קוראים למסמך design doc ולא plan, ומפרידים בין לאן הולכים (המצב הסופי) לבין איך מגיעים לשם. כל לקוח מוסיף לו שאלות משלו.

דוגמה טקטית מעניינת היא PRD של PM. כש-PM מבקש מה-AI "תכתוב את זה", יוצא מסמך של 10 עמודים שהצוות מכנה "AI psychosis slop", והמסמך מנותק מהמצב האמיתי של הקוד. מה שעובד: לתת ל-PM גישה ל-repos (ישירות או דרך GitHub MCP), prompt של שני משפטים, ו-pipeline שמוצא את 7 ה-repos הרלוונטיים מתוך 200. אחר כך ה-pipeline מתחקר את ה-PM בסגנון grill me: איזו בעיה פותרים, איך מודדים הצלחה, ואיך מחליטים לזנוח את הניסוי. Dex מגדיר את המטרה כ-"raising the floor": לתת ל-80% מהצוותים בסיס שעובד.

שאלת עלות שנשארה בלי מענה המגיש שואל איך מונעים מ-1,000 מהנדסים להשתמש במודל הגדול והיקר לכל משימה. התשובה היחידה: יש צוותים שבהם המודל שנשמע כ-"Fable" (כנראה שיבוש ASR של שם מודל) פשוט חסום. צריך להסתדר עם Opus, וגישה זמנית למודל היקר דורשת בקשה מיוחדת [11:18].

4. הייפ מול מציאות: dark factories, token maxing ופי 2-4

המגיש מספר שארגונים רבים בסביבה שלו שואפים ל-dark factory, מפעל תוכנה שעובד בלי אדם בלולאה. Dex עונה ישר: "[16:55] I think most of it is BS." הוא מספר על פגישה עם ראש מחלקה בחברה ותיקה עם אלפי מהנדסים. שני מהנדסי principal שם טובעים ב-PRs של slop, ובאחד מהם 30,000 שורות. ראש המחלקה נשאר לתהות מה עושים בחברות שטוענות ש-90% מהקוד שלהן נכתב ב-Dex. AI מתאר את זה כמשחק פוזות בין מנהלים: בכל פגישת הנהלה שואלים "am I 90th percentile, am I 99th percentile".

המגיש מצטט אדם שנשמע בתמליל כ-Andre Brestloff (השם לא ודאי), שקרא לתקופה הזו "the Instagram phase": הכול מזויף, ומי שאומר שאצלו זה לא עובד שומע "Dex: skill issue" מוסיף את token maxing: מישהו מתגאה שגמר שישה מנויי Codex בשבוע, ובפועל בנה משחקים שאף אחד לא ישחק. הוא אומר שזה בסדר לשחק ככה, אבל לא בסדר להציג את זה כרף שמהנדס צריך לעמוד בו.

— Dex Horthy [20:00] I think every engineer going 2 to 4x faster would be transformational to human society

בעיניו, מי שמנסה להגיע ל-100x או ל-"אין צורך במהנדסים" מוותר על ההשפעה המצטברת: פי 2-3, ואז עוד פי 2-3 על זה. הוא משווה את זה ל-YAGNI. לזרוק את כל מה שהתעשייה למדה ולבנות הנדסה מחדש מאפס זו אותה טעות כמו לתכנן ארכיטקטורה לדרישות שאף אחד לא ביקש.

5. Pull requests, code review ואיך בונים taste

המגיש מצטט מישור בשם Dario מ-AMP (השם והחברה כפי שנשמעו, לא ודאי) שאומר שהצוות שלו פרודוקטיבי כי הם לא עושים PRs ודוחפים ישר ל-Dex. main לא נבהל מזה. בעבודה הראשונה שלו כולם יכלו למזג ל-main, והביקורת הייתה נורמה תרבותית ולא כלל שנאכף. בלשונו: "[21:31] 'culture beats process every single time'. מה שקובע הוא בעלות: אתה היחיד שאחראי על מה שדחפת, גם אם מישור עבר עליו. הוא ממליץ לא להפוך review של design doc לחובה, אבל הוא בעצמו שולח את ה-design docs שלו למייסד השותף וה-CTO, 'the slop-gatekeeper'. הסיבה: לשמוע 'לא' על מסמך של עמוד או שניים זול בהרבה מלגלות ב-PR שכל הארכיטקטורה שגויה.

האם screenshots, וידאו ו-linters יכולים להחליף קריאה של הקוד? Dex מבחין בין שתי סיבות לקרוא קוד: נכונות ו-maintainability. וידאו מוכיח נכונות של הפיצ'ר, אבל לא אומר כלום על maintainability. הצוות שלו משתמש בהרבה linters, וגם ב-cyclomatic complexity, אבל:

— Dex Horthy [24:32] I have never seen a prompt or a deterministic tool that can drive codebase maintainability and fight the model's tendency to write unmaintainable code in any meaningful way without a human in the loop.

הוא משתמש ב-skill של Matt Pocock בשם improve code base architecture, ולפעמים פשוט נותן לו לבצע את שלושת התיקונים הראשונים. אבל לדבריו, הערך של ה-skill מגיע מהאדם שאומר "לא, ככה אני רוצה לעשות את ה-refactor". taste-הבנה נבנה מהפעמים שקמת בשלוש בלילה לתקן תקלה ונשבעת: "I'm never letting anyone merge anything with this pattern in it again." [26:04]. הוא מזכיר חבר בשם Ian (לפי התמליל, הסטארטאפ שלו נרכש על ידי Expo) שהתחנן למעבדות להכניס ל-RL משימות שהמטרה שלהן היא למחוק קוד או לחלץ abstraction טובה.

— Dex Horthy [0:00] Do not mortgage your code base in the hopes that in a year, the model will come out that's smart enough to fix all your problems.

ואיך בונה taste מי שאין לו עשר שנות ניסיון מלפני ה-agents? פעם זה קרה ב-reviewer. PR review טוב לא כתב "תיקנתי לך", הוא שאל למה עשית ככה והפנה לדפוס קיים בקוד. היום, לדבריו, ההערות נשלחות ישר ל-AI, וה-reviewer מרגיש שהוא מדבר עם ה-agent של מישור אחר. הוא מביא את Mitchell Hashimoto, שבפרויקט שנשמע כ-"ghost" (כנראה Ghostty) כותב ב-PRs רק "Dex". [29:08] "ask your claude why this is wrong". מעריך שהלמידה תעבור ל-design docs, כי "[29:08] 'design doc is basically a prompt'". מי שמעיר על design doc מלמד את הכותב לכתוב prompt טוב יותר. HumanLayer שוקלת גם פיצ'ר של שיחה קולית שהתמליל שלה נשמר כ-artifact על המשימה. בעיניו, ה-AI native forge יקשר בין קוד, artifacts ו-sessions.

6. ה-workflow של Dex: one-shots, janitor bots ו-RPI

סוג עבודה	חלק משוער	איך זה עובד
One-shot	50-60% מהעבודה	שינויים של עד כמה מאות שורות, בלי תכנון מראש. מבקשים, בודקים, ולפעמים שולחים עוד prompt או שניים.
Janitor bots	כ-10% מה-PRs	רצים בלילה מול רשימה של איך ה-codebase אמור להיראות ("כמו תרמוסטט"): refactor, חוקי React doctor, הגירה של endpoint ל-framework החדש. לכל סוג, ויש 3-6 סוגים, פתוח לכל היותר PR אחד, והסימון נעשה ב-label.
RPI (research-design-plan)	כ-30% מהקוד	research על ה-codebase ב-session שלא רואה את ה-ticket, אחר כך PRD (רק כשהמשימה עמומה), design טכני, ולפעמים plan שמפרק לשלבים ול-stacked PRs.

על ה-janitor bots הוא אומר: "If the PR is too big, it never gets merged." [33:46]. לכן כל מה שה"מפעל" מייצר קטן בכוונה. PR שלא מוזג חוסם את ההרצה של אותו סוג למחרת, כמו לוח RPI. Kanban התפתח לגרסאות עם שמות כמו "Crispy" (שמות לא ודאיים בתמליל), ו-Dex כבר הפסיק להתעסק בראשי התיבות. בשלב ה-plan העיקרון הוא נקודות ביקורת: קל הרבה יותר להוציא 10,000 שורות אם בודקים כל 1,000. כשמגלים תקלה רק בסוף, שטח הדיבאג עצום.

ב-implementation הם משתמשים ב-sub-agents ומשלבים מודלים: לפעמים מודל אחד מתזמר אחר, ולפעמים Opus משמש גם כ-parent וגם כ-sub-agent. השמות כפי שנשמעו, "Fable" ו-"Soul", לא ודאיים. כדי שמודל של ספק אחד יתזמר sub-agents של ספק אחר, צריך היום לבקש ממנו להפעיל את Codex CLI. לדבריו, harness של HumanLayer יתמוך בזה בקרוב. כלל תרבותי פנימי אצלם: "do not send anyone code for review that you yourself have not read" [36:49]. לפעמים הוא משנה 50% מהקוד, ולפעמים זורק הכול וחוזר ל-design, כי ברור לו ש-Kyle לא יאשר את זה. "I think people underestimate that you can just throw out the output and try again." [36:49].

למה לא לעבוד הרבה במקביל בלילה

המגיש שואל על חזון נפוץ בהנהלות: לתכנן ביום ולהריץ בלילה. Dex: "I don't want to wake up to 10,000 lines of code." [38:21]. אם מתכננים עשר משימות בלילה, את ארבע השעות הראשונות של הבוקר מבלים בקריאת קוד, ואחרי ארבעה ימים חצי מהמשימות עדיין באוויר. לכן "do less things in parallel" [38:21]. הוא מעדיף להתחיל שתי משימות, לעקוב אחריהן ולהעלות אותן ל-production באותו יום. במקום אחד הוא משאיר זמן המתנה: research ב-codebase גדול לוקח 10-20 דקות, והוא מוכן לחכות כדי שה-agent יקבל 90-99% מה-context.

לגבי מנויים: המגיש משלם 200 דולר בחודש ומודה שהמגבלות השבועיות דוחפות אותו להריץ בלילה רק כדי לנצל את המכסה. Dex עונה שכאן הפער הגדול בין Twitter לעולם האמיתי: "no enterprise has subscription plans" [39:53]. גם לקוחות של 15 מהנדסים משלמים לפי token בגלל דרישות security ו-data retention. הוא מעריך שבדיעבד token maxing ייראה כתוצר של מודל ההפצה, כלומר של האופן שבו מוכרים מנויים למפתחים עצמאיים.

Dex Horthy [42:59] You should always have one most important thing, and your top goal is to get — that thing landed

הדימוי שסוגר את השיחה: Dex רוצה לראות סריקות מוח של מי שמריץ שישה Claude במקביל, של מי שעובד עם agent אחד, ושל מי שמושך בידית של מכונת מזל. עבודה נכונה במקביל דומה בעיניו לפוקר, יד אחת בכל פעם עם ריכוז מלא, ולא למכונת מזל.

7. מסקנות ומה לעשות הלאה

- **למיין משימות לשלושה מסלולים:** one-shot לשינויים קטנים, בוטים קטנים לתחזוקה, ותכנון מדורג (research, design, plan) לפיצ'רים גדולים. לא להפעיל את התהליך הכבד על כל משימה.
- **להגדיר design doc כנקודת הביקורת העיקרית:** מסמך של עמוד או שניים שמפריד בין לאן הולכים לבין איך מגיעים. לשלוח אותו ל-reviewer הבכיר לפני שנכתבת שורת קוד.
- **לבנות janitor bots מוגבלים:** רשימה של איך הקוד אמור להיראות, PR קטן אחד פתוח לכל סוג, ועצירה כשאף אחד לא ממזג.
- **לקבוע כלל בעלות:** לא שולחים ל-review קוד שלא קראת בעצמך. ומי שדחף קוד אחראי עליו גם אחרי שמישהו עבר עליו.
- **להשתמש ב-linters כרצפה ולא כתקרה:** הם חוסמים קטגוריות של slop, אבל maintainability עדיין דורשת אדם עם taste.
- **לעבור את ה-PRD מול הקוד:** לתת ל-PM גישה ל-repos או GitHub MCP ו-pipeline שמתחקר אותו, במקום מסמך של 10 עמודים מ-prompt אחד.
- **לעבוד במקביל, אבל עם עדיפות אחת:** כשהמשימה החשובה ביותר משתחררת, עוזבים את כל השאר וחוזרים אליה.

שאלה פתוחה	למה היא נשארה פתוחה
איך מכניסים taste למודלים?	Dex אומר במפורש שזה עוד לא נפתר, וש-RL לא כולל משימות של מחיקה ו-refactor.
איך מכשירים מהנדסים צעירים כשהערות ב-PR עוברות ישר ל-AI?	ההצעה (ללמד דרך design docs) היא כיוון, לא פתרון שנבדק.
האם עבודה במקביל מאיצה את המשימה החשובה ביותר?	אף אחד מהשניים לא מדד. Dex רוצה לראות מחקר על זה.
איך מנתבים בין מודלים זולים ליקרים בארגון של 1,000 מהנדסים?	התשובה היחידה הייתה לחסום את המודל היקר.
האם SlopCodeBench יסקף מודלים עתידיים?	benchmark אחד ממעבדה אחת, יקר להרצה, ולא אומץ ב-model cards.

8. מילון מונחים

מונח	משמעות
Context engineering	תכנון ה-context שמגיע למודל, המונח ש-Dex מזהה עם טביעתו
SlopCodeBench	benchmark שמודד איך איכות הקוד נשחקת כשמוסיפים פיצ'רים בשלבים, בלי לחשוף את כל הבעיה מראש
SWE-bench	benchmark של תיקון issue קטן ב-repo ציבורי לפי טסטים מוסתרים
Slop	קוד או טקסט שנוצר ב-AI באיכות נמוכה ובלי מחשבה
Background / cloud agent	agent שרץ מרחוק, לא על המחשב של המפתח, למשל מ-ticket או מ-Slack ל-PR
Shift left	להקדים החלטות ובדיקות לשלב שלפני כתיבת הקוד
Design doc	מסמך שמגדיר מצב סופי וגישה טכנית לפני המימוש. ב-HumanLayer הוא מחליף את ה-plan
PRD	מסמך דרישות מוצר: איזו בעיה פותרים ואיך מודדים הצלחה
Grill me	סגנון שבו המודל מתחקר את המשתמש בשאלות עד שהדרישה מתבהרת
Dark factory	מפעל תוכנה אוטונומי לחלוטין, בלי אדם בלולאה
Token maxing	ניצול מקסימלי של מכסות מנוי כמדד גאווה, בלי קשר לערך שנוצר
YAGNI	You Aren't Gonna Need It: לא לבנות יכולות שאף אחד עוד לא צריך
Cyclomatic complexity	מדד למספר המסלולים בפונקציה, שמשמש לחסימת קוד מסובך מדי
Taste	שיקול דעת הנדסי שנבנה מניסיון, ובמיוחד מכאב של ארכיטקטורה גרועה
Janitor bot	agent שרץ בלילה ומקדם את ה-codebase בצעדים קטנים לעבר מצב רצוי
RPI	Research, Plan, Implement: התהליך של HumanLayer לפיצ'רים גדולים
Sub-agent	agent משני שה-agent הראשי מפעיל לחלק מהמשימה
Stacked PRs	סדרה של PRs קטנים שכל אחד בנוי על הקודם וניתן למזג בנפרד
Human in the loop	אדם שמאשר או מכון את העבודה בנקודות ביקורת