

מבט לאחור על שנה של Claude Code

ראש Claude Code וראשת המוצר שלו מסכמים שנה: verification, auto mode, routines ו-loop, תפקידים שמתמזגים, ומעבר מ-context engineering למינימליזם בהקשר

2026 Boris Cherny, Cat Wu · Claude · 8

משך: כ-18 דקות · YouTube

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו שיחה מצולמת של 18 דקות שפרסם ערוץ Claude לציון שנה ל-general availability של Claude Code. משתתפים בה Boris Cherny (Head of Claude Code) ו-Cat Wu (Head of Product, Claude Code). אין בה מצגת. שני המשתתפים מראיינים זה את זה ועוברים על מה שהשתנה מאז ההדגמה הראשונה ב-Slack, שקיבלה "שתי תגובות", ועד היום, שבו מספר על "עץ של אלפי agents".

הטענה המרכזית: העבודה עם Claude Code עברה שתי קפיצות. בראשונה המהנדס הפסיק לכתוב קוד ועבר לדבר עם agent. בשנייה הוא מפסיק לדבר עם ה-agent ומדבר עם loop או routine, שמנחים את ה-agent בשבילו. מסביב לטענה הזו הם דנים ב-verification אמיתי (ה-agent מריץ את המוצר בעצמו), ב-auto mode כתחליף ל-permission prompts ול-plan mode, בתפקידים שמתמזגים בארגון, ובמעבר מ-context engineering לגישה של "context minimalism".



צילום רחב של השיחה: Cat Wu ו-Boris Cherny סביב שולחן עבודה במשרד. בהקלטה אין שקפים, יש רק צילומי שיחה

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות והראיות להן, ומה נשאר בלי כיסוי.
- 2. **מקורות ו-verification:** ההתחלה הצנועה, הלולאה של CLAUDE.md ו-skills, ולמה verification אינו unit tests.
- 3. **התפקידים מתמזגים:** PMs, מעצבים, data science ו-finance ב-Claude Code, והאנלוגיה ל-HBR משנות ה-90.
- 4. **Routines ו-agents:** loop שמאזינים לבאגים ומגישים תיקונים, והקפיצה השנייה.
- 5. **Auto mode ואבטחה:** למה Boris ויתר על plan mode, ואיך בנו את האמון ב-classifier.
- 6. **מאות agents ו-voice mode:** Remote Control, agent view, **context minimalism**, וכפות הקשר למודל.
- 7. **מה עושים מחר:** צעדים מעשיים וטבלת שאלות פתוחות.
- 8. **מילון מונחים:** המונחים באנגלית שמופיעים במסמך.

הערת מקור המסמך הופק אוטומטית מהתמליל ומהפריימים של ההקלטה. התמליל הוא ASR ומשבש שמות: Claude מופיע בו כ-Quad, Sonnet כ-Sonnet, QuadMD כ-SONID, ו-Cowork כ-Co-Work. בתמליל אין סימון של הדובר, ולכן שיוך הדברים ל-Boris או ל-Cat הוא הסקה מההקשר. בסרטון אין שקפים, והתמונות הן צילומי שיחה בלבד. הציטוטים מועתקים מילה במילה מהתמליל ובבדקו מול חותמת הזמן שלהם, כולל השיבושים. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

השיחה אינה הכרזת מוצר. היא תיאור של שיטת עבודה, כפי שנראית אצל מי שבונים את הכלי. רוב הטענות מבוססות על אנקדוטות מתוך הצוות, בלי מספרים, אבל הן מתארות כיוון ברור: פחות השגחה אנושית על כל צעד, יותר אוטומציה סביב ה-agent, ומודלים שצריכים פחות הכוונה.

- **Verification הוא ה-agent שמריץ את המוצר, לא lint:** לפי Boris, unit tests ו-type check כבר היו אוטומטיים. השאלה החשובה היא אם ה-agent יכול להריץ את מה שבנה. Cat משתמשת ב-skill שמרים את ה-desktop app המקומי ולוחץ בו בעזרת [1:31] computer use.
- **כל טעות הופכת לידע קבוע:** במקום לתקן את Claude בשיחה, מבקשים ממנו לכתוב את הלקח ל-CLAUDE.md או ל-skill. לדברי Boris, כך Claude יכול "לרוץ לנצח" [0:00].
- **Auto mode החליף את Boris plan mode:** כבר לא משתמש ב-plan mode. לדבריו, השלב הזה היה חשוב מ-Opus עד 4.5, אבל החל מ-4.6, ובוודאות ב-4.7, המודל לא צריך אותו [6:04].
- **Auto mode בטוח יותר מקריאת כל בקשת הרשאה:** כשמאשרים 99% מהבקשות, העין מפסיקה לקרוא. auto mode מנתב את הבקשה למודל נפרד שבודק אבטחה, והוא נבנה על אלפי red teaming, transcripts, ו-evals [9:07] [7:34].
- **Routines הם השימוש המובן הראשון ב-Claude Code כתוכנה:** מהנדס בצוות הריץ routine שמאתר באגים שלא נענו חמש שעות ומגיש להם תיקון. לדברי Boris, ה-routines עושים code review, מתקנים CI ועושים rebase [6:04] [4:34].
- **הקפיצה הבאה: מדברים עם loop ולא עם agent:** לפי Boris, בתוך שנה וחצי היו שתי קפיצות: מכתבת קוד לשיחה עם agent, ומשיחה עם agent לשיחה עם loop או [10:42] routine.
- **התפקידים מתמזגים:** PMs, מעצבים ו-DevRel כותבים קוד, צוות finance מכין תחזיות ב-Claude Code, ומהנדסים מובילים מוצרים מקצה לקצה, כולל העבודה מול legal ו-[13:47] [3:02] marketing.
- **Claude במרכז הארגון, בדומה למעבר למחשבים:** לפי מאמר HBR משנות ה-90 שמצטטים בשיחה, המחשב הביא תועלת רק כשזרקו את ארון התיק. המעבר הוא לקח 10 עד 15 שנה, ועם AI הוא מהיר יותר כי העבודה כבר דיגיטלית [10:42] [12:14].
- **Context minimalism:** נותנים system prompt ו-tools מינימליים ודרך למשוך הקשר, ומשאירים למודל להבין את השאר. גם ה-harness עצמו מצטמצם, כדי להשאיר מקום לפרומפטים של המשתמש [15:17] [16:49].

מה לא נאמר

- **אין מספרים:** לא נמסרו נתוני אימוץ, פרודוקטיביות או מספר משתמשים. גם "אלפי agents" ו"חצי מהעבודה מהטלפון" הם תיאורים אישיים.
- **אין שיעורי שגיאה ל-auto mode:** נאמר שה-"extremely good classifier", אבל לא נמסרו שיעורי false positive או false negative, ולא נאמר איזה מודל מסווג.
- **אין עלות:** לא דובר על עלות tokens או compute כשמריצים עשרות עד מאות agents במקביל.
- **אין מדיניות merge:** ה-routine שמגיש תיקונים ממוזג כשהתיקון "קל לאמת", אבל לא הוגדר מה נחשב קל לאמת ומה עובר ביקורת אנושית.
- **כמעט אין כישלונות:** התקלה הממשית היחידה שהוזכרה היא staging שנפל. לא נאמר מה נשבר ב-routines או ב-auto mode.

2. מקורות ו-verification: מהדגמה ב-Slack לסוכן שבודק את עצמו



Boris Cherny, ראש Claude Code, מסביר למה verification של agent שונה מ-unit tests. פריים מצולם, לא שוקף

הפתיחה צנועה. Boris מספר שהסרטון הראשון של Claude Code שפרסם ב-Slack קיבל שתי תגובות. Cat נזכרת שהכלי היה טוב "במשימות ההנדסיות הקלות מאוד שלי", ו-Boris עונה שזו "דרך נחמדה לומר שהוא לא היה טוב". היום, לדבריו, הוא מנחה agent שמנחה agents, בעץ של אלפים. העיקרון שמחזיק את זה: "I tell it to write it to" [0:00] "the QuadMD or to make a skill or something to do it differently" CLAUE.md. כל תיקון נשמר ב-CLAUE.md. skill, ולא נשאר רק בשיחה אחת.

לגבי Boris, verification, טוען שרוב האנשים מבינים את המושג לא נכון. מדברים על unit tests, lint ו-type check, אבל אלה דברים שכבר היו אוטומטיים ממילא.

Boris Cherny [1:31] But actually, when we talk about verification for agents, it's something slightly —
?different. It's like, can the agent run the thing

Boris מתאר את הרגע שבו זה התחיל לעבוד: עם Opus 4 ביקש מ-Claude לבנות feature ולבודק את עצמו ב-Bash, ו-Claude פתח Claude CLI קטן ובודק את ה-feature שבנה. היום יש loops כאלה ל-iOS simulator, ל-Android simulator ולמחשבי desktop.

Cat מתארת מקרה מהעבודה שלה. מהנדס בצוות כתב desktop development skill שמלמד את Claude להריץ את ה-desktop app המקומי. Claude מרים את האפליקציה, לוחץ עליה בעזרת computer use, מפעיל את ה-UX החדש, בודק edge cases, מתקן ובודק שוב. כש-staging נופל, היא מבקשת מ-Claude לקרוא את Slack ולבודק אם יש תקלה ידועה. בסוף כל debug היא מבקשת ממנו לעדכן את ה-[3:02] skill [1:31]. זה אותו עיקרון שתיאר Boris: הלקח נכתב חזרה לכלי.

3. התפקידים מתמזגים: Claude במרכז הארגון



Cat Wu, ראשת המוצר של Claude Code (זיהוי לפי תיאור הסרטון), מתארת PMs, מעצבים ו-finance שעובדים ב-Claude Code. פריים מצולם, לא שקף

Boris אומר שמעולם לא היה בצוות שבו ה-PM כותב קוד. Cat מסבירה שזה נעשה קל יותר: Claude כותב את הקוד, ולכן מה שחשוב יותר הוא הרעיון. מי שמחזיק את ההקשר העסקי ואת ההקשר של המוצר יגיע לרעיונות טובים יותר. Boris מזכר שנבהל כשראה PRs של Megan, המעצבת (השם כפי שנשמע בתמליל), עד שהסתבר שהיא "רק מתקנת כפתור" והקוד תקין. לפי השיחה, הדפוס חוזר גם אצל לקוחות enterprise: המהנדסים מאמצים ראשונים, והתפקידים הסמוכים מצטרפים אחריהם. "Like our finance team runs in cloud code. They do their projections there." [3:02] גם צוות data science עובד כך [4:34].

לשאלה איך משתנים ארגוני הנדסה, Cat מביאה מאמר Harvard Business Review משנות ה-90, שבדק למה המחשבים האישיים לא הביאו שיפור בפרודוקטיביות. התשובה שם הייתה שלא מספיק להציב מחשב ליד ארון התיק: "Actually, what you have to do is you throw out the filing cabinet." [10:42]. ב-Anthropic, היא מספרת, עובדים חדשים לא שואלים אנשים שאלות אלא את Claude. גם code review, security review ומילוי טפסים (דרך Cowork) עוברים דרכו. לפי השיחה, המעבר למחשבים לקח 10 עד 15 שנה, ועם AI הוא מהיר יותר כי רוב העבודה כבר דיגיטלית ו-Claude יודע להפעיל מחשב, לכתוב קוד ולהריץ אותו [12:14].

— **Cat Wu [12:14]** Everyone's going to be both. I feel pretty strongly that these roles are merging

4. Routines ו-loop: הקפיצה השנייה

Cat מתארת את routines כשימוש שמותח היום את הגבולות. מהנדס שהשיק את voice mode בכל המוצרים הגדיר routine שמאזין לכל GitHub issue, ticket ודיווח באג על voice mode, מגיש תיקון מיוזמתו ושולח לו את ה-PR. אחר כך הרחיב את הרעיון לכל פידבק שלא נענה. Boris מספר שהופתע: עבד על באג ב-feature שהוא עצמו שחרר, ו-Claude שלו גילה שמופע אחר של Claude כבר תיקן אותו.

— **Boris Cherny [4:34]** he says he has another routine that just looks for bug reports that haven't been responded to in five hours and puts up a fix and he merges the ones that are easy to verify

Boris מציג את routines כשימוש המובן הראשון ב-Claude Code כתוכנה. לדבריו, ה-Agent SDK היה הניסיון הראשון בכיוון, אבל לא היה ברור מה עושים איתו. היום routine עושה את כל ה-code review ומשגיח על כל PR:

"You used to have to like fix CI, you used to have to rebase. Like, I haven't done that in a long time."
[6:04]

מכאן הוא מגיע ל-loop. לפני כשנה וחצי המהנדס הפסיק לעבוד מול קוד המקור ועבר לעבוד מול agent. עכשיו מגיעה הקפיצה הבאה: "I don't talk to an agent anymore. I talk to a loop or I talk to a routine and it prompts me." [10:42] quad for me. השכבה שמולה המהנדס עובד עולה עוד רמה: הוא מגדיר תהליך שמנחה את ה-agents, ולא מנחה אותם בעצמו.

5. Auto mode ואבטחה: למה פחות אישורים זה בטוח יותר

Cat שואלת באילו features הוא משתמש כשהוא עובד מול Claude באופן סינכרוני. Boris עונה שבעבר זה היה plan mode, והיום זה auto mode. לדבריו, המודלים החדשים לא צריכים שלב תכנון. השלב הזה היה חשוב מ-Opus 4 עד 4.5, ומ-4.6, ובוודאות ב-4.7, הוא מיותר. יש מי שעדיין משתמשים ב-plan mode בשביל התוצר הכתוב. Boris מפעיל auto mode לכל משימה, עובר ל-Claude הבא ולא צריך לשבת ולצפות [6:04].

לפני שנה וחצי, הוא מסביר, מודל ה-permission prompts היה הטוב ביותר שהיה אפשרי, כי עדיין לא היו classifiers והמודל היה פחות aligned. auto mode מנתב כל בקשה למודל אחר שבדוק אבטחה. אם פקודה נראית חשודה, היא נחסמת, ואפשר לאשר אותה אחר כך [7:34].

Boris Cherny [7:34] It's just human nature when you accept 99% of requests that your eyes just — glaze over when you read it. And so actually we feel that auto mode is more safe than reading every single permission prompt

מצב	איך עובד	מה נאמר עליו בשיחה
Permission prompts	המשתמש מאשר או דוחה כל הפעלה של tool	הפתרון הטוב ביותר שהיה לפני שנה וחצי. מאשרים 99% מהבקשות ולכן מפסיקים לקרוא אותן [7:34]
Plan mode	שלב תכנון נפרד שמייצר תוכנית לפני הביצוע	היה חשוב מ-Opus 4 עד 4.5, ומיותר מ-4.6 ובוודאות ב-4.7. יש מי שמשתמשים בו בשביל התוצר הכתוב [6:04]
Auto mode	מודל נפרד מסווג כל בקשה, מאשר או חוסם, ואפשר לאשר את מה שנחם אחר כך	"It's the best". מאפשר להריץ agent נוסף במקביל בלי להשגיח [6:04] [7:34]

על האבטחה מספרים ששני שלבים באו לפני ההשקה. בשלב הראשון אספו אלפי transcripts, כל אחד מסלול מלא של agent שהסתיים ב-permission prompt, וביקשו מ-auto mode לסווג אם הבקשה בטוחה. בשלב השני ביקשו מ-red teamers ומצוות פנימיים לנסות prompt injection ולפרוץ ל-codebase. מהניסיונות האלה בנו evals ושיפרו את auto mode עד שכל התקיפות נחסמו. לדבריהם, המטרה הייתה להגן גם מפני "the most intelligent attacks that we can construct" [9:07].

Boris מודה שבהתחלה פסל את הרעיון: "route the prompt to a model no way that's not going to work" [9:07]. בפועל זה עבד היטב. המסקנה הכללית שלו היא שחלק ניכר מהאינטואיציות ההנדסיות שצבר לאורך השנים לא עובד כשבונים על מודל, וצריך ללמוד מחדש.

6. מאות agents ו-context minimalism



צילום רחב מהחלק האחרון של השיחה, שעוסק ב-voice mode, Remote Control, agent view ו-context minimalism. פריים מצולם, לא שקף

עד לא מזמן Boris עבד עם שש לשוניות terminal ושישה git checkouts של אותו repo. היום הוא עובד בלשונית אחת עם agent view החדש, ובנוסף ב-desktop app, שמנהל את ה-worktrees בעצמו. את מה שלא היה צופה לפני חצי שנה הוא מתאר כך: חלק גדול מהעבודה ההנדסית שלו מתחיל מהטלפון (התמליל משובש כאן). הוא מתחיל משימה במחשב, מתחבר אליה מהטלפון דרך Remote Control, בודק את ה-agents בזמן שהוא הולך לקפה, ולפעמים פותח agent חדש ב-voice mode באמצע שיחה [13:47] [15:17]. Cat מספרת שראתה את המחשב שלו נשאר פתוח על השולחן ערב אחרי ערב, בזמן שהוא ממשיך למצא PRs, ואז הסתבר שהוא עובד מהספה: "that was the week that remote control got really good" [15:17].

לגבי Boris, context engineering מתאר רצף שמתאים ליכולות המודלים: בתקופת Sonnet 3.5 היה צריך prompt engineering, בתקופת Opus 4 היה צריך context engineering, ועם המודלים של היום לא צריך אף אחד מהם.

Boris Cherny [15:17] You give it the minimal possible system prompt, the minimal possible tools, — and then you let the model figure it out. You just have to give the model some way to pull in the .context

Cat מגדירה את עצמה "context minimalist" ומסבירה ש"when you give the model too much context, it's kind of like you're micromanaging it" [16:49]. לדבריה, המודל מוצא לפעמים דרך טובה יותר להגיע לאותה תוצאה. היא מוסיפה שהצוות מצמצם גם את ה-harness, כדי להשאיר יותר מקום לפרומפטים של המשתמש ולשפר את הציות להם.

בסיום Boris מעריך שבעוד שנה צורת השימוש תהיה שונה לגמרי. agents רצים זמן רב יותר ובאופן אוטונומי יותר, ואצלו כמעט אף פעם לא רץ agent אחד בלבד. הוא לא יודע איך ייראה הממשק לזה, ומצפה שהרעיונות יגיעו מהצוות ומההילה [16:49].

7. מה עושים מחר

- כל תיקון נכתב למקום קבוע: כש-Claude טועה, מבקשים ממנו לעדכן את CLAUDE.md או skill, ולא רק לתקן בשיחה [0:00].
- לכתוב skill שמריץ את המוצר: לכל אפליקציה כדאי skill שמרים אותה מקומית ובודק אותה (computer use, simulator, CLI unit tests לבדם לא מספיקים [1:31]).
- לינוט auto mode במקום לאשר כל בקשה: כדאי לבדוק על repo שאינו קריטי ולעקוב אחרי מה שנחסם. אם עובדים עם Opus 4.6 ומעלה, אפשר גם לבדוק אם plan mode עדיין מוסיף משהו [6:04].

- **להתחיל ב-routine מצומצם:** למשל routine שמאזין ל-issues שלא נענו בתחום אחד ומגיש PR. כדאי לקבוע מראש מה נחשב "קל לאמת" וממוזג, ומה עובר ביקורת אנושית [4:34].
- **לקצץ את CLAUDE.md וה-system prompts:** להשאיר את המינימום ולתת למודל דרך למשוך הקשר בעצמו [15:17].
- **להכניס תפקידים סמוכים:** PMs, מעצבים ו-finance יכולים להתחיל בשינויים קטנים באפליקציה ובתחזיות [3:02].
- **להפסיק לעבוד עם checkouts ידניים:** agent view ו-desktop app מנהלים worktrees במקום שש לשוניות, ו-Remote Control מאפשר לעקוב אחרי ה-agents מהטלפון [13:47].

שאלה פתוחה	למה זה חשוב
מה שיעורי ה-false negative של ה-classifier ב-auto mode?	"בטוח יותר מ-prompts" היא טענה איכותית. ארגון שמאשר את הכלי צריך מספרים
מי מאשר merge של PR שנוצר ב-routine, ולפי איזה קריטריון?	"easy to verify" לא הוגדר, ובלי הגדרה routines עלולים למזג שינויים בלי ביקורת
כמה עולה עץ של מאות או אלפי agents?	עלות ה-tokens וה-compute לא הוזכרה כלל
איך מתאמים בין routines ש"מתחרים" על אותו באג?	Boris מספר שמופע אחר של Claude כבר תיקן את הבאג שלו. זה חיסכון, אבל גם סימן לעבודה כפולה
האם context minimalism עובד גם ב-codebase ארגוני גדול ומורכב?	הדוגמאות בשיחה באות מתוך Anthropic, על המודלים העדכניים ביותר

8. מילון מונחים

מונח	משמעות
Verification	במובן של השיחה: ה-agent מריץ את המוצר בעצמו ובודק שהוא עובד, מעבר ל-tests ול-lint
CLAUDE.md	קובץ הוראות קבוע ש-Claude Code קורא בכל הפעלה. בתמליל מופיע כ-QuadMD
Skill	חבילת ידע והוראות ש-Claude טוען לפי הצורך, למשל איך להריץ את ה-desktop app
Computer use	היכולת של Claude להפעיל ממשק גרפי: ללחוץ, להקליד ולנווט
Auto mode	מצב שבו מודל נפרד מסווג כל בקשת הרשאה ומאשר או חוסם אותה במקום המשתמש
Plan mode	מצב שבו Claude מציג תוכנית לפני שהוא מבצע
Permission prompt	בקשת אישור מהמשתמש לפני הפעלת tool
Routine	תהליך אוטומטי שמפעיל את Claude Code, למשל האזנה ל-issues והגשת תיקונים
Loop	לולאה שמנחה את Claude שוב ושוב, כך שהמשתמש עובד מול הלולאה ולא מול ה-agent
Agent SDK	ה-SDK שמאפשר להשתמש ב-Claude Code מתוך קוד
Agent view	ממשק לניהול agents רבים במקביל מלשונית אחת
Remote Control	שליטה מהטלפון ב-agents שרצים על המחשב
Red teaming / evals	ניסיונות תקיפה מכוונים, ומבחנים אוטומטיים שנבנים מהם
Prompt injection	הוראות זדוניות בתוכן שה-agent קורא, שמטרתן להטות את פעולתו
Context engineering / minimalism	הרכבה מדוקדקת של ההקשר שמקבל המודל, לעומת הגישה של הקשר מינימלי והשארת השאר למודל
Harness	השכבה שעוטפת את המודל: system prompt, tools ועוד
Worktree	git checkout נוסף של אותו repo, כדי שכמה agents יעבדו במקביל