

אנחנו סומכים על AI agents שנשברים כל יום

ארבעה פערים בין שימוש יומיומי ב-agents לבין הרצתם בפרודקשן, ודמו של Temporal:
תשלום ששורד את קריסת ה-workers באמצע

Shy Ruparel (Temporal) · AI Engineer · 29.10.2026

משך: כ-22 דקות · YouTube

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו הרצאה של Shy Ruparel, developer advocate ב-Temporal, מכנס AI Engineer. לפי השקפים היא נערכה בניו יורק ב-13 באוקטובר 2026 כ-Expo Session. ההרצאה בנויה על דוח State of Development השנתי של Temporal (השנה השנייה שלו), שסקר 554 מהנדסים ומובילי הנדסה שעובדים עם AI agents. רובם ותיקים: 48.4% עם 11 שנות ניסיון ומעלה, ועוד כשליש עם 6 עד 10 שנים.

הטענה המרכזית: agents נכשלים כל יום גם אצל צוותים שסומכים עליהם, ואי אפשר לצמצם את זה לאפס. לכן המטרה היא שכישלון לא יעלה כלום. לפי הדובר, הדרך לשם היא durable execution, כלומר הרשומה שבה מתבצע debugging היא אותה רשומה שממנה המערכת מתאוששת. הדמו מראה workflow של תשלום עם מודל מקומי. ה-workers נהרגים פעמיים באמצע, ושום tool call לא רץ פעמיים.

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות עם המספרים, ומה לא נאמר.
- **2-5. ארבעת הפערים:** שימוש מול פרודקשן, אמון מול מציאות, prototype מול enterprise, לראות מול להתאושש. כל פער עם הגרף שלו מהדוח.
- **6. הדמו ו-workflow: durable execution** של תשלום בתשעה שלבים, הריגת workers, וקריאה של ה-event history.
- **7. מה הדמו פתר ומי בפרודקשן:** state כשדות, retry כ-policy, ה-AI כיועץ מוגבל, וכן OpenAI Codex ו-Replit.
- **8. מסקנות ושאלות פתוחות:** מה כדאי לבדוק במערכת שלכם.
- **9. מילון מונחים:** המונחים באנגלית ומה הם אומרים.

הערות מקור המסמך הופק אוטומטית מתמליל ההקלטה ומהשקפים שלה. שמות ומונחים עלולים להשתבש בתמלול האוטומטי (ASR), ובמקומות כאלה ציינו את אי-הוודאות. הציטוטים מועתקים מילה במילה ונבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

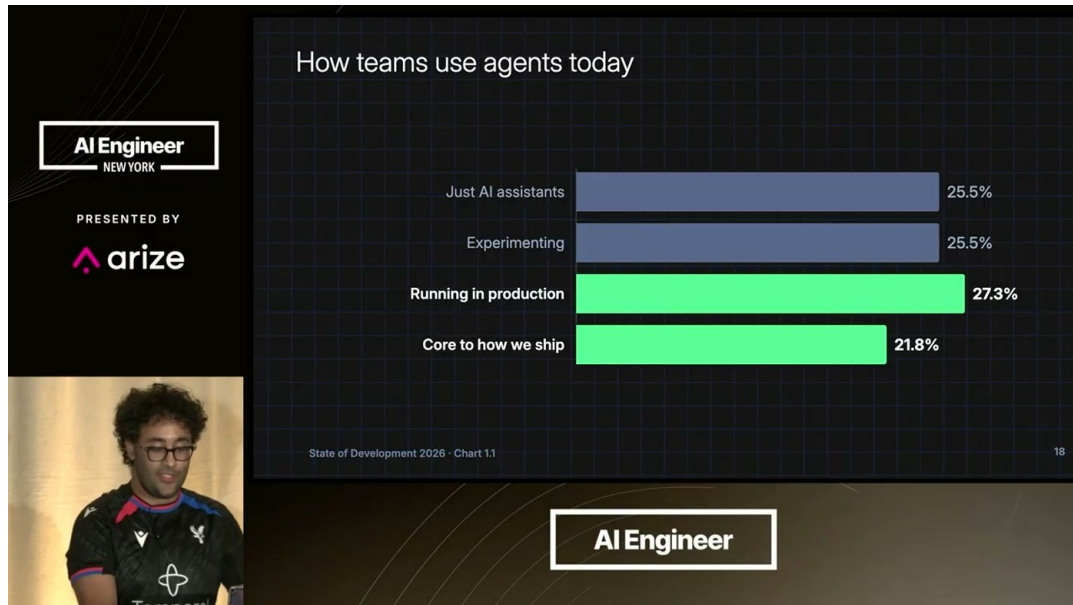
Ruparel פותח בתחום הפיננסים. הוא מצטט דוח agentic AI של מה שנשמע בתמליל כמו KPMC, וכנראה מדובר ב-94% KPMG מחברות ה-fintech אומרות ש-AI הוא חלק מרכזי בפעילות שלהן, ו-62% מאבדות זמן או הכנסות בגלל בעיות אמינות. בפיננסים העבודה בנויה מ-workflows ארוכים (loan origination, KYC/AML, fraud review), שנמשכים כמה ימים, עוברים בכמה מערכות ומערבים כסף אמיתי. משם הוא עובר לנתוני הסקר של Temporal.

- השימוש היומיומי הוכפל, הפרודקשן לא: 80.8% משתמשים ב-agents מדי יום (לעומת 47.3% לפני שנה), אבל רק 49.1% מריצים אותם בפרודקשן או אומרים שהם core לאופן שבו הם משחררים קוד.
- אמן גבוה, תקלות יומיות: 85.5% סומכים על הפלט של ה-agent לפחות במידה מסוימת. במקביל, 41.1% נתקלים בתקלות פעם ביום או יותר, ו-16.4% כל שעה או ברציפות.
- ניסיון לא פותר את הבעיה: צוותים שמגדירים את עצמם מצליחים נתקלים בתקלות בתדירות שעתית או יומית בסיכוי גבוה פי 1.7. מספר ה-agents לא נמצא במתאם עם תדירות התקלות.
- חברות גדולות איטיות פי כמה: 51.3% מגיעים מ-prototype ל-production-ready תוך שעות או פחות. בחברות עם +5,001 עובדים 18.5% אומרים שזה לוקח חודשים, לעומת 5.5% בשאר.
- החסם הוא state, לא המודל: tracking state across services מוביל עם 35.7%, ואחריו isolating and debugging עם 30.9%. 22.2% מאבדים התקדמות בגלל קריסות.
- observability לא מספיקה: רק 13% בטוחים מאוד שהם מסוגלים לצפות ב-AI workflows ולעשות להם debugging בקנה מידה, ומקור העזרה מספר אחת כש-agent נשבר הוא YouTube (47.3%).
- durable execution כפתרון: בדמו כל activity של AI רצה פעם אחת. אישור האדם ואישור הבנק הגיעו כ-signals בזמן שלא רץ אף worker, וה-workflow המשיך מהצעד האחרון שהצליח.
- שבעה חסמים, workflow אחד: לפי הדובר הדמו טיפל ב-debugging, state, עלות, retries, approval, עבודה ארוכה ו-crash recovery.
- לקוחות בפרודקשן: לפי הדובר OpenAI Codex ו-Replit Agent רצים על Temporal.

מה לא נאמר

- אין מספרי ביצועים לשחזור: לא נאמר תוך כמה זמן worker חדש מתאושש. בדמו החי ההתאוששות נתקעה, והדובר עבר לסרטון מוקלט.
- אין מחיר: לא דובר על עלות התשתית של Temporal, על ה-overhead של שמירת event history או על latency.
- ההסבר להצלחה הוא ניחוש: הממצא על פי 1.7 אצל צוותים מצליחים מוסבר כהשערה (אולי הם פשוט מחפשים יותר תקלות), והדובר אומר במפורש שצריך לבדוק את זה.
- החודשים בחברות גדולות לא פורקו: הדובר מייחס אותם ל-auditability ולאמן, אבל אומר שהסקר לא מפרק את הנתון.
- אבטחה נדחתה להרצאה אחרת: 40% מציינים אבטחה כחסם לאוטונומיה, והנושא הופנה להרצאה של Temporal ו-Keycard מיום ה-workshops.
- השוואת state management לא רציפה: 27% שתיארו state management כשביר הגיעו מסקר אחר עם שאלה אחרת, ולכן אין כאן קו מגמה.
- אין חלופות: לא הוזכרו כלי durable execution אחרים, ולא הוצג מקרה שבו הגישה לא מתאימה.

2. פער 1: שימוש מול פרודקשן

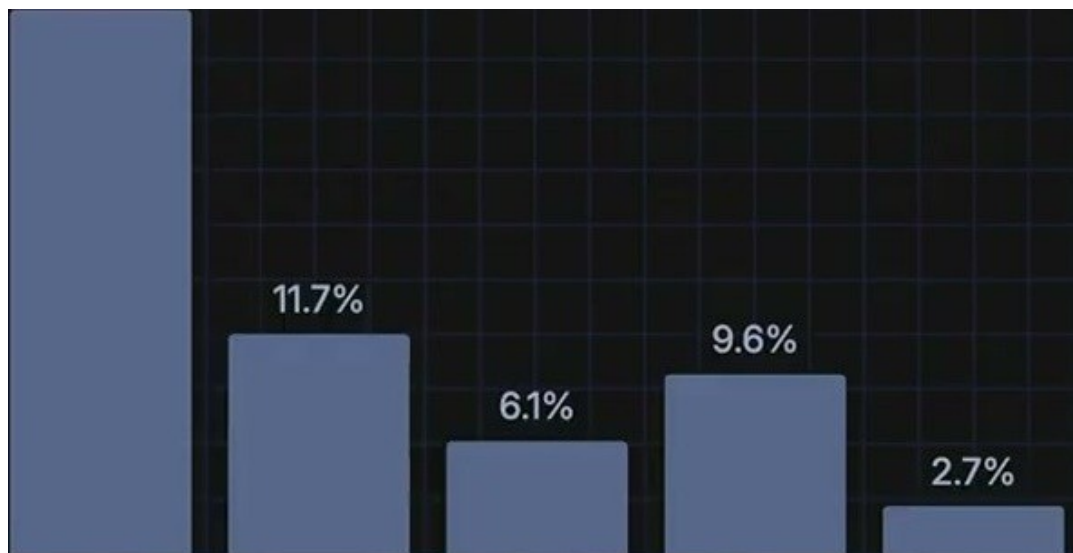


running in production 27.3%, core to how we ship 21.8%, AI assistants 25.5%, ניסויים 25.5%
21.8%

השימוש היומיומי עלה מ-47.3% בסקר של 2025 ל-80.8% היום. אבל כשבדקים מה הצוותים עושים בפועל, רק מחצית מעבירים agents לפרודקשן. בשקף, 27.3% מריצים בפרודקשן ו-21.8% אומרים שה-agents הם core לאופן שבו הם משחררים, יחד 49.1%. החצי השני עדיין מתנסה (25.5%) או משתמש רק ב-(25.5%) assistants.

הדובר מתאר את זה כפער של כ-30 נקודות: "So far fewer people in production than actually using them".
"daily. And something seems to keep breaking on the way to production".

3. פער 2: אמן מול מציאות



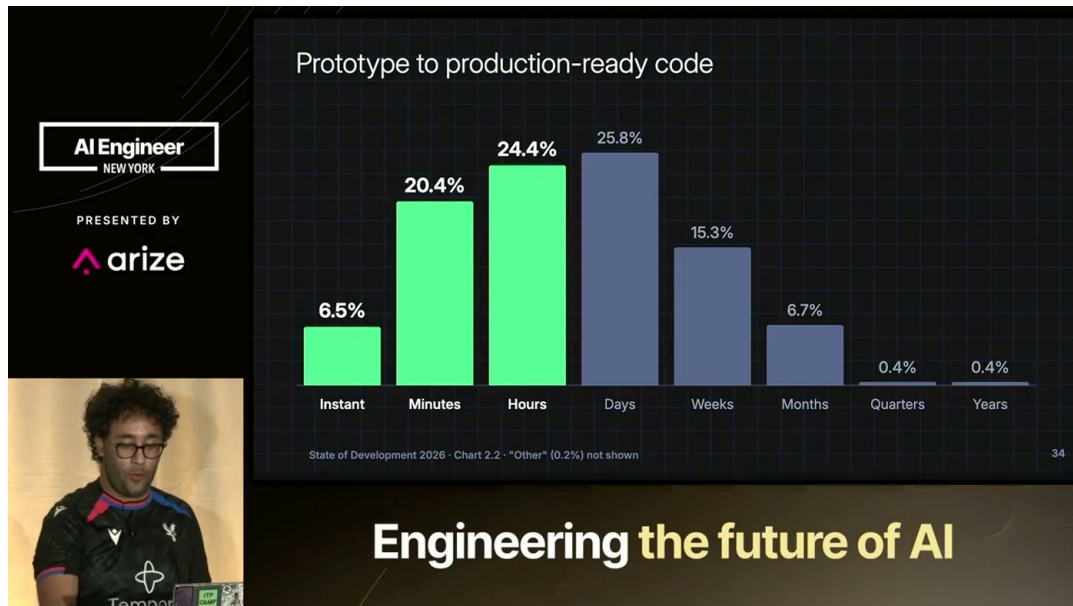
How often do your agents have issues: ברציפות 9.0%, כל שעה 7.4%, מדי יום 24.7%, שבועי 28.7%, חודשי 11.7%, רבעוני 6.1%,
כמה פעמים בשנה 9.6%, לא בפרודקשן 2.7%

רבע מהנשאלים סומכים על הפלט של ה-agent כמעט לחלוטין, רובם סומכים במידה מסוימת, וכמעט אף אחד לא חושד בו. סך הכול 85.5% סומכים לפחות במידה מסוימת. הדובר מעיר שהוא עצמו בקטגוריה החשדנית. במקביל, העמודות הוורודות בגרף (ברציפות, כל שעה, מדי יום) מסתכמות ב-41.1%. אחד מכל שישה נתקל בתקלה כל שעה.

הממצא המפתיע ביותר: צוותים שמגדירים את עצמם מצליחים נתקלים בתקלות שעתיות או יומיות בסיכוי גבוה פי 1.7, ולפי הדוח אין מתאם בין מספר ה-agents לתדירות התקלות. ההשערה של Temporal היא שצוותים כאלה פשוט מחפשים תקלות טוב יותר.

Shy Ruparel So the goal isn't zero failures when we're using agents in production. It's just how do — we make it so these failures don't cost us anything

4. פער 3: prototype מול enterprise

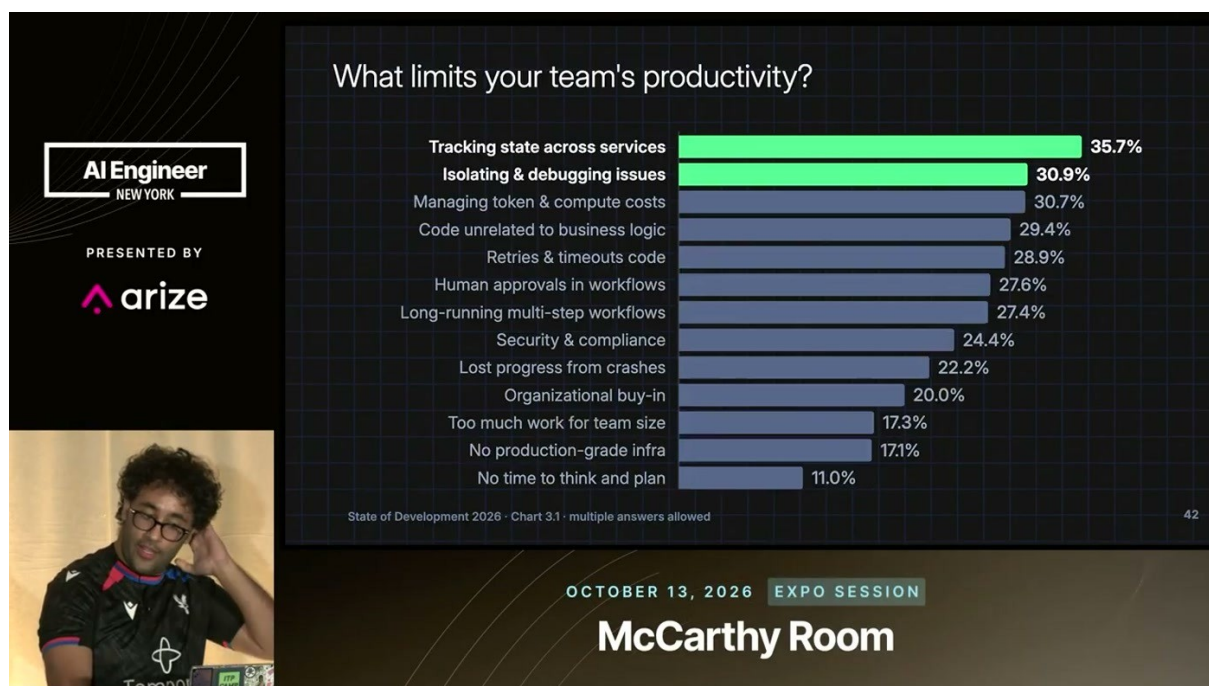


Prototype to production-ready code: מיידית 6.5%, דקות 20.4%, שעות 24.4%, ימים 25.8%, שבועות 15.3%, חודשים 6.7%, רבעונים ושנים 0.4% כל אחד

51.3% מעבירים prototype של AI לקוד production-ready תוך שעות או מהר יותר, כלומר בקצב של סטארטאפ. בחברות גדולות התמונה שונה: לפי השקף "It takes months", 18.5% עובדים מדווחים על חודשים, לעומת 5.5% בכל השאר. בתמליל זה מנוסח כאחד מכל חמישה מול אחד מכל עשרים.

לפי הדובר אלה בעיקר בנקים, חברות ביטוח וחברות תשלומים, שצריכות approvals, audit trails ומערכות שלא מאבדות תשלום. בעיניו לשם הולכים החודשים, אבל הוא מודה שזו פרשנות שלו: "My read is that's where the five months are going into the auditability and the trust of all of those things. And the survey doesn't break that down."

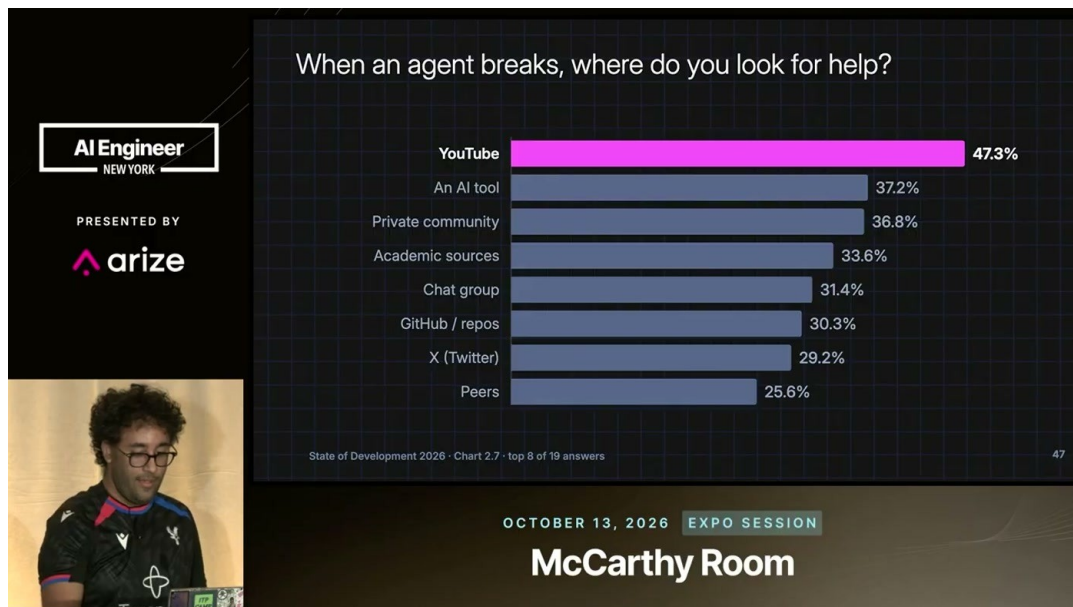
5. פער 4: לראות מול להתאושש



What limits your team's productivity: tracking state 35.7%, debugging 30.9%, token/compute 30.7%, עלויות 30.7%, קוד שלא קשור ללוגיקה העסקית 29.4%, קוד 29.4%, retries/timeouts 28.8%, human approvals 27.8%, long-running workflows 27.4% קוד 27.4%, אובדן התקדמות מקריסות 22.2%

המודלים לא נמצאים בשני המקומות הראשונים ברשימת החסמים. tracking state across services מוביל עם 35.7%, ואחריו isolating and debugging issues עם 30.9%. בנוסף, 22.2% מאבדים עבודה ונאלצים לחזור עליה בגלל קריסות והפעלות מחדש. בסקר אחר מהשנה שעברה 27% תיארו את ה-state management שלהם כשביר, והשנה הוא החסם מספר אחת. הדובר מדגיש שאלה סקרים שונים עם שאלות שונות, ולכן אי אפשר לדבר על קו מגמה. ובכל זאת: "State got more important as agents got more real".

אפשר לראות את התקלה ועדיין לא להתאושש ממנה. ל-observability יש traces, logs וכלי LLM observability, אבל לפי הדובר "Observability right now just tells you your agent failed". רק 13% בטוחים מאוד שהם מסוגלים לצפות ב-AI workflows ולעשות להם debugging בקנה מידה. ובמילים שלו: "the gap between seeing and recovering is a person reading a trace at 2 a.m. in the morning and then fixing their state by hand".



When an agent breaks, where do you look for help: YouTube 47.3%, AI 37.2%, קהילה פרטית 36.8%, מקורות אקדמיים 33.6%, קבוצת צ'אט 31.4%, X 29.2%, GitHub 30.3%, עמיתים 25.6%

6. הדמו ו-durable execution

LIVE DEMO

Kill every worker. Approve anyway.

1. Start the Human approval payout: \$2,500, new recipient, unusual amount
2. Watch the agent investigate and write a brief
3. Stop a worker until 0 are running
4. Click Approve with nothing running
5. Start a worker and watch it pick up

OCTOBER 13, 2026 EXPO SESSION
McCarthy Room

שקף הדמו Kill every worker. Approve anyway: הפעלת payout של \$2,500 לנמען חדש בסכום חריג, ה-agent חוקר וכותב brief, עצירת workers עד שאין אף אחד, אישור בלי worker רץ, הפעלת worker והמשך

ה-workflow כולל תשעה שלבים: validate, reserve funds, FX quote, חקירת הונאה על ידי agent, brief, לאדם, אישור אנושי (human in the loop) והגשה לבנק. בדמו רץ LLM מקומי על הלפטופ. בתמליל השם נשמע כמו "QN36B9B", וכנראה מדובר במודל ממשפחת Qwen, אבל השם המדויק לא ודאי. בתמליל נשמע גם "\$25,000", אבל בשקף ובדמו עצמו הסכום הוא \$2,500.

במערכת רגילה, אחרי קריסה הכסף נשאר שמור במערכת הבנק בלי payout, ה-brief אובד לפני שמגיע למאשר, וצריך לשלם שוב על tokens כדי ליצור אותו מחדש. כל ה-state יושב בזיכרון של התהליך. לכן כותבים קוד recovery, טבלאות state, לוגיקת retry ו-reconciliation jobs. זה קוד שלא קשור ללוגיקה העסקית, ו-30% מהנשאלים מתלוננים עליו. ההגדרה שהדובר נותן: "So durable execution is a temporal concept that code essentially should run to completion, even when the process doesn't".

AI Engineer
NEW YORK

PRESENTED BY

arize

Every step, recorded. Including the AI.

Temporal UI · timeline from a local run of the payout demo

OCTOBER 13, 2026 EXPO SESSION

McCarthy Room

Every step, recorded. Including the AI: timeline של Temporal מריצת ה-payout. כל צעד, כולל ה-tool calls של ה-agent, הוא activity עם תוצאה שמורה, והפס הצהוב הוא ההמתנה לאישור אנושי

החלוקה בארכיטקטורה: קוד ה-workflow מחליט מה הצעד הבא, והוא deterministic. כל מה שאינו deterministic או שיש לו side effects, כמו קריאת LLM או קריאה ל-API של הבנק, רץ בתוך activity. אישור או דחייה של האדם מגיעים כ-signal ונשמרים ב-history כמו כל דבר אחר, גם כשאף worker לא רץ. ה-history יושב ב-Temporal service, ולכן worker חדש ממשיך מהצעד האחרון שהצליח. activity שהייתה באמצע ריצה בזמן הקריסה מורצת שוב, וכל מה שלפניה נשמר. לפי הדוח כ-80% אומרים שעלות מגבילה את השימוש שלהם ב-24.7% agents: 55.1% שזה גורם גדול ו-5.1% שזה גורם במידה מסוימת.

בדמו החי ה-worker לא חזר בזמן, והדובר עבר לסרטון מוקלט של אותו תרחיש. ב-event history שמוצג בסרטון כל activity של AI רצה פעם אחת וניסתה פעם אחת. אישור האדם וההגשה לבנק הגיעו כשלא רץ אף worker, ושניהם עובדו על workers חדשים: "zero workers, the payout waited, the approval landed, and we didn't have to rerun anything from the beginning".

AI Engineer
NEW YORK

PRESENTED BY

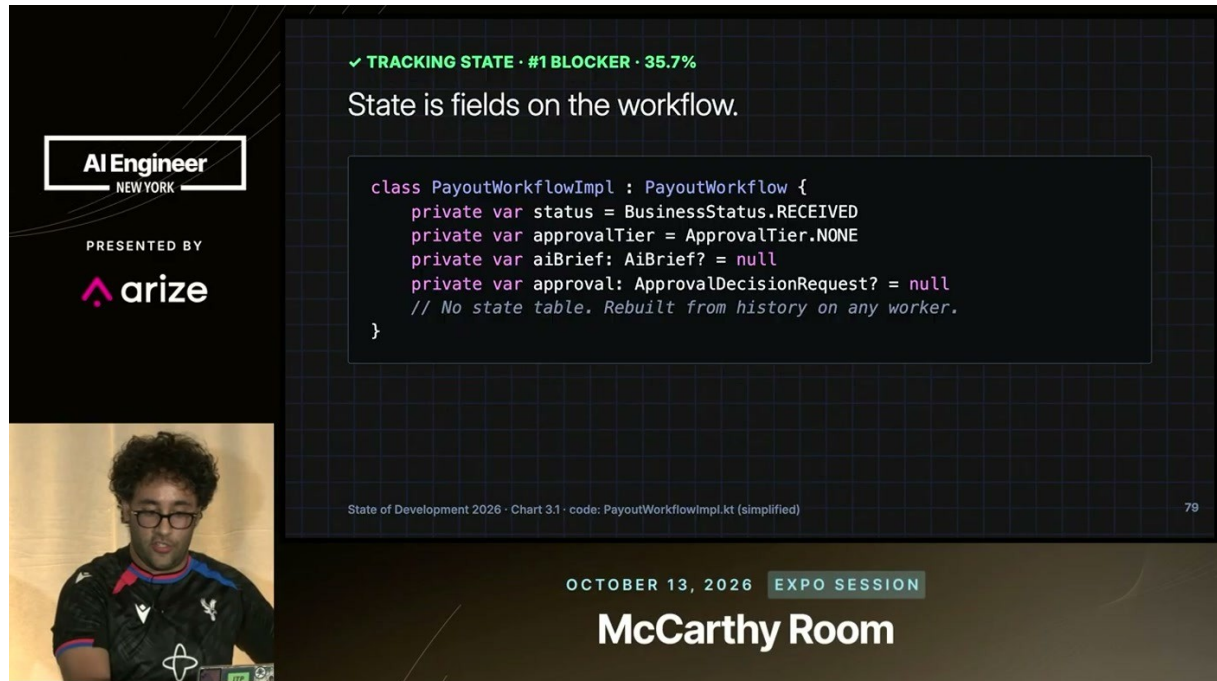
arize

OCTOBER 13, 2026 EXPO SESSION

McCarthy Room

מסך הדמו: payout-transient בממשק של Temporal, עם פרטי הריצה ו-timeline של ה-activities אחרי שה-workers הופעלו מחדש

7. מה הדמו פתר ומי בפרודקשן



✓ TRACKING STATE · #1 BLOCKER · 35.7%

State is fields on the workflow.

```
class PayoutWorkflowImpl : PayoutWorkflow {  
    private var status = BusinessStatus.RECEIVED  
    private var approvalTier = ApprovalTier.NONE  
    private var aiBrief: AiBrief? = null  
    private var approval: ApprovalDecisionRequest? = null  
    // No state table. Rebuilt from history on any worker.  
}
```

State of Development 2026 · Chart 3.1 · code: PayoutWorkflowImpl.kt (simplified)

OCTOBER 13, 2026 EXPO SESSION

McCarthy Room

No state table. Rebuilt from history on every worker. *State is fields on the workflow: מחלקת PayoutWorkflowImpl עם שדות status, approvalTier, approval ו-approval, וההערה No state table.*

הדובר עבר על שבעה חסמים מהסקר ועל המענה שהדמו נתן לכל אחד:

State: ב-workflow ה-state הוא שדות רגילים, בלי database ובלי cache שנבנו במיוחד. אחרי קריסה ה-replay בונה אותם מחדש בדיוק. עלות: ה-agent רץ פעם אחת וה-tool calls נשמרו, כך שלא נשרפו tokens נוספים. "Retries: Retrys themselves were a policy, not a loop" ה-retry policy מוגדר פעם אחת בקונפיגורציה וחל על כל הקריאות, ואין לולאת retry בקוד. Approval ועבודה ארוכה: את הטיימר של האישור האנושי אפשר להגדיר לכל משך, אפילו שנה, באותו קוד. Crash recovery: לא נכתב קוד Debugging. recovery: ה-history שבו מתבצע debugging ב-UI הוא אותו history שה-worker מריץ ב-replay, ולפי הדובר זה מה שסוגר את פער 4.

העיקרון שהדובר הציג כשקף החשוב ביותר לקהל: ה-agent רק מייעץ, וה-workflow הוא שאחראי על הכסף ועל ה-state.

— Shy Ruparel And if the model had failed, the payout would have continued without the brief. A model outage wouldn't have degraded anything.

8. מסקנות ומה לעשות הלאה

- **לתכנן לכישלון, לא לאפס תקלות:** לבדוק מה עולה כל קריסה של agent אצלכם: tokens שנשרפים שוב, כסף שנשאר שמור, עבודה ידנית בלילה.
- **להפריד בין החלטה לביצוע:** קוד orchestration שהוא deterministic, וכל קריאת tool call, LLM או API חיצוני כיחידה נפרדת עם תוצאה שנשמרת.
- **ה-agent מייעץ, ה-workflow אחראי:** לתכנן כך שכשל של המודל יפגע באיכות (למשל אין brief) ולא יעצור את התהליך העסקי.
- **אישור אנושי כאירוע שנשמר:** approval צריך להישמר גם כשאין תהליך פעיל, ולא לחיות בזיכרון.
- **retry כ-policy:** להחליף לולאות retry ידניות בהגדרה אחת. לפי הדובר, פלט גרוע של מודל הוא פשוט עוד "activity failure: Bad model output is just an activity failure, so similar retry policy as well".
- **לאחד observability ו-recovery:** אם ה-traces וה-state נשמרים בשתי רשומות שמתרחקות זו מזו, מישהו יצטרך לחבר ביניהן ידנית.

- לקרוא את הדוח המלא: State of Development של Temporal (temporal.io) מכיל את כל המספרים שהוצגו.

שאלה פתוחה	למה זה חשוב
כמה זמן לוקח ל-worker חדש להתאושש, ומה ה-overhead של replay על history ארוך?	בדמו החי ההתאוששות התעכבה, ולא הוצגו מספרים
למה צוותים מצליחים מדווחים על יותר תקלות?	ההסבר (מחפשים יותר) הוא השערה שעדיין לא נבדקה
לאן בדיוק הולכים החודשים בחברות גדולות?	הסקר לא מפרק את הנתון, וההסבר של auditability הוא פרשנות של הדובר
איך מטפלים בכשל אבטחה של agent אוטונומי?	40% מציינים אבטחה כחסם, והנושא הופנה להרצאה אחרת
מה העלות התפעולית של Temporal מול כתיבת recovery בעצמכם?	לא הוצגה השוואה של עלות או latency

9. מילון מונחים

מונח	משמעות
Durable execution	הרצת קוד עד הסוף גם כשהתהליך שמריץ אותו קורס. ה-state וההתקדמות נשמרים ב-event history
Workflow	קוד ה-orchestration ב-Temporal. הוא deterministic ומחליט מה הצעד הבא
Activity	צעד עם side effects או שאינו deterministic (קריאת LLM, API, tool call של בנק), שהתוצאה שלו נשמרת
Signal	אירוע חיצוני, למשל אישור אנושי, שנשמר ב-history גם כשאין worker פעיל
Worker	התהליך שמריץ בפועל את קוד ה-workflow וה-activities. אפשר להחליף אותו אחרי קריסה
Event history	רשומה של כל הצעדים ותוצאותיהם. משמשת גם ל-debugging וגם ל-replay
Replay	בנייה מחדש של ה-state של ה-workflow מתוך ה-history על worker חדש
Retry policy	הגדרה אחת של ניסיונות חוזרים שחלה על כל הקריאות, במקום לולאה בקוד
Human in the loop	שלב שבו אדם מאשר או דוחה לפני שהתהליך ממשיך
Agentic loop	לולאה שבה המודל בוחר tool calls, למשל שליפת פרופיל לקוח, ובסוף מנסה brief
Observability	traces, logs וכלי ניטור שמראים שמהו נכשל, אבל לא משחזרים את המצב
KYC / AML	הכרת לקוח ומניעת הלבנת הון, תהליכים פיננסיים ארוכים שמוזכרים כדוגמה
FX quote	הצעת מחיר להמרת מטבע, אחד משלבי ה-payout בדמו
Production-ready	קוד שמוכן לשימוש לקוחות אמיתיים, להבדיל מ-prototype