Badanie

SII TESTING LAB

Sprawdzamy „AI Boom” w automatyzacji testów

WPROWADZENIE

AI na dobre weszło do pracy zespołów IT – także w testowaniu oprogramowania. Nie jako jednorazowy przełom, ale jako zmiana, która w krótkim czasie stała się standardem.

W obszarze testowania oprogramowania wpływ AI jest szczególnie widoczny. Wsparcie przy tworzeniu przypadków testowych, porządkowaniu wymagań, analizie dokumentacji, streszczaniu ticketów czy proponowaniu scenariuszy testów manualnych nikogo już nie dziwi.

Wiemy, że w tych obszarach AI realnie pomaga i przyspiesza pracę – to nie kontrowersyjna teza, lecz praktyka, którą wiele zespołów ma już wdrożoną.

Dlatego w naszym badaniu nie próbujemy mierzyć wpływu AI na testowanie jako całości. Zawężamy pole obserwacji do jednego, konkretnego obszaru: tworzenia testów automatycznych. Nie interesuje nas, czy AI pomaga pisać dokumentację lub przygotowywać testy manualne – tam korzyści są już intuicyjne i powszechnie obserwowane. Chcemy zbadać, jak duży skok daje AI tam, gdzie tester musi usiąść i dostarczyć działający kod.

Ten obszar wydaje się szczególnie ciekawy, ponieważ automatyzacja testów to nie tylko „napisanie czegoś, co działa". Liczy się również jakość rozwiązania: jego czytelność, utrzymywalność, stabilność i zgodność z dobrymi praktykami. Nawet jeśli AI pozwala pisać szybciej, warto zadać pytanie – jak dużo szybciej i czy ten zysk nie odbywa się kosztem jakości?

Problem w tym, że mimo ogromnego boomu na AI wciąż dysponujemy zaskakująco małą ilością danych porównawczych, które pokazują skalę tej zmiany. Wiele opinii opiera się na odczuciach:

> „Pracuje mi się szybciej", „Łatwiej zacząć", „AI odblokowuje mnie na trudniejszych zadaniach"

Brzmi to przekonująco, ale nie odpowiada na pytanie, jak AI sprawdza się w kontrolowanych warunkach, przy tych samych zadaniach. Właśnie to chcemy sprawdzić. Dodatkową trudnością jest efekt ruchomego celu (ang. moving target) – próbujemy ewaluować model, który zmienia się w bardzo szybkim tempie, co w skrajnym przypadku dezaktualizuje badanie z każdą kolejną wersją modelu.

CEL **BADANIA**

Punkt wyjścia jest prosty: zakładamy, że AI przyspiesza pracę testera automatyzującego. Widzimy to na co dzień, widzą to zespoły, widzi to rynek. Pytanie nie brzmi więc „Czy AI pomaga?", ale raczej:

Jak duży jest rzeczywisty skok produktywności i co dzieje się z jakością dostarczanego rozwiązania?

Na to właśnie pytanie ma odpowiedzieć nasze badanie. Chcemy zmierzyć wpływ modeli LLM na pracę testerów automatyzujących w dwóch wymiarach:

Efektywność ilościowa

– ile scenariuszy udało się zrealizować, jaki był średni wynik i jak wyglądał rozrzut rezultatów.

Efektywność jakościowa

– czy dostarczone rozwiązania są zgodne z dobrymi praktykami, czy kod nadaje się do utrzymania i czy wzrost szybkości nie prowadzi do pogorszenia jakości.

Nie próbujemy udowadniać samego faktu, że AI pomaga. Chcemy pokazać skalę tej przewagi w sposób mierzalny, w identycznych warunkach i na tych samych zadaniach.

METODYKA

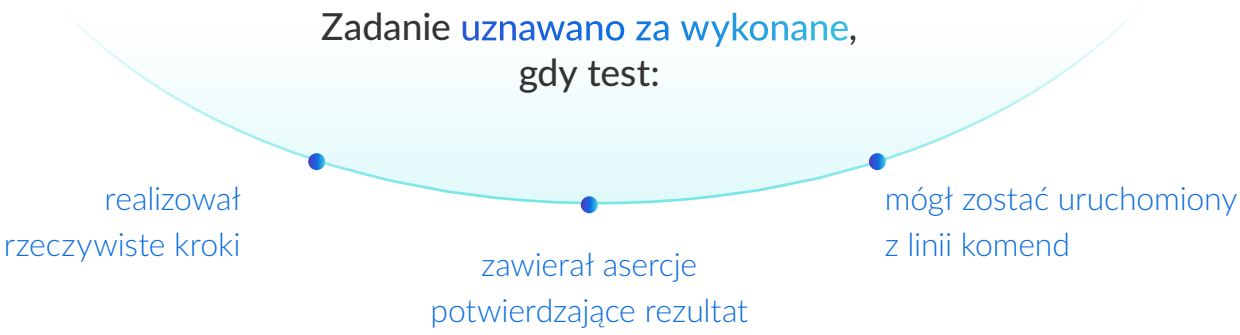
Badanie zaprojektowaliśmy jako kontrolowane porównanie dwóch metod tworzenia rozwiązań do automatyzacji testów. Wzięło w nim udział dziesięć dwuosobowych zespołów, dobranych poziomem doświadczenia tak, aby każdy z nich odpowiadał kompetencjom dwóch regular automation testerów. Zespoły podzielono na dwie grupy:

- AI – zespoły korzystające ze wsparcia narzędzi opartych o LLM.
- Oldschool – zespoły realizujące zadania bez użycia AI.

Każdy zespół otrzymał własne, ale identyczne środowisko przygotowane w oparciu o replikowalne rozwiązania chmurowe. W środowisku nie było żadnego istniejącego frameworka – uczestnicy stanęli przed tzw. greenfieldem: tworzyli rozwiązanie od zera, bez istniejącego kodu, architektury czy ograniczeń wynikających z wcześniejszych decyzji technologicznych.
Uczestnicy nie znali systemu wcześniej – otrzymali jedynie informację, że pracują na sklepie internetowym.

Dokumentację w formie listy przypadków testowych, kryteria oceny oraz pełny dostęp do aplikacji udostępniono dopiero na starcie eksperymentu, co ograniczyło możliwość wcześniejszego przygotowania.

Samo badanie miało formę klasycznego hackathonu – ośmiogodzinnego dnia pracy. Zespoły rozmieszczono w osobnych pokojach projektowych, aby ograniczyć wymianę informacji i zminimalizować zakłócenia.



Taki układ pozwalał porównać nie deklaracje, lecz realny efekt pracy: ile testów zespoły były w stanie dostarczyć i w jakiej jakości.

PRZEBIEG **EKSPERYMENTU**

Obserwacje ogólne

Już w trakcie eksperymentu było widać, że obecność AI nie tylko przyspiesza pracę, ale zmienia jej charakter. Zespoły pracujące w modelu Oldschool większą część czasu spędzały na manualnym kodowaniu, debugowaniu i rozwiązywaniu problemów technicznych krok po kroku. Kiedy pojawiała się trudniejsza przeszkoda – na przykład integracja z XML API albo problem wymagający głębszego dochodzenia technicznego – tempo pracy wyraźnie spadało. Część zespołów zatrzymywała się na dłużej na jednym problemie, co bezpośrednio ograniczało liczbę zadań, które były w stanie zrealizować w ciągu dnia.

W grupie AI przebieg pracy wyglądał inaczej. Problemy techniczne nadal się pojawiały, ale zespoły znacznie szybciej przechodziły od blokady do kolejnej próby rozwiązania. AI skracało czas dochodzenia do poprawnego kierunku, podpowiadało możliwe warianty implementacji i pomagało szybciej wrócić na właściwy tor. W praktyce zespoły AI rzadziej „utknęły", a częściej domykały jeden problem i przechodziły do następnych zadań.

Rola doświadczenia i sposobu pracy z AI

Bardzo wyraźnie ujawniła się różnica w sposobie korzystania z AI. Najlepsze wyniki osiągały zespoły pracujące iteracyjnie: prompt, weryfikacja, poprawka, kolejna iteracja. W takim modelu AI działało jak przyspieszacz pracy inżynierskiej. Z kolei zespoły próbujące generować duże fragmenty kodu jednorazowo częściej traciły czas na poprawianie zbyt szerokich lub niedopasowanych odpowiedzi. Sam dostęp do AI nie gwarantował sukcesu – liczyła się umiejętność pracy z modelem i świadome kierowanie jego podpowiedziami.

Najwyższe rezultaty w grupie AI osiągały zespoły złożone z doświadczonych inżynierów i architektów. To oni potrafili szybko ocenić jakość wygenerowanej propozycji, odrzucić błędny kierunek i poprowadzić model ku lepszemu rozwiązaniu.

AI nie eliminowało znaczenia eksperckiej wiedzy, lecz wzmacniało tych, którzy już potrafili dobrze projektować rozwiązania, oceniać kompromisy i odróżniać dobre podpowiedzi od złych.

Zespoły bez takiego zaplecza korzystały z AI bardziej powierzchownie – model pomagał im pisać szybciej, ale nie kompensował braków w podejmowaniu decyzji architektonicznych. Warto podkreślić, że w grupie AI pojawiły się znacznie większe różnice między zespołami niż w grupie Oldschool.

AI nie wyrównuje różnic kompetencyjnych – wręcz przeciwnie, uwidacznia przepaść między zespołami, które korzystają z niego świadomie, a tymi, które traktują je wyłącznie jako generator kodu.

Ograniczenia modeli

Ciekawa obserwacja pojawiła się na poziomie konkretnych problemów technicznych. Zespoły AI miewały trudność z selektorami dla dynamicznych przycisków. Model próbował rozwiązywać ten problem, ale w niektórych przypadkach wpadał na błędne tropy i brnął w niepoprawne wzorce. Jest to naturalne zachowanie modeli, których rozumowanie opiera się wyłącznie na dostarczonym kontekście. Człowiek w takiej sytuacji prawdopodobnie szybciej podejmie decyzję o zmianie podejścia. Dokładnie tak postępowały zespoły z większym doświadczeniem w automatyzacji i AI. To świetny przykład **ograniczeń modelu i podatności na lokalne „zatrucie kontekstu".**

Co ciekawe, zespoły Oldschool niemal nie zderzyły się z tym problemem w ten sam sposób – selektory dobierały ręcznie i bardziej bezpośrednio, bez wchodzenia w analogiczny tok analizy. Z kolei XML API, będące przeszkodą dla zespołów klasycznych, nie wpłynęło zupełnie na pracę zespołów AI – przeszkoda ta wręcz nie została przez nie zauważona.

Organizacja pracy i zmęczenie

Różnica była też wyraźna na poziomie **organizacji pracy i energii zespołów**. Pod koniec dnia zespoły Oldschool były wyraźnie bardziej zmęczone – większość wysiłku szła w ręczne pisanie i debugowanie kodu.

Zespoły AI pracowały bardziej koncepcyjnie i iteracyjnie. Mniej energii poświęcały na odtwarzanie schematów, więcej na podejmowanie decyzji, ocenę propozycji i sterowanie kolejnymi krokami. To sugeruje, że AI nie tylko przyspiesza pracę, ale przesuwa rolę inżyniera z wykonawcy w stronę operatora procesu i recenzenta rozwiązania, nadając pracy bardziej strategiczny charakter.

WYNIKI

Efektywność ilościowa

Najbardziej uchwytny rezultat eksperymentu widoczny jest w liczbie dostarczonych testów. Różnice były zaskakujące:

ID	Grupa	Liczba testów
A1	AI	196
A2	AI	37
A3	AI	60
A4	AI	9
A5	AI	5
O1	Oldschool	8
O2	Oldschool	5
O3	Oldschool	5
O4	Oldschool	5
O5	Oldschool	5

Zespoły korzystające z AI przygotowały od 5 do niemal 200 testów, podczas gdy zespoły pracujące bez AI dostarczyły od 5 do 8. Skala tej różnicy pokazuje, że użycie **AI nie daje jedynie kosmetycznej poprawy produktywności,** ale może prowadzić do **wielokrotnego wzrostu liczby zrealizowanych scenariuszy w tym samym czasie**.

W grupie Oldschool wyniki były niemal identyczne, co sugeruje, że bez wsparcia AI tempo pracy ograniczały podobne bariery techniczne i koszt ręcznej implementacji. Jedynym wyraźniejszym odstępstwem był zespół, w którym znalazł się architekt – potwierdza to, że doświadczenie techniczne wciąż ma realny wpływ na efekty pracy, nawet bez wsparcia AI.
W grupie AI rozrzut wyników był zdecydowanie większy. Z jednej strony ukazuje to ogromny potencjał narzędzi LLM, z drugiej jasno wskazuje, że sam dostęp do AI nie wystarcza. Zespoły, które potrafiły pracować z modelem iteracyjnie i świadomie, osiągały bardzo wysokie rezultaty; pozostałe zespoły również zyskiwały przewagę, ale znacznie mniejszą. Efektywność ilościowa w środowisku AI zależy więc nie tylko od klasycznych kompetencji technicznych, ale również od umiejętności pracy z modelem.

Z perspektywy liczbowej AI nie tylko zwiększyło tempo dostarczania testów, ale też znacząco skróciło czas tracony na lokalne blokery.

Zespoły bez AI częściej „spalały" dużą część dnia na pojedynczych problemach technicznych, podczas gdy zespoły AI szybciej zamykały temat i przechodziły dalej.

Ten mechanizm wydaje się jednym z głównych powodów tak ogromnej różnicy w liczbie ukończonych scenariuszy.

Efektywność jakościowa

Ocena rozwiązań nie opierała się wyłącznie na liczbie dostarczonych testów. Równie ważne było **to, jak zostały zaprojektowane i zaimplementowane**. Innymi słowy: nie tylko „ile", ale przede wszystkim „jak dobrze".

Dlatego przyjęliśmy osiem kryteriów jakości, które pozwalają spojrzeć na rozwiązania z perspektywy realnej pracy inżynierskiej:

K1 - Zgodność testu z celem biznesowym i pokrycie wymagania.

Czy test weryfikuje istotę scenariusza, a nie jedynie sekwencję akcji? Czy kryterium sukcesu i porażki zdefiniowano jednoznacznie?

K2 – Dane testowe i przygotowanie stanu.

Niezależność testu od ręcznego przygotowania środowiska, sposób tworzenia danych, ich unikalność, deterministyczność oraz cleanup po wykonaniu scenariusza.

K3 – Stabilność rozwiązania.

Odporność testów na wolniejsze środowisko, animacje, opóźnienia i stan DOM; stosowanie właściwych mechanizmów oczekiwania zamiast arbitralnych opóźnień.

K4 – Jakość selektorów i lokalizatorów.

Stabilność, spójność w obrębie projektu i unikanie lokatorów kruchych, nadmiernie zależnych od struktury layoutu.

K5 – Architektura testu i wzorce.

Sensowne zastosowanie Page Object Model, komponentów UI, helperów, fixture'ów i warstw setupowych.

K6 – Jakość asercji.

Czy asercje pojawiają się w kluczowych punktach scenariusza, weryfikują rzeczywisty wynik biznesowy i są odporne na nieistotne zmiany interfejsu?

K7 – Diagnostyka i obsługa błędów.

Użyteczność logów, screenshotów i artefaktów wspierających analizę niepowodzeń.

K8 – Engineering.

Czytelność kodu, spójność stylistyczna, nazewnictwo, struktura projektu, unikanie duplikacji i ogólna dojrzałość rozwiązania.

Każde repozytorium oceniano w skali pięciostopniowej dla każdego z kryteriów, a następnie wyznaczano średnią ocenę jakościową. Ze względu na ograniczony czas pracy zespołów, oceniano przede wszystkim poprawność i dojrzałość ukończonych elementów. Przyjęto, że wydzielenie adresów środowiskowych, URL API czy adresu panelu administracyjnego do osobnego pliku konfiguracyjnego jest rozwiązaniem poprawnym i pożądanym, natomiast obecność sekretów, tokenów i danych użytkowników w repozytorium oceniano negatywnie.

ID	Grupa	K1	K2	K3	K4	K5	K6	K7	K8	Śr.
A1	AI	3	4	4	3	4	3	5	3	3,6
A2	AI	4	3	4	3	5	4	4	5	4,0
A3	AI	3	5	4	4	5	3	4	3	3,9
A4	AI	2	3	3	3	4	2	4	2	2,9
A5	AI	3	4	3	3	4	2	3	3	3,1
O1	Oldschool	4	5	4	4	5	3	4	5	4,3
O2	Oldschool	3	3	3	3	4	2	2	3	2,9
O3	Oldschool	2	3	2	2	3	1	1	2	2,0
O4	Oldschool	3	3	4	3	3	2	4	2	3,0
O5	Oldschool	3	2	4	3	4	2	2	3	2,9

ANALIZA
WYNIKÓW JAKOŚCIOWYCH

Średnia jakościowa grupy AI wyniosła 3,53/5, natomiast grupy Oldschool – 3,02/5. Różnica nie jest tak spektakularna, jak w liczbie testów, ale pozostaje **wyraźna i konsekwentna.**

Przewaga zespołów AI wynikała przede wszystkim z lepszego przygotowania danych testowych i setupu, mocniejszej architektury frameworków, częstszego stosowania uporządkowanych wzorców projektowych oraz lepszej diagnostyki awarii. W repozytoriach AI częściej występowały oddzielne fixture'y, helpery, warstwy page objectów, unikalne dane testowe oraz artefakty wspierające analizę błędów: screenshoty, trace'y i rozbudowane logowanie.
Innymi słowy: AI nie tylko przyspieszało pisanie testów, ale często prowadziło do bardziej ustrukturyzowanych i „inżynierskich" rozwiązań.

Nie oznacza to jednak braku problemów. W części repozytoriów widoczna była nierówna jakość asercji, obecność sekretów w konfiguracji oraz ślady szybkiego, nie w pełni domkniętego wytwarzania. Szybkość nadal miała swoją cenę – choć nie tak wysoką, jak można by się spodziewać.
W grupie Oldschool obraz był bardziej zróżnicowany. Najlepsza pojedyncza próba – repozytorium O1 – osiągnęła wynik 4,3/5 i należała do najmocniejszych technicznie rozwiązań w całym badaniu. O1 wyróżniało się bardzo dobrą separacją warstw, sensownym setupem danych przez API, cleanupem oraz wysoką kulturą engineeringową. Pozostałe repozytoria Oldschool częściej ujawniały słabości: płytkie asercje, słabą diagnostykę, obecność sekretów czy niższą dojrzałość infrastruktury testowej.

Ostatecznie przewaga AI nie sprowadzała się więc wyłącznie do liczby dostarczonych testów – obejmowała również jakość organizacji pracy i konstrukcji samego frameworka.

WNIOSKI

AI realnie i znacząco zwiększa produktywność.

Różnica między 5–8 testami a 5–200 testami nie pozwala mówić o marginalnej poprawie – to zmiana poziomu wydajności pracy. Co istotne, wyższa produktywność nie odbyła się kosztem jakości. Grupa AI wykazała się większą spójnością zarówno pod kątem przygotowania danych, architektury, jak i utrzymywalności kodu — widoczne było to szczególnie w strategii przygotowania danych, podejściu do selektorów i jednolitym traktowaniu asercji.

Skuteczne użycie AI jest kompetencją samą w sobie.

Najlepiej radziły sobie zespoły korzystające z modelu iteracyjnie, małymi krokami, z ciągłą weryfikacją i korektą. Model nie zastępuje procesu inżynierskiego – wzmacnia go, ale tylko wtedy, gdy zespół potrafi nim świadomie sterować.

Doświadczenie odgrywa kluczową rolę.

Najlepsze wyniki osiągały zespoły z architektami, co potwierdza, że AI nie eliminuje znaczenia eksperckiej wiedzy. Wręcz przeciwnie – zwiększa zwrot z jej posiadania.

AI znacząco skraca czas wychodzenia z blokad technicznych.

Tam, gdzie zespoły Oldschool zatrzymywały się na przeszkodach i traciły czas na ręczne dochodzenie do rozwiązania, zespoły AI szybciej odzyskiwały tempo. Zdolność do szybkiego rozwiązywania blokad wydaje się jednym z kluczowych źródeł przewagi.

Modele mają swoje ograniczenia.

Problem z selektorami dynamicznych przycisków ujawnił, że AI potrafi wejść w błędną ścieżkę i przez pewien czas ją wzmacniać, zamiast z niej wyjść. Nawet przy dużym wzroście produktywności techniczna ocena człowieka pozostaje niezbędna. AI przyspiesza, ale nie zwalnia z myślenia.

AI zmienia charakter pracy inżyniera.

W modelu Oldschool dominowały manualne działania wykonawcze: pisanie, poprawianie, debugowanie. W modelu AI większą rolę odgrywały decyzje, iteracja, ocena jakości i dobór kolejnych kroków. Dzięki łatwemu dostępowi do dużej ilości uszeregowanych informacji modele wspierały bardziej przemyślane i spójne decyzje. Doświadczeni automatycy coraz rzadziej ograniczają się do tworzenia kodu, a coraz częściej pełnią rolę architektów interakcji z modelami – definiując problem, weryfikując wyniki i decydując, co trafia do systemu. Tę zmianę można porównać do momentów zmiany paradygmatów, kiedy analogiczne różnice było widać między inżynierami bazującymi na wiedzy książkowej a tymi, którzy korzystali z platform typu Stack Overflow. Sami uczestnicy badania zauważyli, że w jego toku lepiej zrozumieli, jak przekazywać informacje modelom językowym.

Przewaga jakościowa potwierdza obraz ilościowy

Średnia grupy AI wyniosła 3,53/5 wobec 3,02/5 grupy Oldschool. Repozytoria AI częściej wykorzystywały lepszy setup danych, bardziej uporządkowaną architekturę, wyraźniejszą separację warstw i bogatszą diagnostykę. Przewaga ta nie jest jednak absolutna – najlepsza pojedyncza próba w grupie Oldschool osiągnęła 4,3/5, co dowodzi, że jakość rozwiązania silniej zależy od dojrzałości i doświadczenia konkretnego zespołu. AI częściej natomiast pomagała osiągnąć wyższy poziom dojrzałości rozwiązania w tym samym, ograniczonym czasie.

CO DALEJ? KONTYNUACJA EKSPERYMENTU

Pierwsza edycja pokazała jasno: AI potrafi dać ogromną przewagę, ale o wyniku decyduje sposób jego wykorzystania. Różnice między zespołami dowiodły, że to nie sama technologia, lecz podejście do pracy z nią decyduje o wynikach. Dlatego kolejna edycja Testing Lab skupi się na tym, które metody i narzędzia faktycznie przyspieszają pracę, utrzymują wysoką jakość kodu i ograniczają koszty. Jeśli pierwsza edycja odpowiadała na „czy AI działa?", kolejna odpowie na „jak używać AI, by maksymalizować efekty bez kompromisów".

POZNAJ JURY TESTING LAB – AI EDITION:

Krzysztof Bednarski

Solution Architect w Sii Polska. W badaniu koncentrował się na ocenie dojrzałości inżynierskiej rozwiązań, ze szczególnym uwzględnieniem jakości architektury frameworków, ich skalowalności oraz wpływu narzędzi opartych o LLM na codzienną pracę doświadczonych inżynierów testów.

Remigiusz Bednarczyk

Test Manager w Sii Polska. Analizował momenty, w których wykorzystanie AI przynosi zespołom realną przewagę biznesową, identyfikując jednocześnie granice efektywności i obszary, w których nadmierne poleganie na modelach LLM może obniżać jakość lub kontrolę nad rozwiązaniem.

Tomasz Kuran

Solution Architect w Sii Polska. W swojej analizie skupiał się na identyfikacji ryzyk związanych z wykorzystaniem AI oraz na ocenie długoterminowej utrzymywalności rozwiązań. Szczególną uwagę poświęcił ewolucji standardów architektonicznych w kontekście pracy z modelami językowymi.

Masz pytania?

Skontaktuj się z nami

Iuliia Toporova
Business Development Manager