

Dostarczanie oprogramowania w erze sztucznej inteligencji

Perspektywa Sii na skalowanie AI i agentów w cyklu wytwarzania oprogramowania

Streszczenie

Sztuczna inteligencja jest już integralną częścią dostarczania oprogramowania. Nie wszędzie, nie równomiernie i nie zawsze z jasno widocznymi rezultatami - ale kierunek jest już jasny. AI przestaje być narzędziem wspomagającym pracę pojedynczych deweloperów i staje się integralną częścią systemu dostarczania oprogramowania.

Ten raport przeznaczony jest dla liderów, którzy chcą przejść od eksperymentów z AI do mierzalnej poprawy w dostarczaniu oprogramowania. Raport pokazuje, gdzie obecnie znajduje się rynek i identyfikuje sygnały istotne dla AI w cyklu wytwarzania oprogramowania (SDLC).

Jedną z najważniejszych części raportu jest przedstawienie, jak Sii podchodzi do skalowania AI w praktyce: od modelu dojrzałości AI-native SDLC, przez governance i bezpieczeństwo, po wykorzystanie metod kontrolowania pracy agentów. Raport przedstawia także wyniki własnych eksperymentów Sii w tym obszarze.

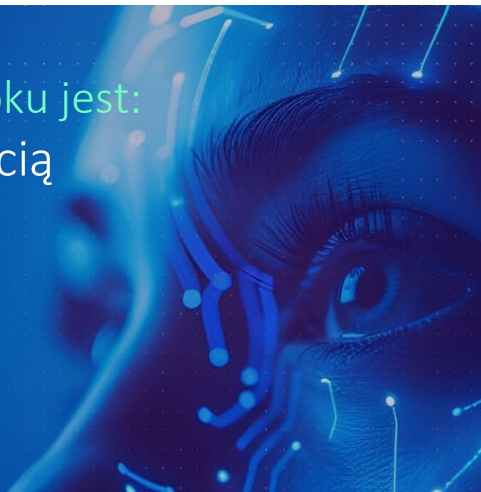
W sekcji: „Plan działań dla liderów na 6-12 miesięcy" znajduje się praktyczny plan do, którego mogą przejść bezpośrednio czytelnicy, którzy poszukują głównie konkretnych kolejnych kroków.

Większość organizacji zaczęła od produktywności deweloperów: generowania kodu, analizy kodu, debugowania, dokumentacji i testów jednostkowych. Te przypadki użycia są ważne. Są widoczne i łatwe do pilotażu. Jednak w sytuacji gdy liderzy patrzą wyłącznie na lokalne przyspieszenie pracy, może prowadzić do fałszywego poczucia postępu.

Dostarczanie oprogramowania wykracza poza samo tworzenie kodu. Jeżeli AI przyspiesza kodowanie, ale przegląd lub testowanie działają na starych zasadach, biznes nie zauważa dużej zmiany.

Tworzy to problem z metrykami sukcesu i nie daje odpowiedzi na pytanie, czy AI faktycznie poprawia przepływ pracy w całym systemie dostarczania oprogramowania.

Głównym wyzwaniem drugiej połowy 2026 roku jest:
różnica między skalą adopcji AI a zdolnością organizacji do wykazania jej wpływu na dostarczaną wartość



Kluczem do sukcesu jest zrozumienie, czy AI faktycznie zmienia cały cykl wytwarzania oprogramowania, czy tylko wzrasta liczba użytych narzędzi AI. Wiele organizacji ma już dostęp do narzędzi AI. Kolejnym wyzwaniem jest zaprojektowanie sposobu, w jaki zespoły pracują i dostarczają oprogramowanie z ich pomocą. Oznacza to szukanie praktycznych sygnałów pokazujących, gdzie AI poprawia przepływ, gdzie tworzy nowe wąskie gardła i gdzie przed skalowaniem potrzebne są silniejsze mechanizmy kontroli.

Na co zwrócić szczególną uwagę:

1. Asystenci oparci na sztucznej inteligencji stają się powszechnym elementem pracy inżynierskiej.
2. Testowanie i kontrola jakości to istotne obszary zastosowań AI w najbliższej przyszłości.
3. Agenci AI przechodzą od fazy demonstracyjnej do wspierania prawdziwych procesów.
4. Skalowanie AI w SDLC wymaga governance: wspólnych zasad bezpieczeństwa i wykorzystania AI.
5. O tym, czy adopcja AI pozostanie zbiorem lokalnych eksperymentów, czy stanie się powtarzalnym modelem pracy, zdecyduje sposób budowy i rozwoju platform deweloperskich.

Realny wpływ AI w SDLC coraz częściej wynika z usprawniania systemu dostarczania oprogramowania, a nie wyłącznie z przyspieszania samego tworzenia kodu.

Główna teza

Realna wartość AI w SDLC pojawia się, gdy poprawa produktywności pojedynczych zadań przekłada się na usprawnianie całego cyklu wytwarzania oprogramowania

1. Sytuacja na rynku

W wielu zespołach deweloperskich AI przestała być rozwiązaniem eksperymentalnym i stała się elementem codziennej pracy. Co ważniejsze, jej zastosowanie wykracza coraz bardziej poza pisanie kodu. Kolejne role w procesie dostarczania oprogramowania wspierane są za pomocą AI.

Od tworzenia przypadków testowych i porządkowania wymagań, przez pracę z dokumentacją systemową, po analizę incydentów. Jest to wyraźny sygnał, że adopcja AI przesuwa się z pojedynczych zadań deweloperskich na szerszy przepływ pracy w SDLC.

> *Ponad 50% respondentów badania Gartnera wykorzystuje sztuczną inteligencję do pisania i rozumienia kodu, debugowania, generowania dokumentów oraz tworzenia przypadków testowych*

Gartner, *How to Maximize the Impact of Agentic AI in the SDLC*, 19 stycznia 2026.

> *Zgodnie z badaniem Gartnera – 66% respondentów wykorzystujących AI w co najmniej 10 przypadkach użycia deklarowało poprawę szybkości wprowadzania zmian na rynek.*

Gartner, *How to Maximize the Impact of Agentic AI in the SDLC*, 19 stycznia 2026.

Wdrożenie narzędzi jest faktem. Sukces jest mniej pewny.

Wiele firm dodało narzędzia AI do procesu dostarczania oprogramowania bez przeprojektowywania sposobu, w jaki ten proces działa. Zespoły mogą szybciej generować kod, tworzyć testy i podsumowywać incydenty, ale często w dalszym ciągu występują czasochłonne przekazania pracy między zespołami lub członkami zespołu, oczekiwanie na zatwierdzenie, kolejki przeglądów i ograniczenia związane z dostarczaniem. W rezultacie mamy do czynienia z fragmentarycznym wdrożeniem, a nie z przemodelowanym systemem dostarczania oprogramowania.

W tej samej organizacji często dochodzi do sytuacji, w której jeden zespół intensywnie korzysta ze sztucznej inteligencji, inny jej unika, a trzeci eksperymentuje poza oficjalnie zatwierdzonymi narzędziami. Polityki są niejasne, a wskaźniki często śledzą wyłącznie wykorzystanie narzędzi, zamiast skupiać się na miarach dotyczących dostarczania oprogramowania. Utrudnia to ustalenie, czy sztuczna inteligencja faktycznie poprawia przepływ, jakość lub kontrolę ryzyka.

W tej sytuacji przydatne jest podejście rozróżniające poziomy adopcji AI w cyklu rozwoju oprogramowania:

Na pierwszym poziomie AI wspiera pojedyncze zadania programistów, ale działa w jasno określonych ramach: z użyciem zatwierdzonych narzędzi, według zasad dotyczących danych i utrzymywanych instrukcji. Odpowiedzialność za rezultat nadal w pełni spoczywa na człowieku.

Kolejny etap zaczyna się wtedy, gdy korzystanie z AI przestaje być indywidualnym nawykiem i staje się wspólną, powtarzalną praktyką zespołu w wybranych fazach SDLC. Zespół korzysta ze wspólnych promptów, instrukcji i kontekstu projektu oraz mierzy efekty wdrożenia AI.

Na najwyższym poziomie agenci realizują pracę wieloetapową, a zasady są egzekwowane technicznie poprzez monitoring, bramki decyzyjne i kontrolę jakości.



Wykonywanie zadań przez agentów

Agenci AI obsługują ograniczone zadania z dostępem do narzędzi: aktualizują zależności, tworzą pull requesty, klasyfikują luki w zabezpieczeniach, utrzymują testy, przygotowują dokumentację. Większość organizacji plasuje się pomiędzy wsparciem zadań a wsparciem przepływu pracy. Niektóre testują realizację zadań przez agentów.

Niewiele jednak jest gotowych na szeroką autonomię. Sii dostrzega ten sam schemat w rozmowach z klientami. Częściej pada pytanie „które narzędzie AI powinniśmy kupić?”, a rzadziej „jak wykorzystać AI bez utraty kontroli nad jakością dostaw, bezpieczeństwem i kosztami?”.

Wniosek

Rynek wyszedł już poza fazę podstawowej ciekawości wobec sztucznej inteligencji. Teraz najtrudniejszym zadaniem jest zaprojektowanie efektywnej adopcji.

2. Pułapka produktywności

Najprostszą narracją o AI jest przyspieszenie kodowania. Jest to jednak niepełny obraz.

To, że deweloper może szybciej zakończyć zadanie dzięki AI, nie oznacza, że organizacja szybciej dostarcza oprogramowanie. Praca nadal musi przejść przez etap przeglądu kodu, testy, kontrole bezpieczeństwa, dostarczanie i przygotowanie do wdrożenia na środowisko produkcyjne.

Przyspieszenie jednego etapu nie usuwa ograniczeń całego procesu. Szybsze kodowanie może po prostu zwiększyć kolejkę do review, większy wolumen zmian obciążyć QA, a późne kontrole bezpieczeństwa wygenerować więcej poprawek tuż przed wydaniem.

Podobnie ręczne zatwierdzenia mogą nadal spowalniać release, nawet jeśli wcześniejsze etapy zostały skutecznie zautomatyzowane.

W ten sposób wiele wdrożeń AI traci wiarygodność raportując adopcję narzędzi, liczbę promptów lub zaoszczędzony czas w izolowanych zadaniach. Liderzy pytają wówczas, dlaczego czas realizacji, wskaźniki defektów ani przewidywalność wdrożeń nie uległy poprawie.

Odpowiedź jest zazwyczaj prosta: AI była stosowana tam, gdzie było najłatwiej, nie tam, gdzie system miał wąskie gardła.



Zespół ze słabym pokryciem testami automatycznymi może zyskać więcej na wspieranym przez AI generowaniu testów, niż na autouzupełnianiu kodu. Zespół pracujący ze starszymi systemami (legacy) odniesie większe korzyści z wyjaśniania kodu i wsparcia w modernizacji. Zespół w środowisku podlegającym ścisłym regulacjom doceni pomoc w generowaniu dokumentacji i zapewnianiu zgodności.

Wniosek

Szybsze kodowanie ma wartość tylko wtedy, gdy reszta systemu dostarczania jest w stanie je obsłużyć.

3. CO MIERZYĆ?

Metryki dotyczące AI często skupiają się na niewłaściwych obszarach. Liczba promptów, zaakceptowanych sugestii czy wygenerowanych linii kodu pokazują aktywność, ale nie wartość. Nie mówią liderom, czy oprogramowanie szybciej trafiło do użytkowników, czy wzrosła jakość, ani czy zespoły tracą mniej czasu na poprawki.



Praktyczny model pomiaru powinien obejmować cztery obszary.

Jakość i ryzyko: Czy zmienił się wskaźnik przepuszczanych błędów? Czy spadła konieczność wprowadzania poprawek? Czy wzrosła stabilność testów?

Adopcja: Kto używa AI? Jakich narzędzi? W jakich zespołach? Do jakich zadań w SDLC?

Przepływ (Flow): Czy skrócił się czas realizacji zadania? Czy spadł czas przeglądu kodu? Czy poprawił się cykl dostarczania? Czy zadania krócej czekają w kolejkach?

Doświadczenie deweloperów i wyniki biznesowe: Czy deweloperzy zgłaszają mniej żmudnej pracy? Czy poprawił się proces wdrażania pracowników? Czy klienci szybciej otrzymują zmiany?

W przypadku przepływów agentowych konieczne są dodatkowe kontrole:

- Jak często zmiany wygenerowane przez AI przechodzą weryfikację za pierwszym podejściem?
- Jak często człowiek musi poprawiać wyniki pracy AI?
- Czy kontekst projektu używany przez AI jest aktualny?
- Które procesy agentowe posiadają jasne kryteria akceptacji?
- Które akcje są logowane i możliwe do audytu?

Wniosek

Celem nie jest tworzenie obciążeń biurokratycznych, lecz zaprzestanie mylenia aktywności z postępem.

4. Od asystentów do agentów

Asystenci AI pomagają ludziom wykonywać zadania. Agenci AI próbują samodzielnie realizować ograniczony zakres prac.

Różnica jest znacząca. Asystent potrafi wyjaśnić metodę, zasugerować test czy przygotować szkic dokumentacji. Człowiek zachowuje pełną kontrolę na każdym etapie.

Agent potrafi przeanalizować zgłoszenie, zbadać pliki, wprowadzić zmiany, uruchomić testy i przygotować pull request. Może działać w tle, podczas gdy deweloper skupia się na innym zadaniu. Zmienia to rolę sztucznej inteligencji z pomocnika na aktywnego uczestnika przepływu pracy.

W połowie 2026 r. typowe scenariusze agentowe obejmują:

- przygotowywanie pull requestów;
- generowanie i utrzymywanie testów;
- aktualizację zależności;
- naprawianie luk bezpieczeństwa;
- podsumowania z przeglądów kodu;
- aktualizacje dokumentacji;
- podsumowania incydentów;
- propozycje refaktoryzacji.

Nie oznacza to, że cykl SDLC staje się w pełni autonomiczny. Ograniczenia pozostają wyraźne: niepełny kontekst, słabe zrozumienie domeny, ryzyka związane z bezpieczeństwem i licencjonowaniem, niedopasowanie architektoniczne oraz trudności w audytowaniu decyzji podejmowanych przez maszyny.

Ścieżka dojrzałości w standardzie Sii AI-native SDLC:

Poziomy i kryteria przejścia są elementem standardu Sii AI-native SDLC. Służą do podejmowania decyzji o zwiększaniu autonomii w ocenianym zakresie.

1

L1 Assisted:

Wsparcie zadań przy zatwierdzonych narzędziach, zasadach danych, utrzymywanych instrukcjach i ludzkiej odpowiedzialności.

2

L2 Integrated:

Wspólna, powtarzalna praktyka zespołu w wybranych fazach SDLC, utrzymywane zasoby i pomiar względem baseline'u.

3

L3 AI-native:

Wieloetapowa praca agentowa, techniczne egzekwowanie standardu, monitoring, bramki człowieka oraz trwałe dowody jakości i wyników.

Poziomy są kumulatywne. L3 wymaga m.in. co najmniej dwóch kolejnych okresów pomiarowych bez wzrostu wskaźników defektów i reworku względem baseline'u oraz bez incydentu produkcyjnego przypisanego zmianie AI, która przeszła review. Dowód dotyczy ocenianego zakresu.

Dojrzałość usługi i autonomia zadania to odrębne decyzje. Dobrze działająca usługa może celowo pozostać na L1 lub L2, a pojedyncza osoba korzystająca z zaawansowanego agenta nie czyni zespołu AI-native.

Wniosek

Agenci sprawdzają się najlepiej tam, gdzie zadanie ma jasne granice, pełny kontekst jest dostępny, a proces weryfikacji przez człowieka jest wbudowany w system.

5. Model kontra „uprząż”

Najlepsze modele AI coraz mniej różnią się między sobą pod względem jakości. W praktyce oznacza to, że sam wybór dostawcy przestaje być głównym źródłem przewagi konkurencyjnej. Wiodące modele osiągają dziś zbliżony poziom w zadaniach programistycznych - różnice częściej dotyczą sposobu działania i szczegółów niż samej klasy możliwości.

Coraz większe znaczenie ma więc korporacyjna warstwa, która steruje pracą modelu i osadza go w kontekście konkretnego projektu: dostarcza wiedzę domenową, zasady architektoniczne, standardy pracy, ograniczenia, mechanizmy kontroli, budżety oraz reguły kierowania zadań. Tę warstwę określa się mianem „harnessu”, lub „meta harnessu” albo po prostu „uprzęży”.

> *„Przewaga nie jest w modelu AI, a w sposobie jego integracji z procesem, narzędziami i wiedzą”*
>
> *Wniosek z 11-dniowego eksperymentu Sii: 70 ticketów, 3 projekty produkcyjne, pełna telemetria.*

Wraz z dojrzewaniem modeli rośnie znaczenie harnessu

Rozróżnienie między modelem a harnessem ma bardzo praktyczne znaczenie. Bez harnessu praca z agentem przypomina rozmowę: człowiek zadaje pytanie, agent odpowiada, a człowiek ocenia wynik. Wszystko odbywa się krok po kroku i bez gwarancji, że ten sam proces zostanie wykonany w powtarzalny sposób.

Z harnessem zadanie przechodzi przez zdefiniowany proces: planowanie, implementację, testy i przegląd, z jasno określonymi punktami kontroli między kolejnymi etapami. Model odpowiada za rozumowanie, a harness pilnuje, by jego działania pozostały w określonych ramach i nie prowadziły do niekontrolowanych zmian w kodzie.

Harness odpowiada za sześć rzeczy: obserwację otoczenia, dostarczanie kontekstu, kontrolę działań, ich wykonanie, pamięć stanu i weryfikację wyników.

Skuteczny harness opiera się na trzech filarach.



Kontrakty i bramki

Odpowiedź agenta ma z góry ustaloną formę (np. JSON), a nie swobodny opis. Jeśli czegoś brakuje lub wynik jest niejednoznaczny, proces się zatrzymuje — agent nie zgaduje. Człowiek zatwierdza tylko dwa momenty: plan i scalenie kodu. Pozostałe kroki są zautomatyzowane, ale nie pozostają bez nadzoru.



Routing i ekonomia

Każdą rolę — planowanie, implementację, przegląd — obsługuje model dobrany do zadania i do budżetu. Koszt jednego zadania ma twardy limit, a jego przekroczenie przerywa pracę, zamiast otwierać nieograniczony rachunek.



Audytowalność

Po każdym przebiegu zostaje pełny zapis: plan, podjęte decyzje, koszt i log działań. Nadzór nie jest więc zapisem w polityce, ale naturalnym efektem tego, jak zbudowano proces.

Dwie warstwy – model i harness - muszą działać razem: agent wykonuje zadanie, a harness wyznacza granice i sprawdza rezultat.

Liczba bramek ludzkich powinna wynikać przede wszystkim z poziomu ryzyka, zakresu uprawnień i stopnia izolacji środowiska. W izolowanym harnessie, w którym agent działa w granicach zatwierdzonego planu, liczba wymaganych decyzji człowieka może być mniejsza. W środowisku deweloperskim lub przy szerszych uprawnieniach potrzebne są dodatkowe punkty kontroli.

Wniosek

Harness nie zastępuje modelu — steruje jego działaniem.
To harness w dużej mierze decyduje o skuteczności agenta.

Harness dopasowany do potrzeb organizacji

Jak pokazują badania, skuteczność narzędzia leży nie tyle w samym modelu AI, co w zbudowanym wokół niego harnessie i dopasowaniu do domeny projektu (Guo i in., 2026, arXiv:2606.20683). Wnioski te potwierdza eksperyment przeprowadzony wewnątrz Sii, gdzie rozbudowaliśmy Claude Code – harness dostarczany przez Anthropic – o własne devkity, integracje źródeł danych, subagentów oraz integrację wewnętrznych narzędzi. W ten sposób 86% ticketów przeszło pełny proces bez interwencji człowieka poza zdefiniowanymi bramkami, wobec 33% przy bazowej konfiguracji Claude Code.



Wskaźnik dostarczenia: 86%

60 z 70 ticketów przeszło pełny proces zgodnie ze standardową ścieżką. Pozostałe zostały zatrzymane na konkretnym etapie, z jasno wskazaną przyczyną — żaden nie zakończył się bez wyjaśnienia.



Koszt: Średnio 6,11 USD za dostarczony ticket

Mediana kosztu wyniosła 2,96 USD, a najdroższy ticket kosztował 31,21 USD. Wyższy koszt nie wynikał jednak z większej trudności, lecz z szerszego zakresu zadania. Dlatego podział dużych ticketów przed rozpoczęciem pracy jest jedną z najprostszych i najtańszych metod optymalizacji procesu.



Skala autonomii: 9,7×

536 wywołań agentów przy 55 decyzjach człowieka. Z naszych obserwacji wynika, że w modelu „agent + prompt” proporcja zbliżona jest do 1:1, ponieważ niemal każdy krok agenta wymaga reakcji człowieka. Harness automatyzuje tę pętlę i ogranicza liczbę punktów, w których potrzebna jest interwencja.



Zero cichych awarii

Każde zatrzymanie miało jasno wskazaną przyczynę techniczną. Proces działa w trybie fail-closed: brak pozytywnej weryfikacji oznacza zatrzymanie, a agent nie zatwierdza własnej pracy.

Najlepiej sprawdza się w projektach typu Brownfield

Istniejący produkt, stały napływ ticketów i wiele osób pracujących równolegle. Najważniejsze są jakość, przewidywalność i kontrola nad każdym zgłoszeniem - nie szybkość pierwszej zmiany w kodzie.

Nie zawsze warto: Greenfield solo

Jedna osoba, szybki prototyp i prosty cel: jak najszybciej uruchomić działającą aplikację. W takim scenariuszu bramki, dodatkowe artefakty i limity mogą generować więcej narzutu niż realnej wartości.

Harness nie rozwiązuje problemu słabo przygotowanych ticketów ani zadań wymagających eksploracji — te nadal pozostają po stronie ludzi. Zmienia się natomiast rola dewelopera: mniej czasu poświęca na pisanie kodu, a więcej na precyzyjne definiowanie zadań i ocenę pracy agenta. Odpowiedzialność przesuwa się tam, gdzie człowiek nadal ma przewagę: w określaniu intencji i podejmowaniu decyzji. Eksperyment pokazuje, że kontrolowany proces agentowy może działać na projektach produkcyjnych, z interwencją człowieka ograniczoną do określonych bramek oraz pomiarem kosztów na poziomie ticketu. Nie porównuje jednak czasu ani pełnego kosztu dostarczenia z procesem bez agentów i nie stanowi benchmarku dla innych typów projektów.

Wniosek

Harness rozszerza możliwości modelu, zapewniając kontrolę, powtarzalność i dostęp do kontekstu domenowego — szczególnie w większych, złożonych projektach.

Cztery filary skalowania agentów SDLC

Kompetencje: Rola dewelopera zmienia się - mniej czasu może zajmować samo pisanie kodu, a więcej - precyzyjne definiowanie zadań, weryfikacja rezultatów i audyt pracy agenta.

Governance: Zasady działania agenta muszą być zapisane w procesie: kto zatwierdza plan, kiedy agent może rozpocząć pracę, co wymaga akceptacji człowieka i co zostaje zapisane do audytu.

Security: Agent powinien mieć dostęp tylko do tego, czego naprawdę potrzebuje — konkretnych repozytoriów, narzędzi i środowisk. Wrażliwe dane mogą pozostać wewnątrz organizacji, a wybrane zadania mogą być kierowane do modeli lokalnych.

Koszty: Budżet powinien działać jak bezpiecznik: limit na zadanie, kontrola kosztu na każdym etapie i automatyczne zatrzymanie po przekroczeniu ustalonego progu. Koszt powinien być znany przed uruchomieniem, nie dopiero na koniec.

W praktyce przy skalowaniu agentów wracają te same pytania:

- Kto zatwierdza plan, zanim agent zacznie działać?
- Co się dzieje, gdy agent przekroczy budżet?
- Czy da się odtworzyć, dlaczego dana zmiana trafiła na produkcję?
- Czy wrażliwe dane mogą pozostać lokalnie, zamiast trafiać do chmury?
- Czy koszt zadania jest znany przed jego uruchomieniem?

Wybór modelu jest decyzją taktyczną. Warunki skalowania – strategiczną.

Wniosek

Wraz z dojrzewaniem modeli coraz większa część przewagi przesuwa się do systemu wokół modelu - harnessu, kontekstu, kontroli i sposobu organizacji pracy.

6. Co działa w SDLC

AI może wspierać cały SDLC, ale jej wartość, dojrzałość i ryzyko różnią się w zależności od etapu procesu.



Odkrywanie i wymagania

Na tym etapie AI pomaga przede wszystkim uporządkować informacje: streszcza materiały od interesariuszy, grupuje feedback i wychwytuje niejasności. Może też przygotować szkic historyjek użytkownika lub kryteriów akceptacji, ale decyzje produktowe nadal wymagają oceny człowieka.



Architektura i projektowanie

W pracy architektonicznej AI przydaje się do analizy dokumentacji, przygotowania szkiców ADR-ów, mapowania zależności i oceny wpływu zmian. Ostateczna decyzja należy jednak do architekta, bo wybory projektowe przekładają się na koszt, bezpieczeństwo i łatwość utrzymania systemu.



Kodowanie

Najbardziej widoczny efekt AI pojawia się dziś w codziennej pracy z kodem: jego generowaniu, wyjaśnianiu legacy, refaktoryzacji, debugowaniu i tworzeniu testów jednostkowych. Korzyść szybko znika, jeśli pozornie poprawny kod trafia dalej bez rzetelnego przeglądu.



Testowanie i jakość

Testowanie daje szerokie pole do zastosowania AI - od generowania przypadków i danych testowych po szkice automatyzacji, analizę błędów i utrzymanie testów. To praktyczny obszar także dlatego, że rezultat można stosunkowo łatwo zweryfikować i zmierzyć.

Bezpieczeństwo



W obszarze bezpieczeństwa AI może przyspieszyć analizę zależności, klasyfikację luk, modelowanie zagrożeń i przygotowanie zaleceń dotyczących bezpiecznego kodowania. Te zastosowania wymagają jednak szczególnie dokładnej weryfikacji, bo błędna rekomendacja może zwiększyć ryzyko zamiast je ograniczyć.

DevOps i operacje



W DevOps AI sprawdza się przede wszystkim tam, gdzie trzeba szybko przetworzyć dużą ilość informacji: przygotować opis dostarczenia, podsumować incydent, wesprzeć postmortem lub przeanalizować dane obserwowalności. Największa wartość wynika dziś z ograniczenia powtarzalnej analizy i szybszego dotarcia do istotnych informacji, a nie z pełnej autonomii.

Najlepszy punkt startowy jest zależny od potrzeb organizacji. Firma produktowa ze słabym odkrywaniem wymagań może zacząć od wsparcia rozpoznawania potrzeb użytkowników. Instytucje finansowe od wsparcia compliance. Duża organizacja inżynierska od wsparcia testowania, dokumentacji lub przeglądów kodu.

Wniosek

Przypadki użycia należy dobierać według ograniczeń w procesie dostarczania, nie według aktualnych trendów.

7. Testowanie jako praktyczny punkt wejścia

Testowanie zasługuje na osobną uwagę. Ma powtarzalny charakter, jasne kryteria weryfikacji i mierzalne wyniki. Wpływa też bezpośrednio na przepływ dostarczania - gdy testowanie jest wolne lub niestabilne, wzrost szybkości kodowania niewiele pomaga.

AI może pomagać zespołom QA w:

- generowaniu przypadków testowych;
- tworzeniu danych testowych;
- przygotowywaniu skryptów automatyzacji;
- analizowaniu nieudanych testów;
- utrzymywaniu zestawów regresji;
- identyfikowaniu luk w pokryciu kodu;
- wspieraniu samonaprawiania się automatyzacji testów.

Publiczne materiały Sii oparte o wewnętrzne eksperymenty i rozwiązania związane z testowaniem wskazują właśnie te kierunki. Skupiamy się na automatycznym tworzeniu testów i jakości testów wspieranych przez AI - nie wyłącznie na szybkości. Wygenerowany test musi skutecznie weryfikować zachowanie systemu. Szybki test sprawdzający niewłaściwe zachowanie wprowadza szum, nie wartość.

- Badanie Sii Testing Lab – sprawdzamy „AI Boom" w automatyzacji testów
- Od automatyzacji do autonomii: Ewolucja w kierunku Agentic Testing na przykładzie mabl
- AI w pracy Test Developera – co naprawdę zmieniają Codex, Claude Design, Claude Opus 4.7, Codex Security i Claude Mythos
- Leveraging AI and Playwright for Efficient UI & API Testing



AI nie eliminuje testerów, ale zmienia ich pracę. Testerzy spędzają mniej czasu na powtarzalnych zadaniach, a skupiają się bardziej na ryzyku, pokryciu kodu, logice walidacji i przeglądzie.

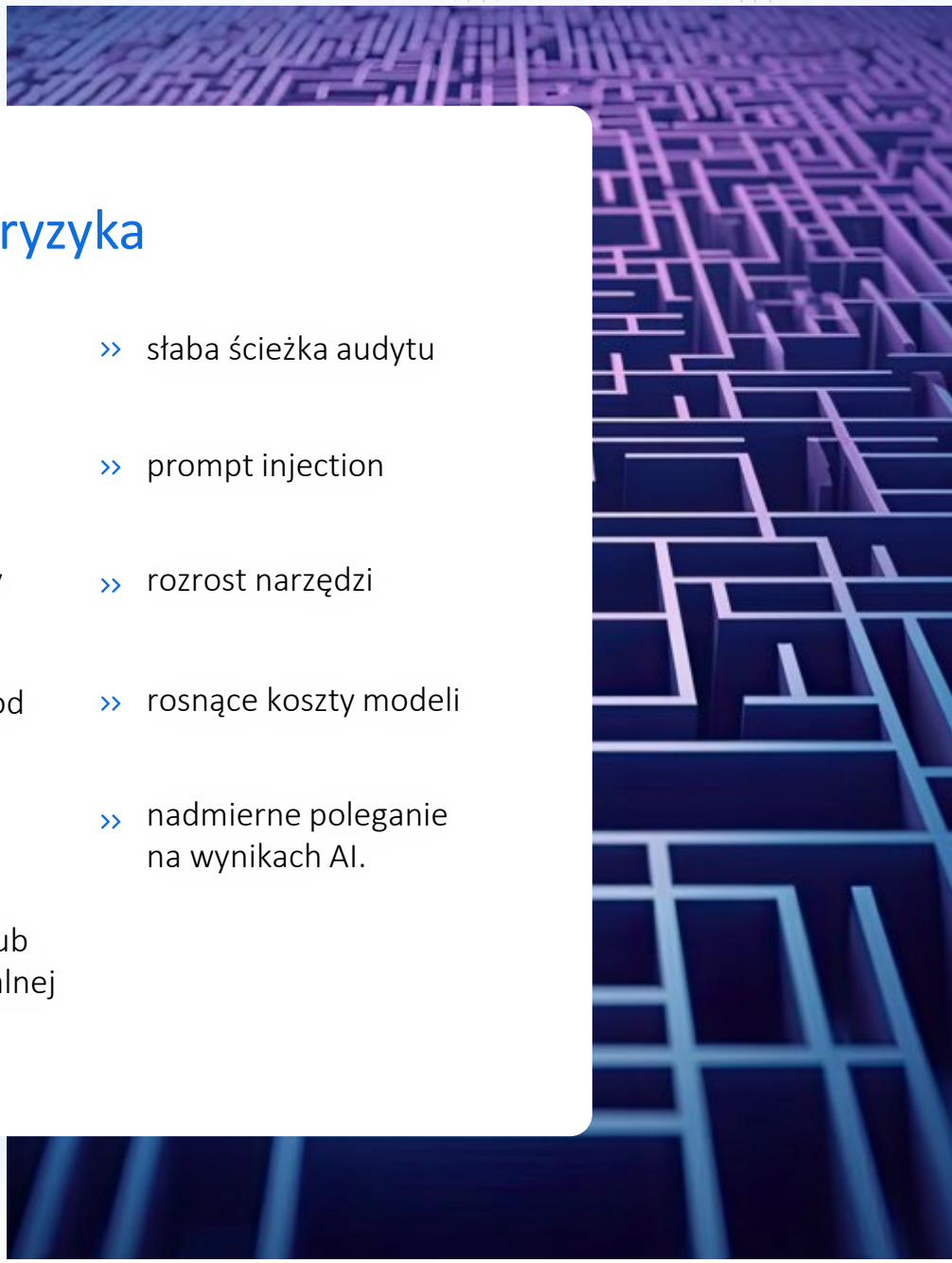
Testowanie może być dobrym pierwszym obszarem skalowania tam, gdzie stanowi istotne ograniczenie przepływu. Jego zaletą są powtarzalne zadania, stosunkowo jasne kryteria weryfikacji i możliwość pomiaru rezultatów.

Wniosek

Testowanie wspierane przez AI działa najlepiej, gdy testerzy pozostają odpowiedzialni za logikę jakości.

8. Zarządzanie i nadzór sprawowany przez ludzi

Oprogramowanie wytwarzane z pomocą AI nie może skalować się wyłącznie na zaufaniu. Ryzyko rośnie wraz z autonomią – sugestia kodu to jedno, agent zmieniający pliki, wywołujący narzędzia czy tworzący pull requesty to zupełnie co innego. Może wpływać na dostarczane artefakty, a pośrednio i na systemy produkcyjne.



Najważniejsze ryzyka

- ukryte zależności
- niepoprawny kod
- niespójne standardy
- niezabezpieczony kod
- wyciek danych
- naruszenie licencji lub własności intelektualnej
- słaba ścieżka audytu
- prompt injection
- rozrost narzędzi
- rosnące koszty modeli
- nadmierne poleganie na wynikach AI.

Użyteczny model zarządzania odpowiada na konkretne pytania:

- Które narzędzia są zatwierdzone?
- Jakich danych można używać?
- Jakie przypadki użycia są dozwolone?
- Które zadania wymagają zatwierdzenia przez człowieka?
- Które wyniki wymagają automatycznych kontroli?
- Jak rejestrowane są zmiany wspierane przez AI?
- Kto odpowiada za ryzyko?



Nadzór przez człowieka powinien być dostosowany do charakteru pracy.

Praca powtarzalna i niskiego ryzyka może funkcjonować w modelu human-on-the-loop. Praca średniego ryzyka wymaga punktów kontrolnych. Praca wysokiego ryzyka wymaga zatwierdzenia przez człowieka (human-in-the-loop).

Systemy krytyczne wymagają ścisłych zasad, standardów i kontroli.

Ma to szczególne znaczenie w środowiskach regulowanych, systemach ewidencji, logice finansowej, logice autoryzacji, przetwarzaniu danych wrażliwych i kluczowej własności intelektualnej.

Wniosek

Im większą autonomię otrzymuje sztuczna inteligencja, tym bardziej przejrzyste muszą być mechanizmy kontroli.

9. Governance dla Sii AI-native SDLC

Polityka ma wartość wtedy, gdy przekłada się na decyzję, narzędzia lub człowieka. Każde istotne ryzyko ma przypisaną kontrolę i właściciela — już w fazie pilotażu.

Trzy grupy mechanizmów zamieniają zasady w decyzje.

1

Dane i dostęp

Zatwierdzony punkt dostępu do modelu, ograniczony zakres kontekstu, zakaz przekazywania sekretów. Agent pracuje na tożsamości z minimalnymi uprawnieniami.

2

Zatwierdzanie i kontrola wyniku

Działanie wymagające zgody ma wskazanego człowieka, zakres akceptacji i zapis decyzji przed wykonaniem. Wynik sprawdzają testy, skanery i niezależne kryteria akceptacji.

3

Zatrzymanie i odtworzenie przebiegu

Limity prób, czasu i kosztu przerywają pracę. Zapis przebiegu obejmuje wersję modelu i konfiguracji, źródła, wywołania narzędzi, wyniki kontroli i decyzje — bez sekretów w logach.

Odpowiedzialność powinna być przypisana imiennie: Delivery Manager odpowiada za usługę i rollout, Solution Architect za stronę techniczną, a eksperci domenowi, QA i bezpieczeństwo dostarczają kryteria oraz ocenę. Agent nie przejmuje odpowiedzialności za zaakceptowany rezultat.

Wniosek

Zabezpieczeniem nie jest instrukcja w prompcie, lecz uprawnienia i proces akceptacji.

10. Bezpieczeństwo w pracy z agentami

Agent, który czyta pliki, wywołuje narzędzia i tworzy pull requesty, rozszerza powierzchnię ataku systemu dostarczania. Filtr odpowiedzi modelu nie jest wystarczającą kontrolą.

Jednym z istotnych ryzyk praktycznych jest prompt injection: dokument, zgłoszenie lub plik repozytorium może zawierać instrukcję próbującą przekierować działanie agenta.



Gdzie powstaje ryzyko

- prompt injection
- wyciek sekretów
- nadmiar uprawnień
- nieznane integracje
- dane poza firmą
- dostęp do produkcji
- zatrucie kontekstu
- podatne zależności
- brak śladu audytu
- sekrety w logach
- brak oceny po zmianie modelu

Cztery mechanizmy, które ograniczają skutki błędu agenta.

Uprawnienia: agent działa na tożsamości z minimalnymi uprawnieniami; jawnie dopuszczone polecenia, repozytoria i integracje.

Izolacja: odseparowane środowisko, bez dostępu do produkcji i sekretów. Git, CI i uprawnienia trzyma harness.

Integracje: API i MCP wymagają osobnej kontroli uprawnień. Samo połączenie nie nadaje prawa do działania.

Dane: zatwierdzony punkt dostępu do modelu, ograniczony zakres kontekstu, zakaz sekretów. Routing dopuszcza modele lokalne.

Praktyki, które utrzymują te mechanizmy skutecznymi:

- Treść pobierana przez agenta to dane o określonym poziomie zaufania.
- Ślad audytowy obejmuje przebieg i decyzje, ale nigdy sekretów.
- Po zmianie modelu, instrukcji lub integracji powtórz zestaw ocen.
- Brak zgody na przetwarzanie danych zatrzymuje pracę na danych klienta.
- Powrót do nadzorowanego trybu pracy musi być możliwy w każdej chwili.

Bezpieczeństwo agentowe zależy przede wszystkim od uprawnień, izolacji, konfiguracji i mechanizmów egzekwowania zasad. Prompt może wspierać poprawne zachowanie, ale nie powinien być podstawowym mechanizmem bezpieczeństwa.

Wniosek

Filtr odpowiedzi modelu nie wystarcza — liczą się uprawnienia, izolacja i akceptacje.

11. Kontekst, kontrola i specyfikacja

Wraz ze wzrostem autonomii agentów rośnie znaczenie wiarygodnego kontekstu, deterministycznych kontroli i jasnych kryteriów akceptacji.

Trzy praktyki zyskują na znaczeniu.

1

Kontekst dla agentów AI (context)

Agenci potrzebują decyzji architektonicznych, standardów kodowania, reguł domeny, polityk bezpieczeństwa i instrukcji specyficznych dla repozytorium. Bez kontekstu mogą generować kod działający w izolacji, który nie pasuje do systemu.

2

Kontrole i guardrails (checks)

Artefakty wygenerowane przez AI powinny przechodzić deterministyczne kontrole: testy, analizę statyczną, skanowanie bezpieczeństwa, sprawdzanie zależności, walidację polityk i kontrole audytowe.

3

Specyfikacje (specification-driven development

W miarę jak AI przejmuje więcej zadań wykonawczych, wymagania i kryteria akceptacji muszą być bardziej precyzyjne. Jeśli organizacja nie potrafi opisać i zweryfikować pracy, agent nie jest w stanie jej niezawodnie zrealizować.

Dokumentacja staje się w ten sposób infrastrukturą. Decyzje architektoniczne, standardy i kryteria akceptacji stają się zasobami używanymi zarówno przez ludzi, jak i narzędzia.

Zanim zwiększysz autonomię - popraw kontekst, kontrole i specyfikacje.

Wniosek

Jakość wyników AI zależy od jakości kontekstu, w którym AI działa.

12. Wewnętrzna platforma deweloperska jako podstawa skalowania

Oddolne eksperymentowanie z AI to dobry pierwszy krok, ale niewystarczający do osiągnięcia skali w organizacji. Bez wspólnych wzorców zespoły rozwiązują te same problemy osobno: dostęp do narzędzi, dobre praktyki, kontekst, reguły przeglądów, kontrole bezpieczeństwa, polityki danych i kontrola kosztów. To prowadzi do fragmentacji, którą platforma deweloperska może ograniczyć.

Wewnętrzna platforma deweloperska wspierana przez AI może zapewniać:

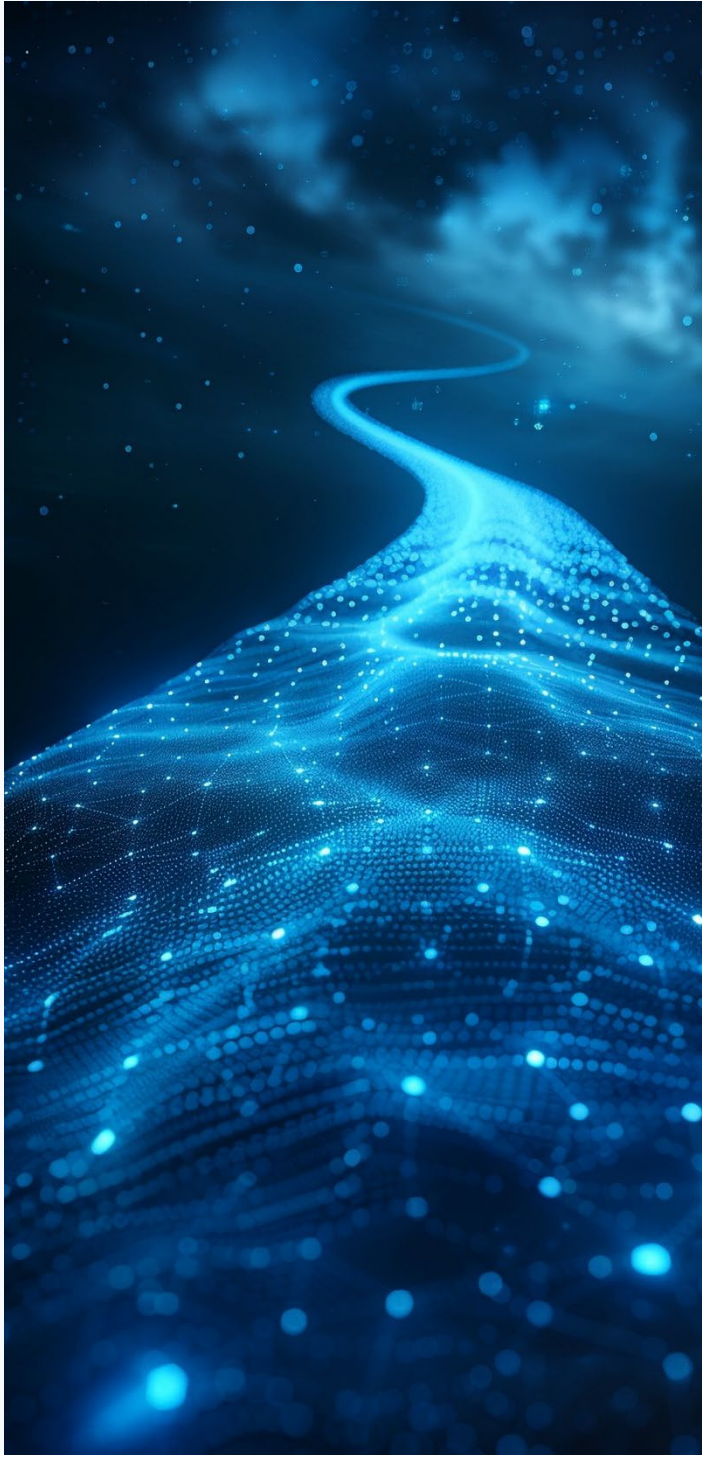
- zatwierdzone narzędzia i modele;
- wielokrotnie używalne przepływy pracy;
- wskazówki dotyczące repozytoriów;
- standardy kodowania i testowania;
- bezpieczne wartości domyślne;
- kontrole CI/CD;
- polityki jako kod (policy-as-code);
- logi audytowe;
- monitorowanie kosztów;
- integrację z portalem deweloperskim.



Cel jest prosty: sprawić, by bezpieczna ścieżka była jednocześnie ścieżką najłatwiejszą.

W przypadku agentów ma to jeszcze większe znaczenie. Agenci potrzebują kontrolowanego dostępu do repozytoriów, systemów śledzenia zgłoszeń, pipeline'ów i dokumentacji. Bez warstwy platformowej przepływy agentowe stają się trudne do monitorowania i zarządzania.

Zespoły platformowe mogą pakować przypadki użycia AI jako wielokrotnie używalne przepływy pracy, np.: klasyfikacja luk w zabezpieczeniach, generowanie testów, aktualizacje zależności, informacje o wydaniach czy aktualizacje dokumentacji.



Wniosek

Adopcja AI skaluje się, gdy zespoły otrzymują wielokrotnie używalne przepływy pracy - nie tylko licencje na narzędzia.

13. Plan działań dla liderów na 6–12 miesięcy

Liderzy nie muszą czekać na pełną autonomię. Potrzebują krótkiej i praktycznej agendy.

0–3 miesiące: Mapowanie i koncentracja

- Zmapuj, gdzie AI jest już teraz używana. Uwzględnij zatwierdzone narzędzia i nieuprawnione zastosowania (shadow AI).
- Zidentyfikuj rzeczywiste wąskie gardła. Sprawdź czas przeglądów kodu, opóźnienia w testowaniu, kolejki kontroli bezpieczeństwa, zatwierdzenia wydań i zakres poprawek.
- Wybierz kilka przypadków użycia, które są powtarzalne, weryfikowalne i mierzalne. Dobrzy kandydaci to: generowanie testów, objaśnianie kodu, dokumentacja, doprecyzowywanie wymagań, klasyfikacja luk, informacje o wydaniach.
- Ustal metryki bazowe przed skalowaniem.

3–6 miesięcy: Pilotaż i governance

- Wprowadź kontrolowane pilotaże. Na początku unikaj najbardziej ryzykownych przepływów pracy.
- Określ zatwierdzone narzędzia, zasady dotyczące danych, reguły przeglądów i sprawdzania wymagań.
- Twórz wielokrotnie używalne przepływy pracy. Nie pozwól, by każdy zespół wymyślał własne podejście.
- Mierz wyniki, a nie tylko wykorzystanie tokenów i narzędzi.

6–12 miesięcy: Skalowanie sprawdzonych rozwiązań

- Sprawdzone przypadki użycia skaluj w całej organizacji.
- Połącz przepływy AI z repozytoriami, CI/CD i bramkami jakości.
- Uruchom ograniczone przepływy agentowe tam, gdzie ryzyko jest niskie, a wyniki łatwe do weryfikacji.
- Popraw zarządzanie kontekstem, automatyczne kontrole i specyfikacje.

Pytania do zadania już teraz

- Gdzie AI jest już teraz używana w naszym SDLC?
- Które zastosowania są zatwierdzone, a które to shadow AI?
- Które wąskie gardła ograniczają dostarczanie?
- Które przypadki użycia AI skracają czas oczekiwania lub ograniczają poprawki?
- Które wyniki wymagają zatwierdzenia przez człowieka?
- Czy artefakty wygenerowane przez AI są weryfikowane?
- Czy adopcja AI jest wspierana przez wspólną platformę deweloperską?
- Czy mierzymy wpływ AI na dostarczanie?

Wniosek

Kolejny etap AI w SDLC powinien być praktyczny, kontrolowany i mierzony.



Podsumowanie:

Od eksperymentów do dostarczania opartego na AI

Zwycięzcami będą te firmy, które powiążą AI z pełnym przepływem dostarczania wartości, jakością, bezpieczeństwem i wynikami biznesowymi. Nie te, które jako pierwsze wdrożyły najnowsze narzędzia AI.

AI może ograniczać powtarzalną pracę. Może przyspieszać analizę, testowanie, dokumentację i naprawę błędów. Ale może też tworzyć nowe ryzyka: słabe wyniki, ukryte luki, nadmierne poleganie na AI i trudności z weryfikacją rezultatów, braki zgodności z regulacjami i wymaganiami.

Jednocześnie ekonomika pracy agentów pokazuje skalę potencjału. W eksperymencie Sii koszt dostarczonego ticketu wyniósł 6,11 USD (mediana: 2,96 USD). AI nadal wymaga kontroli i udziału człowieka, ale możliwość wykonywania coraz większej części pracy inżynierskiej przy tak niskim koszcie zmienia ekonomię wytwarzania oprogramowania.

Podejście Sii jest praktyczne: zaczynaj od rzeczywistych ograniczeń SDLC, wybieraj konkretne przypadki użycia, mierz wyniki, zachowuj ludzką odpowiedzialność i skaluj przez wielokrotnie używalne wzorce dostarczania.



Gotowi na kolejny krok?

Porozmawiajmy o AI w SDLC

Poznaj nasze podejście, ofertę i sposób pracy naszych zespołów.



Marcin Mosiołek
AI Competency Center Lead
mmosiolek@sii.pl



Łukasz Biegański
AI Strategist
lbieganski@sii.pl