Komisja Architektury i Urbanistyki
Oddziału Polskiej Akademii Nauk we Wrocławiu
wykład wygłoszony 24.04.2026 r. w siedzibie oddziału

Tytuł roboczy (PL): *Architekt jako orkiestrator. Programowanie agentowe w projektowaniu parametrycznym i kształtowaniu przestrzeni 3D*

Tytuł roboczy (EN): *The Architect as Orchestrator. Agent-Based Programming in Parametric Design and 3D Space Shaping*

Autor: mgr inż. arch. Przemysław Nowak, Politechnika Wrocławska

Abstract

This paper describes a qualitative shift in the toolchain of computational design, brought about by the widespread availability of large language models (LLMs) capable of code generation and by agent-based frameworks enabling the orchestration of multiple specialised model instances. The emerging paradigm is described as a three-layer stack: the language model as interpreter of design intent, parametric code as an intermediate deterministic representation, and agentic orchestration as the coordination layer. In this configuration, the role of the architect shifts from operator of software to orchestrator of a pipeline, while the core of professional competence remains architectural. The feasibility of the method is demonstrated by four original applications, each illustrating a different layer of the agent pipeline: Stairgen with deterministic normative validation, Archidesign with domain-knowledge grounding in a corpus of historical treatises, GenCAD Live with a minimal three-agent pipeline and multimodal control, and Handcad with an embodied gesture-tracking interface. The paper also discusses four structural limitations of the method (hallucinations, lack of deterministic reproducibility, non-transferable legal responsibility of the designer, and dependence on the model provider) and four educational directions for architectural faculties (prompt engineering, pipeline literacy, critical evaluation of model output, and legal awareness). The agentic stack constitutes a qualitatively new layer of the design toolchain. Its integration into Polish architectural education requires programmatic reflection undertaken early, to reduce the gap between academia and practice.

Keywords: agent-based programming, parametric design, large language models, architectural education, deterministic validation, multimodal interfaces, Polish architectural discourse.

1. Wprowadzenie

Parametryzm jako paradygmat projektowy zdominował dyskurs architektoniczny dwóch ostatnich dekad (Schumacher, 2008, 2011). Jego pełna realizacja techniczna pozostała jednak zarezerwowana dla wąskiego kręgu specjalistów, projektantów posługujących się środowiskami parametrycznymi (Grasshopper, Dynamo, Generative Components) albo programowaniem w językach Python, C# lub RhinoScript. Bariera kompetencyjna, oddzielająca architekta od narzędzia parametrycznego, stała się przedmiotem krytyki zarówno z perspektywy historyczno-kulturowej (Picon, 2010), jak i metodologicznej (Carpo, 2017).

Sytuacja zmieniła się jakościowo wraz z upowszechnieniem dużych modeli językowych (Large Language Models, LLM) zdolnych do generowania kodu, od Codex (2021), przez GPT-4 (2023), po Claude Opus 4.x (2025) i Gemini 3.x (2025). Drugą warstwą zmiany jest programowanie agentowe, sposób organizowania pracy, w którym wiele wyspecjalizowanych instancji modelu współpracuje mając odrębne role. Fundamenty tego podejścia stanowią prace nad pętlą rozumowania i działania (ReAct: Yao et al., 2023), samokrytyką agenta (Reflexion: Shinn et al., 2023) oraz elementami orkiestracji (AutoGen: Wu et al., 2023).

Tekst niniejszy ma charakter raportu identyfikującego pole do prac badawczych i dysputy naukowej: diagnozuje zmianę narzędziową, definiuje pojęcia robocze i prezentuje cztery wdrożone narzędzia własne autora jako ilustracje praktycznej wykonalności prezentowanej metody. Każde z nich rozwiązuje inny typ zadania projektowego, ale wszystkie korzystają z wariantów wspólnego podejścia: orkiestracji wielu agentów osadzonych w wiedzy dziedzinowej.

Strukturę raportu wyznacza diagnoza obecnego stanu, opis zmiany narzędziowej, refleksja nad nową rolą architekta, prezentacja czterech demonstracyjnych prototypów, konsekwencje dla edukacji architektonicznej, granice metody, perspektywa rozwojowa i wnioski. Wszystkie cztery omawiane narzędzia, Stairgen, Archidesign, GenCAD Live oraz Handcad, są publicznie dostępne z otwartym kodem źródłowym.

2. Diagnoza obecnego stanu

Parametryzm funkcjonuje w literaturze w dwóch znaczeniach, stylistycznym (Schumacher, 2008) i technicznym, przy czym dla niniejszego raportu istotne jest wyłącznie drugie. Projektowanie parametryczne jako technika operacyjna, czyli sposób opisywania geometrii poprzez relacje i parametry zamiast bezpośredniego rysunku, ugruntowało się w warsztacie

architektonicznym wraz z upowszechnieniem środowisk takich jak Grasshopper (od 2007) czy Dynamo (od 2011). Mimo dostępności tych narzędzi, ich operacyjne opanowanie pozostało zadaniem trudnym. Wymaga ono kompetencji hybrydowych, łączących myślenie architektoniczne z programowaniem w sensie algorytmicznym, a często także ze znajomością geometrii różniczkowej (Pottmann et al., 2007) lub składni języków takich jak Python czy C#.

Skala wymagań jest dobrze udokumentowana w literaturze warsztatowej. Tedeschi (2014) w referencyjnym podręczniku *AAD: Algorithms-Aided Design* przedstawia kilkaset stron technik, których opanowanie wymaga znaczącego wysiłku poznawczego i czasu. Rezultatem tej bariery jest rozwarstwienie środowiska zawodowego: z jednej strony elitarne pracownie i ośrodki akademickie operujące pełnym warsztatem komputacyjnym, z drugiej, zdecydowanie liczniejsza, codzienna praktyka projektowa pozostająca w paradygmacie Revit, ArchiCAD lub AutoCAD, bez warstwy generatywnej.

Polskie środowisko akademickie podejmowało problem projektowania parametrycznego od dawna. Słyk (2012) w monografii *Źródła architektury informacyjnej* postawił tezę, że metodą architektury informacyjnej nie jest projektowanie rozwiązań, lecz programowanie procesów. Środowisko Wydziału Architektury Politechniki Białostockiej zajmowało się cyfrowymi mediami w architekturze od lat dziewięćdziesiątych (Asanowicz i Jakimowicz, 1998) i kontynuuje ten kierunek w nowej formule konferencji *Digital Architecture Research* (Jakimowicz i Śliwecki, 2023). Mimo tego dorobku, transfer wniosków z badań akademickich do codziennej praktyki projektowej pozostaje w wąskiej niszy oddolnych działań.

Punkt zwrotny przyniosło upowszechnienie LLM zdolnych do generowania kodu. Modele te likwidują kluczowy element bariery opisanej w poprzedniej części tekstu, czyli wymóg programistyczny. Architekt opisujący swoje intencje w języku naturalnym otrzymuje skrypt parametryczny gotowy do uruchomienia w środowisku takim jak Blender, Rhino czy Grasshopper. Jeszcze ważniejsze jest to, że kolejna warstwa, programowanie agentowe, pozwala dekomponować zadanie projektowe na wyspecjalizowane role i delegować je odrębnym instancjom modelu. W rezultacie architekt nie pisze już kodu, ale opisuje intencję, ramy operacyjne i kryteria walidacji. Bariera, która oddzielała architekta od projektowania komputacyjnego, była w istocie barierą kompilacji intencji w kod. Modele językowe usuwają tę barierę, przejmując pracę tłumaczenia intencji na kod.

3. Zmiana stosu narzędziowego

W niniejszym raporcie nowy paradygmat narzędziowy opisywany jest jako trójwarstwowy stos. Pierwszą warstwę tworzy LLM, pełniący funkcję interpretera intencji projektanta, generatora

kodu i krytyka wyniku. Drugą warstwę stanowi kod parametryczny, czyli pośrednia reprezentacja deterministyczna geometrii i procesu projektowego, należąca do tradycji cyfrowego projektowania, której meta-teorię proponują Oxman i Oxman (2014). Trzecią warstwą jest orkiestracja agentowa, czyli koordynacja wielu wyspecjalizowanych instancji modelu, z których każda ma własną rolę, kontekst pracy i zakres odpowiedzialności. Te trzy warstwy współpracują szeregowo i zwrotnie. Programowanie agentowe jest dla projektowania komputacyjnego tym, czym BIM był dla projektowania budowlanego: nową warstwą metodologiczną, która zmienia, kto może wykonać pracę i przy jakim koszcie poznawczym.

Stan badań nad zastosowaniami programowania agentowego w branży architektury, inżynierii i budownictwa (Architecture, Engineering, Construction; AEC) potwierdza, że opisywany w niniejszym raporcie paradygmat nie jest spekulatywny. W *Journal of Computing in Civil Engineering* opublikowany został framework Text2BIM (Du et al., 2026), w którym kilku wyspecjalizowanych agentów LLM współpracuje przy generowaniu modeli BIM z opisu w języku naturalnym, a wyniki ich pracy są walidowane przez moduł sprawdzający reguły. W *Automation in Construction* zaprezentowany został BIM-GPT (Zheng i Fischer, 2023), system łączący GPT z dynamicznym promptem do pobierania informacji z baz BIM przez naturalne zapytania. Forth i Borrmann (2024) wykazali w *Journal of Building Engineering*, że model językowy dotrenowany na zbiorze tekstów z dziedziny budownictwa pozwala automatycznie wzbogacać semantycznie modele BIM dla potrzeb symulacji energetycznych. Yang i Zhang (2024) zademonstrowali w *Automation in Construction*, że inżynieria promptowa (ang. prompt engineering) nad GPT umożliwia automatyczne tłumaczenie postanowień norm budowlanych na język logiki, co stanowi krok w stronę automatycznego sprawdzania zgodności projektów.

4. Nowa rola architekta: orkiestrator

Architekt pracujący ze stosem agentowym nie staje się przez to programistą ani autorem promptów. Pozostaje orkiestratorem przestrzeni, którego intencja kształtuje całość projektu. Zmienia się medium tej pracy: narzędzia, dotąd wymagające znacznego wysiłku poznawczego, stają się dla projektanta niewidoczne.

Tak rozumiana orkiestracja wymaga, w ujęciu niniejszego raportu, pięciu kompetencji warsztatowych: dekompozycji semantycznej problemu projektowego, *role-promptingu* (formułowania precyzyjnych instrukcji systemowych dla każdego agenta), osadzenia w wiedzy dziedzinowej, projektowania warstwy walidacyjnej oraz krytycznej ewaluacji wyniku. Każdą z nich ilustrują cztery wdrożone narzędzia autora, omawiane w dalszej części raportu.

Pięć kompetencji wymienionych powyżej, w ujęciu autora raportu, stanowi rozwinięcie

warsztatu praktykowanego na wydziałach architektury od pokoleń, nie zaś nowy katalog wymagań. Dekompozycja semantyczna problemu, podział zadań między specjalistów zespołu projektowego, lektura traktatów i norm, weryfikacja zgodności rysunku z przepisami oraz krytyczna ocena wyniku w dyskusji to operacje obecne w klasycznej dydaktyce projektowej: na korektach studenckich i w referencyjnych podręcznikach metody projektowej. Zmienia się medium, w którym są wykonywane, nie ich istota. Architekt operujący stosem agentowym wykonuje znaną sobie pracę warsztatową, ale w środowisku, w którym przekład intencji w narzędzia nie wymaga już ręcznej kompilacji. Zdjęcie tego kosztu poznawczego pozwala skupić uwagę architekta na tym, co stanowi rzeczywisty rdzeń jego zawodu: na decyzji przestrzennej i kompozycyjnej.

Tak rozumiana rola architekta wymaga umiejscowienia wobec wcześniejszych diagnoz roli sztucznej inteligencji w architekturze. Punktem odniesienia jest tom Leacha (2022) *Architecture in the Age of Artificial Intelligence*. Leach zarysował w nim wizję systemów opartych na sztucznej inteligencji, mających w niedalekiej przyszłości zaprojektować za nas budynki i miasta. W przedmowie zapowiedział też tom drugi, *The Death of the Architect*, jako dopełnienie tej wizji od strony jej zagrożeń. Argumentacja niniejszego raportu odnosi się do diagnozy Leacha krytycznie, ale na innym poziomie. Tom z 2022 r. opisuje pierwszą falę zastosowań sztucznej inteligencji w architekturze, w której narzędziem były generatywne sieci neuronowe wytwarzające obrazy formy (GAN, modele dyfuzyjne typu Midjourney lub Stable Diffusion). Stos agentowy opisywany w niniejszym raporcie działa inaczej: wyspecjalizowane instancje modelu językowego wytwarzają i walidują kod parametryczny, a kontrola nad procesem pozostaje po stronie architekta-orkiestratora. W tym ujęciu autonomiczna AI, projektująca samodzielnie zamiast architekta, nie jest dla projektowania komputacyjnego ani konieczna, ani potrzebna. Architekt nie zostaje wyparty: zostaje wzmocniony.

Praktyczną konsekwencją powyższego ujęcia jest dekompozycja zadania projektowego na role. Próba zlecenia pojedynczemu modelowi językowemu złożonego polecenia obejmującego naraz badanie literatury, przekształcenie danych geometrycznych, weryfikację składni biblioteki programistycznej, napisanie kodu i połączenie komponentów prowadzi w praktyce do błędów kumulujących się w kolejnych iteracjach. Powtarzanie tego samego promptu z prośbą o naprawę błędu nie usuwa źródła problemu, bo polecenie pozostaje przeciążone. Rozwiązaniem jest rozdzielenie odpowiedzialności pomiędzy wyspecjalizowane instancje modelu, z których każda otrzymuje jedno wąsko zdefiniowane zadanie: badacz przeszukuje literaturę, matematyk formalizuje zależności, programista zapisuje je w kodzie, walidator sprawdza wynik. Kontrast tych dwóch sytuacji przedstawia figura 1. Tak działająca koordynacja jest odzwierciedleniem opisywanego w poprzednich sekcjach podejścia. Cztery

aplikacje omówione w dalszej części raportu są wariantami tej samej zasady, dostosowanymi do różnych klas zadań projektowych.

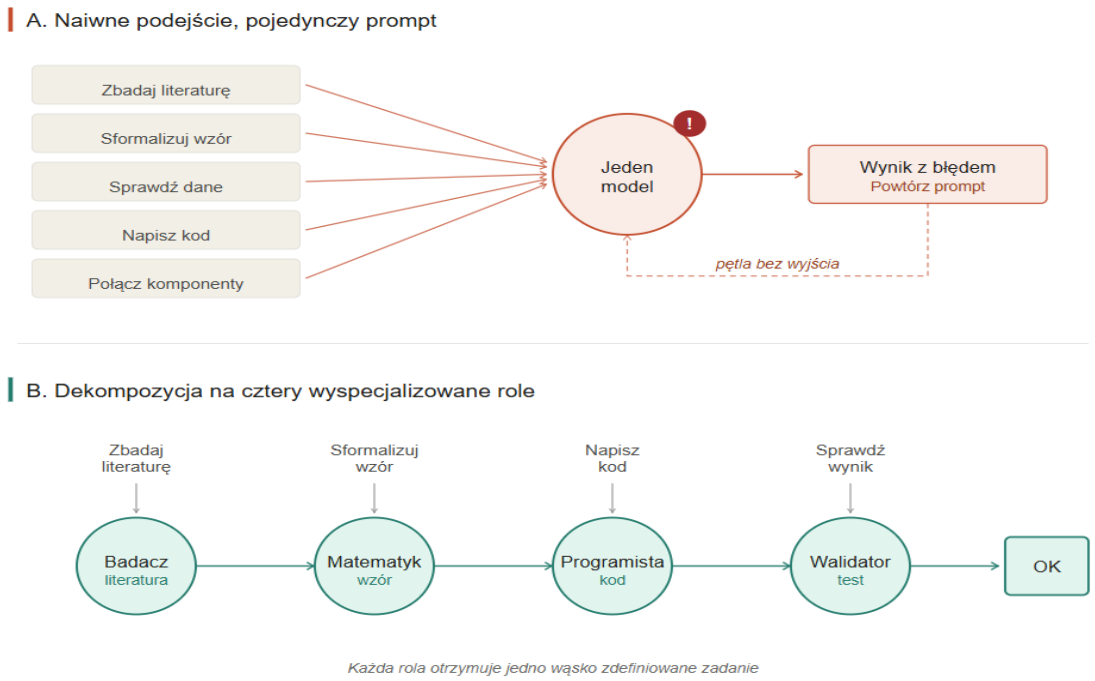

Figura 1.

A: pojedynczy model językowy obciążony pięcioma zadaniami zwraca wynik z błędem, a powtarzanie tego samego polecenia tworzy pętlę bez wyjścia. B: te same zadania rozdzielone pomiędzy cztery wyspecjalizowane role agentowe (badacza, matematyka, programistę i walidatora) pracujące sekwencyjnie z domknięciem walidacyjnym.

5. Demonstracje wykonalności na przykładach autorskich prototypów

Cztery aplikacje przedstawione poniżej dobrane zostały tak, by każda zilustrowała inną warstwę pipeline'u agentowego – procesu zautomatyzowanego wykonywania zadań opartego na wzajemnej komunikacji i transferze danych. Ich kolejność odzwierciedla wzrastającą złożoność problemu. Stairgen ilustruje warstwę walidacji deterministycznej. „Archidesign” demonstruje osadzenie wiedzy dziedzinowej, w którym agenty pracują nad korpusem traktatów. „GenCAD” realizuje połączenie trzech wyspecjalizowanych agentów ze sterowaniem multimodalnym, łączącym wejście głosowe i tekstowe. „Handcad” sięga poza warstwę agentową, w stronę interfejsu między architektem a maszyną. Wszystkie cztery są autorskimi narzędziami z otwartym kodem źródłowym.

Stairgen jest parametrycznym konfiguratorem schodów kręconych. Aplikacja przyjmuje na wejściu około 65 parametrów geometrycznych: od wysokości całkowitej i kąta obrotu marszu (do 720°), przez kierunek wznoszenia, geometrię słupa nośnego i balustrady, po szczegóły profilu stopnia. Wprowadzanie danych odbywa się poprzez regulację suwaków widocznych na fig. 2. Na wyjściu zwraca trójwymiarowy model schodów w otwartym formacie glTF, w którym wszystkie parametry konfiguracji są zachowane. Pozwala to wielokrotnie wracać do projektu i go modyfikować bez utraty wcześniejszych ustawień. W tle pracuje moduł walidacji deterministycznej, sprawdzający zgodność konfiguracji z polskimi warunkami technicznymi (Rozporządzenie Ministra Infrastruktury z 12 kwietnia 2002 r., tekst jednolity Dz.U. 2022 poz. 1225). Walidator stosuje progi z § 68 (wymiary stopni dla różnych typów zabudowy: mieszkaniowej, użyteczności publicznej, pomocniczej) oraz wzór Blondela z § 69 ust. 4 ($2h+s \in [60, 65]$ cm, gdzie h to wysokość stopnia, s jego szerokość). W obecnej wersji Stairgen nie zawiera komponentu modelu językowego. Jego rolą w argumentacji raportu jest zilustrowanie warstwy walidacji jako gotowego, deterministycznego filtra zgodności normatywnej, który w pełnym stosie agentowym wychwytywałby halucynacje modelu.

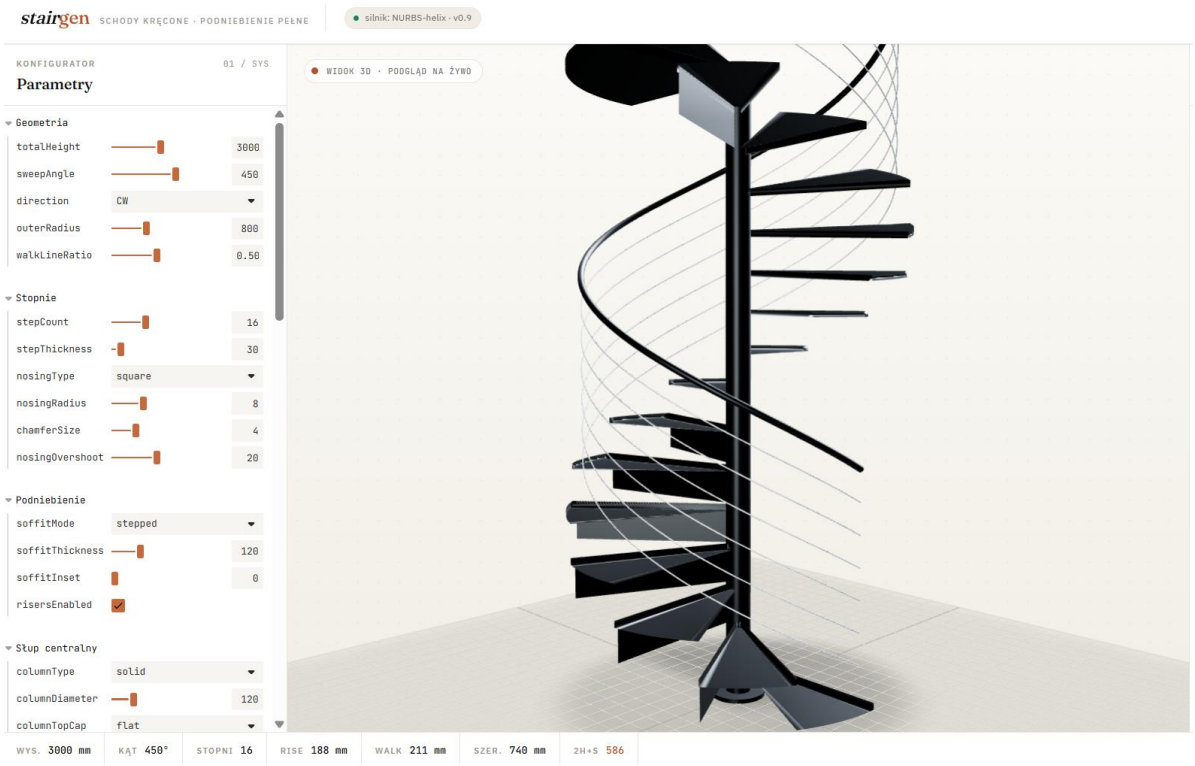

Figura 2.

Interfejs programu Stairgen z widocznymi suwakami oraz parametrycznie wygenerowanym modelem.

Idea automatycznego sprawdzania zgodności projektu z normami budowlanymi jest rozwijana od dawna (Eastman et al., 2009), a najnowsza fala pracy nad zastosowaniem modeli językowych do tego zadania dopiero powstaje (Yang i Zhang, 2024). Walidator nie zastępuje architekta, towarzyszy mu, sygnalizuje naruszenia normatywów i sugeruje korekty, pozostawiając ostateczną decyzję człowiekowi, który dokumentację podpisuje.

Archidesign jest hybrydowym systemem generującym sparametryzowane modele 3D budynków w stylu klasycystycznym. Pipeline agentowy obsługuje w nim wiedzę dziedzinową: dwa wyspecjalizowane agenty (Bibliotekarz przekształcający skany historycznych traktatów w ustrukturyzowany plik tekstowy oraz Architekt ekstrahujący z niego reguły proporcji kompozycji) zasilają bazę wiedzy stanowiącą nadrzędne źródło wszystkich parametrów architektonicznych. Sam model trójwymiarowy budowany jest z tej bazy obliczeniowo, w sposób deterministyczny: warstwa agentowa odpowiada za reguły, warstwa geometryczna za ich algorytmiczne wykonanie. Korpus traktatów obejmuje cztery kanoniczne pozycje: Witruwiusza (*De architectura libri decem*), Palladia (*I Quattro Libri dell'Architettura*), Vignolę (*Regola delli cinque ordini d'architettura*) i Serlia (*Sette libri dell'architettura*). Specyfikacja systemu nakazuje, by każda wartość liczbowa miała uzasadnienie w traktatach historycznych: parametry wpisywane na stałe w kodzie nie są dopuszczalne. Archidesign wpisuje się w linię badań nad formalizacją reguł palladiańskich, którą otworzył Wittkower (1949) analizą harmonii proporcji, sformalizowali Stiny i Mitchell (1978) gramatyką kształtów, a kontynuowali Spallone i Calvano (2022) parametrycznymi eksperymentami nad willami Palladia. Różnica wprowadzona w niniejszej aplikacji polega na zastosowaniu agentów LLM do pozyskiwania wiedzy z traktatów: zamiast ręcznej kodyfikacji reguł przez badacza, korpus wiedzy budowany jest przez wyspecjalizowane modele językowe pracujące nad oryginalnymi tekstami źródłowymi.

GenCAD Live jest interaktywnym systemem CAD generującym i modyfikującym modele trójwymiarowe w czasie rzeczywistym na podstawie poleceń w języku polskim. Wejście aplikacji jest multimodalne: użytkownik może wpisać polecenie z klawiatury lub wypowiedzieć je głosowo, a syntezator mowy zwraca głosową odpowiedź potwierdzającą. Sercem aplikacji jest pipeline trzech wyspecjalizowanych agentów opartych na modelu Gemini, przekazujących sobie ustrukturyzowane wiadomości (przykład widoczny na fig. 3).

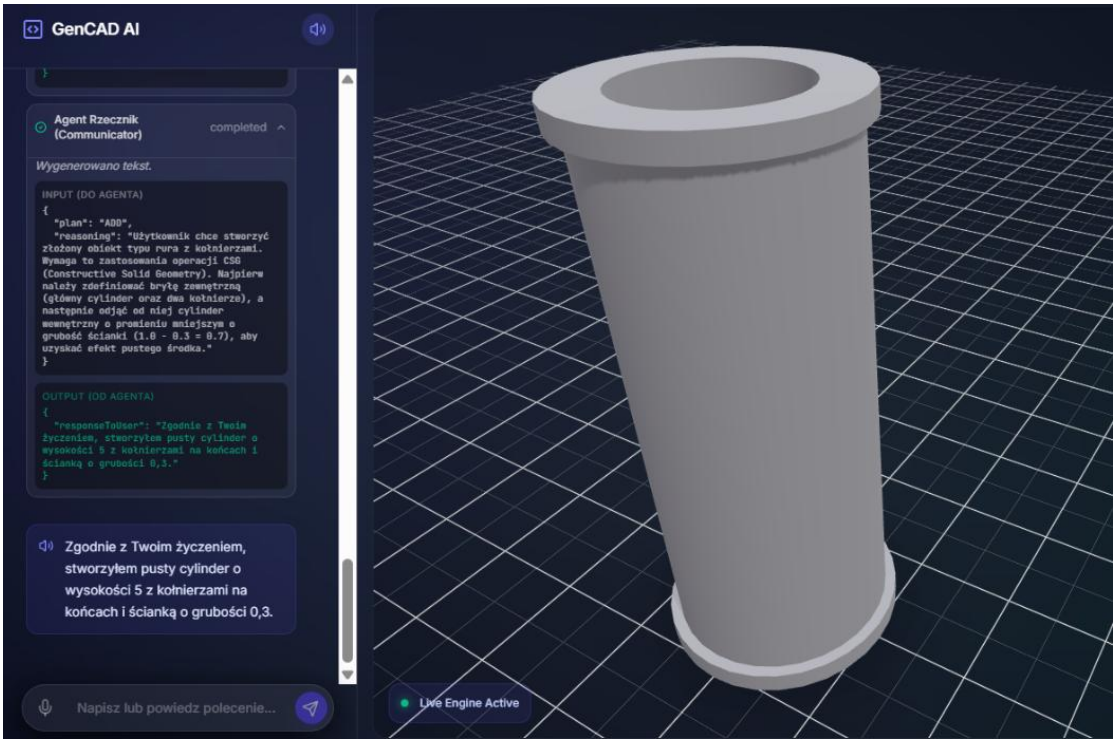

Figura 3.

Interfejs programu GenCAD Live z uwidocznionym procesem rozumowania oraz wygenerowaną geometrią.

Pierwszy agent (Analityk) analizuje semantykę polecenia i dedukuje przestrzenne skutki, zwracając plan kolejnych operacji modelowania. Drugi agent (Konstruktor) wykonuje kalkulacje matematyczne i transformacje macierzowe, zwracając nową tablicę stanów obiektów z wektorami pozycji, rotacji i skali. Trzeci agent (Rzecznik) generuje zwięzły komunikat tekstowy potwierdzający wykonane akcje, dostosowany do syntezatora mowy. Tak skonstruowana architektura odpowiada wzorcom *orchestrator-workers* i *prompt chaining* w terminologii opracowania inżynierskiego Anthropic (Schluntz i Zhang, 2024), a w szerszej perspektywie wpisuje się w nurt frameworków multi-agentowych dla modeli językowych (Wu et al., 2023). Aspekt multimodalny aplikacji odnosi się do długiej tradycji badań nad interfejsami łączącymi mowę i gest, której początek wyznacza praca Bolta (1980) z MIT. Rolą GenCAD Live w argumentacji raportu jest pokazanie, że minimalny pipeline trzech agentów wystarcza do realizacji operacyjnego CAD w czasie rzeczywistym.

Handcad jest interaktywnym systemem CAD do modelowania trójwymiarowego, w którym jedynym kanałem wejściowym jest optyczne śledzenie gestów dłoni w czasie rzeczywistym. Aplikacja przetwarza strumień obrazu z kamery w przeglądarce, mapując ruchy palców na

operacje na bryłach prostopadłościennych oraz kontrolę wirtualnej kamery (proces widoczny na fig. 4).

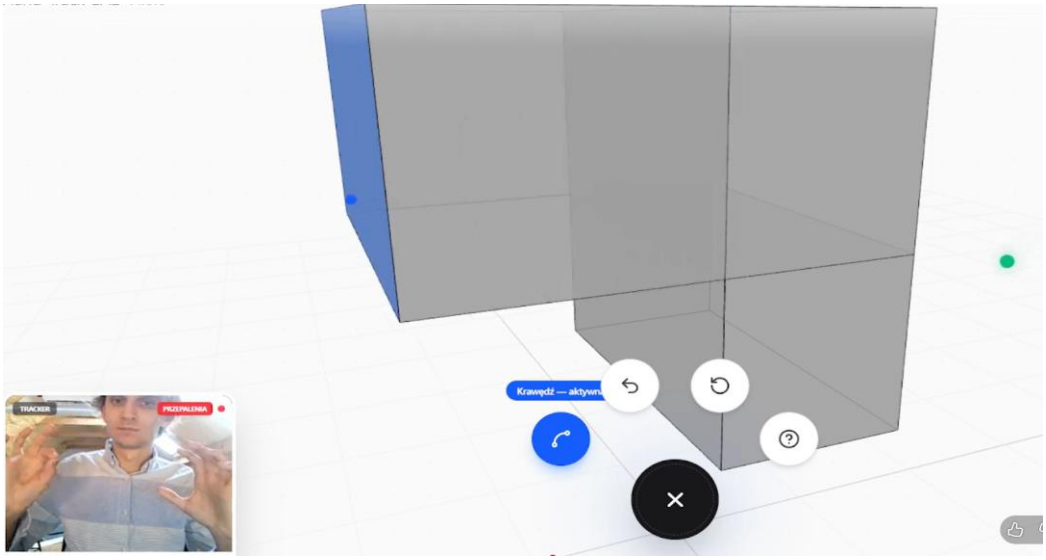

Figura 4.

Interfejs programu Handcad, w lewym dolnym rogu widoczny obraz z kamery, na którym autor dokonuje manipulacji bryłą w czasie rzeczywistym poprzez poruszanie rękami.

Rozpoznawanie pozy dłoni realizuje model HandLandmarker frameworku MediaPipe (Lugaresi et al., 2019; preprint arXiv). Szum wysokoczęstotliwościowy w sygnale śledzenia (powodujący drżenia i brak precyzji) redukuje filtr One Euro (Casiez et al., 2012), adaptacyjnie dostosowujący ruch kursora do prędkości ruchu dłoni: niska przy bezruchu (eliminacja drżenia kursora) i wysoka przy szybkim ruchu (minimalizacja opóźnienia). Mapowanie interfejsu obejmuje cztery operacje: lewa dłoń w zwarciu palców wytłacza wybraną ścianę bryły, prawa dłoń w zwarciu obraca scenę wokół orbity, obie dłonie w zwarciu sterują skalą widoku, a otwarta lewa dłoń wywołuje menu podręczne. Topologia edytowanego obiektu nie opiera się na klasycznych operacjach bryłowych, lecz na drzewie uproszczonym drzewie figur prostopadłościennych, co upraszcza obliczenia i pozwala na szybkie cofanie operacji. Handcad nie zawiera komponentu modelu językowego ani pipeline'u agentowego. Jego rolą w argumentacji raportu jest pokazanie nowych metod wprowadzania danych i kontroli urządzenia jako kierunku, w którym warstwa interakcji człowiek-maszyna może się rozwijać niezależnie od warstwy agentowej, by w przyszłości połączyć się z nią w pełną multimodalność, w której architekt nie pisze, lecz mówi i gestykuluje.

Cztery przedstawione aplikacje pokrywają różne warstwy stosu narzędziowego architekta-orkiestratora: walidację normatywną, osadzenie wiedzy dziedzinowej, koordynację wielu

agentów w czasie rzeczywistym oraz interfejs sterowania. Pytanie, jak architekta przygotować do pracy w tak ukształtowanym pejzażu, podjęte zostaje w sekcji następnej.

6. Konsekwencje dla dydaktyki architektonicznej

Konsekwencje dla dydaktyki architektonicznej są bezpośrednie. Stos agentowy wymaga od architekta kompetencji, których wydziały architektury nie obejmują systematycznie. Warto wskazać cztery konkretne kierunki.

Pierwszy to praca z promptem, rozumianym jako warsztatowe ćwiczenie precyzji języka. Sformułowanie instrukcji systemowej dla wyspecjalizowanego agenta nie różni się formalnie od pisania notatki do współpracownika w biurze: trzeba podać kontekst, zadanie, ograniczenia, kryterium sukcesu. Ta umiejętność daje się ćwiczyć od pierwszego roku studiów, bez żadnej infrastruktury technicznej.

Drugi to rozumienie budowy pipeline'u: czym jest agent, czym rola, jak działa łańcuch promptów, jaką funkcję pełni walidator. To nie jest wiedza programistyczna, to jest myślenie o przepływie zadań i odpowiedzialności w zespole. Z tym architekt ma do czynienia od zawsze, tylko w innym medium.

Trzeci, najtrudniejszy, to krytyczna ocena tego, co model zwraca. Halucynacja modelu wygląda tak samo jak prawda: odpowiedź jest spójna, gramatyczna, z pozoru udokumentowana. Rozpoznać błąd potrafi tylko ktoś, kto sprawdza wynik z normą, z literaturą, z własnym doświadczeniem. To jest najbliższe rdzeniowi zawodu architekta i jednocześnie najtrudniejsze do wyuczenia, bo wymaga już ukształtowanej intuicji projektowej.

Czwarty to świadomość prawna. Polskie prawo budowlane nakłada odpowiedzialność na osobę podpisującą dokumentację, niezależnie od narzędzi, którymi powstała. Architekt korzystający z modelu językowego pozostaje pełnym autorem projektu, ze wszystkimi tego konsekwencjami.

Polski dyskurs nad sztuczną inteligencją w architekturze istnieje. Politechnika Gdańska bada zastosowania AI w dydaktyce projektowej (Cudzik i Nyka, 2024), a Politechnika Warszawska zajmuje się łączeniem modeli językowych z BIM dla niskoemisyjnego projektowania (Płoszaj-Mazurek i Ryńska, 2024). W skali międzynarodowej powstał systematyczny przegląd zastosowań modeli generatywnych w edukacji architektonicznej (Alamasi i Asfour, 2026), wskazujący na brak badań nad długoterminowym wpływem tych narzędzi na kompetencje krytyczne studentów. Polska edukacja architektoniczna ma okazję, być może w niewielkim

oknie czasowym, włączyć te kompetencje do programów w sposób systematyczny. Im wcześniej zostanie podjęta refleksja programowa, tym mniejsza luka między uczelnią a praktyką.

7. Granice metody i ryzyka

Pipeline agentowy ma swoje ograniczenia. Cztery z nich, zidentyfikowane w toku prac nad opisanymi aplikacjami i potwierdzone w literaturze przedmiotu, zasługują na osobne omówienie. Świadomość granic narzędzia jest częścią odpowiedzialności projektanta, nie dodatkiem do niej.

Pierwszym ograniczeniem są halucynacje. Model językowy generuje wynik, który wygląda na prawdziwy niezależnie od tego, czy nim jest (Ji et al., 2023). Dla projektanta oznacza to ryzyko wymyślonych norm, fikcyjnych wzorów geometrycznych, nieistniejących bibliotek programistycznych. Stairgen pokazał jedno z rozwiązań: walidator deterministyczny, który wychwytuje takie błędy na wyjściu modelu.

Drugim ograniczeniem jest brak deterministycznej reprodukowalności. Tradycyjny warsztat projektowy, od rysunku odręcznego po skrypt parametryczny w Grasshopperze, opiera się na tym, że ten sam zestaw wejściowych decyzji daje zawsze ten sam wynik. Stos agentowy łamie tę zasadę: model językowy generuje tekst probabilistycznie, więc dwa wywołania z tym samym promptem mogą dać różne wyniki. Dla projektanta oznacza to, że proces, w którym powstała dokumentacja, jest nieodtwarzalny w pełni. Rejestrowanie promptów, użytych modeli i parametrów łagodzi problem, ale go nie usuwa.

Trzecim ograniczeniem jest to, że odpowiedzialności prawnej projektanta nie można przenieść na model. Art. 20 Prawa budowlanego (Ustawa z 7 lipca 1994 r., z późn. zm.) wymaga od projektanta opracowania projektu zgodnie z obowiązującymi przepisami i zasadami wiedzy technicznej oraz dołączenia oświadczenia o jego sporządzeniu. Użycie modelu językowego przy tworzeniu dokumentacji nie zwalnia projektanta z tych obowiązków. Każdy element dokumentacji wymaga jego weryfikacji przed podpisem.

Czwartym ograniczeniem jest zależność od dostawcy modelu. Komercyjni dostawcy LLM (OpenAI, Anthropic, Google) mogą wprowadzać zmiany cenników, polityki dostępu lub wycofywać modele z eksploatacji. Każda taka zmiana może podważyć stabilność procesu projektowego opartego na konkretnej wersji modelu. Częściowym rozwiązaniem są modele z otwartymi wagami (Llama, Mistral, DeepSeek), które można uruchomić lokalnie, choć kosztem niższej jakości wyniku w porównaniu z głównymi modelami komercyjnymi.

8. Perspektywa rozwojowa

Stos agentowy jest narzędziem na wczesnym etapie rozwoju. Cztery kierunki jego dojrzewania zarysowują się już teraz.

Pierwszy to multimodalność jako pełnoprawny interfejs projektowy, w którym głos, gest, szkic i tekst stanowią równorzędne kanały wprowadzania intencji. Drugi to lokalne modele wyspecjalizowane dla branży AEC, dotrenowane na korpusach normatywnych i konstrukcyjnych, działające poza chmurą komercyjnego dostawcy. Trzeci to ustalenie standardów interoperacyjności między modelami językowymi a formatami danych projektowych (IFC, BIM). Czwarty to skonsolidowanie programów akademickich uczących pracy ze stosem agentowym jako osobnego przedmiotu studiów architektonicznych.

Cztery kierunki opisane powyżej nie są spekulatywne. Każdy z nich ma swoje pierwsze realizacje opisywane w cytowanej w niniejszym raporcie literaturze oraz w aplikacjach omówionych w sekcji 5. Pełna dojrzałość stosu agentowego w warsztacie architekta wymaga jednak dwóch typów pracy: inżynieryjnej i dydaktycznej. Wymiar inżynieryjny dotyczy integracji istniejących komponentów w spójne środowisko projektowe oraz powstania lokalnych modeli wyspecjalizowanych dla branży AEC, niezależnych od polityk komercyjnych dostawców. Wymiar dydaktyczny dotyczy konsolidacji programów akademickich uczących pracy ze stosem agentowym jako osobnej kompetencji warsztatowej. Programy te muszą obejmować również świadomość czterech granic omówionych w sekcji 7. W tym podziale środowisko akademickie odgrywa rolę szczególną: jest miejscem, w którym myśl projektowa i kompetencje warsztatowe spotykają się z refleksją badawczą i programową.

9. Wnioski

Warsztat architektoniczny przechodził w swojej historii kolejne przekształcenia narzędziowe: od kreski ołówka po skrypt parametryczny. Stos agentowy jest kolejnym z nich, ale pierwszym, w którym narzędzie potrafi zinterpretować intencję projektanta w języku naturalnym. Niniejszy raport powstał w momencie, w którym ta zmiana dopiero przeszła z laboratorium do operacyjnego użycia, i opisuje pole otwierające się dla architekta gotowego nim zarządzać.

Bibliografia

Alamasi, R., i Asfour, O. S. (2026). Applications of generative AI in architectural design education: A systematic review and future insights. *Digital*, *6*(1), art. 6.

https://doi.org/10.3390/digital6010006

Asanowicz, A., i Jakimowicz, A. (red.). (1998). *CAAD: Computer-Aided Architectural Design. Materiały konferencji*. Politechnika Białostocka.

Bolt, R. A. (1980). "Put-that-there": Voice and gesture at the graphics interface. *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '80)*, 262–270. https://doi.org/10.1145/800250.807503

Carpo, M. (2017). *The second digital turn: Design beyond intelligence*. MIT Press.

Casiez, G., Roussel, N., i Vogel, D. (2012). 1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*, 2527–2530. https://doi.org/10.1145/2207676.2208639

Cudzik, J., i Nyka, L. (2024). Artificial intelligence in architectural education — Green campus development research. *Global Journal of Engineering Education*, *26*(1), 20–25.

Du, Y., Liu, T., i Borrmann, A. (2026). Text2BIM: Generating building information models from text instructions using multi-agent large language models. *Journal of Computing in Civil Engineering*. [dane bibliograficzne do uzupełnienia w finalnej redakcji]

Eastman, C., Lee, J.-M., Jeong, Y.-S., i Lee, J.-K. (2009). Automatic rule-based checking of building designs. *Automation in Construction*, *18*(8), 1011–1033. https://doi.org/10.1016/j.autcon.2009.07.002

Forth, K., i Borrmann, A. (2024). Semantic enrichment of BIM models for energy performance simulation using domain-adapted language models. *Journal of Building Engineering*. [dane bibliograficzne do uzupełnienia w finalnej redakcji]

Jakimowicz, A., i Śliwecki, P. (red.). (2023). *Digital Architecture Research. Materiały konferencji*. Politechnika Białostocka.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., i Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, *55*(12), 1–38. https://doi.org/10.1145/3571730

Leach, N. (2022). *Architecture in the age of artificial intelligence: An introduction to AI for architects*. Bloomsbury.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang,

C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., i Grundmann, M. (2019). *MediaPipe: A framework for building perception pipelines* [preprint arXiv]. https://arxiv.org/abs/1906.08172

Oxman, R., i Oxman, R. (red.). (2014). *Theories of the digital in architecture*. Routledge.

Picon, A. (2010). *Digital culture in architecture: An introduction for the design professions*. Birkhäuser.

Płoszaj-Mazurek, M., i Ryńska, E. (2024). Artificial intelligence and digital tools for assisting low-carbon architectural design: Merging the use of machine learning, large language models, and building information modeling for life cycle assessment tool development. *Energies*, *17*, art. 2997. https://doi.org/10.3390/en17122997

Pottmann, H., Asperl, A., Hofer, M., i Kilian, A. (2007). *Architectural geometry*. Bentley Institute Press.

Schluntz, E., i Zhang, B. (2024). *Building effective agents* [opracowanie inżynierskie]. Anthropic. https://www.anthropic.com/engineering/building-effective-agents

Schumacher, P. (2008). Parametricism as style — Parametricist manifesto. *Architectural Design*. [pierwsza publikacja manifestu]

Schumacher, P. (2011). *The autopoiesis of architecture, Volume I: A new framework for architecture*. Wiley.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., i Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, *36*.

Słyk, J. (2012). *Źródła architektury informacyjnej*. Oficyna Wydawnicza Politechniki Warszawskiej.

Spallone, R., i Calvano, M. (2022). Parametric experiments on Palladio's 5 by 3 villas. *Nexus Network Journal*, *24*(2), 287–313. https://doi.org/10.1007/s00004-022-00592-1

Stiny, G., i Mitchell, W. J. (1978). The Palladian grammar. *Environment and Planning B*, *5*(1), 5–18. https://doi.org/10.1068/b050005

Tedeschi, A. (2014). *AAD: Algorithms-Aided Design — Parametric strategies using Grasshopper*. Le Penseur.

Wittkower, R. (1949). *Architectural principles in the age of humanism*. Alec Tiranti.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., i Wang, C. (2023). *AutoGen: Enabling next-gen LLM applications via multi-agent conversation* [preprint arXiv]. https://arxiv.org/abs/2308.08155

Yang, X., i Zhang, S. (2024). Automated translation of building regulations into logic using prompt engineering with GPT. *Automation in Construction*. [dane bibliograficzne do uzupełnienia w finalnej redakcji]

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., i Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations (ICLR)*.

Zheng, J., i Fischer, M. (2023). Dynamic prompt-based virtual assistant framework for BIM information search. *Automation in Construction*, *155*, art. 105067. https://doi.org/10.1016/j.autcon.2023.105067

Akty prawne

Ustawa z dnia 7 lipca 1994 r. Prawo budowlane. (1994). *Dziennik Ustaw*, 1994 nr 89 poz. 414, z późn. zm.

Rozporządzenie Ministra Infrastruktury z dnia 12 kwietnia 2002 r. w sprawie warunków technicznych, jakim powinny odpowiadać budynki i ich usytuowanie. (2002). *Dziennik Ustaw*, 2022 poz. 1225 (tekst jednolity).