

OKIEM EKSPERTA

Wzorce architektoniczne systemów wieloagentowych

Sprawdzone praktyki tworzenia GenAI, agentów, RAG, LLMOps oraz skalowalnych systemów AI



dr Ali Arsanjani
Juan Pablo Bustos



Tytuł oryginału: Agentic Architectural Patterns for Building Multi-Agent Systems:
Proven design patterns and practices for GenAI, agents, RAG, LLMOps,
and enterprise-scale AI systems

Tłumaczenie: Grzegorz Werner

ISBN: 978-83-289-4180-9

Copyright © Packt Publishing 2026. First published in the English language under the title
'Agentic Architectural Patterns for Building Multi-Agent Systems' – (9781806029570).

Polish edition copyright © 2027 by Helion S.A.

All rights reserved. No part of this book may be reproduced or transmitted in any
form or by any means, electronic or mechanical, including photocopying, recording
or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości
lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione.
Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie
książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie
praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi
bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje
były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich
wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych
lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności
za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

helion.pl/user/opinie/wzarai

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: helion@helion.pl

WWW: helion.pl (księgarnia internetowa, katalog książek)

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- [Lubię to! » Nasza społeczność](#)

*Chciałbym zadedykować tę książkę swojej rodzinie,
która jest dla mnie źródłem inspiracji*

— dr Ali Arsanjani

*Chciałbym zadedykować tę książkę swojej żonie, Cinthii,
oraz dzieciom, Penny i Andrew*

— to Wy motywujecie mnie do działania

— Juan Pablo Bustos

| Spis treści

O autorach	21
O recenzentach	22
Słowo wstępne	23
Przedmowa	25

CZĘŚĆ 1. Podstawy i kluczowe koncepcje

ROZDZIAŁ 1

GenAI w przedsiębiorstwie: krajobraz, dojrzałość

i podejście agentowe	33
Transformacyjny potencjał GenAI	34
Przegląd zastosowań biznesowych	37
Zastosowania horyzontalne (międzyfunkcyjne przypadki użycia)	37
Zastosowania wertykalne lub dziedzinowe	39
Wprowadzenie do agentowych systemów AI	41
Anatomia agentowej AI	41
Główne komponenty	42
Kluczowe cechy architektoniczne	44
Model dojrzałości GenAI: droga do systemów agentowych	46
Nowy stos agentowy	50
Komunikacja między agentami: od narzędzi do współpracy	50
Wewnętrzna struktura agenta (dla uproszczenia wspólna dla A i B)	51
Trudności we wdrażaniu GenAI klasy produkcyjnej	53
Podsumowanie	55

ROZDZIAŁ 2

LLM-y gotowe do pracy z agentami: wybór, wdrażanie i adaptacja 58

Rola LLM-ów w systemach agentowych	59
Wybór modelu: jak dobrać odpowiedni fundament?	63
Rozmiar okna kontekstowego	64
Rozmiar modelu a specjalizacja w kontekście agentów	66

Natywna obsługa wywoływania narzędzi i funkcji	69
Odporność, niezawodność i bezpieczeństwo modelu	70
Adaptowalność i potencjał dostrajania	72
Inne ważne kryteria wyboru	74
Wdrażanie i optymalizacja wydajności agentów	77
Architektury udostępniania agentowych LLM-ów	77
Strategie optymalizacji wydajności	79
Optymalizacja pod kątem interakcji z narzędziami	82
Aspekty bezpieczeństwa przy wdrażaniu LLM-ów na użytek agentów	83
AgentOps: zarządzanie LLM-ami w systemach agentowych	85
Podsumowanie	89

ROZDZIAŁ 3

Spektrum adaptacji LLM-ów do potrzeb agentów:

od RAG do dostrajania 93

Od ogólnych LLM-ów do wyspecjalizowanych agentów	94
Inny model dojrzałości agentowej AI	96
Ziarnistość agentów	96
Hierarchiczna architektura agentowa do automatyzacji procesów biznesowych	99
Podstawowe komponenty: agenty i ich możliwości	100
Struktura hierarchiczna: orkiestratory i specjaliści	102
Nadzór i obserwowalność z użyciem wywołań zwrotnych	103
Wzbogacanie kontekstowe: etap 1., wzbogacanie z użyciem RAG	104
Agent obsługi klienta oparty na RAG	105
Agent do analiz finansowych wykorzystujący RAG do pozyskiwania aktualnych informacji rynkowych	108
Agent ds. zgodności zapewniający przestrzeganie wymogów w monitorowaniu transakcji	111
Analiza scenariuszy	114
Dostrajanie pod kątem zdolności agentowych	116
Specjalizacja dziedzinowa	116
Spektrum dostrajania: od dostrajania efektywnego parametrycznie do dostrajania pełnego	117
Adaptacja agenta przez uczenie kontekstowe	120
Kompleksowy przykład: agent analizujący opinie o produkcie z użyciem ICL	122
Ugruntowywanie wyników modelu	124
Podsumowanie	126

CZĘŚĆ 2. Agentowa AI: architektura i wzorce projektowe

ROZDZIAŁ 4

Architektura agentowej AI: komponenty i interakcje 131

Definicja agenta: podstawowe pojęcia i możliwości	132
LLM-y	133
Zautomatyzowane przepływy pracy	134
Agenty AI	134
Anatomia agenta	135
Studium przypadku: agent do planowania podróży	137
Studium przypadku: agentowy system przetwarzania wniosków kredytowych	137
Magazyny danych i kontekst środowiskowy dla agentów	142
Modele interakcji agentów i ich kluczowe cechy	145
Cechy architektoniczne	145
Modele interakcji między agentami	145
Aspekty techniczne architektur agentowych	149
Podsumowanie	149

ROZDZIAŁ 5

Wzorce koordynacji wielu agentów 151

Praktyczny przewodnik po implementowaniu wzorców koordynacji	152
Systemy wieloagentowe: koordynacja podstawowa (poziom 4.)	154
Zaawansowana koordynacja i autokorekta wielu agentów (poziomy 5. – 6.)	154
Wzorzec routera agentów (routing oparty na intencji)	156
Kontekst	156
Problem	156
Rozwiązanie	157
Przykład: routing żądania dotyczącego zgodności	157
Przykładowa implementacja	157
Konsekwencje	159
Wskazówki implementacyjne	160
Modele delegowania zadań	160
Architektura nadzorcy (scentralizowana orkiestracja)	161
Architektura roju (emergentna koordynacja zdecentralizowana)	165
Topologie kompozycji agentów	168
Tablicowe centrum wiedzy	169
Rynek kontraktowy (mediator + oferty)	172
Drzewo nadzoru ze strzeżonymi zdolnościami	175
Planowanie wieloagentowe	178
Kontekst	178
Problem	178

Rozwiązanie	179
Przykład: generowanie raportu z analizy rynkowej	179
Przykład implementacji	180
Konsekwencje	181
Wskazówki implementacyjne	181
Współdzielenie wiedzy	181
Kontekst	182
Problem	182
Rozwiązanie	182
Przykład: współdzielenie rozwiązań w obsłudze klienta	183
Przykładowa implementacja	184
Konsekwencje	184
Wskazówki implementacyjne	185
Routing narzędzi w kontekście systemów wieloagentowych	185
Kontekst	186
Problem	186
Rozwiązanie	186
Przykład: inteligentny asystent osobisty	186
Przykładowa implementacja	187
Konsekwencje	188
Wskazówki implementacyjne	189
Konsensus	189
Kontekst	189
Problem	190
Rozwiązanie	190
Przykład: debata na temat prognozy finansowej	190
Przykładowa implementacja	192
Konsekwencje	193
Wskazówki implementacyjne	193
Negocjacje między agentami	194
Kontekst	194
Problem	194
Rozwiązanie	195
Przykład: negocjowanie dostępu do współdzielonego zasobu	195
Przykładowa implementacja	196
Konsekwencje	197
Wskazówki implementacyjne	198
Alokacja zasobów	198
Kontekst	198
Problem	198
Rozwiązanie	199
Przykład: autonomiczna alokacja robotów w inteligentnej fabryce	199
Przykładowa implementacja	201
Konsekwencje	202
Wskazówki implementacyjne	202

Rozwiązywanie konfliktów	203
Kontekst	203
Problem	204
Rozwiązanie	204
Przykład: rozwiązywanie konfliktu w korporacyjnym przepływie pracy	206
Przykładowa implementacja	207
Konsekwencje	208
Wskazówki implementacyjne	208
Sterowanie formacją	210
Kontekst	210
Problem	210
Rozwiązanie	211
Przykład: rój dronów rolniczych	211
Przykładowa implementacja	212
Konsekwencje	214
Wskazówki implementacyjne	214
Podsumowanie	214

ROZDZIAŁ 6

Wzorce wyjaśnialności i zgodności w systemach agentowych 217

Praktyczny przewodnik po implementacji wzorców wyjaśnialności i zgodności z przepisami	218
Audyt wierności instrukcjom	219
Kontekst	219
Problem	219
Rozwiązanie	219
Przykład: audyt stosowania rabatów w sklepie internetowym	219
Przykładowa implementacja	221
Konsekwencje	222
Wskazówki implementacyjne	222
Fraktalne osadzanie łańcucha myśli	223
Kontekst	223
Problem	223
Rozwiązanie	223
Przykład: synteza badań	224
Przykładowa implementacja	225
Konsekwencje	226
Wskazówki implementacyjne	226
Trwałe zakotwiczenie instrukcji	227
Kontekst	227
Problem	227
Rozwiązanie	228
Przykład: utrzymywanie ograniczeń w sprawozdawczości finansowej	228

Przykładowa implementacja	229
Konsekwencje	230
Wskazówki implementacyjne	231
Współdzielona pamięć epistemiczna	231
Kontekst	231
Problem	231
Rozwiązanie	232
Przykład: zakłócenie w łańcuchu dostaw	232
Przykładowa implementacja	233
Konsekwencje	235
Wskazówki implementacyjne	235
Komponowanie wzorców w celu zapewnienia niezawodności systemu	236
Podsumowanie	237

ROZDZIAŁ 7

Wzorce odporności i tolerancji na błędy 238

Strategiczny przewodnik po wdrażaniu wzorców odporności	239
Odporność agenta jako spektrum pięciu poziomów	239
Architektura integracji systemów: jak wzorce ze sobą współdziałają?	241
Łączenie wzorców w praktyce: przykład przetwarzania wniosku kredytowego	241
Mierzenie odporności: kluczowe wskaźniki	242
Konsensus w wykonywaniu równoległym	243
Kontekst	244
Problem	244
Rozwiązanie	244
Przykład: walidacja oceny zdolności kredytowej	244
Przykładowa implementacja	245
Konsekwencje	246
Wskazówki implementacyjne	247
Strategia opóźnionego przekazywania	247
Kontekst	247
Problem	248
Rozwiązanie	248
Przykład: niski poziom pewności podczas sprawdzania zgodności z przepisami	248
Przykładowa implementacja	249
Konsekwencje	250
Wskazówki implementacyjne	251
Nadzorca z limitem czasu	251
Kontekst	251
Problem	252
Rozwiązanie	252
Przykład: zapobieganie zawieszaniu się agenta analitycznego	252
Przykładowa implementacja	253

Konsekwencje	255
Wskazówki implementacyjne	255
Adaptacyjne ponawianie z mutowaniem promptu	256
Kontekst	256
Problem	256
Rozwiązanie	256
Przykład: naprawianie nieudanej ekstrakcji danych	257
Przykładowa implementacja	258
Konsekwencje	259
Wskazówki implementacyjne	260
Automatyczna resuscytacja agenta	260
Kontekst	260
Problem	261
Rozwiązanie	261
Przykład: ponowne uruchomienie agenta przetwarzania danych po awarii	261
Przykładowa implementacja	262
Konsekwencje	263
Wskazówki implementacyjne	264
Przyrostowe tworzenie punktów kontrolnych	264
Kontekst	265
Problem	265
Rozwiązanie	265
Przykład: wieloetapowy potok przetwarzania dokumentów	265
Przykładowa implementacja	266
Konsekwencje	269
Wskazówki implementacyjne	269
Głosowanie większościowe między agentami	270
Kontekst	270
Problem	270
Rozwiązanie	271
Przykład: finalizowanie decyzji w sprawie wniosku kredytowego	271
Przykładowa implementacja	273
Konsekwencje	274
Wskazówki implementacyjne	274
Graf zależności przyczynowych	275
Kontekst	275
Problem	275
Rozwiązanie	275
Przykład: audyt decyzji o przyznaniu kredytu	276
Przykładowa implementacja	276
Konsekwencje	278
Wskazówki implementacyjne	279

Samoobrona agenta	279
Kontekst	280
Problem	280
Rozwiązanie	280
Przykład: neutralizacja ataku na narzędzie do streszczania opinii	280
Przykładowa implementacja	282
Konsekwencje	283
Wskazówki implementacyjne	283
Siatkowa obrona agentów	284
Kontekst	284
Problem	284
Rozwiązanie	285
Przykład: zapobieganie dostępowi czatbota do bazy danych	285
Przykładowa implementacja	286
Konsekwencje	288
Wskazówki implementacyjne	288
Izolacja środowiska wykonawczego (sandboxing)	289
Kontekst	289
Problem	289
Rozwiązanie	289
Przykład: izolowanie złośliwego interpretera kodu	290
Przykładowa implementacja	290
Konsekwencje	292
Wskazówki implementacyjne	293
Optymalizacja narzutu związanego z translacją	293
Kontekst	293
Problem	293
Rozwiązanie	294
Przykład: streszczanie obszernego dokumentu	294
Przykładowa implementacja	295
Konsekwencje	296
Wskazówki implementacyjne	297
Wywoływanie z ograniczeniem częstotliwości	297
Kontekst	297
Problem	297
Rozwiązanie	297
Przykład: zarządzanie API biura kredytowego	298
Przykładowa implementacja	299
Konsekwencje	300
Wskazówki implementacyjne	300
Wywoływanie modelu zapasowego	301
Kontekst	301
Problem	301
Rozwiązanie	302

Przykład: zapewnienie ciągłej dostępności chatbota	302
Przykładowa implementacja	303
Konsekwencje	304
Wskazówki implementacyjne	305
Erozja i ocena zaufania	305
Kontekst	306
Problem	306
Rozwiązanie	306
Przykład: samoopptymalizujące się streszczanie wiadomości	306
Przykładowa implementacja	307
Konsekwencje	309
Wskazówki implementacyjne	310
Testowanie agenta kanarkowego	310
Kontekst	310
Problem	310
Rozwiązanie	311
Przykład: bezpieczne aktualizowanie agenta streszczającego	311
Przykładowa implementacja	312
Konsekwencje	313
Wskazówki implementacyjne	314
Podsumowanie	314

ROZDZIAŁ 8

Wzorce interakcji między człowiekiem a agentem 316

Strategiczny przewodnik po implementacji wzorców interakcji między człowiekiem a agentem	317
Poziomy interakcji między człowiekiem a agentem	317
Architektura integracji systemów: jak wzorce ze sobą współdziałają?	317
Łączenie wzorców w praktyce: przykład rezerwowania podróży służbowej	319
Mierzenie sukcesu: wskaźniki ewaluacji według wzorca	320
Agent wywołuje człowieka (przekazywanie do człowieka)	321
Kontekst	321
Problem	321
Rozwiązanie	321
Przykład: rozstrzyganie niejednoznaczności we wniosku kredytowym	322
Przykładowa implementacja	323
Konsekwencje	325
Wskazówki implementacyjne	325
Człowiek deleguje do agenta	325
Kontekst	326
Problem	326
Rozwiązanie	327
Przykład: delegowanie badania rynku	327

Przykładowa implementacja	328
Konsekwencje	329
Wskazówki implementacyjne	330
Człowiek wywołuje agenta	330
Kontekst	330
Problem	330
Rozwiązanie	331
Przykład: sprawdzanie statusu zamówienia	331
Przykładowa implementacja	331
Konsekwencje	333
Wskazówki implementacyjne	333
Agent deleguje do agenta	334
Kontekst	334
Problem	334
Rozwiązanie	334
Przykład: kompleksowa analiza finansowa	335
Przykładowa implementacja	336
Konsekwencje	337
Wskazówki implementacyjne	337
Agent wywołuje agenta pośredniczącego	338
Kontekst	338
Problem	338
Rozwiązanie	338
Przykład: międzyfirmowy program lojalnościowy	339
Przykładowa implementacja	340
Konsekwencje	341
Wskazówki implementacyjne	342
Podsumowanie	342

ROZDZIAŁ 9

Wzorce na poziomie agenta 344

Strategiczny przewodnik po implementacji wzorców na poziomie agenta	345
Komponenty wymagane do implementacji zdolności agenta	345
Wewnętrzna architektura agenta: jak poszczególne wzorce łączą się ze sobą?	347
Bazowy wzorec pojedynczego agenta	347
Kontekst	348
Problem	348
Rozwiązanie	348
Przykład: prosty agent do zatwierdzania wniosków kredytowych	349
Przykładowa implementacja	350
Konsekwencje	351
Wskazówki implementacyjne	351

Kontekst i pamięć specyficzne dla agenta	351
Kontekst	352
Problem	352
Rozwiązanie	352
Przykład: stanowy agent konwersacyjny do obsługi kredytów	353
Przykładowa implementacja	353
Konsekwencje	355
Wskazówki implementacyjne	355
Percepcja z użyciem RAG	356
Kontekst	356
Problem	356
Rozwiązanie	356
Przykład: agent kredytowy z obsługą RAG	357
Przykładowa implementacja	357
Konsekwencje	360
Wskazówki implementacyjne	360
Ustrukturyzowane rozumowanie i autokorekta	360
Kontekst	361
Problem	361
Rozwiązanie	361
Przykład: samokorygujący się agent kredytowy	361
Przykładowa implementacja	363
Konsekwencje	365
Wskazówki implementacyjne	365
Multimodalne dane sensoryczne	365
Kontekst	365
Problem	366
Rozwiązanie	366
Przykład: przetwarzanie wniosku kredytowego na podstawie obrazu	367
Przykładowa implementacja	367
Konsekwencje	369
Wskazówki implementacyjne	369
Wskazówki dotyczące wdrożeń w środowisku korporacyjnym	370
Mierzenie sukcesu: wskaźniki ewaluacyjne dla poszczególnych wzorców	370
Podsumowanie	370

ROZDZIAŁ 10

Wzorce systemowe dla gotowości produkcyjnej 372

Strategiczny przewodnik po implementacji wzorców systemowych	373
Odwzorowanie wzorców systemowych na model dojrzałości GenAI	373
Architektura integracji systemów: plan osiągnięcia gotowości do zastosowań produkcyjnych	374

Łączenie wzorców w praktyce: przykład automatyzacji łańcucha dostaw	376
Wskazówki dotyczące wdrożeń w środowisku korporacyjnym	377
Rejestr narzędzi i agentów	378
Kontekst	378
Problem	378
Rozwiązanie	379
Przykład: dynamiczny system zaopatrzeniowy	379
Przykładowa implementacja	380
Konsekwencje	382
Wskazówki implementacyjne	382
Monitorowanie zgodności z przepisami w czasie rzeczywistym	383
Kontekst	383
Problem	383
Rozwiązanie	383
Przykład: agent opieki zdrowotnej zgodny z HIPAA	384
Przykładowa implementacja	385
Konsekwencje	387
Wskazówki implementacyjne	387
Uwierzytelnianie i autoryzacja agenta	387
Kontekst	388
Problem	388
Rozwiązanie	388
Przykład: wielodziałowy system analityczny	389
Przykładowa implementacja	390
Konsekwencje	391
Wskazówki implementacyjne	391
Kluczowe zasoby i literatura uzupełniająca	392
Reaktywność sterowana zdarzeniami	393
Kontekst	393
Problem	393
Rozwiązanie	394
Przykład: zautomatyzowany system obsługi klienta	394
Przykładowa implementacja	395
Konsekwencje	397
Wskazówki implementacyjne	398
Podsumowanie	398

ROZDZIAŁ 11

Zaawansowana adaptacja: budowanie agentów, które się uczą 400

Koło zamachowe samodoskonalenia: model ciągłego uczenia się	401
Model R ⁵ : ramy operacyjne dla agentów produkcyjnych	402
Architektura hybrydowa (planista + oceniający)	403
Kontekst	404
Problem	404

Rozwiązanie	404
Przykład	405
Przykładowa implementacja	406
Konsekwencje	407
Wskazówki implementacyjne	407
Niestandardowe miary ewaluacji	407
Kontekst	408
Problem	408
Rozwiązanie	408
Przykład (STEPScore)	409
Przykładowa implementacja	409
Konsekwencje	411
Wskazówki implementacyjne	411
Generowanie danych syntetycznych z kontrolą preferencji	411
Kontekst	412
Problem	412
Rozwiązanie	412
Przykład	413
Przykładowa implementacja	414
Konsekwencje	415
Wskazówki implementacyjne	415
Zaawansowane wzorce dostrajania modeli	415
Podstawowe dostrajanie modeli (SFT i PEFT)	416
Dostrajanie preferencyjne (DPO)	416
Koewolucyjny trening agentów	417
Kontekst	418
Problem	418
Rozwiązanie	418
Przykład	420
Przykładowa implementacja	420
Konsekwencje	421
Wskazówki implementacyjne	422
Testowanie adversaryjne i red teaming	422
Kontekst	422
Problem	423
Rozwiązanie	423
Przykład	423
Przykładowa implementacja	424
Konsekwencje	425
Wskazówki implementacyjne	425
Zarządzanie kosztami i tokenomika	426
Kontekst	426
Problem	426
Rozwiązanie	426

Przykład	427
Przykładowa implementacja	428
Konsekwencje	429
Wskazówki implementacyjne	429
Mierzenie wartości biznesowej (ROI)	429
Kontekst	430
Problem	430
Rozwiązanie	430
Przykład	431
Przykładowa implementacja	432
Konsekwencje	433
Wskazówki implementacyjne	433
Odwzorowywanie zasad R ⁵ na wzorce samodoskonalenia	434
Podsumowanie	434

CZĘŚĆ 3. Wdrożenie: strategia, przypadki użycia i przyszłość

ROZDZIAŁ 12

Praktyczny plan: wdrażanie wzorców agentowych

według poziomu dojrzałości 439

Poziom 1. System fundamentalny (dojrzałość podstawowa)	440
Kluczowa zasada architektoniczna	441
Wzorce do zaimplementowania	441
Główne kwestie implementacyjne	442
Wynikowy system i konsekwencje	443
Poziom 2. Usługa gotowa do zastosowań produkcyjnych (dojrzałość średnio zaawansowana)	443
Kluczowa zasada architektoniczna	444
Wzorce do zaimplementowania	445
Główne kwestie implementacyjne	446
Wynikowy system i konsekwencje	447
Poziom 3. Samodoskonalący się ekosystem (dojrzałość zaawansowana)	447
Kluczowa zasada architektoniczna	448
Wzorce do zaimplementowania	448
Główne kwestie implementacyjne	449
Wynikowy system i konsekwencje	450
Twoja agentowa mapa drogowa: przewodnik do strategicznej refleksji	450
Początek podróży: skąd wyruszasz?	451
Budowanie fundamentów: czym jest Twój agent o minimalnej koniecznej funkcjonalności?	451

Budowanie z myślą o skalowalności: jak wygląda Twoja droga do systemu produkcyjnego?	452
Dążenie do autonomii: czym jest Twoja gwiazda przewodnia?	453
Tabela podsumowująca mapę drogową	453
Podsumowanie	453

ROZDZIAŁ 13

Przypadek użycia: pojedynczy agent do obsługi

wniosków kredytowych 456

Wymagania techniczne	457
Wyzwanie: krytyczny przepływ pracy	457
Kierowanie „umysłem” agenta z użyciem FCoT	458
Projektowanie monolitycznego agenta	459
Budowanie agenta w notatniku Colab	460
Konfiguracja i zależności	461
Definiowanie narzędzi	462
Konfigurowanie „umysłu” agenta	465
Wykonanie i analiza	470
Wykonywanie optymistycznej ścieżki	472
Wykonywanie ścieżki wyjątku: obsługa odrzucenia wniosku	474
Przejście z poziomu 5. na 6.: mapa doskonalenia	477
Zwiększanie odporności i niezawodności	477
Poprawa łatwości konserwacji poprzez specjalizację	478
Większa złożoność dzięki modularności	478
Podsumowanie	479

ROZDZIAŁ 14

Przypadek użycia: system wieloagentowy do obsługi

wniosków kredytowych 481

Wymagania techniczne	482
Hierarchiczne architektury agentowe	482
Budowanie systemu wieloagentowego	484
Narzędzia dla specjalistów	484
Tworzenie zespołu wyspecjalizowanych agentów	486
Dodawanie zabezpieczeń produkcyjnych	487
Przebudowa „umysłu” orkiestratora	489
Wykonanie i analiza	490
Praktyczna analiza wzorców	496
Wzorzec: architektura nadzorcy (orkiestrator AI lub metaagent)	496
Wzorzec: agent deleguje do agenta	497
Wzorzec: FCoT	498

Odwzorowanie przypadku użycia na poziomy agentowej AI	499
Poziom 3. Funkcjonalny, ale kruchy monolit	499
Poziom 4. Odporny, współpracujący zespół	500
Ponad poziom 4. — przyszłość współpracy między agentami	500
Podsumowanie	501

ROZDZIAŁ 15

Platformy agentowe — przypadek użycia: system wieloagentowy do obsługi wniosków kredytowych z CrewAI i LangGraphem 503

Wymagania techniczne	504
Agent Development Kit (ADK) firmy Google	504
CrewAI	505
LangGraph	506
Podobieństwa i różnice między trzema platformami	507
Podobieństwa	508
Główne różnice	508
Analiza porównawcza: ADK, CrewAI i LangGraph	510
Reimplementacja agenta kredytowego: praktyczne porównanie	510
Implementacja 1. CrewAI (współpracujący zespół)	516
Implementacja 2. LangGraph (maszyna stanów)	520
Obserwowalność i odpowiedzialna AI	528
Obserwowalność w praktyce	528
Odpowiedzialna AI	529
Zalecenia dotyczące wyboru platformy	530
Platformy jako katalizatory dojrzałości agentowej	532
Podsumowanie	534

ROZDZIAŁ 16

Podsumowanie: wytyczanie własnej ścieżki

w świecie agentowej AI 536

Studia przypadków	536
Studium przypadku 1. Zautomatyzowany agent ds. zgodności z przepisami finansowymi	537
Studium przypadku 2. Agent do naprawiania infrastruktury IT	537
Podsumowanie kluczowych wniosków	538
Model dojrzałości GenAI jako mapa drogowa	538
Agenty to coś więcej niż prompty	540
Wzorce to Twoje plany architektoniczne	540
Platformy przyspieszają projektowanie, ale go nie zastępują	541
Systemy produkcyjne wymagają holistycznego podejścia	541

Osiąganie wyższych poziomów dojrzałości agentowej	541
Oceń obecny stan swojej organizacji	542
Zidentyfikuj przypadki użycia o największym wpływie	542
Zdefiniuj architekturę, zaczynając od wzorców	543
Wprowadź nadzór i zabezpieczenia	544
Iteruj i ulepszaj	544
Plan działania dla praktyków	545
Opanuj jedną platformę agentową	545
Myśl wzorcami	546
Rozwiń umiejętności w zakresie AgentOps	547
Promuj odpowiedzialną AI	548
Plan działania dla praktyków	549
Uwagi końcowe	550
Słowo od autorów	551

Architektura agentowej AI: komponenty i interakcje

Rozdział

4

W części I tej książki wyłożyliśmy podstawy niezbędne do zrozumienia generatywnej AI i opisaliśmy jej rozwój w kierunku bardziej zaawansowanych, rozproszonych systemów oraz ewolucję ku większej autonomii. Zbadaliśmy transformacyjny potencjał GenAI w środowisku korporacyjnym, wyznaczyliśmy ścieżkę ku coraz bardziej zaawansowanym możliwościom za pomocą modelu dojrzałości agentowej AI, a następnie skupiliśmy się na kluczowej roli LLM-ów jako silników kognitywnych napędzających te inteligentne systemy.

W rozdziałach 2. i 3. szczegółowo omówiliśmy kluczowe aspekty wyboru, wdrażania, optymalizacji i adaptacji LLM-ów — od RAG po dostrajanie — tak aby były one naprawdę gotowe do pracy z agentami. Ustaliliśmy, że przygotowanie LLM-a do zadań agentowych z pewnością obejmuje inżynierię promptów, ale wykracza daleko poza tę technikę. Obejmuje również definiowanie sposobu, w jaki model interpretuje intencje użytkownika i wchodzi w interakcję z narzędziami (API) poprzez wywoływanie funkcji. Agent wymaga starannego przygotowania, aby mógł służyć jako niezawodny rdzeń rozumowania w ramach większej struktury operacyjnej.

Położywszy fundamenty pod przygotowanie kognitywnego rdzenia agenta, w tym rozdziale przenosimy uwagę na praktyczną architekturę niezbędną do budowania solidnych i skutecznych systemów agentowej AI.

Rozdział rozpoczniemy od analizy fundamentalnej architektury agentowej AI, koncentrując się na głównych komponentach, obowiązkach i funkcjach, które odróżniają agenty od prostszych interakcji z AI. W poprzednich rozdziałach ustaliliśmy znaczenie LLM-ów jako silników rozumowania, zdolnych do adaptacji w reakcji na dane sensoryczne napływające z otoczenia i odpowiedniego korygowania planów działania.

Następnie zmienimy perspektywę: LLM, niezależnie od swoich możliwości, nie jest produktem końcowym, lecz krytycznym komponentem w ramach szerszej, rozproszonej struktury operacyjnej. Oznacza to przełomową zmianę paradygmatu projektowego: *od systemów monolitycznych do dynamicznych ekosystemów inteligentnych agentów AI opartych na rolach*.

Dalej przeanalizujemy anatomię agenta, badając jego podstawowe elementy składowe — jak otrzymuje cele i postrzega swoje środowisko oraz jak na nie oddziałuje. Zbadamy kluczową rolę magazynów danych i kontekstu środowiskowego, od których zależy inteligentne zachowanie agentów. Na koniec przedstawimy różne modele interakcji między

agentami oraz kluczowe cechy definiujące sposób działania i współpracy tych systemów. Zrozumienie tej architektury to pierwszy krok ku projektowaniu i implementacji zaawansowanych wzorców agentowych, które będziemy eksplorować w części II.

W tym rozdziale omówimy następujące tematy:

- definiowanie agenta: podstawowe koncepcje i możliwości;
- anatomia agenta;
- magazyny danych i kontekst środowiskowy;
- modele interakcji agentów i ich kluczowe cechy;
- aspekty techniczne architektur agentowych.

Zacznijmy od jasnego zdefiniowania, czym jest agent AI.

Definicja agenta: podstawowe pojęcia i możliwości

W rozdziale 1. wprowadziliśmy koncepcję agentowych systemów AI. W tym miejscu kluczowe jest ujednoznacznienie naszej definicji agenta AI.

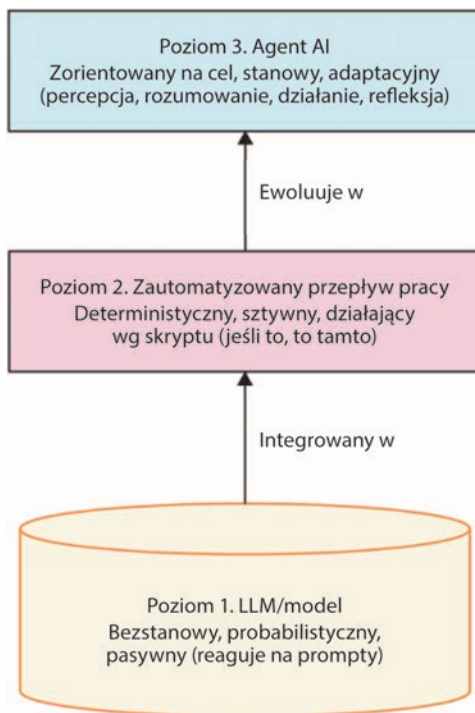
Agent AI to system, często oparty na LLM-ach, zaprojektowany do postrzegania swojego otoczenia, podejmowania decyzji i wykonywania działań w celu osiągnięcia określonych celów. Ta definicja opisuje bardziej autonomiczny byt z bieżącymi celami, który wykacza poza proste reagowanie na prompty.

Prawdziwe agenty AI mają kilka kluczowych cech, które odróżniają je od prostszych interakcji z LLM-em:

- **Autonomia.** Agenty mają pewien stopień samodzielności, który pozwala im działać niezależnie w dążeniu do celów bez ciągłej bezpośredniej ingerencji człowieka. Po wyznaczeniu celu agent potrafi sam określić kroki potrzebne do jego osiągnięcia.
- **Reaktywność (czyli zdolność percepcji).** Agenty są w stanie postrzegać swoje środowisko operacyjne i reagować na zachodzące w nim zmiany lub zdarzenia. To „wyczuwanie” otoczenia jest kluczowym aspektem zachowania agentowego.
- **Proaktywność (czyli orientacja na cel).** Agenty nie tylko reagują — podejmują inicjatywę i działają w sposób ukierunkowany na cel. Dążą do realizacji wyznaczonych zadań, co może obejmować złożone planowanie i wykonywanie zadań rozłożonych w czasie.
- **Zdolności społeczne (opcjonalne, ale powszechne, szczególnie w systemach wieloagentowych).** Wiele agentów, zwłaszcza tych, które działają w systemach wieloagentowych, potrafi wchodzić w interakcje i komunikować się z innymi agentami i ludźmi. Wykorzystują uzgodnione języki i protokoły do współpracy, negocjacji lub koordynacji działań.

Istotne jest wyraźne odróżnienie agentowej AI zarówno od prostych interakcji z LLM-ami, jak i od tradycyjnych zautomatyzowanych przepływów pracy. Choć we współczesnych

systemach korzysta się ze wszystkich trzech rozwiązań — LLM-ów, zautomatyzowanych przepływów pracy i agentów AI — to reprezentują one zasadniczo różne poziomy możliwości i autonomii, jak ilustruje hierarchia z rysunku 4.1.



Rysunek 4.1. Hierarchia autonomii

LLM-y

LLM-y stanowią rdzeń rozumowania agenta, jego „mózg” — silnik umożliwiający rozumienie, planowanie i generowanie treści. Na potrzeby tej książki używamy zamiennie terminów LLM i MMM, choć różnica między nimi jest technicznie istotna.

Modele multimodalne (ang. *multi-modal model*, **MMM**) stanowią ważny etap ewolucji w tej warstwie. W przeciwieństwie do wczesnych modeli trenowanych wyłącznie na tekście współczesne modele multimodalne (takie jak Google Gemini 3) są trenowane na ogromnych zbiorach danych obejmujących tekst, kod, obrazy, audio i wideo. Dzięki temu potrafią rzutować różne modalności na wspólną przestrzeń semantyczną, co umożliwia natywne rozumowanie na różnych typach danych wejściowych. Na przykład model multimodalny może przeanalizować zrzut ekranowy pulpitu nawigacyjnego w celu zdiagnozowania błędu systemowego albo odsłuchać plik audio, aby wyekstrahować odczucia — bez polegania na osobnych, rozłącznych modelach translacyjnych.

Jednak MMM sam w sobie *nie jest* agentem. Pomimo zaawansowanych zdolności percepcyjnych model pozostaje z natury bezstanowy i pasywny. Przetwarza dane wejściowe

i przewiduje dane wyjściowe, ale nie inicjuje działań, nie utrzymuje długoterminowej pamięci z własnej woli ani nie realizuje samodzielnie celów. Reaguje wyłącznie wtedy, gdy zostanie o coś zapytany.

Kontrastuje to ze scentralizowanymi, reaktywnymi modelami z rozproszonymi systemami agentowymi, które działają kooperatywnie w rojach, koalicjach lub społecznościach. O ile model dostarcza surową inteligencję, to dopiero otaczająca go architektura agentowa przekształca ten pasywny potencjał w aktywne, celowe działanie.

Zautomatyzowane przepływy pracy

Zautomatyzowany przepływ pracy, czyli orkiestracja AI, to sekwencja zadań, z których część jest z góry określona i deterministyczna, a pozostałe mogą być niedeterministyczne — opierają się na interpretacji intencji i wywoływaniu funkcji.

Można to sobie wyobrazić jako skrypt wykonujący serię kroków w stylu „jeśli to, to tamto”. Choć realizuje konkretne akcje, jest deterministyczny i sztywny — podąża ustaloną ścieżką. Nie potrafi rozumować, dynamicznie planować ani dostosowywać swojego działania do nieprzewidzianych okoliczności.

Agenty AI

Agent AI jest najbardziej zaawansowanym spośród tych trzech rozwiązań. To kompletny system, który wykorzystuje LLM jako swój „mózg” do autonomicznego osiągnięcia celów. Agent jest zorientowany na cel, ma stan wewnętrzny i potrafi się adaptować. W odróżnieniu od prostego przepływu pracy potrafi analizować swoje otoczenie, tworzyć dynamiczny plan działania, używać narzędzi do realizowania tego planu oraz uczyć się na podstawie wyników, aby w przyszłości działać skuteczniej.

LLM może na przykład napisać szkic e-maila po otrzymaniu odpowiedniego promptu. Zautomatyzowany przepływ pracy może wysyłać wcześniej przygotowanego e-maila w każdy poniedziałek o 9:00. Agent AI natomiast może zostać oddelegowany do zarządzania całą skrzynką odbiorczą — aktywnie klasyfikuje wiadomości, używa narzędzi do planowania spotkań na podstawie treści e-maili i powiadamia użytkownika o pilnych sprawach, jednocześnie ucząc się jego preferencji z biegiem czasu.

Aby jeszcze lepiej zobrazować różnice między tymi rozwiązaniami, porównaliśmy je w tabeli 4.1.

Zasadniczo agent to system, który integruje i przenosi na wyższy poziom możliwości dwóch pozostałych podejść. Wykorzystuje LLM jako silnik rozumowania, ale osadza go w strukturze umożliwiającej autonomiczne, zorientowane na cel wykonywanie zadań — w sposób, jakiego nie jest w stanie zapewnić statyczny przepływ pracy.

Po omówieniu tych fundamentalnych pojęć i różnic między nimi przyjrzymy się teraz wewnętrznej strukturze, która umożliwia realizację tych zaawansowanych funkcji agentowych.

Tabela 4.1. Porównanie LLM-ów, zautomatyzowanych przepływów pracy i agentów AI

Cecha	LLM	Zautomatyzowany przepływ pracy	Agent AI
Główna funkcja	Generuje tekst, odpowiada na pytania, syntetyzuje informacje na podstawie promptu	Wykonuje predefiniowaną, statyczną sekwencję zadań	Autonomicznie realizuje określony cel
Podjmowanie decyzji	Dopasowywanie wzorców i probabilistyczne generowanie tekstu. Nie podejmuje decyzji	Oparte na sztywno zakodowanej logice „jeśli to, to tamto”. Nie rozumuje	Rozumuje, planuje i podejmuje dynamiczne decyzje, aby radzić sobie ze złożonymi problemami
Stan i pamięć	Bezstanowy: każda interakcja to nowy początek (choć kontekst może być przekazany w prompcie)	Zasadniczo bezstanowy; nie pamięta poprzednich przebiegów	Stanowy: utrzymuje pamięć krótko- i długoterminową, aby uczyć się i adaptować
Adaptacyjność	Nie adaptuje się, chyba że zostanie dostrojony lub wsparty nowymi technikami promptowania	Sztywny: każda zmiana procesu wymaga przeprogramowania przepływu pracy	Wysoce adaptacyjny; uczy się na doświadczeniu i reaguje na zmiany w swoim otoczeniu
Model interakcji	Otrzymuje prompt, zwraca odpowiedź	Uruchamiany przez zdarzenie, wykonuje ustalony skrypt	Działa w ciągłej pętli percepcja – rozumowanie – planowanie – działanie, dążąc do realizacji swoich celów
Obsługa błędów	Pasywna/brak: może generować wiarygodnie brzmiące, ale niepoprawne informacje (halucynacje) bez świadomości błędu	Sztywna/krucha: kończy się twardym błędem lub uruchamia predefiniowaną ścieżkę wyjątku (np. „zatrzymaj się i powiadom administratora”), gdy naruszone zostaną określone reguły	Odporna/samokorygująca: potrafi wykrywać błędy, przeanalizować ich przyczynę i autonomicznie ponowić próbę z inną strategią lub narzędziem

Anatomia agenta

Jak już wspomnieliśmy, funkcjonalność agenta AI można zrozumieć poprzez jego podstawowe komponenty. Podczas gdy wcześniejsze omówienie stanowiło ogólne wprowadzenie, w tym podrozdziale wracamy do anatomii agenta z perspektywy architektonicznej.

Tym razem przedstawimy te komponenty nie tylko jako koncepcje, ale jako funkcjonalne elementy konstrukcyjne, które umożliwiają ciągłą pętlę operacyjną agenta: postrzeganie otoczenia, rozumowanie prowadzące do sformułowania planu oraz działanie zmierzające do osiągnięcia celów. Zrozumienie tej struktury jest niezbędne do implementacji wzorców projektowych opisanych w dalszej części.

Każdy agent ma wewnętrzną strukturę złożoną z poniższych bloków budulcowych, które zostały wymienione w tabeli 4.2 wraz z ich rolami architektonicznymi.

Tabela 4.2. Role architektoniczne komponentów agenta

Komponent	Podstawowa funkcja (przypomnienie)	Rola architektoniczna i implementacja
Cele (określane przez instrukcje)	Zadania lub pożądane rezultaty, które agent stara się osiągnąć	Definiują funkcję celu agenta i kierują jego planowaniem wysokopoziomowym. Implementowane jako parametry konfiguracyjne lub dynamiczny stan, który może być aktualizowany
Percepcja	Zbiera informacje i dane z otoczenia (cyfrowego lub fizycznego)	Pełni rolę warstwy wejściowej. Implementowana za pomocą listenerów API, procesorów strumieni danych lub standardowych protokołów, takich jak Model Context Protocol (MCP)
Rozumowanie	Główna jednostka przetwarzania, w której analizowane i interpretowane są zebrane informacje	Rdzeń poznawczy, z którym zintegrowany jest LLM przystosowany do pracy agentowej. Interpretuje dane wejściowe, ocenia je w kontekście celów i formułuje strategie wysokopoziomowe
Planowanie	Opracowuje sekwencję działań na podstawie wyników rozumowania, aby osiągnąć cele	Warstwa taktyczna, również napędzana przez LLM. Rozbija strategię wysokopoziomową z komponentu rozumowania na konkretną, uporządkowaną sekwencję wykonalnych kroków lub wywołań narzędzi
Działanie	Wykonuje zaplanowane akcje w środowisku, korzystając z dostępnych narzędzi	Pełni rolę warstwy wyjściowej. Implementowane poprzez wywoływanie narzędzi: zapytania do zewnętrznych API, wykonywanie kodu, wysyłanie poleceń do aktyuatorów lub generowanie odpowiedzi
Pamięć	Przechowuje wiedzę, doświadczenia i stan agenta, dostarczając kontekstu do podejmowania decyzji	Zarządza stanem. Implementowana za pomocą zmiennych krótkoterminowych dla bieżącego zadania oraz długoterminowych magazynów trwałych, takich jak bazy wektorowe na użytek RAG czy przechowywanie preferencji użytkownika
Koordinacja	Interakcja z innymi agentami w celu uzgadniania działań i współpracy nad wspólnymi celami (głównie w systemach wieloagentowych)	Warstwa komunikacji między agentami. Ten komponent zarządza pełnym cyklem życia zadania, śledząc ustandaryzowane stany komunikacji A2A, takie jak: submitted (zgłoszone), working (w trakcie), input-required (wymagane dane wejściowe) i completed (zakończone). Implementowany za pomocą protokołów takich jak Agent2Agent (A2A), umożliwia agentom delegowanie pracy i deterministyczną synchronizację stanu w rozproszonym środowisku

Te elementy konstrukcyjne nie są statyczne — działają w ciągłym cyklu. Agent *odbiera sygnały* ze swojego środowiska, *rozumuje* o nowych informacjach w kontekście swoich celów i *pamięci*, formułuje *plan*, a następnie podejmuje *działanie* w środowisku.

Wyniki tych działań są analizowane w kolejnej iteracji, co tworzy pętlę sprzężenia zwrotnego, która pozwala agentowi uczyć się i dostosowywać zachowanie w czasie. To właśnie ta dynamiczna pętla operacyjna (zbudowana na opisanych komponentach architektonicznych) stanowi fundament bardziej złożonych zachowań i wzorców projektowych, które będziemy omawiać dalej w tej książce.

Aby przejść od teorii do praktyki, przyjrzymy się teraz dwóm konkretnym studiom przypadku. Celowo dobraliśmy je tak, aby pokazać anatomię agenta działającego na różnych poziomach złożoności.

Na początek przeanalizujemy agenta do planowania podróży — przystępny system jednoagentowy, który wyraźnie ilustruje, jak podstawowe elementy składowe — *cele, percepcja, rozumowanie, planowanie, działanie i pamięć* — współdziałają ze sobą, aby zrealizować żądanie użytkownika od początku do końca.

Następnie przeanalizujemy agentowy system przetwarzania wniosków kredytowych, który prezentuje bardziej zaawansowaną architekturę wieloagentową. Ten przykład klasy korporacyjnej pokaże, jak wiele wyspecjalizowanych agentów współpracuje ze sobą, koordynuje działania i dzieli kontekst, aby obsłużyć złożony, kompleksowy proces biznesowy.

Razem te studia przypadków stanowią praktyczną perspektywę, która pozwala zrozumieć zarówno podstawowe komponenty pojedynczego agenta, jak i zasady architektoniczne systemu wieloagentowego.

Studium przypadku: agent do planowania podróży

Aby lepiej zilustrować, jak te elementy składowe współdziałają ze sobą, rozważmy praktyczny przykład: autonomicznego agenta do planowania podróży. Głównym celem agenta jest opracowanie kompletnego planu podróży z wykorzystaniem różnych systemów zewnętrznych na podstawie zapytania użytkownika wyrażonego w języku naturalnym.

Poszczególne elementy anatomii agenta w kontekście tego przykładu wymieniliśmy w tabeli 4.3.

Ten przykład pokazuje, jak abstrakcyjne elementy anatomii agenta przekładają się na konkretne operacje. Ciągła pętla wykrywania potrzeb użytkownika, analizowania dostępnych opcji, planowania kolejnych kroków i wykonywania działań za pomocą narzędzi — to właśnie ona pozwala agentowi przejść od prostego zapytania do złożonego, pomyślnie zrealizowanego celu.

Studium przypadku: agentowy system przetwarzania wniosków kredytowych

W tym punkcie przedstawiamy praktyczne zastosowanie schematu z rysunku 1.1 do scenariusza przetwarzania wniosków kredytowych z wykorzystaniem każdego opisanego wcześniej komponentu architektury agentowej. Przykład ten przekłada teorię na

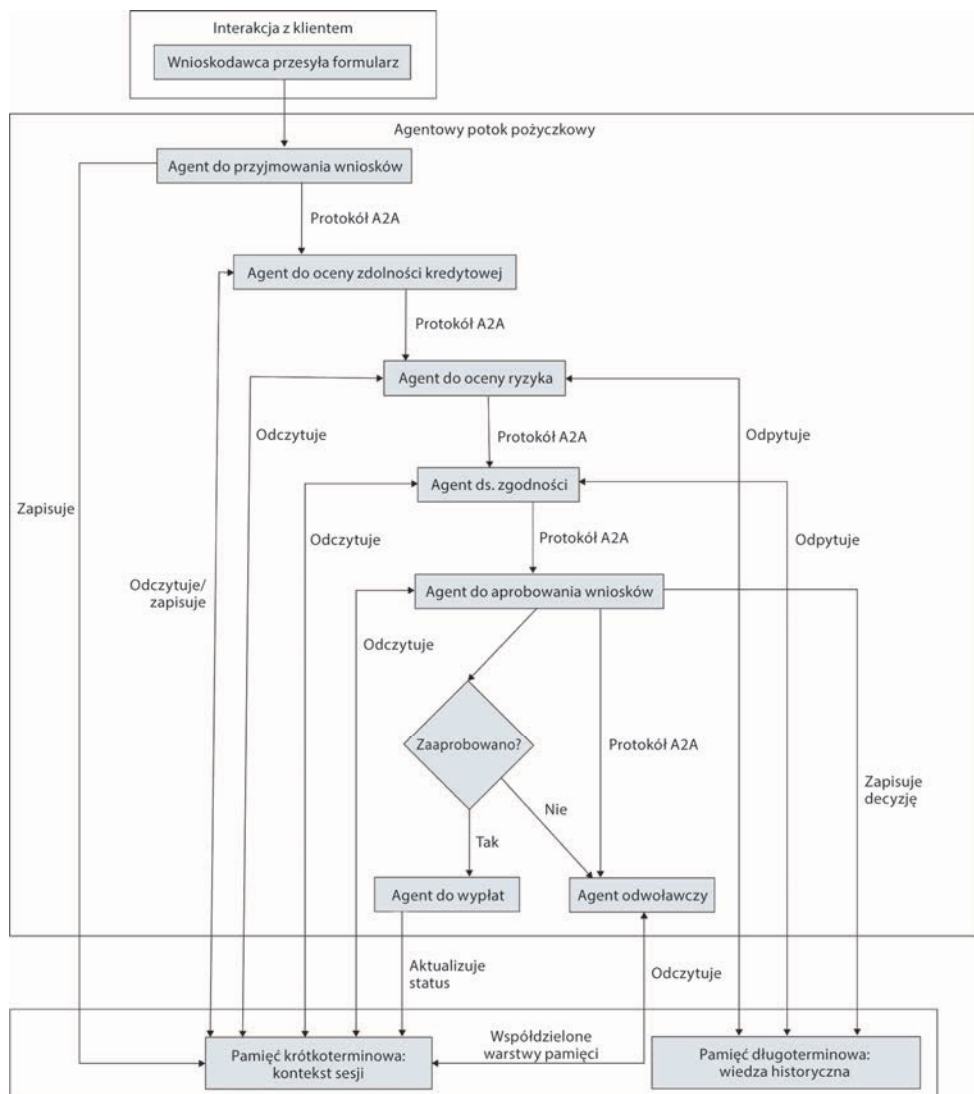
Tabela 4.3. Komponenty agenta w przypadku użycia

Komponent anatomii	Przykład agenta do planowania podróży
Cele	Głównym zadaniem agenta jest spełnienie żądania użytkownika: „Zarezerwuj lot w obie strony do Paryża i 4-gwiazdkowy hotel na 5 nocy w przyszłym miesiącu, tak aby łączny koszt wyniósł poniżej 2500 dolarów”. Cel ten jest dynamiczny i może być aktualizowany, jeśli użytkownik zmieni zdanie lub poda nowe ograniczenia.
Percepcja	Agent zbiera wstępne informacje, przetwarzając zapytanie użytkownika w języku naturalnym. Kontynuuje „postrzeganie”, odbierając dane zwrotne z systemów zewnętrznych, takie jak lista dostępnych lotów z API linii lotniczych czy informacje cenowe z serwisu rezerwacji hoteli.
Rozumowanie	Rdzeń LLM analizuje jawne preferencje użytkownika („Paryż”, „4-gwiazdkowy”, „5 nocy”) oraz jego niejawne intencje. Interpretuje ustrukturyzowane dane z wyszukiwań lotów i hoteli, porównuje opcje z ograniczeniem budżetowym i przeprowadza złożone wnioskowanie, aby określić optymalny plan podróży.
Planowanie	Na podstawie swojego rozumowania agent opracowuje sekwencję kroków: 1. Rozbicie żądania użytkownika na podzadania (rezerwacja lotu, rezerwacja hotelu). 2. Wyszukanie lotów. 3. Wyszukanie hoteli. 4. Analiza wyników w celu znalezienia prawidłowej kombinacji spełniającej wszystkie ograniczenia. 5. Przedstawienie ostatecznego planu podróży użytkownikowi do zatwierdzenia. 6. Wykonanie końcowych działań rezerwacji.
Działanie	Agent realizuje swój plan, korzystając z dostępnych narzędzi. Obejmuje to wywołanie API wyszukiwania lotów (np. <code>search_flights(destination="CDG", month="next")</code>) oraz API hoteli (np. <code>find_hotels(city="Paris", rating=4, nights=5)</code>). Po uzyskaniu potwierdzenia agent ponownie podejmuje działanie: wywołuje funkcje rezerwacji, generuje końcową wiadomość z potwierdzeniem oraz aktualizuje długoterminowy profil użytkownika o nowe preferencje na potrzeby przyszłych interakcji.
Pamięć	Agent wykorzystuje pamięć krótkoterminową do przechowywania preferencji użytkownika, znalezionych opcji lotów i hoteli oraz historii rozmowy dla bieżącego zadania. Może korzystać z pamięci długoterminowej, aby przypomnieć sobie preferowaną linię lotniczą lub sieć hoteli użytkownika z poprzednich interakcji i personalizować przyszłe sugestie.
Koordinacja	W systemie wieloagentowym agent podróży mógłby koordynować działania z innymi wyspecjalizowanymi agentami. Na przykład po zarezerwowaniu głównej podróży mógłby delegować zadanie do „agenta ds. wycieczek” poprzez wysłanie wiadomości w stylu „Znajdź i zarezerwuj bilet do Luwru”, współpracując w ramach wspólnego celu, jakim jest zaplanowanie całej podróży.

praktykę i pokazuje, jak systemy agentowe działają w ramach pełnego stosu technologicznego w wieloagentowym procesie zatwierdzania kredytów.

Institucja finansowa wdraża system wieloagentowy w celu zautomatyzowania i zoptymalizowania procesu obsługi wniosków kredytowych. Przepływ pracy obejmuje cały cykl — od pierwszej interakcji z klientem po finalną wypłatę środków — i wiąże się z weryfikacją danych, oceną zdolności kredytowej, oceną ryzyka, kontrolą zgodności z przepisami oraz ostatecznym zatwierdzeniem. Każdy agent specjalizuje się w odrębnym

etapie tego potoku, wykorzystując protokół A2A do komunikacji i koordynacji zadań, a jednocześnie korzystając ze współdzielonej warstwy pamięci w celu zachowania spójnego kontekstu (rysunek 4.2 i tabela 4.4).



Rysunek 4.2. Przykładowy scenariusz: agentowe przetwarzanie wniosków kredytowych

Przykładowy przepływ: cykl przetwarzania wniosku kredytowego

1. Klient przesyła formularz wniosku o pożyczkę → agent do przyjmowania wniosków parsuje formularz i zapisuje go we współdzielonej pamięci.
2. Zostaje uruchomiony agent do oceny zdolności kredytowej → agent pobiera wynik FICO oraz historię kredytową przez API.

Tabela 4.4. Komponenty anatomii agenta w procesie obsługi wniosków kredytowych

Komponent	Przykład agenta do przetwarzania wniosków kredytowych
Cele	Każdy agent ma wyspecjalizowany cel dopasowany do swojej roli: - Agent do przyjmowania wniosków — zbiera kompletne dane wnioskodawcy. - Agent do oceny zdolności kredytowej — sprawdza historię kredytową i sygnalizuje anomalie. - Agent do oceny ryzyka — przyznaje punktację profilu ryzyka wnioskodawcy. - Agent ds. zgodności — weryfikuje przestrzeganie przepisów prawnych. - Agent do aprobowania wniosków — podejmuje ostateczną decyzję kredytową na podstawie całościowych danych wejściowych.
Percepcja	Agenty zbierają dane poprzez: - formularze wniosków (język naturalny, dane ustrukturyzowane); - wywołania API do biur informacji kredytowej; - wewnętrzne bazy CRM i KYC; - OCR dokumentów; - dane rynkowe czasu rzeczywistego (dla pożyczek o zmiennym oprocentowaniu). Każdy agent korzysta z MCP, aby kontekstowo uzyskiwać dostęp do spójnych danych strukturalnych i niestukturalnych.
Rozumowanie	Agenty napędzane przez LLM rozumują na podstawie: - semantyki dokumentów finansowych (np. odcinki wypłat, wyciągi bankowe); - zasad i wytycznych dotyczących ryzyka; - intencji wywnioskowanych z zapytań wnioskodawcy; - sprzeczności w danych (np. rozbieżności w dochodach). Rdzeń kognitywny ocenia dane wejściowe w odniesieniu do celów agenta.
Planowanie	Plany są specyficzne dla każdego agenta: - Agent do przyjmowania wniosków planuje działania następcze w przypadku braku niektórych informacji. - Agent do oceny zdolności kredytowej planuje, które biura informacji kredytowej odpytać (np. Equifax, TransUnion) i w jakiej kolejności, a także definiuje strategię obsługi awarii API lub rozbieżności w danych. - Agent do oceny ryzyka wybiera model scoringowy w zależności od typu kredytu. - Agent ds. zgodności dobiera wymagane kontrole w zależności od jurysdykcji. - Agent do aprobowania wniosków buduje drzewo decyzyjne integrujące wyniki pracy pozostałych agentów.
Działanie	Agenty działają poprzez: - wysyłanie wiadomości weryfikacyjnych; - aktualizowanie statusu wniosku w systemie CRM; - wywoływanie API w celu odczytania lub zapisania danych; - generowanie ścieżek audytu; - uruchamianie kolejnych agentów po zakończeniu zadania.
Pamięć	Dwie warstwy: krótkoterminowa — pamięć sesji specyficzna dla danego wnioskodawcy; długoterminowa — zagregowane wzorce oszustw, historyczne decyzje, zmiany przepisów. Pamięć umożliwia doskonalenie oparte na uczeniu się (np. dostosowywanie systemu do nowych wskaźników ryzyka).

Tabela 4.4. Komponenty anatomii agenta w procesie obsługi wniosków kredytowych — ciąg dalszy

Komponent	Przykład agenta do przetwarzania wniosków kredytowych
Koordynacja	Koordynacja jest realizowana przez protokół A2A (ang. <i>Agent2Agent Interoperability</i>) opracowany przez Google’a i pełniący rolę warstwy komunikacyjnej: ■ Zapewnia bezpieczny i standardowy kanał umożliwiający agentom delegowanie zadań i wymianę informacji. ■ Pozwala agentom raportować status zadania (np. w trakcie realizacji, zakończone, niepowodzenie) do agenta delegującego. ■ Zapewnia zgodność operacyjną — agenty zbudowane na różnych platformach mogą współpracować bez udostępniania swojej wewnętrznej pamięci czy logiki.

3. Agent do oceny ryzyka analizuje dane kredytowe, dochody i kwotę pożyczki → przypisuje scoring ryzyka.
4. Agent ds. zgodności weryfikuje zgodność z KYC, AML, GDPR oraz lokalnymi przepisami bankowymi.
5. Agent do aprobowania wniosków integruje dane wejściowe, przeprowadza rozumowanie dotyczące konfliktów i podejmuje decyzję o przyjęciu lub odrzuceniu wniosku.
6. W przypadku odrzucenia wniosku agent odwoławczy może zaproponować alternatywne produkty (np. pożyczkę zabezpieczoną).
7. Agent do wypłat realizuje transfer środków i aktualizuje systemy zaplecza.

Koordynacja wielu agentów przez A2A

Agenty współpracują ze sobą, wysyłając ustrukturyzowane zadania za pomocą zdefiniowanego protokołu, takiego jak A2A. Jest to forma bezpośredniej komunikacji, w której agent wysyłający jawnie deleguje zadanie do konkretnego agenta odbierającego.

Na przykład zamiast publikować wynik na ogólnej szynie danych, agent orkiestrator najpierw wysyła zadanie do agenta do oceny ryzyka. Po otrzymaniu wyniku wysyła kolejne zadanie do agenta do aprobowania wniosków, dołączając ocenę ryzyka jako część danych. Tworzy to przejrzysty, audytowalny łańcuch delegacji.

Spory lub przypadki graniczne (np. wyniki kredytowe na pograniczu progów) mogą uruchomić pętlę deliberacji. Obsługiwana jest ona przez sekwencję wiadomości A2A, w ramach których agenty mogą wymieniać się propozycjami, kontrofertami lub przekazywać zadanie do innego agenta bądź człowieka w celu rozstrzygnięcia.

Korzyści z zastosowania anatomii agentowej w procesie obsługi pożyczek obejmują:

- **Reaktywność.** Agenty reagują na zmiany danych w czasie rzeczywistym (np. wahania scoringu kredytowego).
- **Zgodność z przepisami.** Modułowe reguły osadzone wewnątrz agentów oraz współdzielone warstwy zasad.
- **Przejrzystość.** Dzienniki i zarejestrowane ślady rozumowania umożliwiają przeprowadzanie audytów i kontroli regulacyjnych.

- **Adaptowalność.** Agenty mogą ewoluować niezależnie od siebie — np. agent do oceny ryzyka może zostać zastąpiony nowszym LLM-em.

Korzyści te pokazują, w jaki sposób wewnętrzna struktura agenta pomaga mu działać inteligentnie i w sposób odporny na zakłócenie. Jednak inteligencja nie wynika wyłącznie z wewnętrznego projektu agenta — jest krytycznie zależna od bogactwa i trafności danych, do których agent ma dostęp w swoim środowisku. W kolejnym podrozdziale przyjrzymy się różnym magazynom danych i źródłom kontekstu, które pozwalają agentom skutecznie postrzegać otoczenie, rozumować i podejmować działania.

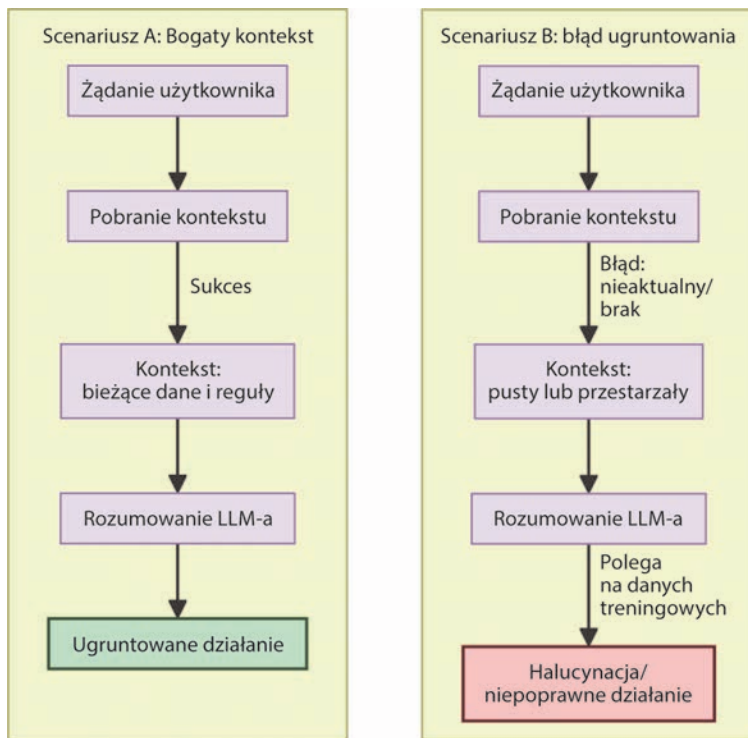
Magazyny danych i kontekst środowiskowy dla agentów

Jak podkreślaliśmy w rozdziale 1., *kontekst jest królem*, a zasada ta jest szczególnie istotna w przypadku systemów agentowych, które muszą podejmować świadome decyzje i wykonywać odpowiednie działania. Aby skutecznie postrzegać, wnioskować i działać, agenty polegają na różnych źródłach danych oraz informacjach kontekstowych ze swojego środowiska.

Tę krytyczną dynamikę zilustrowaliśmy na rysunku 4.3. Zestawia on poprawny, ugruntowany przepływ pracy — w którym bogaty kontekst umożliwia trafne wnioskowanie — ze scenariuszami awarii, w których nieaktualna pamięć lub nieudane pobranie danych prowadzą do błędnych działań.

Kontekst środowiskowy agenta można ogólnie podzielić na następujące kategorie:

- **Cyfrowy kontekst biznesowy.** Obejmuje wszystkie istotne cyfrowe źródła danych, z którymi agent może wchodzić w interakcję lub z których może czerpać informacje w środowisku korporacyjnym bądź online. Do kluczowych typów należą:
 - **Dane nieustrukturyzowane.** Ogromne ilości informacji zawartych w dokumentach tekstowych (raporty, e-maile, artykuły), obrazach, plikach audio i wideo. LLM-y są szczególnie sprawne w przetwarzaniu i rozumieniu nieustrukturyzowanego tekstu.
 - **Bazy wektorowe.** Te wyspecjalizowane bazy danych mają kluczowe znaczenie dla wydajnego wyszukiwania podobieństw opartych na osadzeniach. Gdy agent musi znaleźć informacje semantycznie zbliżone do zapytania, bazy wektorowe umożliwiają szybkie wyszukiwanie na podstawie znaczenia, a nie tylko słów kluczowych. Jest to podstawowy komponent wielu systemów RAG. Implementacje dzielą się zazwyczaj na dwie kategorie: biblioteki i bazy otwartoźródłowe (np. FAISS, Weaviate, Chroma) zapewniające lokalną kontrolę oraz umożliwiające samodzielny hosting i komercyjne usługi zarządzane (np. Pinecone, Google Vertex AI Vector Search) oferujące skalowalność i łatwość zarządzania.
 - **Dane ustrukturyzowane.** Uporządkowane dane, które zazwyczaj znajdują się w relacyjnych bazach danych lub arkuszach kalkulacyjnych. Dostarczają precyzyjnie zdefiniowane punkty danych, które agenty mogą odczytywać



Rysunek 4.3. Kontekstowe dane wejściowe a błędy ugruntowania

w ramach konkretnych zadań, takich jak pobieranie rekordów klientów czy informacji o produktach.

- **Grafy wiedzy.** Reprezentują informacje jako sieć encji i relacji między nimi, zapewniając ustrukturyzowane, semantyczne rozumienie danej dziedziny. Grafy wiedzy są szczególnie przydatne dla agentów, które muszą przeprowadzać złożone wnioskowanie obejmujące wzajemnie powiązane koncepcje.
- **Fizyczny kontekst środowiskowy.** W przypadku agentów zaprojektowanych do interakcji ze światem rzeczywistym, takich jak roboty lub systemy IoT, obejmuje następujące elementy:
 - **Sensory.** Urządzenia takie jak kamery, mikrofony, czujniki temperatury i lokalizatory GPS dostarczają danych o fizycznym otoczeniu.
 - **Aktywatory.** Komponenty takie jak ramiona robotyczne, silniki lub przełączniki pozwalają agentowi wykonywać fizyczne działania lub manipulować obiektami w otoczeniu.

Efektywne agenty często muszą integrować informacje z wielu typów magazynów danych i syntezować informacje z różnorodnych kontekstów, aby budować kompleksowe rozumienie sytuacji.

Aby uczynić te koncepcje bardziej namacalnymi, rozważmy agenta do zarządzania łańcuchem dostaw, którego zadaniem jest monitorowanie oraz optymalizowanie zapasów i logistyki w czasie rzeczywistym. Agent ten musi przetwarzać informacje z różnych systemów cyfrowych i potencjalnie wchodzić w interakcję z fizycznymi komponentami magazynu, aby reagować na zakłócenia.

W tabeli 4.5 pokazaliśmy, w jaki sposób ten agent wchodziłby w interakcję z różnymi magazynami danych i kontekstami środowiskowymi podczas wykonywania swoich obowiązków.

Tabela 4.5. Przykłady magazynów danych i kontekstów środowiskowych

Kontekst/ magazyn danych	Przykład agenta do zarządzania łańcuchem dostaw
Cyfrowy kontekst biznesowy	To cyfrowy świat, w którym działa agent, obejmujący wszystkie dane operacyjne firmy.
Dane nieustrukturyzowane	Agent przetwarza manifesty wysyłkowe (pliki PDF), maile z powiadomieniami o opóźnieniach dostaw oraz artykuły prasowe o strajkach portowych lub zdarzeniach pogodowych, aby zrozumieć potencjalne zakłócenia.
Bazy wektorowe	Gdy wykryte zostanie nowe zdarzenie geopolityczne, agent wykorzystuje bazę wektorową do przeprowadzenia wyszukiwania semantycznego wewnętrznych opracowań na temat tego, jak podobne wydarzenia w przeszłości wpływały na szlaki żeglugowe, i pobiera najbardziej istotny kontekst historyczny.
Dane ustrukturyzowane	Agent odpytuje relacyjną bazę danych, aby uzyskać aktualne poziomy zapasów dla konkretnych identyfikatorów SKU, sprawdza oczekiwane daty dostaw i pobiera informacje o pojemności magazynu.
Grafy wiedzy	Agent wykorzystuje graf wiedzy do zrozumienia złożonych, wielopoziomowych relacji między dostawcami, zakładami produkcyjnymi, centrami dystrybucji i końcowymi punktami sprzedaży detalicznej. Pozwala mu to wywnioskować, że opóźnienie komponentów od jednego dostawcy wpłynie na trzy konkretne linie produktowe.
Fizyczny kontekst środowiskowy	Obejmuje elementy świata rzeczywistego, które agent może wykrywać i na które może oddziaływać, często za pośrednictwem urządzeń IoT.
Sensory	Agent otrzymuje ciągły strumień danych z lokalizatorów GPS na ciężarówkach dostawczych, czujników temperatury w kontenerach chłodniczych zapewniających integralność produktów oraz skanerów RFID na rampach załadunkowych magazynu potwierdzających przyjęcie towaru.
Aktywatory	Jeśli agent przewiduje poważne zakłócenie na głównej trasie, może automatycznie uruchomić aktywator w inteligentnym magazynie, taki jak taśmociąg lub ramię robotyczne, aby przekierować odpowiednie palety na inną rampę załadunkową przeznaczoną dla alternatywnej trasy wysyłkowej.

Ten przykład łańcucha dostaw pokazuje, że skuteczny agent musi płynnie integrować informacje z wielu różnorodnych źródeł — od ustrukturyzowanych baz danych magazynowych, przez nieustrukturyzowane kanały informacyjne, po dane z czujników w rzeczywistym świecie. To właśnie zdolność postrzegania i wnioskowania w tych różnych kontekstach umożliwia mu inteligentne i aktywne działanie.

Zbadawszy wewnętrzną anatomię agenta oraz zewnętrznych danych, na których się opiera, przyjrzymy się teraz cechom architektonicznym, które wynikają z zastosowania tych komponentów, a także wzajemnym interakcjom między agentami.

Modele interakcji agentów i ich kluczowe cechy

Omówione wcześniej komponenty architektoniczne i świadomość kontekstowa dają początek szeregowi użytecznych cech, które są nieodłączną częścią dobrze zaprojektowanych systemów agentowych. Cechy te określają sposób działania agentów, możliwości ich strukturyzacji oraz ich interakcje zarówno między sobą, jak i z otoczeniem. Zrozumienie tych właściwości pozwala w pełni docenić zalety podejścia agentowego.

Aby szczegółowo przyjrzeć się tym zaletom, w niniejszym podrozdziale omówimy dwa kluczowe obszary. Zaczniemy od analizy „wrodzonych” cech architektonicznych wynikających z samego projektu agenta, takich jak modularność, skalowalność i adaptacyjność. Następnie przejdziemy do praktycznych modeli interakcji między agentami — zbadamy, w jaki sposób agenci komunikują się bezpośrednio i pośrednio, a także przedstawimy wyłaniający się stos technologiczny, który umożliwia tę współpracę.

Cechy architektoniczne

Jak już wcześniej pokrótce wspomnieliśmy, dobrze zaprojektowana architektura agentowa daje początek kilku użytecznym cechom, które przyczyniają się do jej elastyczności, odporności i inteligencji. Aby uniknąć powtórzeń, w tym punkcie nie będziemy ponownie definiować tych pojęć, lecz pokażemy, jak przekładają się one na praktyczne korzyści, na przykładzie systemu agentowego do zarządzania łańcuchem dostaw.

W tabeli 4.6 pokazaliśmy, jak każda kluczowa cecha architektoniczna znajduje zastosowanie w praktyce.

Tabela ta pokazuje, jak te „wrodzone” cechy przejawiają się w praktycznym zastosowaniu. Przejdźmy teraz od tych ogólnych właściwości do konkretnych modeli, które określają sposób interakcji agentów między sobą.

Modele interakcji między agentami

Gdy wiele agentów działa w tym samym środowisku, sposób ich wzajemnej interakcji staje się kluczowym aspektem projektowania systemu. Od wybranego modelu zależy, jak agenci koordynują swoje działania, dzielą się informacjami i realizują wspólne cele.

Aby umożliwić tę koordynację, systemy agentowe zazwyczaj wykorzystują dwa podstawowe modele interakcji (rysunek 4.4), które różnią się sposobem wymiany informacji między agentami:

Tabela 4.6. Cechy architektoniczne agenta do zarządzania łańcuchem dostaw

Cecha architektoniczna	Przykład w systemie zarządzania łańcuchem dostaw
Modularność	System łańcucha dostaw wymaga wdrożenia skuteczniejszego wykrywania oszustw w fakturach wysyłkowych. Zamiast przebudowywać istniejącego agenta InventoryAgent, tworzy się nowego, wyspecjalizowanego agenta InvoiceFraudAgent, który zostaje dodany do przepływu pracy. Jeśli model wykrywania oszustw wymaga aktualizacji, dotyczy to wyłącznie tego konkretnego agenta.
Skalowalność	W szczycie sezonu świątecznego system obsługuje ogromny wzrost ilości danych GPS z ciężarówek dostawczych, dynamicznie uruchamiając wiele równoległych instancji agenta LogisticsAgent. Moduł równoważenia obciążenia rozdziela zadania optymalizacji tras pomiędzy te instancje, zapewniając szybkie reakcje systemu.
Adaptowalność	SupplierCommsAgent zauważa, że e-maile od Dostawcy X zawierające frazę „opóźnienie produkcji” konsekwentnie poprzedzają krytyczne braki magazynowe. Wykorzystując tę wyuczoną korelację, agent dostosowuje swoje zachowanie — automatycznie uznaje wszystkie przyszłe wiadomości z tymi słowami kluczowymi za „pilne” i natychmiast powiadamia agenta InventoryAgent.
Interakcja multimodalna	W magazynie agent ReceivingDockAgent najpierw przetwarza tekstowy manifest wysyłkowy (PDF), a następnie za pomocą kamery wizualnie sprawdza otrzymaną paletę pod kątem uszkodzonych kartonów (dane obrazowe). Agent działa na podstawie tych połączonych informacji, aktualizując system magazynowy tylko wtedy, gdy inspekcja wizualna potwierdza zgodność z opisem w manifeście.
Współpraca	DisruptionMonitoringAgent wykrywa zamknięcie portu na podstawie kanału informacyjnego i aktualizuje status we współdzielonej bazie danych. InventoryAgent wykrywa tę zmianę i wysyła bezpośrednią wiadomość do LogisticsAgent z prośbą o alternatywne trasy wysyłki, inicjując wspólne rozwiązywanie problemu przez oba agenty.

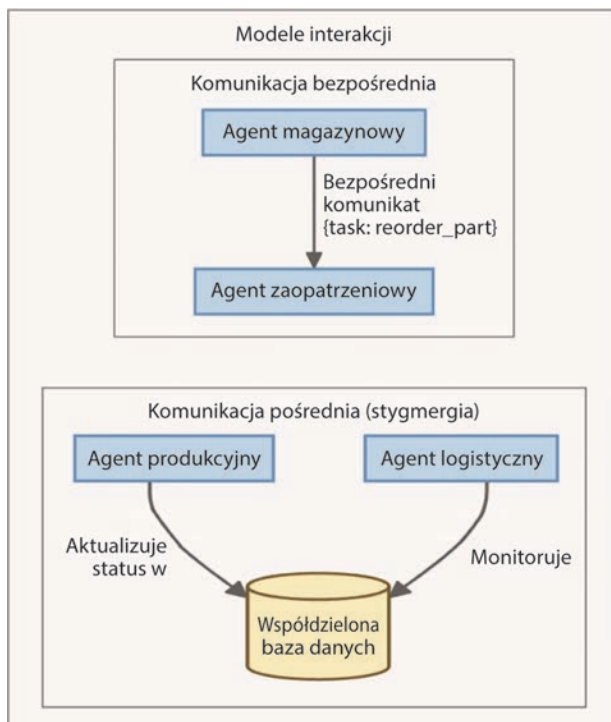
- **Komunikacja bezpośrednia.** W tym modelu agenty jawnie wysyłają do siebie wiadomości, korzystając ze wspólnego języka i protokołu.

Przykład. InventoryAgent wykrywa niski stan magazynowy i wysyła komunikat {"task": "reorder_part", "part_id": "XYZ-123", "quantity": 500} do agenta ProcurementAgent.

- **Komunikacja pośrednia (stygmergia).** Agenty mogą również wchodzić w interakcje pośrednio, obserwując i modyfikując współdzielone środowisko, takie jak baza danych.

Przykład. ManufacturingAgent aktualizuje rekord we współdzielonej bazie danych, ustawiając {"status": "complete"}. LogisticsAgent, który monitoruje tę bazę, wykrywa zmianę statusu i inicjuje proces wysyłki.

Choć modele te opisują koncepcyjne podejście do komunikacji, ich praktyczna implementacja opiera się na rozwijającym się stosie technologicznym (rysunek 4.5). Jak wspomnieliśmy wcześniej, stos ten składa się z komplementarnych warstw umożliwiających



Rysunek 4.4. Modele interakcji między agentami

różne formy interakcji. Wróćmy teraz do tych warstw, aby szczegółowo opisać, w jaki sposób ułatwiają one praktyczną komunikację agentów z narzędziami i innymi agentami:

- **Warstwa 1: wywoływanie funkcji.** To podstawowa warstwa interakcji, w której LLM agenta wywołuje lokalne narzędzie. Pozwala ona modelowi inteligentnie określić, jakiego narzędzia użyć, kiedy je wywołać i z jakimi parametrami — wszystko to w ramach jednego środowiska uruchomieniowego aplikacji. Był to pierwszy istotny krok w kierunku uczynienia z LLM-ów czegoś więcej niż generatorów tekstu poprzez umożliwienie im podejmowania działań. Głównym ograniczeniem jest to, że deweloper odpowiada za hosting, uruchamianie i zabezpieczanie narzędzia w tym samym środowisku.

Przykład. LLM agenta `LogisticsAgent` stwierdza, że musi znaleźć najbardziej efektywną trasę dostawy i generuje wywołanie swojej wewnętrznej funkcji `calculate_optimal_route()`.

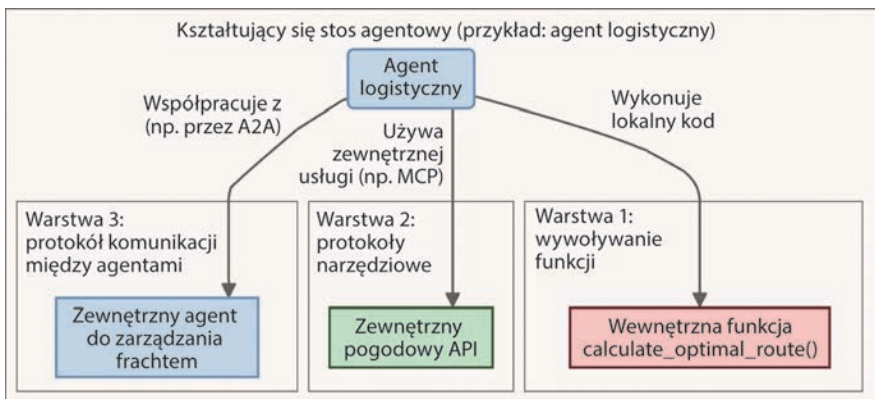
- **Warstwa 2: protokoły narzędziowe.** Zapewnia ustandaryzowany sposób odkrywania i używania zewnętrznych narzędzi jako zgodnych operacyjnie usług. Protokoły takie jak **Model Context Protocol (MCP)** od firmy Anthropic oddzielają hosting i wykonywanie narzędzia od samego agenta. Stanowi to ważny krok ewolucyjny w porównaniu z rozwiązaniami specyficznymi dla platform, takimi jak `ToolExecutor` w `LangChain` czy routery `LangGraph`. Podczas gdy te narzędzia otwartoźródłowe zarządzają wykonaniem w ramach środowiska

uruchomieniowego konkretnej aplikacji (często wymagając współdzielonych zależności), protokoły takie jak MCP pozwalają definiować narzędzia za pomocą schematów i hostować je niezależnie. Dzięki temu dowolny kompatybilny system może je odkrywać i wywoływać ponad granicami procesów czy sieci, skutecznie zamieniając narzędzia w przenośne usługi — podobnie jak REST API.

Przykład. Aby uwzględnić warunki pogodowe, `LogisticsAgent` wykorzystuje protokół narzędziowy do odkrycia i połączenia się z zewnętrznym API serwisu pogodowego, pobierając w czasie rzeczywistym dane o burzach i uwzględniając je w obliczeniach trasy — bez ścisłego powiązania z konkretną implementacją tego API.

- **Warstwa 3: protokoły komunikacji między agentami.** To najwyższy poziom, skupiony na współpracy między niezależnymi agentami, które mogą działać na różnych platformach lub w różnych organizacjach. Choć istnieją własnościowe lub specyficzne dla danej platformy metody delegowania zadań, protokoły takie jak **agent-to-agent (A2A)** koncentrują się na uniwersalnych standardach współpracy agentów. A2A zapewnia standard delegowania ustrukturyzowanych zadań, zarządza asynchronicznymi przepływami pracy oraz umożliwia złożoną koordynację wielu agentów — pełniąc w praktyce rolę uniwersalnego tłumacza, swojego „SMTP dla agentów AI”.

Przykład. Po obliczeniu trasy `LogisticsAgent` musi zarezerwować fracht morski. Wykorzystuje protokół A2A, aby wysłać ustrukturyzowane zadanie do zupełnie odrębnego agenta `FreightForwarderAgent`, który jest udostępniany przez zewnętrzną firmę logistyczną.



Rysunek 4.5. Kształtujący się stos agentowy

Wiele zaawansowanych systemów stosuje podejście hybrydowe, wykorzystując wywoływanie funkcji do akcji wewnętrznych, a protokoły wyższego poziomu do współpracy z systemami zewnętrznymi.

Aspekty techniczne architektur agentowych

Budowanie solidnych i skutecznych systemów agentowych, takich jak omawiane w tym rozdziale, jest zadaniem wymagającym technicznie. Jak wskazaliśmy wcześniej w ogólnym przeglądzie wyzwań produkcyjnych, pomyślne wdrożenie takich agentów wymaga pokonania kilku kluczowych przeszkód technicznych.

Zamiast powtarzać te wyzwania, w tej sekcji powiążemy je bezpośrednio z anatomią agenta, którą właśnie szczegółowo omówiliśmy. W tabeli 4.7 zostały odwzorowane kluczowe zagadnienia techniczne na konkretne komponenty architektoniczne, na które mają one największy wpływ, aby jaśniej pokazać, gdzie te wyzwania ujawniają się w projekcie systemu.

Tabela 4.7. Odwzorowanie wyzwań technicznych na komponenty agenta

Aspekt techniczny	Główne komponenty architektoniczne, których dotyczy
Przetwarzanie i integracja danych	Percepcja, pamięć
Reprezentacja wiedzy	Rozumowanie, pamięć
Integracja i orkiestracja LLM-mów	Rozumowanie, planowanie, koordynacja
Niezawodne mechanizmy użycia narzędzi	Działanie
Zarządzanie stanem i pamięcią	Pamięć
Skalowalność populacji agentów	Koordynacja, ogólna architektura systemu
Efektywność komunikacji między agentami	Koordynacja
Bezpieczeństwo i nadzór	Rozumowanie (wstrzykiwanie promptów), działanie (sandboxing), pamięć (prywatność), koordynacja (AuthN/AuthZ)

Skuteczne rozwiązywanie tych problemów technicznych to proces stopniowego doskonalenia poszczególnych zdolności — ścieżka, która odpowiada etapom modelu dojrzałości.

Przedstawiona analiza anatomii agentów, ich zależności od danych oraz modeli interakcji stanowi fundament do zrozumienia bardziej złożonych zachowań agentowych. W kolejnych rozdziałach będziemy budować na tym fundamencie architektonicznym, badając konkretne wzorce projektowe, które wykorzystują te komponenty do rozwiązywania typowych wyzwań związanych z tworzeniem zaawansowanych systemów AI.

Podsumowanie

W tym rozdziale wyszliśmy od fundamentalnych koncepcji omówionych w części I i przedstawiliśmy szczegółowy plan architektoniczny systemów agentowych opartych na LLM-ach. Przeszliśmy od teorii do struktury, definiując, co sprawia, że system jest „agentowy”, analizując kluczowe komponenty umożliwiające inteligentne zachowanie oraz odwzorowując wyzwania techniczne bezpośrednio na nowy model architektoniczny.

Najważniejsze wnioski:

- **Agenty to coś więcej niż LLM-y.** Agent AI to kompletny system definiowany przez swoją autonomię, orientację na cel oraz zdolność do postrzegania, rozumowania i działania. Wykorzystuje LLM jako swój rdzeń poznawczy, ale różni się zasadniczo od prostej, bezstanowej interakcji z modelem językowym.
- **Anatomia agenta.** Opisaliśmy podstawowe elementy konstrukcyjne agenta — *cele, percepcję, rozumowanie, planowanie, działanie, pamięć i koordynację* — przez pryzmat ich ról architektonicznych w ramach ciągłej pętli operacyjnej, pokazując, jak agent przechodzi od postrzegania do działania ukierunkowanego na cel.
- **Kontekst jest kluczowy dla inteligencji.** Skuteczność agenta zależy w krytycznym stopniu od jego środowiska. Musi on czerpać z bogatego kontekstu cyfrowych źródeł danych (takich jak bazy danych i grafy wiedzy) oraz fizycznych źródeł danych (takich jak czujniki), aby podejmować trafne decyzje.
- **Interakcje definiują system.** Systemy agentowe charakteryzują się takimi cechami jak modularność i skalowalność. Agenty komunikują się za pomocą bezpośrednich i pośrednich modeli komunikacji, które są coraz częściej wspierane przez praktyczny stos protokołów do wywoływania funkcji, korzystania z narzędzi oraz współpracy między agentami.
- **Architektura jest fundamentem wzorców.** Zrozumienie architektury bazowej przedstawionej w tym rozdziale stanowi niezbędny warunek do projektowania i implementowania konkretnych, powtarzalnych wzorców agentowych rozwiązujących realne problemy biznesowe, które będziemy eksplorować w dalszej części książki.

W tym rozdziale ustaliliśmy plan architektoniczny dla pojedynczych agentów. Możemy teraz zbadać, jak te agenty mogą być organizowane w wydajne, współpracujące systemy. W następnym rozdziale przybliżymy pierwszy zestaw kluczowych wzorców projektowych: wzorce koordynacji wielu agentów.

Wzorce te dostarczają strategii i rozwiązań potrzebnych do zarządzania interakcjami, współdzieleniem informacji oraz współpracą między agentami w celu sprostania wyzwaniom wykraczającym poza możliwości pojedynczego agenta.

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

Buduj agenty AI, które działają — nie tylko rozmawiają

Organizacje na całym świecie poszukują sprawdzonych metod, by wyjść poza fazę eksperymentów i wdrożyć agentowe rozwiązania na skalę produkcyjną. Właśnie w tym miejscu pojawia się ta książka — jako inżynierski przewodnik po budowie inteligentnych, wieloagentowych systemów AI gotowych na wymagania środowisk enterprise.

Publikacja prowadzi czytelnika od modelu dojrzałości GenAI, przez dobór i adaptację modeli LLM, aż po zaawansowane wzorce architektoniczne dla systemów wieloagentowych. Omówiono tu koordynację agentów, wyjaśnialność decyzji, odporność na awarie i interakcje człowiek – agent. Szczegółowo opisano hierarchiczną architekturę wieloagentową z orkiestratorami i wyspecjalizowanymi podagentami, protokoły MCP i A2A, a także praktyczne wdrożenia z użyciem Google ADK, LangGraph i CrewAI. Całość uzupełniają kompletne przypadki użycia systemów.

W książce między innymi:

- Model dojrzałości GenAI i ścieżka od RAG do autonomicznych systemów wieloagentowych
- Wzorce koordynacji, odporności i wyjaśnialności w architekturze agentowej
- Protokoły komunikacji między agentami MCP i A2A
- Hierarchiczne architektury wieloagentowe z orkiestratorami i podagentami
- Praktyczne wdrożenia z Google ADK, LangGraph i CrewAI
- AgentOps, LLMOps i zarządzanie cyklem życia systemów agentowych

Dr Ali Arsanjani — dyrektor do spraw inżynierii stosowanej AI w Google Cloud, twórca centrum doskonałości GenAI Blackbelts, wcześniej szef ML w AWS i dyrektor techniczny w IBM. Współtwórca protokołu A2A, wykładowca akademicki, autor ponad 4800 cytowanych publikacji naukowych.

Juan Pablo Bustos — strateg AI w Google, specjalista do spraw wdrożeń agentowej AI dla korporacji z listy Fortune 50, ekspert z doświadczeniem w Google, Stripe i AWS.

Arsanjani i Bustos stworzyli niezbędny przewodnik architektoniczny dla liderów, którzy chcą skutecznie poruszać się w tym złożonym obszarze i budować inteligentne systemy w sposób pewny, metodyczny i skalowalny.

— Thomas Kurian, CEO, Google Cloud

Helion 	KOD KORZYŚCI Sięgnij po więcej! 
 helion.pl  HELION S.A. ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	ISBN 978-83-289-4180-9  9 788328 941809
Cena: 149,00 zł	

<packt>