LG CNS

DevOn AI-Driven Development

Service Offering

LG CNS

Table of contents

전체 목차

Global AI 활용 Trend (1/5)

HOW MANY COMPANIES USE AI?

78% of global Companies use AI

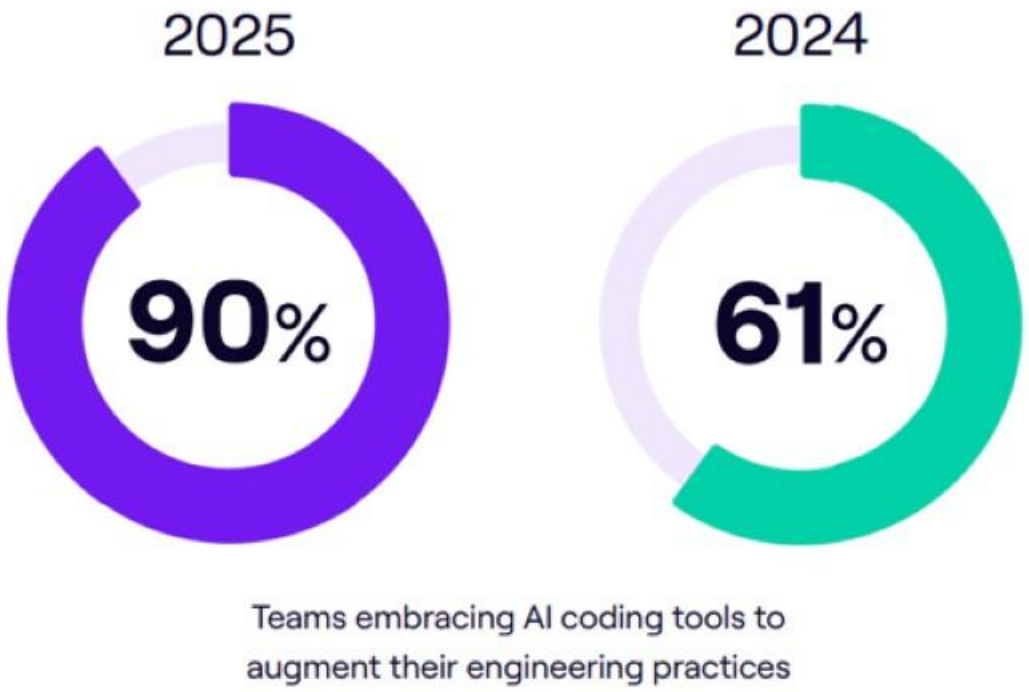
...as well as **99%** of Fortune 500 companies

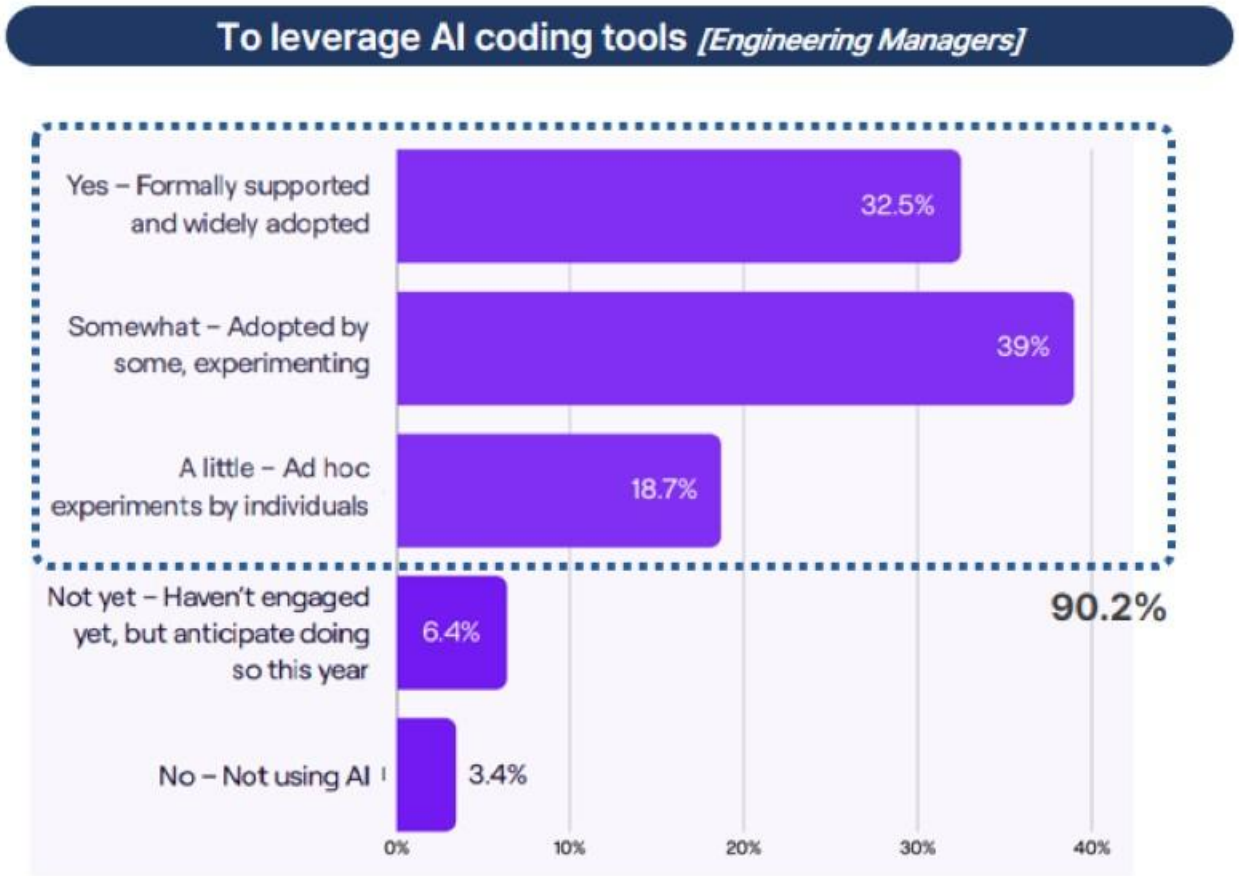
[source : designrush.com]

[source : Designrush]

Global AI 활용 Trend (2/5)

Engineering's AI Era



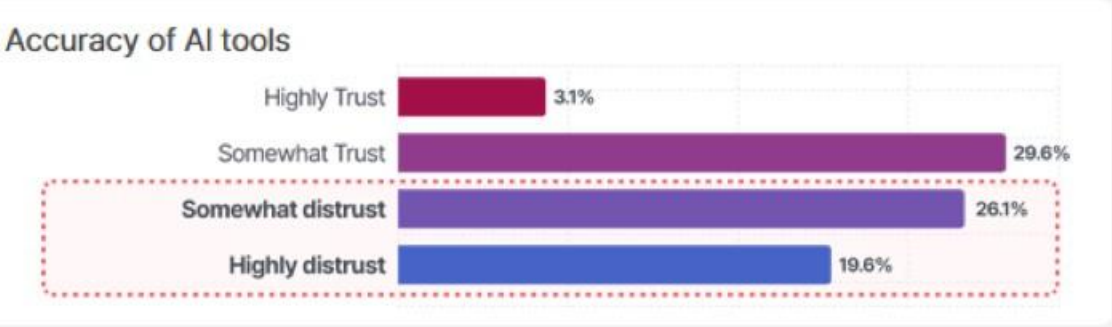


[source : Jellyfish]

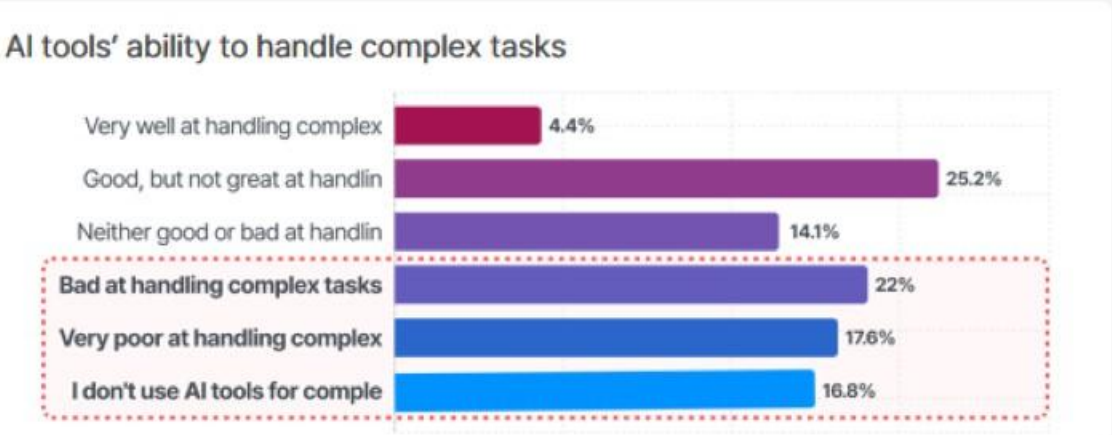
Global AI 활용 Trend (3/5) - AI활용 현황 설문 결과

AI 활용 현황

✓ **Trust-by-Design, 사람의 double Check (중복 검증) 필수**

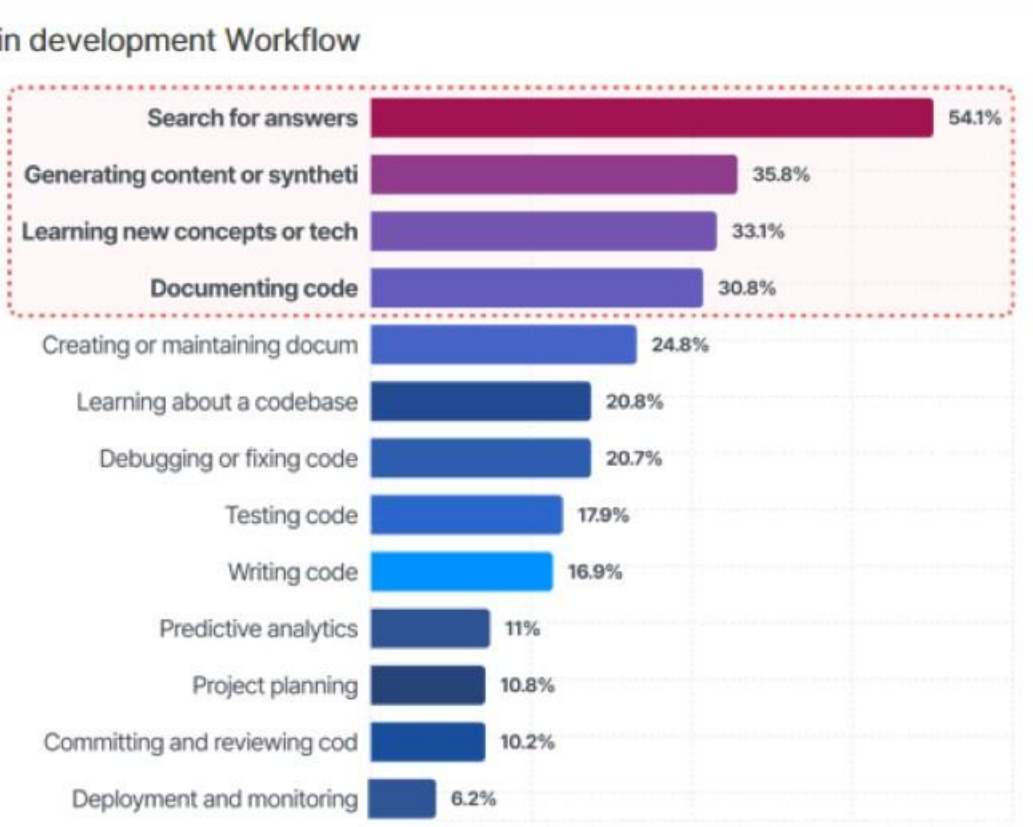


✓ **복잡성이 높은 업무는 활용도가 떨어짐**



AI 주 사용 Use Case

✓ **검색, 학습, 문서화,(mockup)데이터 생성**



[source : Stackoverflow Survey 2025]

Global AI 활용 Trend (4/5) - Agentic으로의 진화

단순한 기술의 발전을 넘어 코드 자동화를 위한 자율적 실행으로 진행 중

기술 발전 방향

AI Artificial Intelligence

인간의 지능을 모방하여
학습, 추론, 판단 등의
인지 능력을 구현한 기술

AX AI-Driven Development

AI 기술을 활용하여 업무 방식,
프로세스, 비즈니스 모델 혁신

ML/DL

패턴 학습을 통한 분류/예측수행

AlphaGo, Google translate

GenAI

텍스트 이미지 생성·요약 대화

ChatGPT, Gemini

AI Agent

목표 기반 계획 행동 검증/ 수행

Operator, Computer Use

Agentic AI

자율적 학습/ 개선 및 협업

1 Optimization

분석/예측 강화를 통한
반복/단순 업무의 자동화, 최적화

2 Generation

대규모 언어모델을 통한
자연어 기반 콘텐츠 생성

코드 생성 중심

3 Execution

레거시 시스템과의 연계를 통한 단위
업무의 자율 수행

AI와 함께 코드 개발

4 Autonomy

목표 중심의 전체 Workflow 설계/ 및
실행 체계로의 운영방식 재정의

Spec 중심

Global AI 활용 Trend (4/5) - Agentic으로의 진화

사례로 살펴보는 Assistant와 Agentic의 차이

Assistant

'SQL 생성 위해
적극적 인간 관여'

VS

Agentic

'SQL 생성 위해
적극적 인간 관여'

AI assistants VS. AI agents

Key difference

Global AI 활용 Trend (5/5) - Agentic으로의 진화

AI 에이전트의 두가지 핵심

리즈닝

'문제해결 위한 계획 수립'

스키마 분석 필요
스키마 분석 SQL 생성

SQL 실행
SQL 실행 결과 판단

연결

'외부자원 : 데이터, 앱, 서비스'

DevOn AI-Driven Development

DevOn AI-Driven Development는 AI 기술을 분석,설계, 개발, 단위테스트까지 전 개발 공정에 적용하여 이행 프로세스를 최적화 하는 LG CNS의 대표적인 개발 도구입니다

Point 1 Seamless
개발 공정 중단 없는 연결

"프로젝트 전체 공정을 지원하고
개발 플랫폼과 연계"

중단 없는 연결 및 업무 자동화

- 요구사항 기반 설계 자동 생성
- 설계 도구 기반 코드 자동 생성
- 개발 도구 기반 테스트 케이스 자동 생성

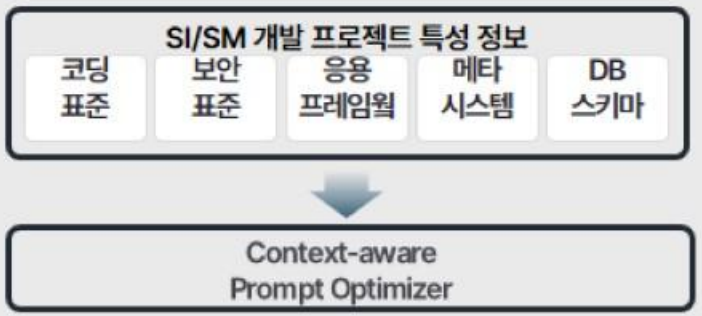
공정/품질 가시화

- 단계별 산출물 자동화
- Dashboard

Point 2 Optimization
SI/SM 특성 최적화

"SI/SM 프로젝트 특성에 최적화된
소스 코드를 생성"

SI/SM 개발 프로젝트 정보 자동 적용

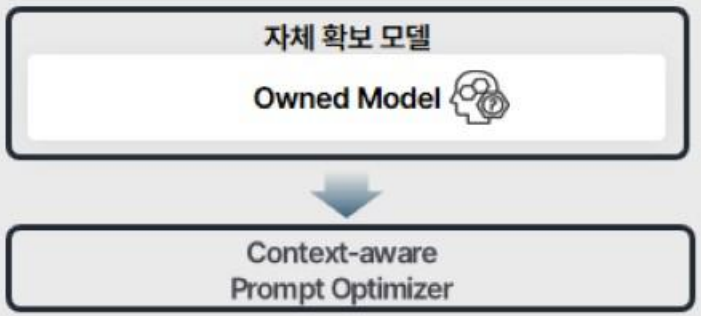


- 프로젝트 표준 및 코드 샘플 설정 템플릿 제공
- 코드 생성 시 템플릿 자동 참조 및 변환 기능 제공

Point 3 Safety
On-Premise환경 개발 지원

"고객사 인프라 환경 어디에서나
설치 가능한 AI"

자체 확보한 모델 기반 On-Premise 제공



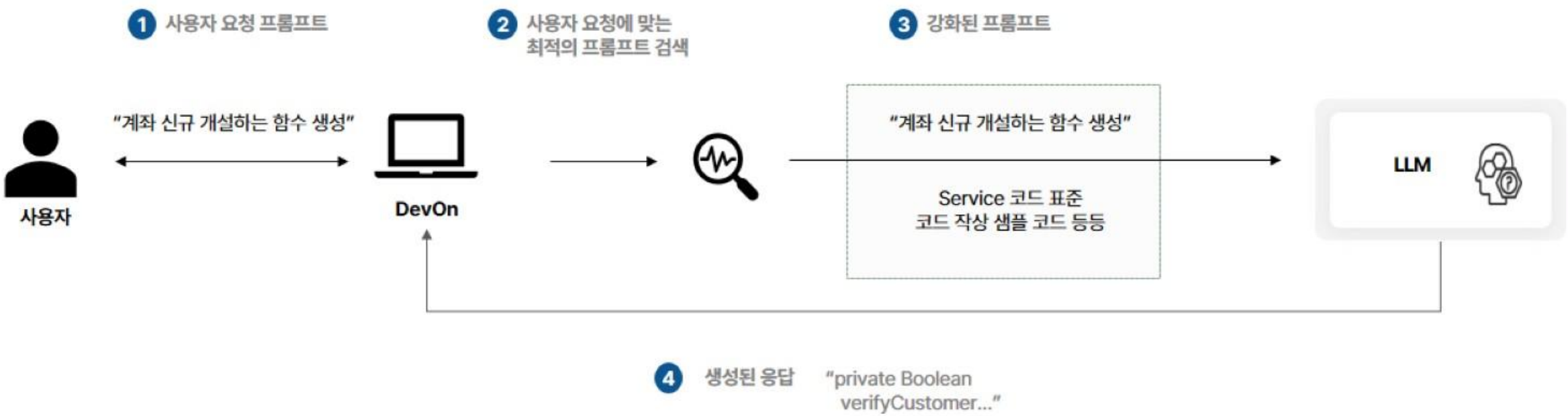
- 고객사 인프라 환경 어디에나 서비스 제공 가능
- 다양한 금융, 공공 프로젝트 Reference 보유

미래에셋생명, 새마을금고, KB Card, 한국자산관리공사, 하나은행 등 600여개 프로젝트에서 On-Premise 환경에서 적용 중

DevOn AI-Driven Development - ② Optimization

프로젝트 표준 및 코드 샘플 설정 템플릿 제공 + 코드 생성시 템플릿 자동 참조 = 마치 학습한 코드와 같은 효과를 낼 수 있습니다.

프롬프트를 활용한 프로젝트 특성 정보 자동 생성 프로세스



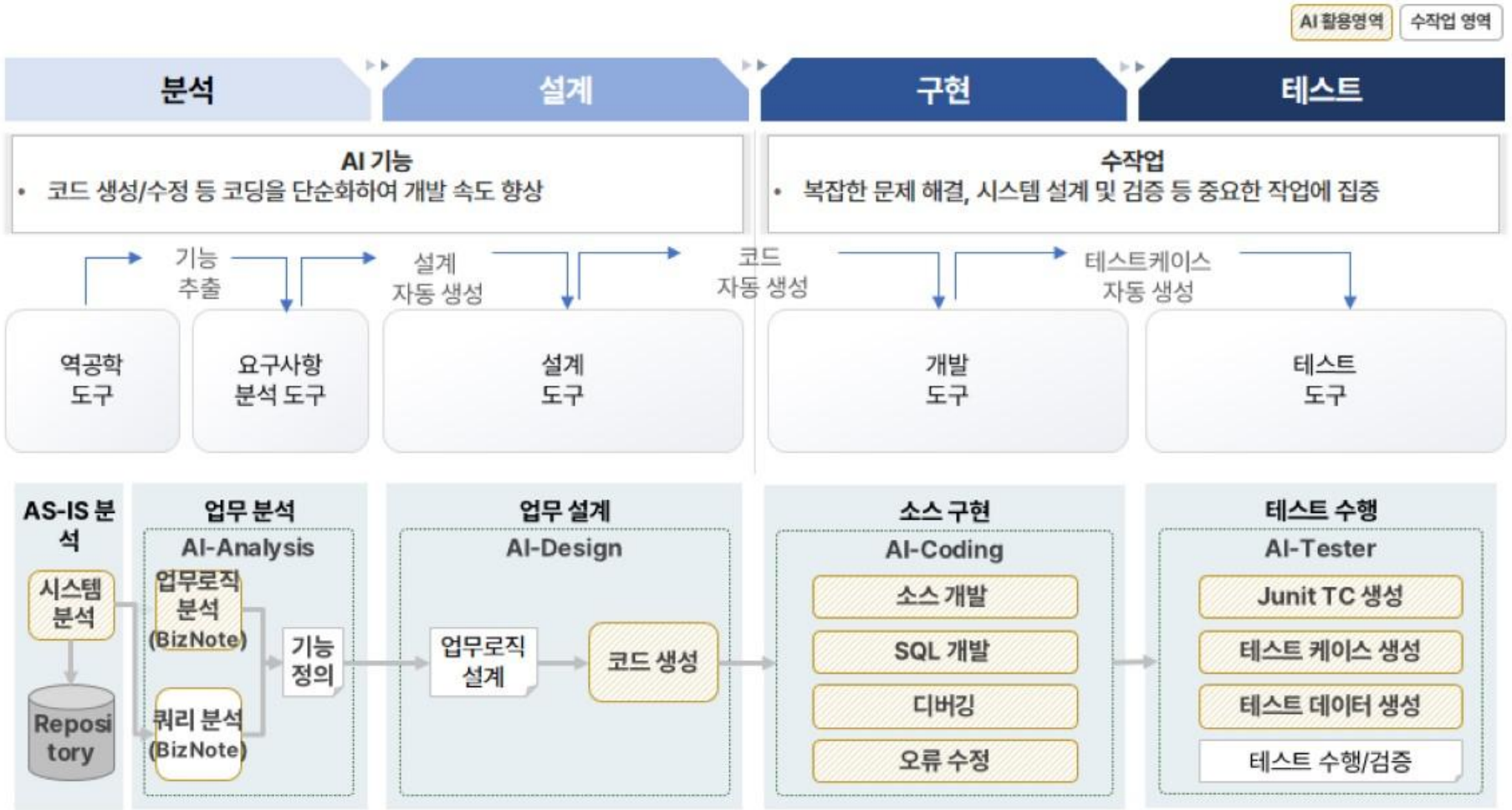
DevOn AI-Driven Development 절차 상세

AI 기술을 분석,설계, 개발, 단위테스트까지 전 개발 공정에 적용하여 이행 프로세스를 최적화하는 LG CNS의 개발 도구입니다

DevOn AI-Driven Development 특징

1. 전 개발 공정 중단 없는 작업 수행
2. 단계별 생성 자동화
3. 프로젝트 특성 자동 반영 및 생성
4. On-Premise 환경의 서비스 생성
5. 산출물 자동 생성

"전 개발 공정을 중단 없이(Seamless) 연결, 자동화(Automation), 가시화(Visualization)"



유첨. 분석/설계 Demo Sample (1/2)

기능 상세

구분	기능	내용
AI-Trans	블럭단위 코드변환	As-Is 소스 코드를 패턴화 하여 개발자가 선택한 부분을 To-Be 개발 표준에 적합한 코드로 변환
	단순 SQL 변환	오라클 SQL을 ANSI 표준 SQL로 변경
AI-Analysis	코드 분석	As-Is 코드에서 Biz Spec을 추출하여 As-Is 코드를 이해하기 쉬운 형태의 설명문으로 요약

유첨. 분석/설계 Demo Sample (2/2)

고려사항

AI-Analysis

- 개발자 변환 관리 강화
  - ✓ 업무 플로우 추출 시 용어사전·메타데이터 불일치 가능 → 도메인 전문가 검증 필요
  - ✓ 레거시 코드 주석·표준 미비 시 → LLM 분석 정확도 저하 → 추가 데이터 라벨링/정제 필요
- 분석 도구 연계성 강화
  - ✓ 코드 분석 전 리파지토리와 연동해 맥락 기반 분석 필요 ◀
- 분석 품질 보증
  - ✓ AI 분석 결과는 신규 개발자 학습 가이드 및 도메인 지식 축적 자료로 재활용 가능

AI-Trans

- 선도 개발(Pre-Development) 필요
  - ✓ 프로젝트 착수 시 To-Be 표준 구조·코드 템플릿 사전 제공 → 설계·개발 표준 정착
  - ✓ 신규 개발 모듈의 기준 구현(Reference Implementation) 제공
- 변환 품질·정합성 보증
  - ✓ Context Window 한계 극복 필요 → 블록 단위 변환 ◀
- 검증 체계 강화
  - ✓ 변환 결과물은 개발자 수동 검증 필수

기대효과

이해도 향상

- 분석·설계 자동화 : DevOn SD/Modeler 기반 **설계 산출물 자동 생성**으로 To-Be 설계 속도 향상
- 코드 이해 증진 : 타 언어 개발자도 AI 분석으로 레거시 코드 구조/로직 **신속 이해**

품질 제고

- 표준 준수 강화 : 기업 표준 가이드라인·산출물 규격에 맞춘 **일관성 있는 아키텍처 확보**
- 도메인 규칙·용어사전 기반 분석 품질 강화 : 표준화된 비즈니스 용어와 규칙을 AI 분석에 반영하여, **중복·모호성을 사전에 차단**

생산성 증대

- 반복 업무 자동화 : 설계 단계의 **반복 작업 자동화**를 통한 생산성 증대
- 오래된 언어, 프레임 워크 등 유지보수 비용이 크고 확장성이 떨어지는 **구형 IT 시스템 전환 용이**

유첨. 개발/테스트 Demo Sample (1/2)

기능 상세

구분	기능	내용
AI-Coding	주석 기반 코드 생성	사용자가 입력한 주석을 기반으로 등록, 조회, 수정, 삭제 등 패턴화 된 코드를 개발 표준에 맞게 생성
AI-Coding	다음 코드 추천	임의의 위치에서 다음으로 이어질 한 줄 또는 여러 줄의 코드를 개발 표준에 맞게 추천
AI-Coding	Comment Chat	소스 코드의 기존 로직을 수정하거나 새로운 내용을 추가
AI-Coding	SQL 생성	DB의 테이블 및 컬럼을 기반으로, 사용자 요구사항을 반영하여 쿼리문을 생성하고 실행까지 가능
AI-Coding	SQL 튜닝	쿼리 실행 계획을 기반으로 성능 상의 한계를 탐지하고 해결 방안을 제시
AI-Coding	Exception Helper	오류 로그를 기반으로 오류의 원인을 파악하고 개발자가 즉시 활용 가능한 해결책을 제시
AI-Coding	오류 자동 수정	오류를 파악하고 메서드 내 해당 오류를 AI 자동 수정
AI-Coding	Auto AI	단순한 기능을 One-Click으로 개발(코드/SQL 생성 > 테스트 > 품질 검증 및 오류 수정) 완료
AI-Coding	인스펙션	*Secure Coding, 프로젝트 표준 관점의 코드 인스펙션 결과 제공*
AI-Test	테스트 케이스 생성	소스 코드의 기능별로 실행 가능한 수준의 단위 테스트 케이스와 테스트 데이터를 자동 생성
AI-Play Chat		실시간으로 다양한 질의를 제출하고 AI가 즉시 응답하는 도구 내 채팅 인터페이스

유첨. 개발/테스트 Demo Sample (2/2)

고려사항

AI-Coding

- 업무 분석 자동화 한계 보완
  - ✓ 전 과정(요구사항 → 설계 → 개발 → 테스트)에 일관성 있게 적용
  - ✓ 신규 투입 개발자 및 외부 협력사 인력이 빠르게 적응할 수 있도록 개발 가이드에 제공
- 업무 연계 및 리소스 고려
  - ✓ SWA, AA 등 기존 업무와 병행 시 리소스 중복·과부하 발생 가능
  - ✓ 핵심 개발 인력은 본연의 과제에 집중 → 전담 지원 인력 별도 투입 필요

AI-Test

- Mock 데이터 활용 및 한계 보완
  - ✓ 초기 단위 테스트·예외 처리 검증에 Mock 데이터 활용 → 빠른 피드백과 반복 검증 가능
  - ✓ 공통 함수는 Mock 데이터 기반 단위 테스트로 충분히 증빙 가능
  - ✓ 외부 연계는 Mock 데이터로 초기 인터페이스 검증 후, 실제 연계 테스트로 단계 확장
- Mock → 실데이터 전환 로드맵
  - ✓ 1단계: Mock 데이터로 기능·구조 검증 (개발 효율성 극대화)
  - ✓ 2단계: 샘플링된 실데이터 기반 보완 테스트 (업무 규칙 반영)
  - ✓ 3단계: 실제 운영 데이터 기반 E2E 및 성능/부하 검증 (최종 품질 보증)

기대효과

생산성 향상

- 개발 속도 가속화 : 반복 작업 자동화로 **핵심 로직 개발에 집중**
- 워크플로우 최적화 : 개발 **리드타임 단축** 및 수정 시간 절감
- 개발자 **온보딩 단축**: 다양한 언어/프레임워크 자동 가이드·템플릿 제공

품질 제고

- 규제 대응 자동화 : 표준 기반 점검·검증 자동화로 **규제 준수** 용이
- 테스트 **품질 향상** : 테스트 케이스·데이터 자동 생성으로 체계적 검증 수행
- 안정성 강화 : 잠재적 오류·보안 취약점 **조기 탐지 및 예방**

리스크 감소

- 휴먼 에러 최소화 : 패턴화된 자동 코드 생성으로 **코드 편차 및 실수 감소**
- Mock 데이터 기반 초기 검증 후 → 실데이터 교차 검증으로 **리스크 조기 차단**
- 코드 리뷰 및 품질 검증 자동화를 통한 **일정 지연 가능성 최소화**

별첨. Use Case

Use Case 리스트 상세

구분	프로젝트 명	규모	환경	적용 내역
SI	서울OO보험	1225억	AOAI	• 약 23,000개 대량의 쿼리를 AI-Trans를 이용하여 변환 및 제공(Oracle SQL -> ANSI SQL)
	XX생명	1170억	On-Prem	• 분석/설계 단계 레거시 분석 및 개발단계 준비 중
	B카드 페이북	88억	On-Prem	• Struct F/W의 연계소스/로직 변환 검토 및 SQL 변환 부분에 활용
	K 선진예보 구축	63억	On-Prem	• 기존 개발 서버에 GPU를 추가하여 AI기능 사용
	하나은행 프로젝트 First	776억	싱글렉스	• 화면 및 코드 분석 중심
	하이케어솔루션 CSMS2.0	42억	싱글렉스	• 코드 및 SQL 변환, 코드 분석
	T Sales Portal2.0_LGEUS	15억	싱글렉스	• 개발 소스(1030개) DevOn F/W -> SpringBoot F/W 변환
	S증권 Project	862억	On-Prem	• Proframe C소스 블록 단위 부분 변환 및 SQL 변환 수행, 개발가이드 AI 활용 반영
	K 증권 정보계 차세대 구축	300억	On-Prem	• 마케팅 허브 업무를 JAVA로 신규 개발하는 프로젝트에 Coding AI 적용하여 진행
	E공제조합	182억	On-Prem	• 비즈니스 유형에 따라 빠르게 개발을 진행 할 수 있도록 적용
	K공사	192억	On-Prem	• 레거시 분석 정보 기반 TO-BE 모델 설계/개발 진행 중
SM	KB 국민카드	194억	On-Prem	• 코드 품질 및 코드 분석 / 영향도 파악 중심
	S 응용서비스 기본계약	30억	싱글렉스	• 코드 품질 및 코드 분석 / 영향도 파악 중심
계열사 SI/SM 프로젝트에 다수 적용 중			싱글렉스	• 600여개 계열사 프로젝트에 적용 중

별첨. XX생명 프로젝트 맞춤형 AI-Driven Development

적용 사례

- 분석/설계 단계 개발자들이 프롬프트를 만들고 BP 공유
  - ✓ AS-IS 소스코드 분석기능 : 변수, 한글명 추출기능, 쿼리추출기능, 조건이 있는 다이나믹 쿼리문 추출, 상품약관 추출 등
  - ✓ 산출물 작성 기능 : 소스코드에서 업무기능분해도 Activity추출, 화면정의서 생성, 인터페이스 설계서 생성 등
  - ✓ 품질체크 : As-Is화면목록 취합 및 사용여부 체크, 산출물 파일명/버전/개정이력 일괄작업 등
- 개발 단계 개발자들이 프롬프트를 만들고 BP 공유
  - ✓ MSA 적용 서비스 분리에 따른 소스코드 분리 및 API 구현
  - ✓ SP(PL/SQL)의 java 소스코드 변환
  - ✓ 화면 테스트 시나리오 작성

적용 효과

- 분석/설계 단계
  - ✓ As-Is 소스코드의 일괄 역공학 작업 없이 개발자들이 필요한 작업이 가능함
  - ✓ 소스코드로부터 산출물 작업을 수행함으로써 데이터의 최신화가 쉬움
- 개발 단계
  - ✓ 아키텍처가 Monolith 에서 MSA로 변경 시, 업무간 서비스 분리가 반영된 To Be 범위 (소스코드 목록) 파악 용이
  - ✓ Java소스코드 변환기능 이용으로, 로직을 살리면서 소스코드 형식 변환이 가능

별첨. XX생명 프로젝트 개발자 패스티벌

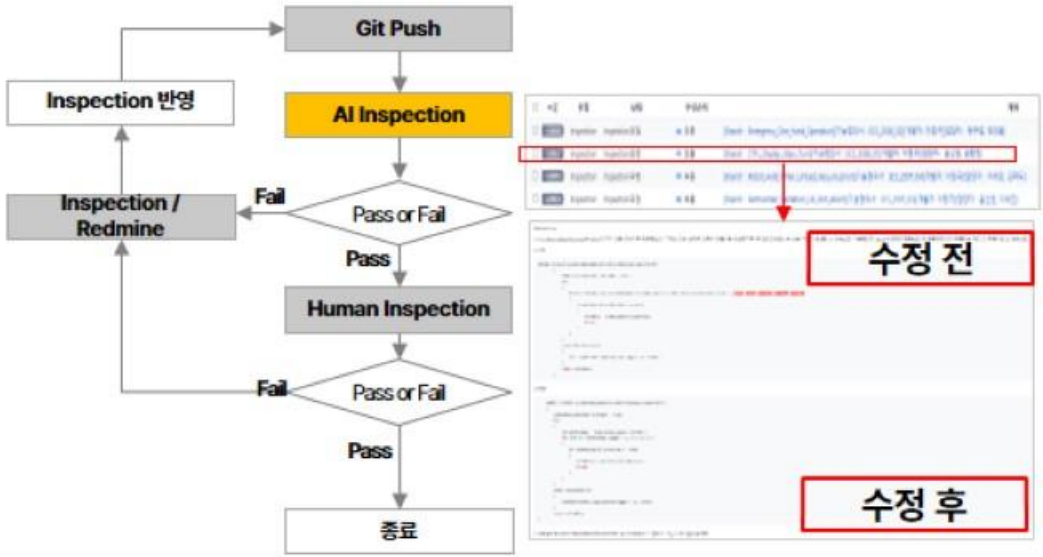
프로젝트 업무를 수행할때 AI Play를 이용하여 개발자들이 필요한 프롬프트를 만들고 WIKI 공유

구분	주요 업무	BP 사례	BP 공유자
분석/설계 단계 업무 수행	AS-IS 소스 분석	• BP0005 VO java 소스 내에서 변수 한글명 추출	김O근
		• BP0004 DQM 소스에서 쿼리 추출 후 쿼리원 활용	윤O주
	분석/설계 단계 산출물 작성	• BP0002 Java 소스 내에서 업무기능분해도 Activity 추출	강O구
		• BP0007 화면정의서 작성	강O구
	관리성 업무	• BP0009 AS-IS 화면목록 취합 및 사용여부 체크 자동화	문O성
		• BP0003 일자별 화면정의서 산출물 취합 자동화	문O성
		• BP0013 업무별 맞춤 엑셀 수식 생성	
개발 단계 준비	MSA 적용으로 서비스 분리에 대한 소스 분리, API 구현	• BP0012 MSA 전문가 AI-Driven Development	문O연
	SP 를 Java 소스로 변환	• BP0006 SP(PL/SQL) 을 Java 소스로 변환	신O빈
	배치 File 에서 도출된 레이아웃으로 향후 개발단계 파일을 검증	• BP0011 배치 인터페이스 파일 레이아웃 도출 및 생성파일 검증	신O현
	테스트 시나리오 작성	• BP0010 화면 테스트시나리오 작성	문O정

별첨. D SLA 생산설비제어 프로젝트 Code Inspection 활용사례

적용 사례

- 개별 공장으로 운영중인 시스템에 SI 프로젝트로 인해 변경된 소스코드의 주기적 유입으로 인해 운영 파악 및 소스코드 분석을 위해 Inspection을 강화하여 적용
  - ✓ 개발 유형에 따른 프롬프트를 정의/공유하여 정확한 답변 유도
  - ✓ 개발 공정에 AI Inspection 수행을 필수 공정으로 정의
  - ✓ AI Inspection 결과 및 수정 전/후 소스코드를 관리 Tool에 기록하고
  - ✓ 제3자 Inspection 회의 시 검토함



적용 효과

- Inspection 소요시간 단축 평균 1.5시간 -> 30분
- 결함 발견 수치 평균 60% -> 80%

별첨. K공사 프로젝트 맞춤형 AI-Driven Development 활용 사례

적용 사례

- 개발 Life Cycle에 다양한 기능 적재적소 적용

현행 분석 → 요구사항 정의

- As-Is 소스코드로 부터 Biz Spec 추출
- As-Is META 생성
- As-Is 화면 /소스코드 /SQL /VO분석

To-Be 기능정의 → 어플리케이션 설계

- As-Is 소스코드로 부터 LG CNS 설계 도구(SD) 모델 생성
- SQL 변환

Sprint #1 ~ N

- SQL 포맷 정련
- As-Is To-Be 매핑정의서를 기반으로 SQL To-be 변환
- 서버프로그램 테스트 케이스, 데이터 및 JUnit테스트코드 생성
- 화면(Jsp/js) 소스코드로부터 테스트케이스 및 데이터 생성
- 주석, 신규 메서드, 로그, 오류처리 등 프로젝트 표준이 반영된 소스코드 생성
- 소나큐브, PMD로 검출된 품질 이슈에 대해 AI가 개선코드 추천 사용

업무 효율 향상	3.7	반복 업무 단축 체감
업무 품질 향상	3.7	개발 품질 개선 우세
프로젝트 적용 유리	3.6	전반적으로 '유리'
타프로젝트 사용 의향	3.3	사용 의지 보통 이상

적용 효과

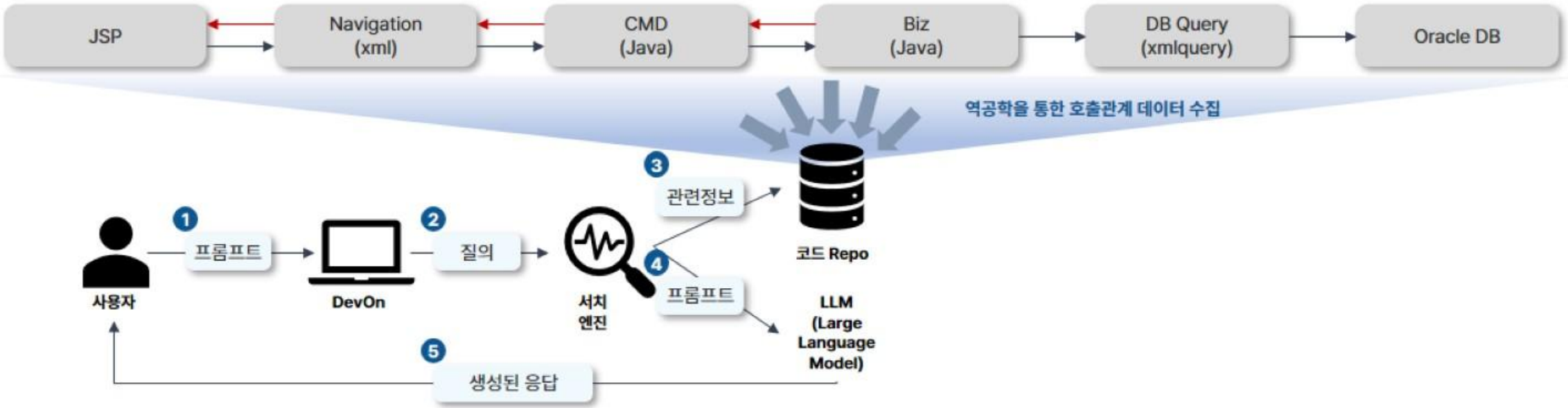
- 폐쇄망에서 휴대폰 검색 없이 개발 중 발생한 오류 및 수정 등 다양한 정보 취득
- As-Is 소스코드로 부터 설계모델 변환하여 To-Be 설계에 활용. SQL(196개) 자동 변환으로 설계단계 생산성 향상
- 표준화된 소스코드/Junit 코드 생성, SQL쿼리 변환, 오류 소스코드 개선코드 추천 등으로 개발 생산성 향상
- AI-Driven Development 활용도 및 만족도(5점 만점, 프로젝트 인원 설문)

항목	점수	시사점
업무 필수성	3.1	'필수' 로 정착
사용 빈도	상	매일 31%, 주 3-4회 28%
AI Play/Chat	3.6	'도움 된다' 우세
JUnit Test	3.3	테스트케이스 생성 등 단위 테스트시 도움
CoQ Fix	3.0	품질이슈 해결시 도움
표준 코드 생성	3.6	'도움 된다' 우세
SQL 변환	3.9	매우 큰 도움

별첨. SM AI Play와 연동한 역공학 데이터 활용 사례 I (1/2)

적용 사례

- 다양한 고객사와의 연동 및 고객별 특화 로직 적용으로 프로그램 변경에 따른 영향도 파악 요청이 많음. 유사도 기반 검색의 한계 타파
- 웹소스 호출 연관도 분석을 위한 역공학 모델을 적용하여 AI Play로 프로그램 로직 분석 질의에 대해 정확하게 호출관계까지 응대



적용 효과

- 정확한 영향도 분석 기반으로 처리 순서분석 가능

별첨. SM AI Play와 연동한 역공학 데이터 활용 사례 I (2/2)

적용 사례

You 2025-09-05

```
@ActionSummary /qn "/a/aa/aaa50Nav/AAA50RetrieveExcel.dev"
```

AI Play 20

[명령어 설명]
- /qn 옵션을 사용하여 **경로를 검색어로 사용합니다.**
- /action 옵션을 사용하여 **액션 경로를 검색 대상으로 사용합니다.**

[검색 대상]
- Action 호출관계에 따른 관련 Command 오퍼레이션
  - lgd.xxscs.a.aa.cmd.AAA50RetrieveExcelCmd.process

Action 분석을 진행합니다.

- **[01/01] '[sscs] lgd.xxscs.a.aa.cmd.AAA50RetrieveExcelCmd.process'**

[Action 요약]
1개의 Command 호출 내역과 Navigation 정보를 종합하여 Action 요약 정보를 생성합니다.

```
[Action 명]
  - /a/aa/aaa50Nav/AAA50RetrieveExcel.dev

[Action 흐름]
  <action name="AAA50RetrieveExcel">
    <command>lgd.xxscs.a.aa.cmd.AAA50RetrieveExcelCmd.process</command>
    <return name="FALSE">/a/aa/aaa50Nav/AAA50RetrievePage.dev</return>
    <return name="SUCCESS">/jsp/common/excel/excel_topLinePrint.jsp</return>
  </action>


[전체 호출 관계]
1.[sscs] lgd.xxscs.a.aa.cmd.AAA50RetrieveExcelCmd.process (command)
   1.1.[sscs] lgd.xxscs.a.aa.biz.AAA50Biz.retrievePOListToExcel (biz)
      1.1.1.[sscs] /a/aa/aaa50/retrievePOListToExcel2 (query)
         1.1.1.1.[TABLE]  R : SSCS_PO_INQUIRY_D, SSCS_PRICE_MASTER_SE_D
         1.1.1.2.[sscs] GET_CUSTOM_COMPANY_OF_CLIENTID (procedure)
            1.1.1.2.1.[TABLE]  R : XXSCS_COM_CODE_DTL
2.<return name="FALSE">/a/aa/aaa50Nav/AAA50RetrievePage.dev</return> (return)
3.<return name="SUCCESS">/jsp/common/excel/excel_topLinePrint.jsp</return> (return)

[목적]
  - 주문 정보를 조회하고, 조건에 따라 데이터를 필터링하여 엑셀 파일로 다운로드하는 action

[입력값]
  - supplierCode : 공급자 코드
  - customCompany : 사용자 지정 회사 코드
```

별첨. SM AI Play와 연동한 역공학 데이터 활용 사례 II (1/3)

AIMDD

적용 사례

로직 검색

MDD

로직 검색

```
#3.1.1.1.1.1.1 (ALICODE) 공카드 재고 상세 조회를 위한 승인요청 처리
#[S:_IrRuQBLGEfOa5o8vPq0fCg]
    [1승인요청처리모듈].[기관코드] = "001";

    //인수
    If ([1공카드재고상세조회].get([1업데이트건수]).[공카드입출유형코드1] == [카드공통Const].[인수])
    {
    [1승인요청처리모듈].[책임자승인사유코드] = [책임자승인사유코드Const].[카드인수취소];
    }
    //입고-33.카드사업팀자재입고
    else If ([1공카드재고상세조회].get([1업데이트건수]).[공카드입출사유코드1] == [카드공통Const].[카드사
    {
    [1승인요청처리모듈].[책임자승인사유코드] = [책임자승인사유코드Const].[카드사업팀자재입고취소];
    }
    //인도
    else If ([1공카드재고상세조회].get([1업데이트건수]).[공카드입출유형코드1] == [카드공통Const].[인도])
    {
    [1승인요청처리모듈].[책임자승인사유코드] = [책임자승인사유코드Const].[카드인도취소];
    }
```

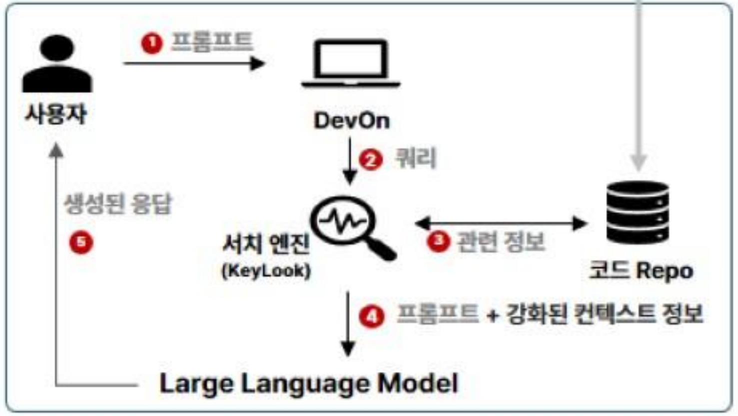
You

카드인도취소 사유로 책임자승인이 이루어지는 경우는?

Illustrative

AI Play

공카드입출유형1이 인도조건일 경우 카드인도 취소로 책임자 승인이 이루어 집니다
[출처] ***모델절대경로Path***



별첨. SM AI Play와 연동한 역공학 데이터 활용 사례 II (2/3)

AIMDD

적용 사례

RAG Pain Point

- 유사도 기반 검색의 한계로 정밀 검색 시 효율성 저하
- 검색 결과가 많을수록 정확도 감소
- 단순 질의 보다는 영향도 분석과 연계 검색 선호(운영조직)

대응 전략

- RDB 연동을 통한 정형 데이터 기반 정확도 강화
- 호출 관계 및 연관 정보 연계로 문맥 기반 검색 품질 향상
- 업무에서 사용 가능한 RAG 질의 BP 제공

해결 과제

- LLM의 Context 한도 내 **요약·압축** 설계
- 큰 Context 처리를 위한 Deep Search 기능 개발

STEP 1 목적 코드를 찾기 위해 유사 검색(RAG)
STEP 2 목적 코드를 찾은 다음 아래 3가지 정보를 AI Play에 질의
AI Play Multi-turn 추가 질의

호출관계

Illustrative

```
/*********기준정보관리/01.법인정보관리/번호상호조회Cpbi/[1]함수
    └──── /********* /05.인터페이스정의서/08.공통인터페이Ebi/[2]함수
    └──── /********* /03.공통서비스/02.공통인터페이Ibi/[3]함수
        └──********** /05.청구서비스Cpbi/[4]함수
    └──── /********* /05.인터페이스정의서/08.공통인터페이Cpbi/[5]함수
```

비즈니스로직(MDD논리코드)

Illustrative

```
[1]함수
   MDD논리코드정보가 제공됩니다

---------- 이후 호출관계에 참여하는 서비스의 논리코드가 함께 제공됩니다(중복 제거)
[2]함수 / [3]함수 / [4]함수 / [5]함수
```

비즈니스로직(SQL)

별첨. SM AI Play와 연동한 역공학 데이터 활용 사례 II (3/3)

AIMDD

적용 사례

로직 검색

You

["고객상태구분코드"]가 "3"인 경우 발생시키는 오류 code는?

Illustrative

AI Play

["고객상태구분코드"]가 "3"인 경우 발생시키는 오류 코드는 다음과 같습니다:

분석을 요청하는 거래코드에 없는 코드를 호출관계 분석을 통해서 통합 검색 결과 제공

1. CI연계정보고객번호상호조회(Cpbi::CI연계정보고객번호상호조회):
 - [CI연계정보고객번호상호조회].[고객상태구분코드]가 "3" 또는 "5"인 경우:
  - code="A4600533"
  - tcode="TGAAZ644"

2. 법적생년월일산출(Cpbi::법적생년월일산출):
 - [s고객상태구분코드]가 "3"인 경우:
  - [법적생년월일]을 [법적생년월일]로 설정.
  - 이후 [법적생년월일산출].[법적생년월일]로 설정.
  - 하지만, "3"인 경우에 대한 명시적인 오류 코드는 제공되지 않았습니다.

따라서, 주어진 정보에서 "고객상태구분코드"가 "3"인 경우에 발생하는 오류 코드는 "A4600533"입니다
[출처] ***모델절대경로Path***

도입 체크리스트

AI-Driven Development 도입 전에 필요한 준비사항을 체크하여 도입 효율성 제고를 위한 고객사의 준비사항을 알려드립니다.

Point 1

조직적 준비
Organizational

경영진·현업 공감대 확보
→ KPI 연계, 추진 의지 명문화

Change Management
→ 파일럿·챔피언 그룹 운영, 온보딩 교육

역할·책임(R&R) 정립
→ 운영 총괄·품질 검증자·교육 담당자 구체화

Point 2

기술적 준비
Technical

인프라 준비
→ GPU 사양·용량·TCO 검토, 구매/임대/클라우드 계획

개발 환경 정합성
→ 프레임워크 호환성, 마이그레이션 난이도 점검

검증 체계 연동
→ 코드 품질·보안 점검 프로세스와 교차 검증

사내 지식 자산 준비
→ 코드 가이드·스키마·용어사전 정리, 학습 자산화

Point 3

운영적 준비
Operational

단계별 로드맵
→ PoC → Pilot → 확산, 목표·평가 지표 정의

운영 조직 구성
→ 전담 운영팀(총괄·지원·검증), Feedback Loop 운영

교육·가이드
→ 기능별 교육(Trans/Analysis/Test), BP 공유

Agentic으로의 진화 – 일하는 방식의 변화

기존 개발 공정

분석
업무 전문가

설계
설계자

개발
개발자

전개

Agentic AI 개발공정

Knowledge Foundation

System Development

전개

분석·설계

AI 분석•설계자 / Agent

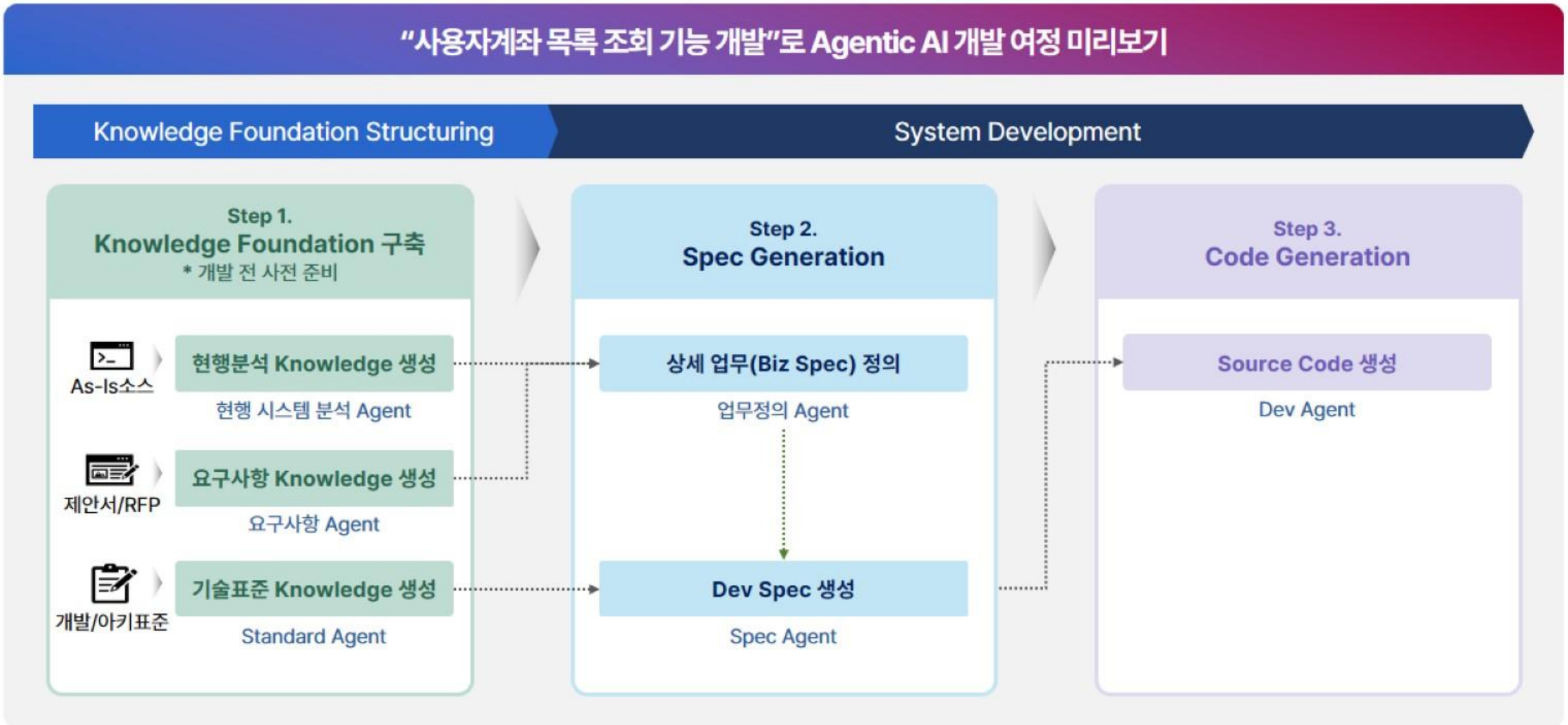
기존 분석 업무의 데이터화 통한
지식 기반 구축
→ Knowledge 기반 Agent 수행

개발·전개

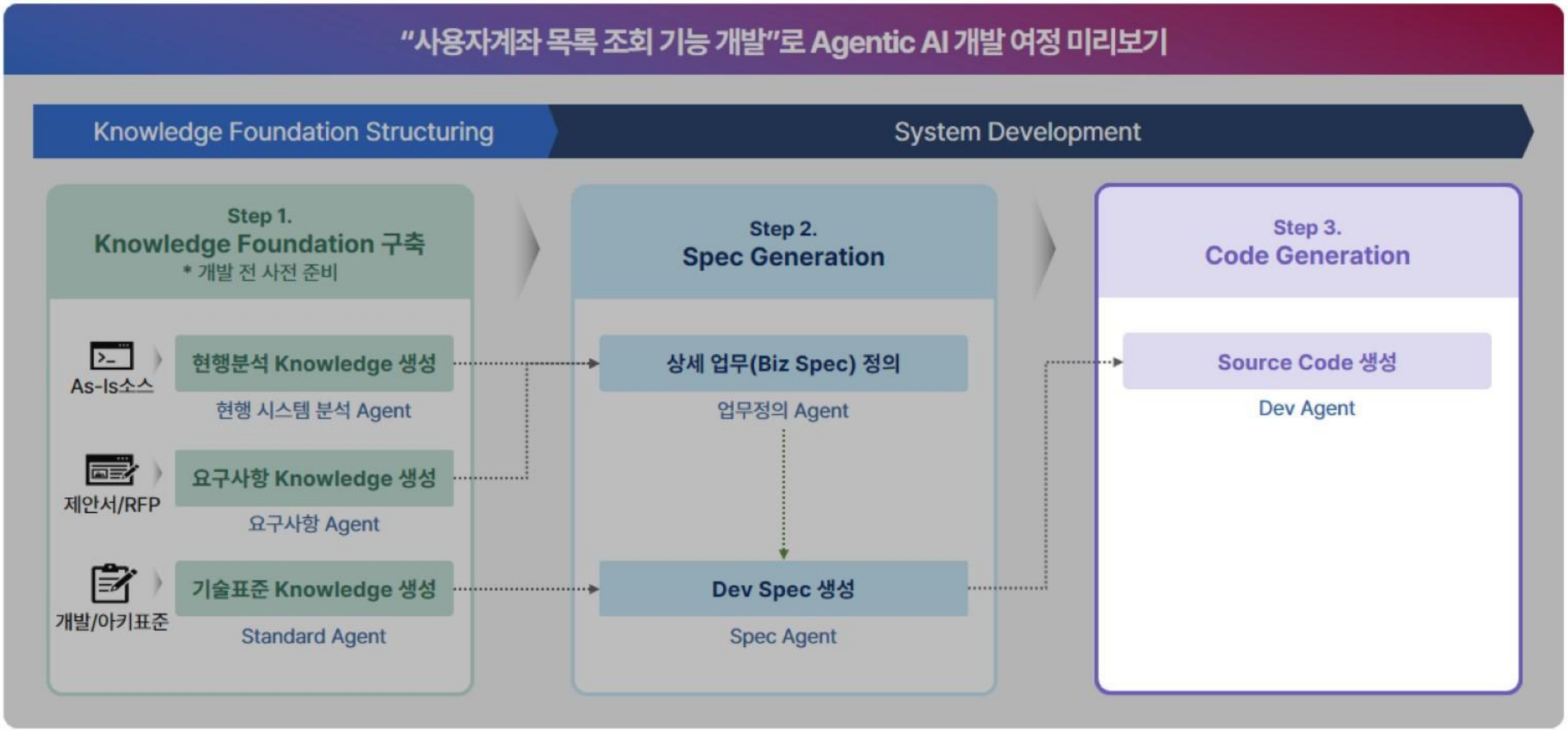
AI App 개발자 / Agent

Agent가 중심이 되어 업무 수행,
사람은 검토·의사결정 집중

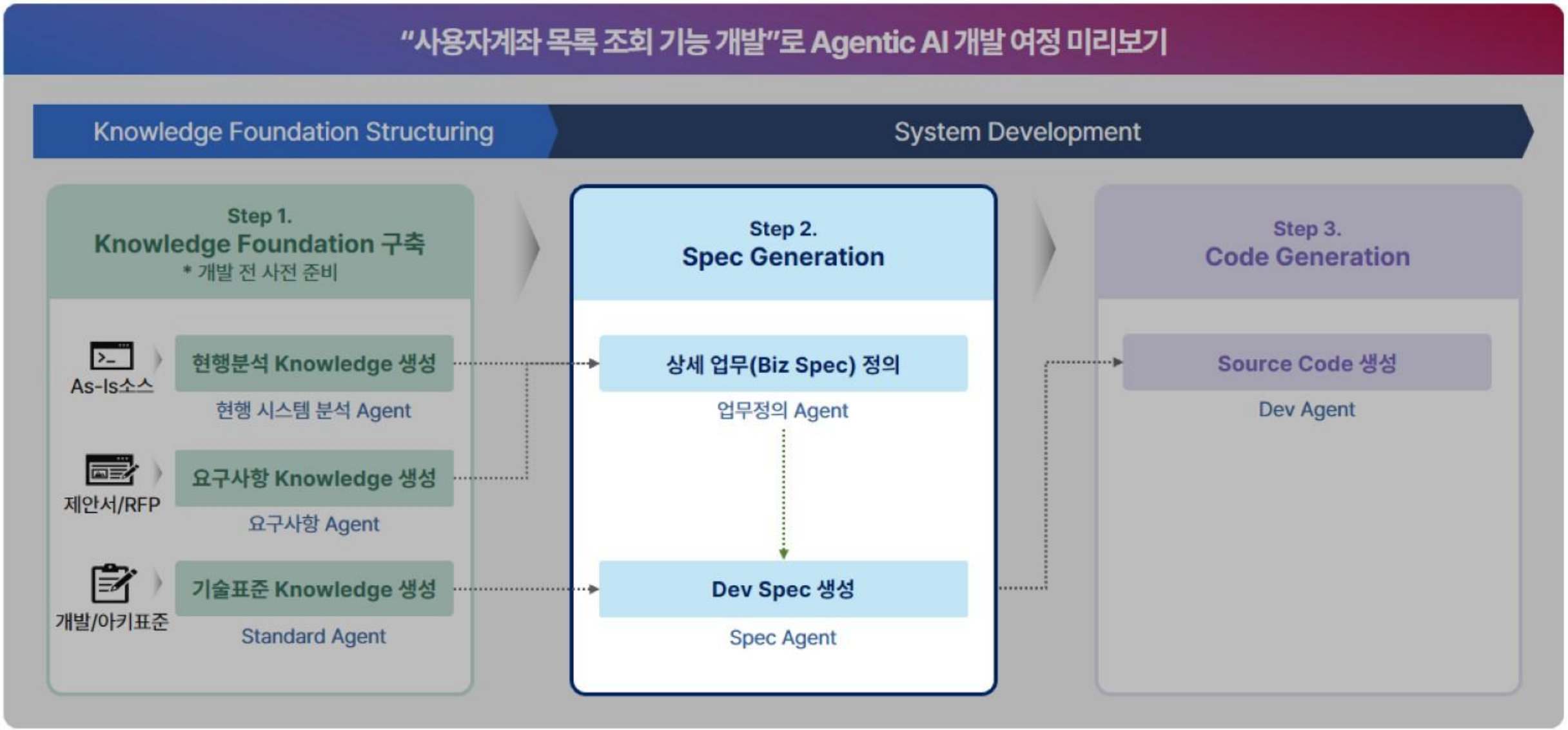
Agentic AI 개발 여정 미리보기



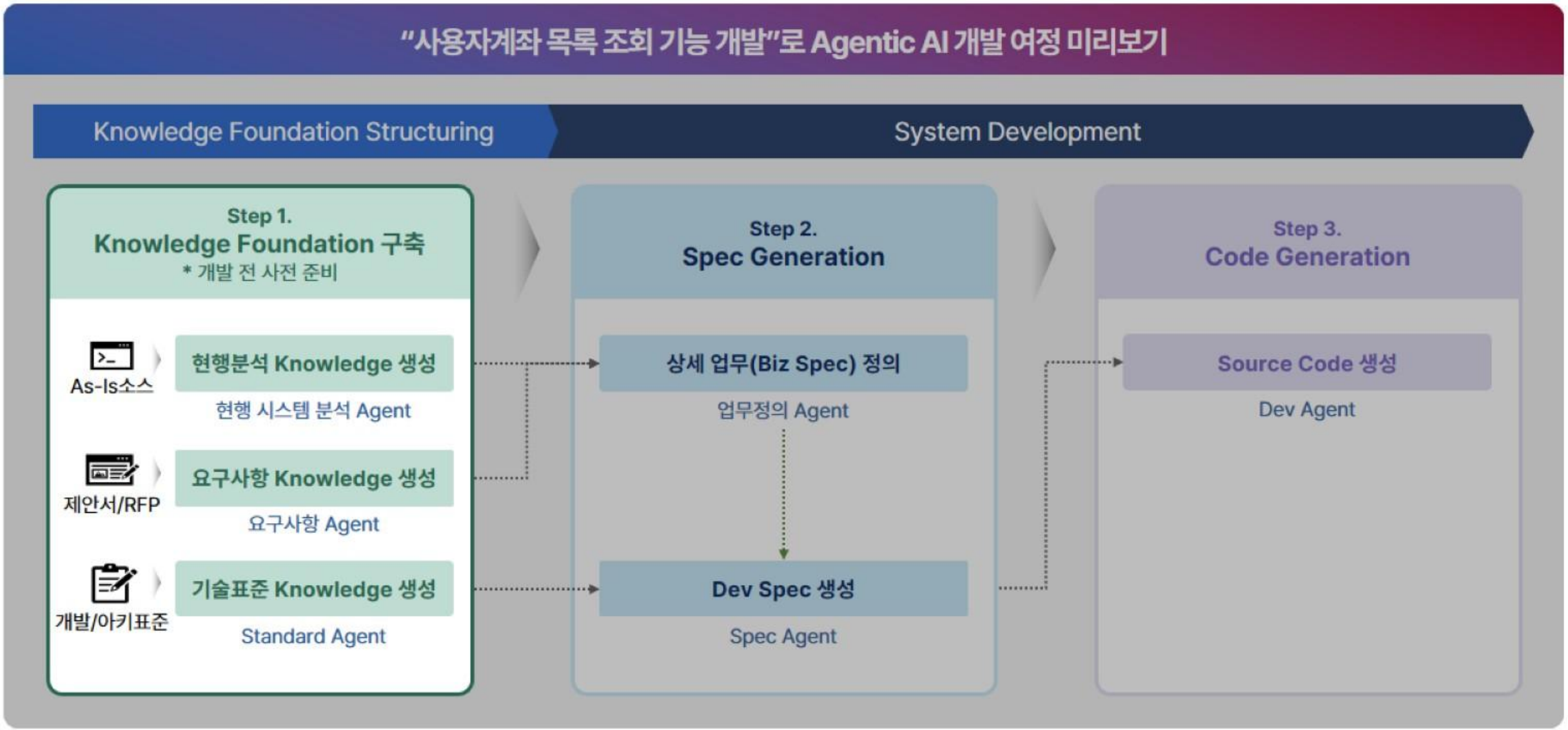
Agentic AI 개발 여정 미리보기



Agentic AI 개발 여정 미리보기



Agentic AI 개발 여정 미리보기



DevOn
AI-Driven Development

Step 1.
Knowledge Foundation 구축

- **현행분석 Knowledge 생성**
- 요구사항 Knowledge 생성
- 기술표준 Knowledge 생성

Step 2.
Spec Generation

- 상세업무 (Biz Spec) 정의
- Dev Spec 생성

Step 3.
Code Generation

- Source Code 생성

LG CNS

DevOn
AI-Driven Development

Step 1.
Knowledge Foundation 구축

현행분석 Knowledge 생성
- **요구사항 Knowledge 생성**

기술표준 Knowledge 생성

Step 2.
Spec Generation

상세업무 (Biz Spec) 정의

Dev Spec 생성

Step 3.
Code Generation

Source Code 생성

LG CNS

DevOn
AI-Driven Development

Step 1.
Knowledge Foundation 구축

현행분석 Knowledge 생성
요구사항 Knowledge 생성
기술표준 Knowledge 생성

Step 2.
Spec Generation

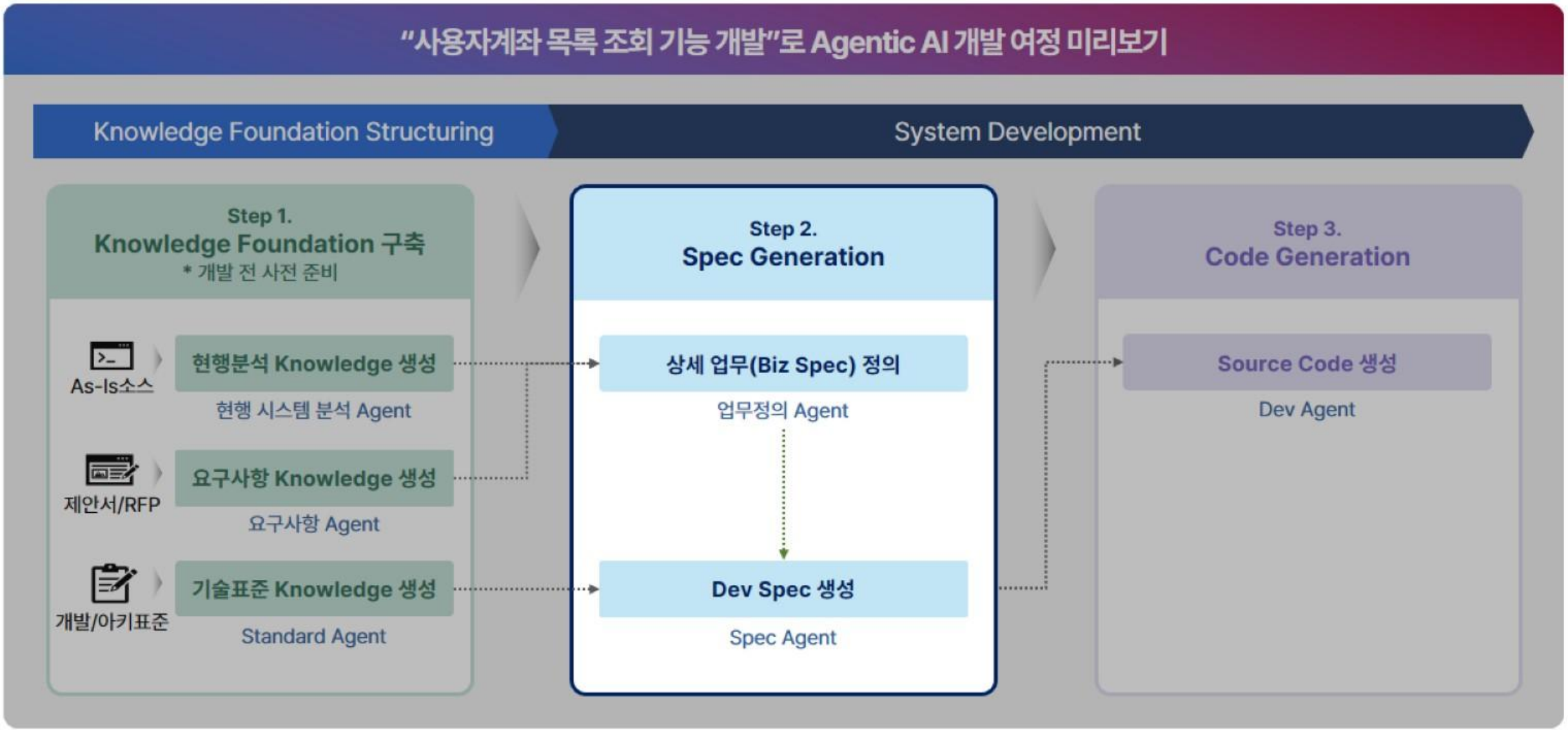
상세업무 (Biz Spec) 정의
Dev Spec 생성

Step 3.
Code Generation

Source Code 생성

LG CNS

Agentic AI 개발 여정 미리보기



DevOn
AI-Driven Development

Step 1.
Knowledge Foundation 구축

현행분석 Knowledge 생성
요구사항 Knowledge 생성
기술표준 Knowledge 생성

Step 2.
Spec Generation

- **상세업무 (Biz Spec) 정의**

Dev Spec 생성

Step 3.
Code Generation

Source Code 생성

LG CNS

DevOn
AI-Driven Development

Step 1.
Knowledge Foundation 구축

현행분석 Knowledge 생성
요구사항 Knowledge 생성
기술표준 Knowledge 생성

Step 2.
Spec Generation

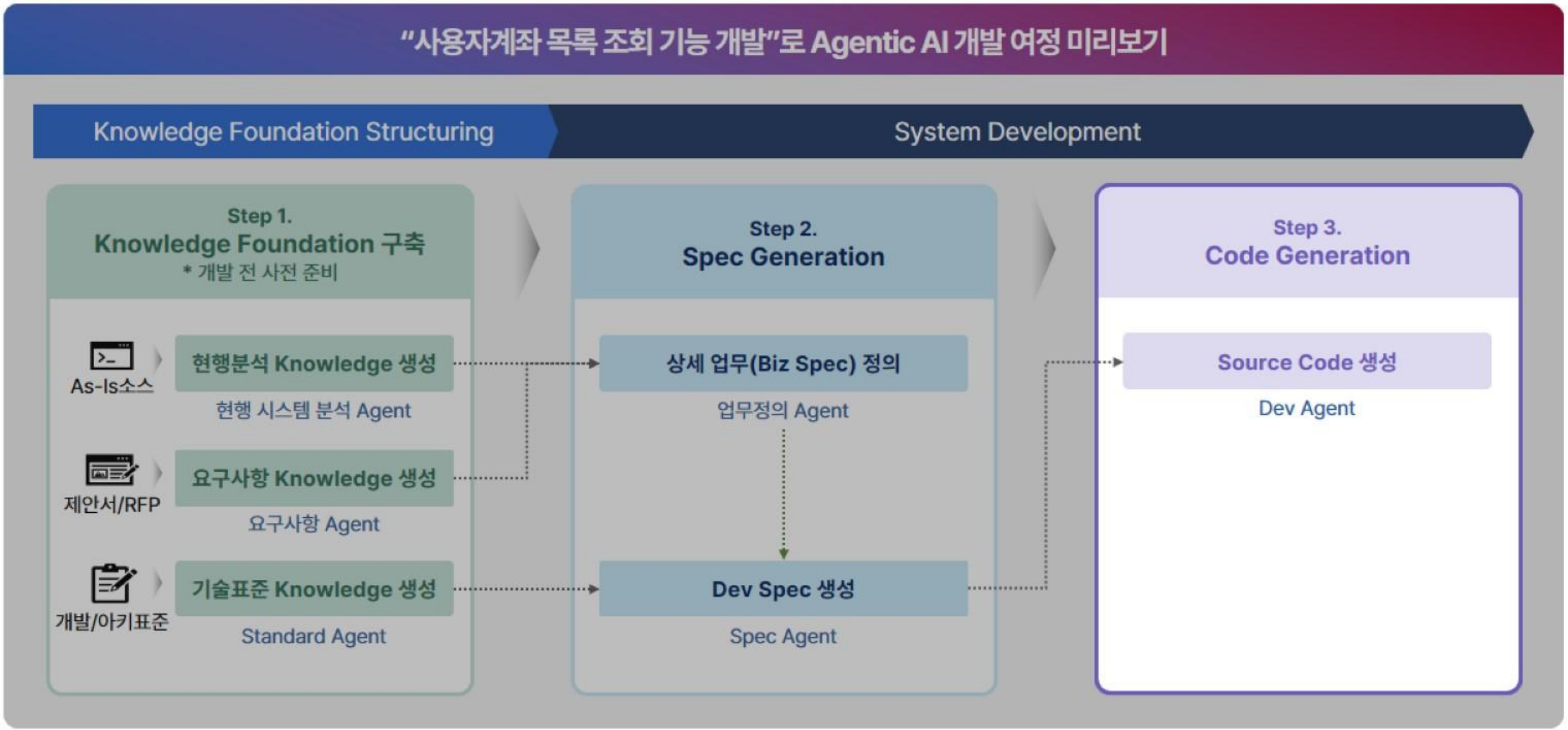
상세업무 (Biz Spec) 정의
‣ **Dev Spec 생성**

Step 3.
Code Generation

Source Code 생성

LG CNS

Agentic AI 개발 여정 미리보기



DevOn
AI-Driven Development

Step 1.
Knowledge Foundation 구축

현행분석 Knowledge 생성
요구사항 Knowledge 생성
기술표준 Knowledge 생성

Step 2.
Spec Generation

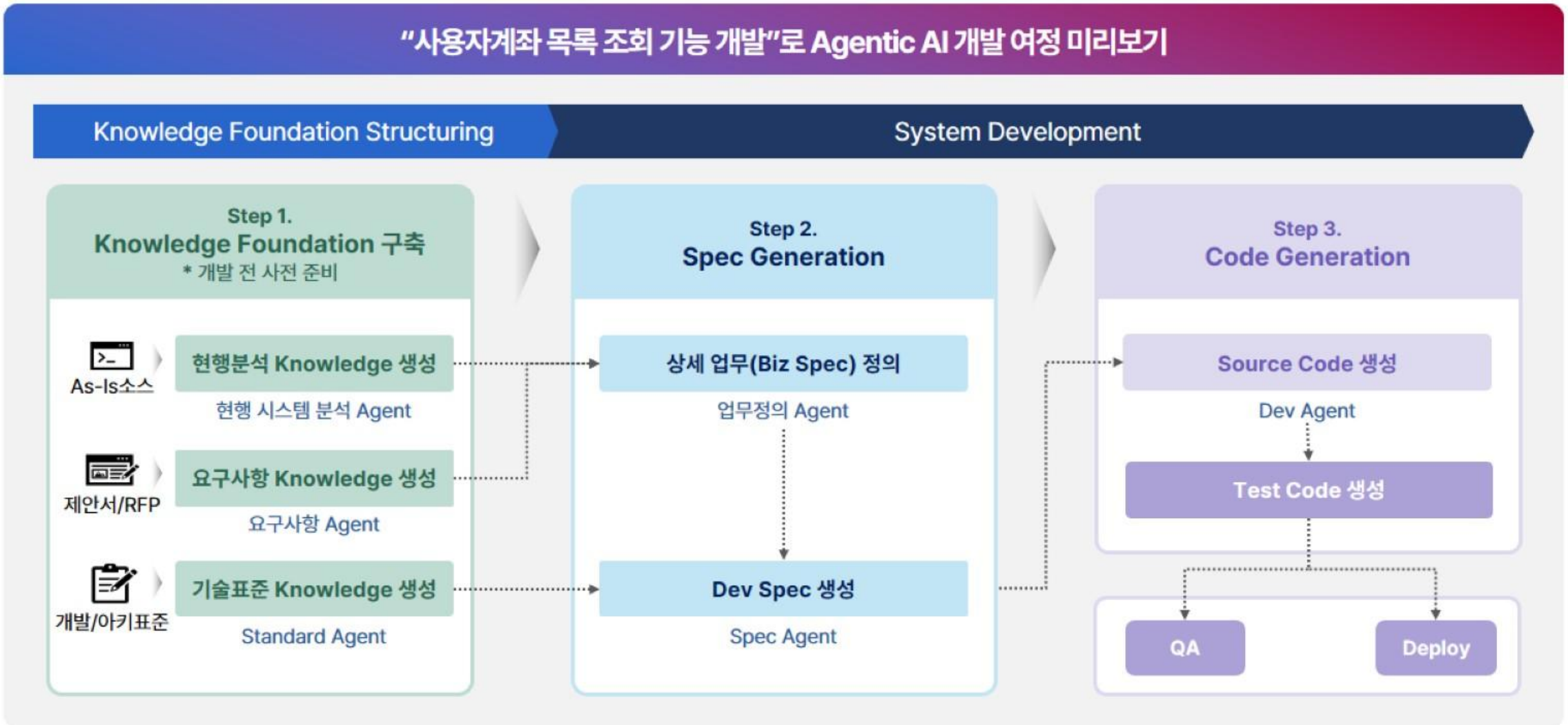
상세업무 (Biz Spec) 정의
Dev Spec 생성

Step 3.
Code Generation

- **Source Code 생성**

LG CNS

Agentic AI 개발 여정 미리보기



Summary

DevOn AI-Driven Development는...

01

개발 효율성 향상을 위한 필수 도구

- ✓ 코드 자동화, 인스펙션, 테스트케이스 생성 등을 통해 개발 품질과 속도 획기적 향상
- ✓ 반복적인 작업 자동화 →개발자가 창의적인 작업에 집중할 수 있게 도와줌 → 품질 개선에 기여함

02

보안 문제로 인한 소스코드 유출 방지

- ✓ 보안이 중요한 환경에서도 신뢰성 있게 소스코드를 생성하고 관리할 수 있도록 설계됨
- ✓ 고객사의 다양한 인프라 환경 어디에서든 안전하게 작동할 수 있음

03

Knowledge Foundation 기반의 Agentic AI

- ✓ Knowledge Foundation 기반으로 효율적인 AI 주도 시스템 개발 가능
- ✓ Spec 중심의 접근으로 일관된 설계와 품질 향상
- ✓ 분석/설계/개발/테스트 전 공정을 통합하고 실시간으로 변경사항 적용 및 지속적 개선

Agentic AI 검증을 위한 PoC 진행

여러 프로젝트의 PoC를 통해 Agentic AI의 효율성과 신뢰성을 검증하고 있으며,
이를 기반으로 지속적인 개선과 최적화 작업 진행 중

DevOn AI

개발자의 일상이 되다

Driven Development

Q&A

Thanks

LG CNS

별첨. RAG

SI/SM 프로젝트에 최적화된 RAG 제공

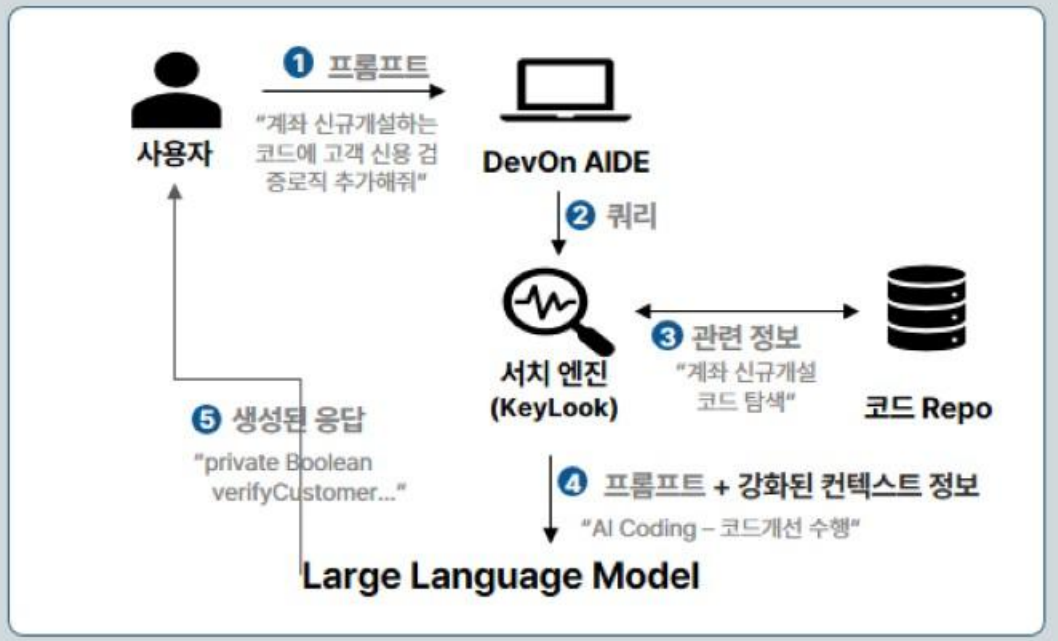
- SI/SM 프로젝트에 최적화된 RAG 구성을 위한 기술 및 솔루션 제공
- 코드 검색, DB 스키마, 변환 패턴 등의 영역별 RAG 구성 Know-How 확보
- 다양한 데이터 원천(PPT, Word, Excel, PDF 등)으로부터 Text 추출 기술 확보

예시) 코드 검색 RAG

- 자연어(한글)로 원하는 코드 block 검색
- Top-3 검색 성능 95%
- 다양한 AI 모델(상용/오픈)과 연결 가능

✓ 코드 검색 RAG 유스케이스

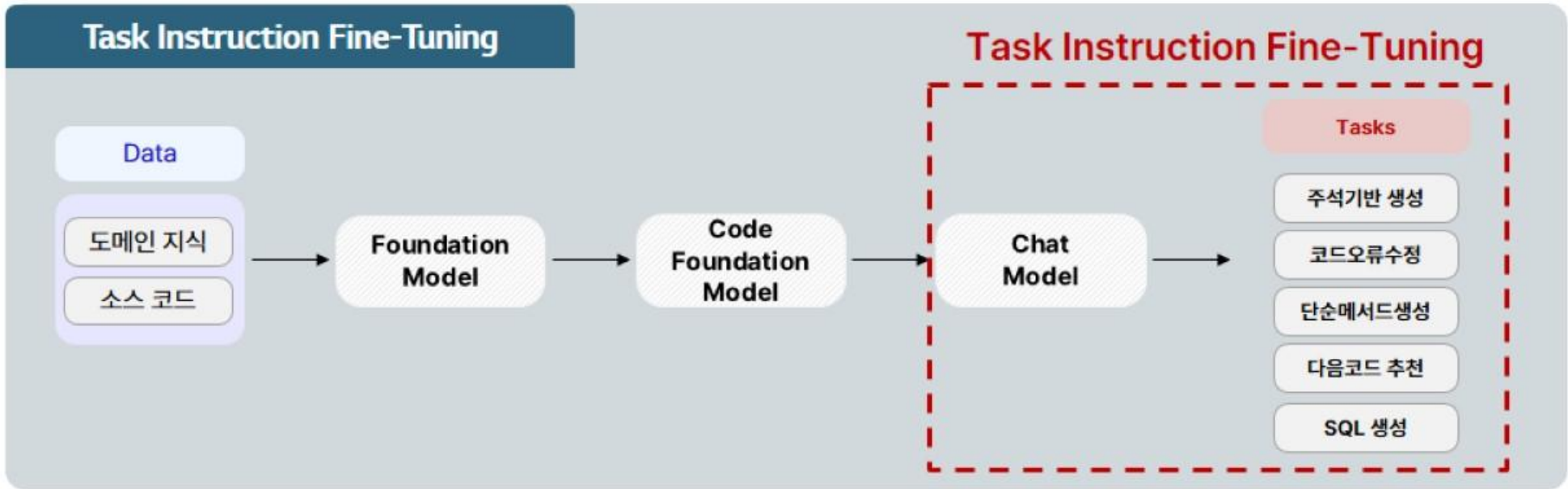
"계좌 신규 개설하는 코드에
고객 신용 검증 로직 추가해줘"



별첨. Fine-Tuning

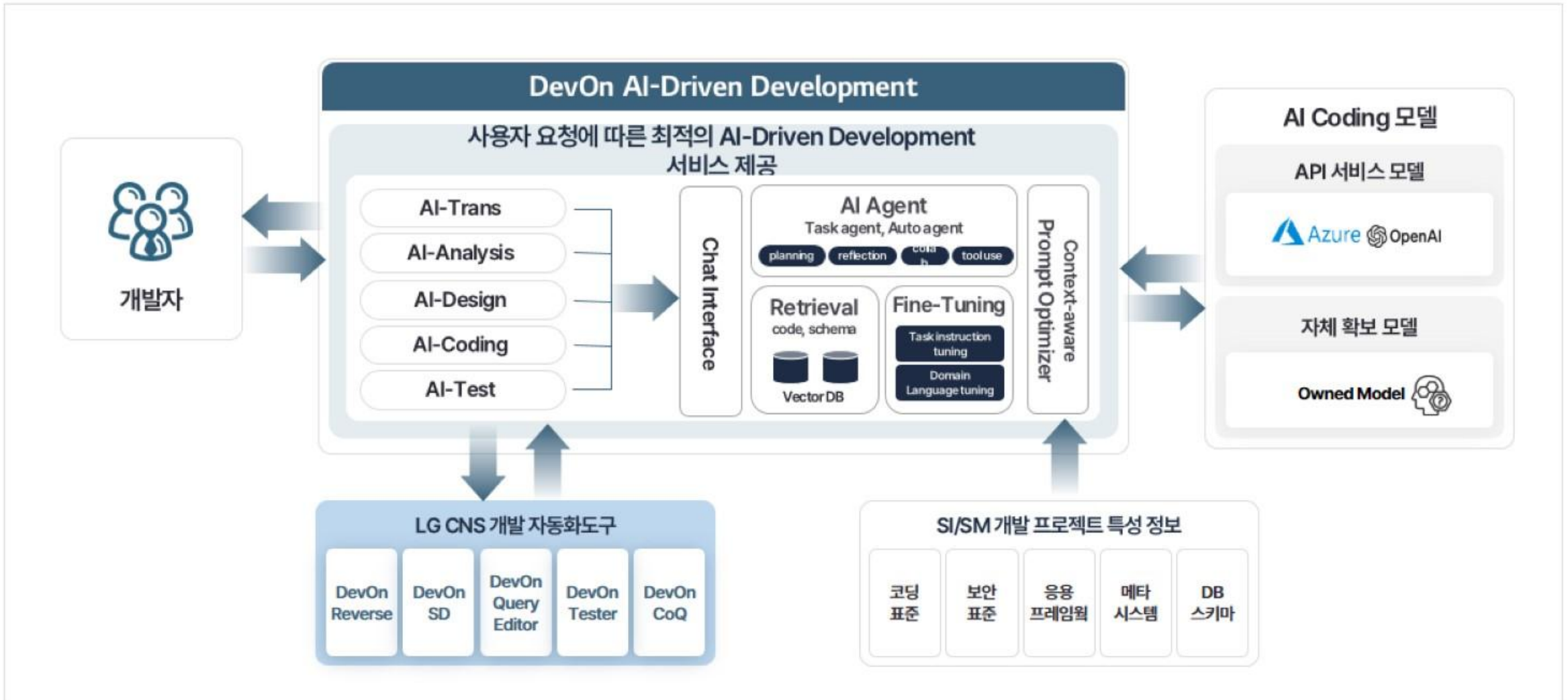
Task 중심 Instruction Fine-Tuning

- AI-Coding Task 중심으로 Instruction Fine-Tuning 수행을 통한 최적의 모델 확보
- Task Instruction Fine-Tuning을 위한 프로세스(데이터 → 학습 → 평가) 체계화
- 소스코드로부터 각 Task별 학습 데이터 자동 생성 및 전문가 검수
- Task 별 Performance Measurement 지표 개발 및 자동화된 성능측정 프로세스 구축
- 특정 Task 전용 초경량 모델 확보 가능 (ex. Next Code 추천 전용 AI 모델)



별첨. DevOn AI-Driven Development 구성

DevOn AI-Driven Development 구성도



별첨. Context Window에 따른 적용 가능 LOC

What is Prompt Engineering

[60토큰 1LOC 가정했을 경우 입력 가능한 LOC]

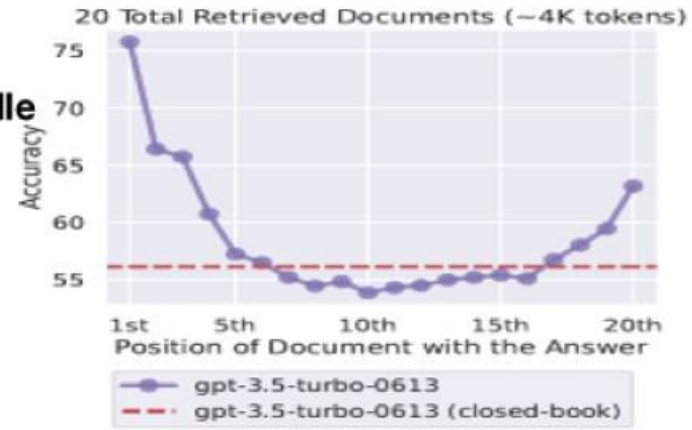
Context Window	전체 LOC 수용량	입력 가능한 LOC(50%)	특징
32k	~530 LOC	~260 LOC	한두 개 함수 단위 분석 적합
128k	~2,120 LOC	~1,060 LOC	소스 전체 파일 분석 가능
256k	~4,240 LOC	~2,120 LOC	다중 소스 파일/헤더까지 커버
512k	~8,480 LOC	~4,240 LOC	복잡한 파일 일괄 처리 가능

[60토큰 1LOC 샘플]

pfmHdrSetUIErrMsg(HDR_MSG_TYPE_L, HDR_MSG_OWF_N, "MA8983", "사업자등록번호는 필수 입력 항목이며, 공백 또는 NULL일 경우 오류로 처리됩니다. 반드시 10자리 숫자로 입력해야 합니다.");

Context Window↑

→ Lost In the Middle



[실험 과제]

- **멀티 문서 질의응답(Multi-document QA): 여러 문서 중 정답이 있는 문서를 찾아 답변.**
- **Key-Value Retrieval: JSON 형태의 Key-Value 쌍에서 특정 키의 값을 찾는 단순 검색 과제**

[Kep Point]

- **모델 성능은 입력 맨 앞 또는 맨 끝에 정답이 있을 때 가장 높음.**
- **중간에 정답이 있으면 성능이 크게 떨어짐 → "Lost in the Middle" 현상.**
- **심지어 긴 컨텍스트 윈도우를 가진 모델(예: GPT-3.5 16k, Claude 100k)도 기존 모델과 성능 차이가 거의 없음.**

Lost in the Middle: How Language Models Use Long Contexts

→ https://aclanthology.org/2024.tacl-1.9.pdf

별첨. On-Premise 인프라 준비

대략적인 수치이고, 추론 내용/토큰수에 따라 차이가 발생할 수 있습니다

Illustrative

GPU	①TPS(Throughput/Sec)	②목표 tps	동시 사용자 수 = ① / ②	DAU (피크 동접률 3%)	DAU (피크 동접률 5%)
H100 80GB	63.26	10	6.33명	211명	127명
	63.26	20	3.16명	105명	63명
A100 80GB	40.35	10	4.04명	135명	81명
	40.35	20	2.02명	67명	40명
RTX6000 Ada 96GB	28.33	10	2.83명	94명	57명
	28.33	20	1.42명	47명	28명

① GPU Throughput (토큰/초): LLM이 해당 GPU에서 생성 가능한 초당 토큰 수
② 목표 tps: 한 사용자가 요구하는 평균 생성 속도(토큰/초)
③ 동접 사용자 수 계산 : ① GPU Throughput / ② 목표 tps
④ DAU 계산: ③ 동접 사용자 수 / 목표 피크 동접률

▪ **AI-Driven Development 적용을 위한 최소 사양(개발자 IDE)**

	Eclipse	VSCode	IntelliJ
최소 사양	4.4+ (2014.03+) Luna+ JDK 1.8+	1.85+ (2023.11+) JDK 17+	Community 2023.3.4+ Ultimate 2023.3.3+ JDK 17+

▪ **적용 유형별 최소 VRAM**

적용 유형	VRAM
Assistant	80G
Agent	1200G

별첨. GPU/LLM 별 TPS

대략적인 수치이고, 추론 내용/토큰수에 따라 차이가 발생할 수 있습니다

GPU	VRAM	Model	tokens/sec (tps)
A100	40GB	qwen2.5-coder:32b	35.97
RTX A6000	48GB	qwen2.5-coder:32b	28.33
A100	80GB	qwen2.5-coder:32b	40.35
A100	80GB	qwq:32b	31.33
A100	80GB	llama3.3:70b	23.92
H100	80GB	qwen2.5-coder:32b	63.26
H100	80GB	qwq:32b	59.58
H100	80GB	llama3.3:70b	38.50
H200	141GB	qwen2.5-coder:32b	63.08
H200	141GB	qwq:32b	58.21
H200	141GB	llama3.3:70b	39.35