

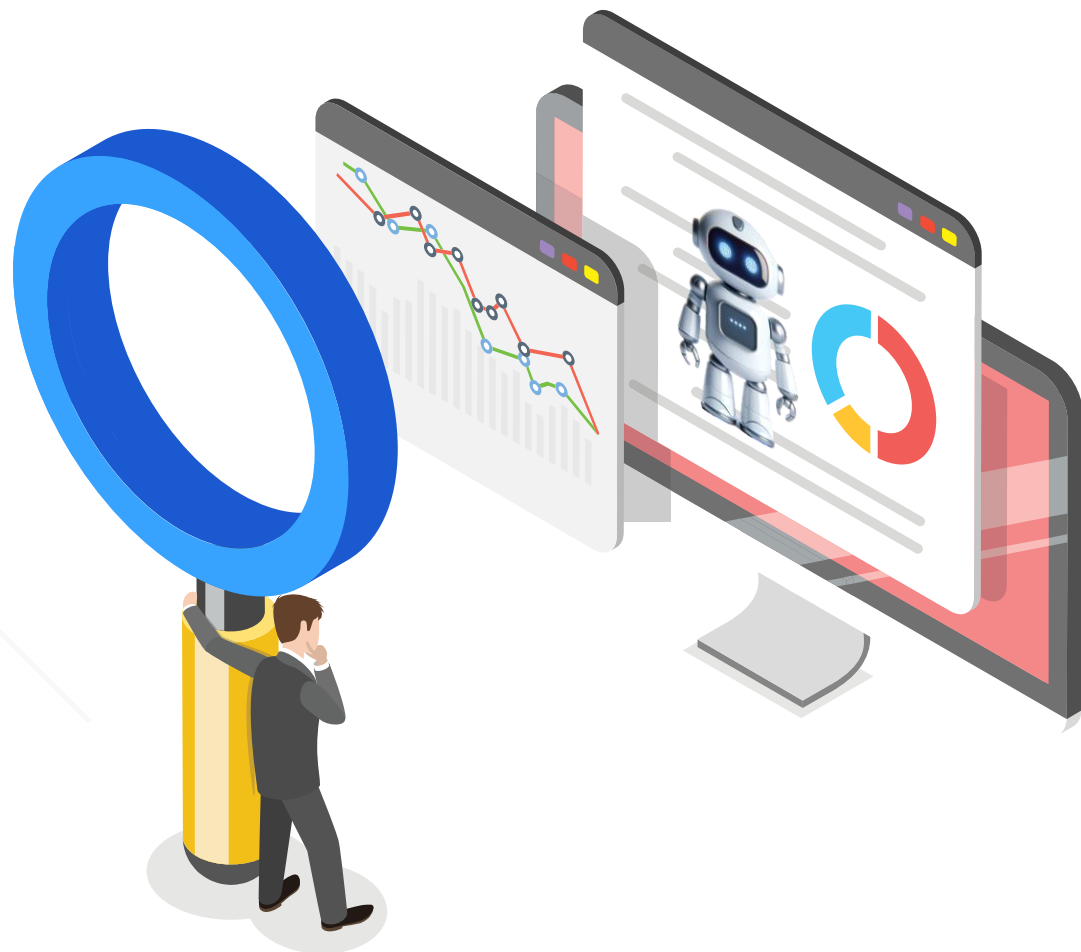


엔터프라이즈 LLM을 활용한

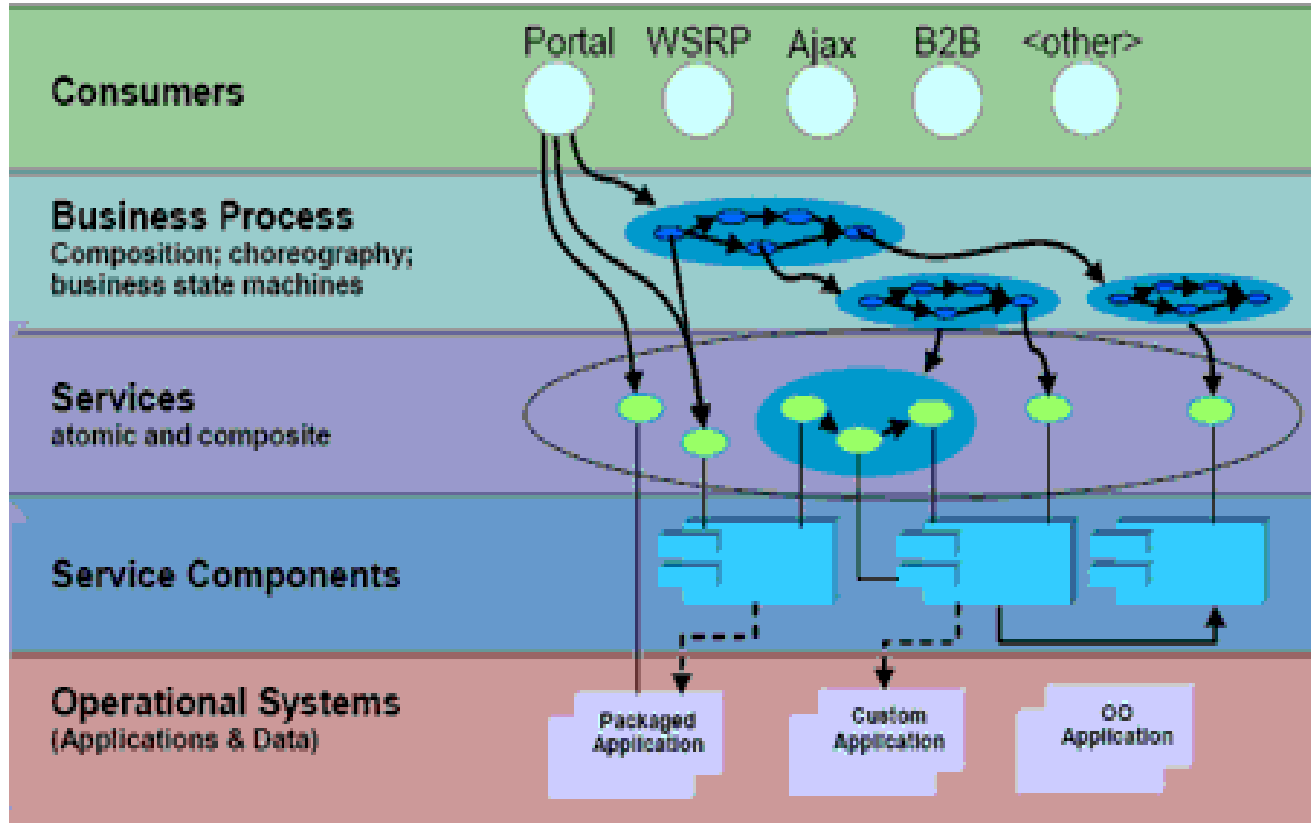
비즈니스 프로세스 관리 혁신 상세 방안

Today's Key Takeaways

- ① 코딩에 의한 정보 시스템 개발 비용은 너무 높다
- ② BPMS는 전통적으로 현업에 의한 프로세스 자동화를 통하여 개발 비용을 낮추려 한 시도
- ③ LLM을 기업 정보 시스템에 얼마나 활용할 수 있을까?
- ④ BPMS와 LLM을 결합하면 프로세스 자동화 비용을 낮출 수 있을까?



☑ 기업 정보 시스템의 구성 요소



Source: IBM SOA Foundation Reference Architecture solution view

☑ UI

☑ 비즈니스 프로세스

☑ 비즈니스 로직

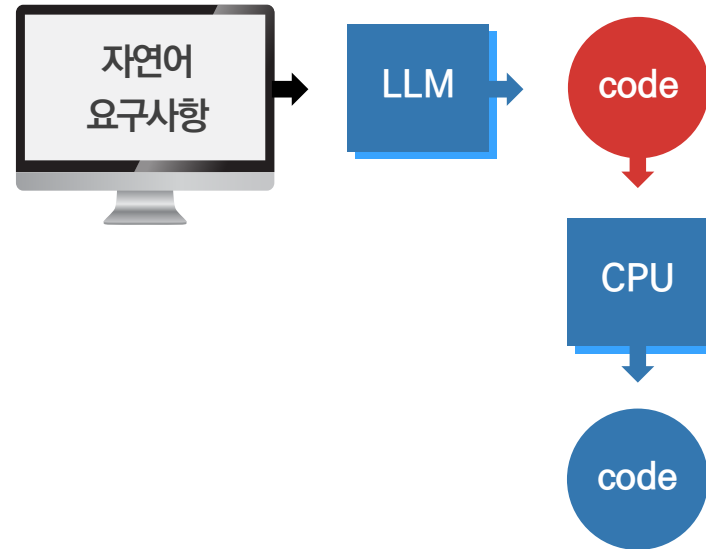
☑ 데이터베이스

☑ 외부 시스템

☑ AI를 기반한 시스템 구축 방법 두가지

접근 1

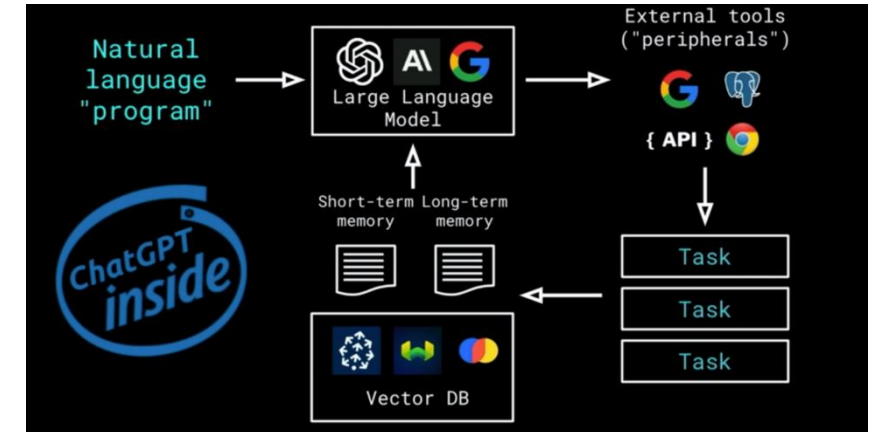
- ☑ 코드를 생성
→ Compliant System



- Copilot / Cursor IDE
- (Devin like) Autogen / MetaGPT

접근 2

- ☑ 자연어를 바로 실행
→ Flexible System



- RPA / BPM
- No Code Tools

☑ 구현 방법의 비교

구분

코드 생성을 통한 애플리케이션 구축
(Deterministic Code)

대화형 LLM을 이용한 정보시스템 구축
(Flexible Application)

프로세스

정해진 프로세스와 흐름을 따르며,
오류가 적고 일관성이 높음

자연어 입력을 통해 동적으로 프로세스가
생성되며, 유연하게 조정 가능

UI

사용자 인터페이스가 사전 설계되어 일관된 경험 제공

대화형 인터페이스를 통해 사용자 질의에 맞춰
동적으로 UI가 변화

비즈니스 로직

복잡한 비즈니스 로직 구현 가능,
높은 정확성과 규칙 준수

간단하고 유연한 로직 적용 가능,
빠르게 변화하는 요구사항에 적응

데이터베이스

고정된 스키마에 따라 데이터베이스 설계,
데이터 일관성과 정확성 유지

자동으로 데이터베이스 스키마 생성 및 조정,
확장성과 유연성 강조



☑ AI Coding 적용 시 고려사항

☑ 기계에게 요구사항을 어떻게 명확하게 전달할 것인가?

☑ 문제 영역을 어떻게 쪼개고 요구사항 범위를 작게 쪼갤 것인가?



☑ 도메인 주도 설계

- 도메인 모델을 먼저 만들어서 개발자(기계)와 도메인에 대한 이해의 갭을 최소화
- 도메인 언어(업무 용어)를 이용한 애플리케이션 설계
- 멀티 도메인으로 쪼개어 업무 범위를 제한하여 구현 복잡도를 한정

☑ 테스트 주도 개발

- 테스트 코드를 먼저 작성하여 요구사항 이해를 코드 기반(명확한 언어)으로
- 인수(완성) 여부를 기계가 자가로 판단할 수 있음



위험도 case	하한나이 condition1	상한나이 condition2	건강상태 condition3	통원유무 condition4	음주빈도 condition5
A rank	age>20	age<41	g	n	1
A rank	age>20	age<41	g	n	2
A rank	age>40	-	g	n	1
B rank	age>20	age<41	g	y	1
B rank	age>40	-	g	n	2
B rank	age>20	age<41	b	n	1
B rank	age>20	age<41	g	y	2
C rank	age>20	-	-	-	-
C rank	age>20	-	-	-	-
C rank	age>40	-	-	-	-
C rank	age>40	-	-	-	-
C rank	age>40	-	-	-	-
D rank	age>20	-	-	-	-
D rank	age>40	-	-	-	-
D rank	age>40	-	-	-	-

예시 입력값 대 기대값들

위의 예시를
JUnit Test로
작성해줘

위의 테스트를
통과하는
구현체를 만들어줘

```
import static org.junit.Assert.*;
import org.junit.Test;

public class InsuranceProductTest {

    @Test
    public void testARankAgeBetween20And40() {
        InsuranceProduct product = new InsuranceProduct();
        String result = product.evaluate(30, "g", "n"); // age, condition3, condition4
        assertEquals("A rank", result);
    }

    @Test
    public void testARankAgeOver40() {
        InsuranceProduct product = new InsuranceProduct();
        String result = product.evaluate(45, "g", "n"); // age, condition3, condition4
        assertEquals("A rank", result);
    }

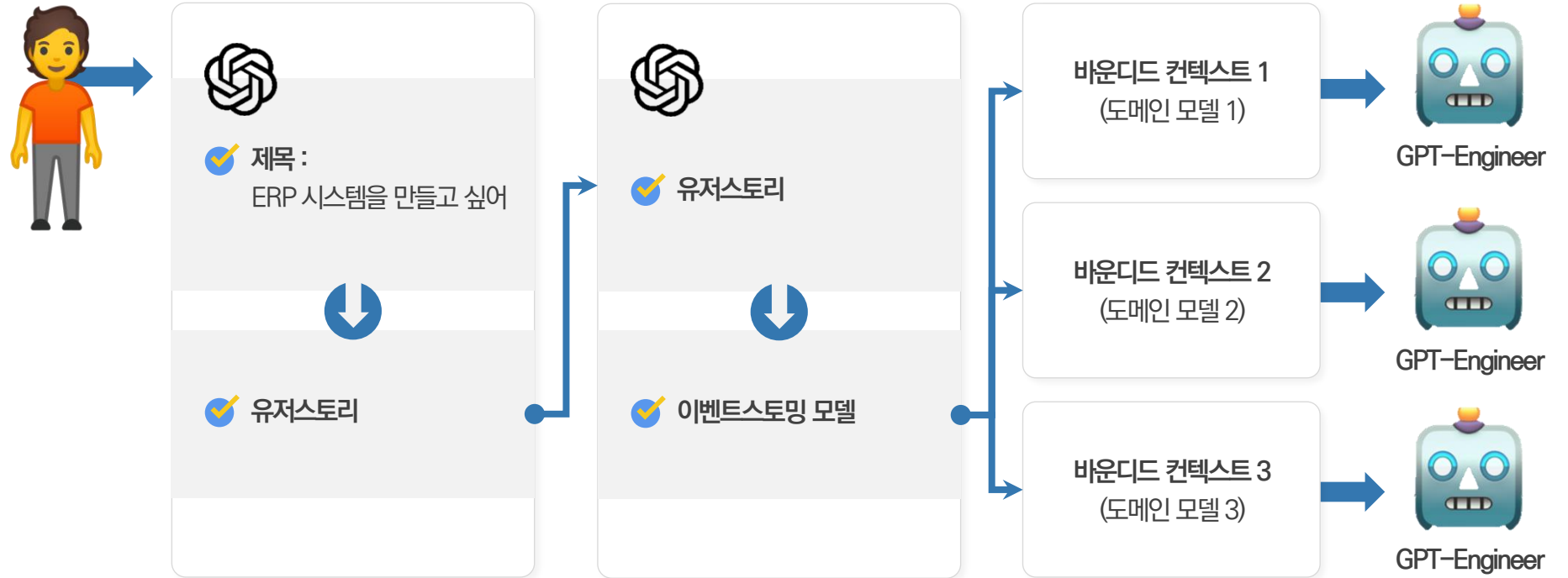
    // ... 추가적인 테스트 메소드들 ...

    @Test
    public void testDRankAgeOver40Condition3BCondition4() {
        InsuranceProduct product = new InsuranceProduct();
        String result = product.evaluate(45, "b", "y"); // age, condition3, condition4
        assertEquals("D rank", result);
    }
}
```

예시 입력값 대 Test
= 코드로 표현된 명확한 요구사항 기대값들

```
public class InsuranceProduct {
    public String evaluate(int age, String condition3, String condition4) {
        if (age == 20 && age < 41 && "g".equals(condition3) && "n".equals(condition4)) {
            return "A rank";
        } else if (age == 40 && "g".equals(condition3) && "n".equals(condition4)) {
            return "A rank";
        } else if (age == 20 && age < 41 && "g".equals(condition3) && "y".equals(condition4)) {
            return "B rank";
        } else if (age == 40 && "g".equals(condition3) && "n".equals(condition4)) {
            return "B rank";
        } else if (age == 20 && age < 41 && "b".equals(condition3) && "n".equals(condition4)) {
            return "C rank";
        } else if (age == 40 && "b".equals(condition3) && "n".equals(condition4)) {
            return "C rank";
        } else if (age == 20 && age < 41 && "g".equals(condition3) && "y".equals(condition4)) {
            return "D rank";
        } else if (age == 40 && "g".equals(condition3) && "y".equals(condition4)) {
            return "D rank";
        }
        // ... 추가적인 조건에 대한 처리 ...
        return "Unknown rank";
    }
}
```

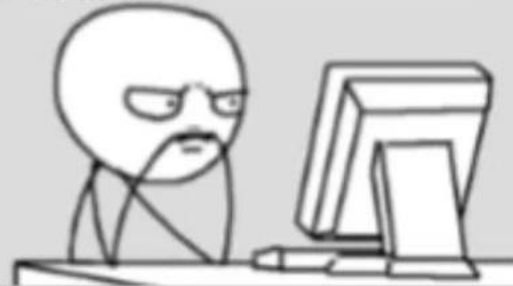
구현코드도출



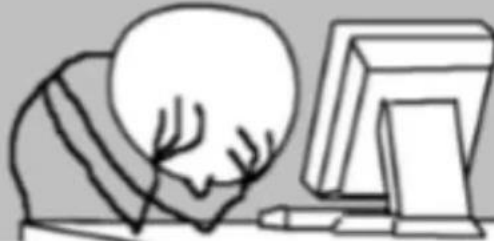
☑ Anyway, 코드 = 부담

Days before OpenAI

Developer coding
- 2 hours



Developer debugging
- 6 hours

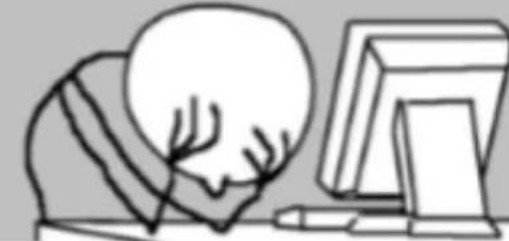


Days after OpenAI

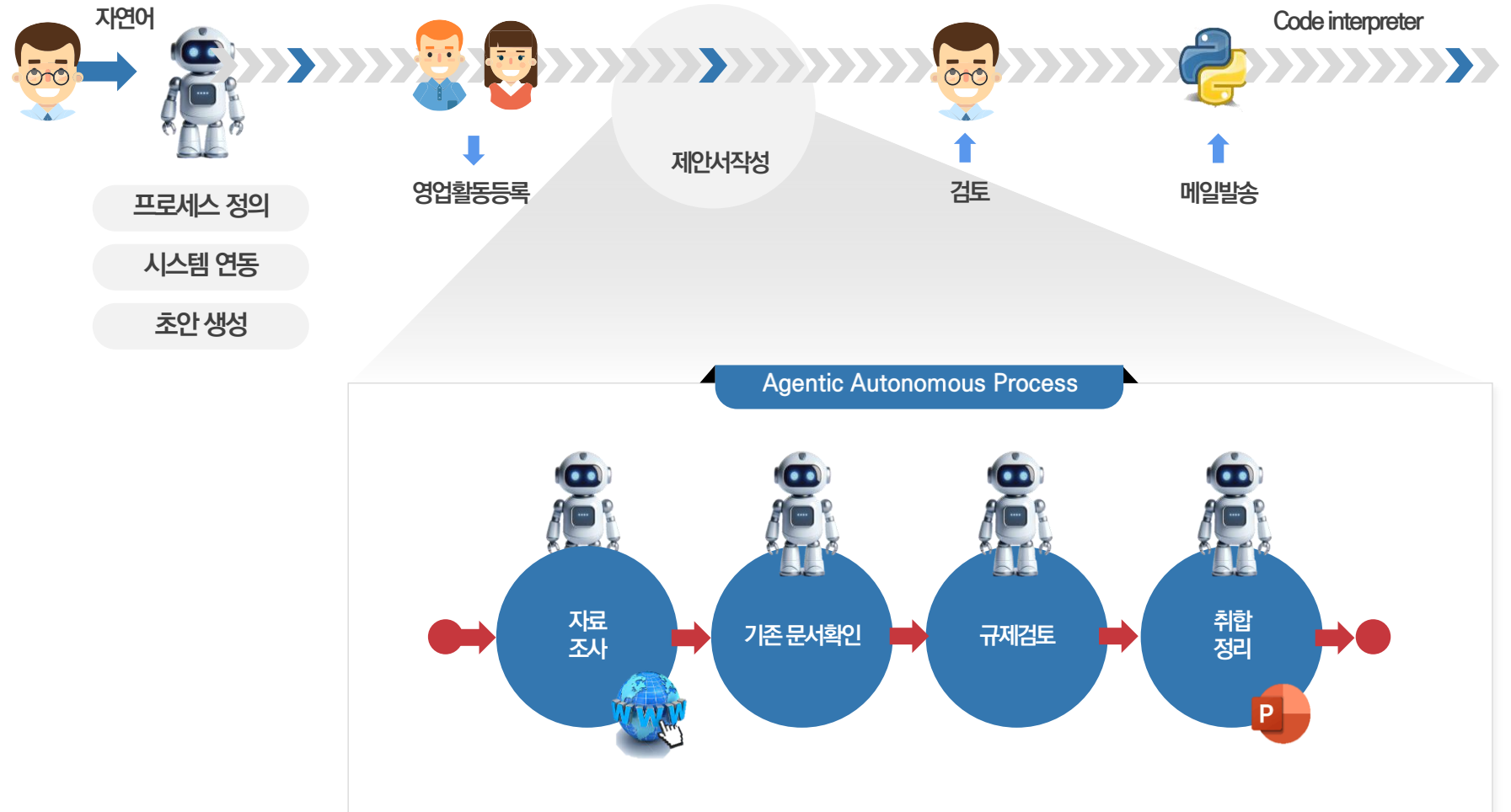
ChatGPT generates
Codes - 5 min



Developer debugging
- 24 hours



✓ 접근2 LLM as a 비즈니스 프로세스 실행 엔진



☑ 대화로 프로세스 정의

- 생성형 AI를 통한 프로세스 정의 생성
- 생성과 함께 가시화
- 생성 후에 관리자가 세부 수정
- 세부 수정사항도 자연어로 기술할 수 있으며 BPMN 프로세스 모델러가 익숙한 사람은 직접 수정



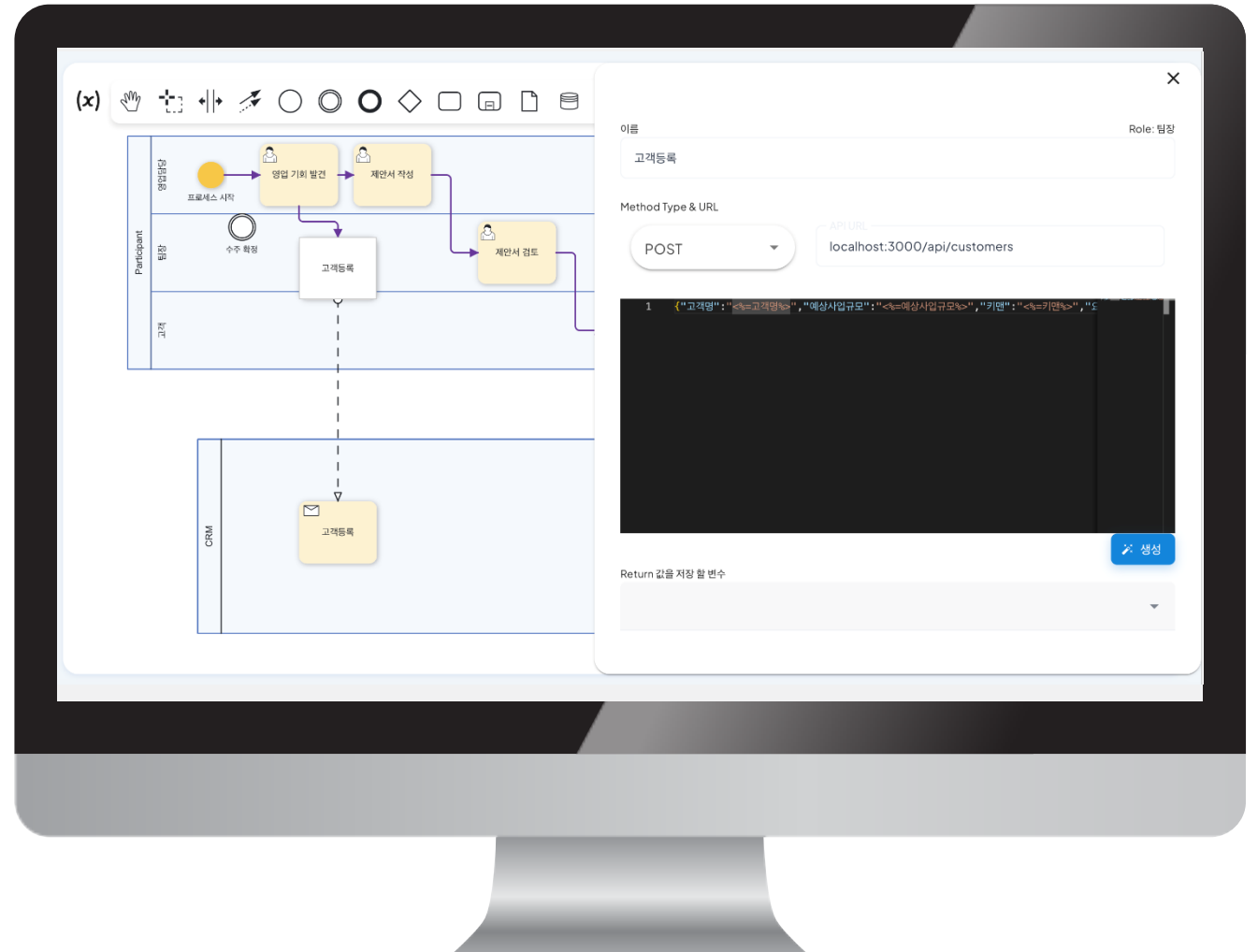
☑ 대화로 UI 생성

- 기존 UI 개발은 우수한 프론트엔드 기술자를 요구
- AI를 기반하면 최신 프레임워크 기반의 멀티스크린에 최적화된 업무 화면을 즉시 생성할 수 있고 생성된 화면을 원하는대로 수정할 수 있음



☑ 대화로 시스템 연동

- 기존 정보 시스템 간 연동은 다양한 기술적/정치적 이유로 시간과 노력이 많이 소요
- AI를 기반하면 내부 시스템들의 Open API Spec을 해석하여 프로세스 데이터 값들과 어떻게 연동하면 될지를 자동으로 생성



☑ LLM을 통한 유저 인터페이스

☑ 대화형 인터페이스를 통한 유연성과 동적으로 생성된 UI를 기반한 정형적 인터페이스의 혼합



☑ 유저는 내가 해야 할 업무를 위하여
기능이 어디있는지 (어떤 메뉴인지)
찾을 필요가 없어야 한다.

☑ 유저는 생성된 정형적인 UI 폼에 입력할 수도 있고
자연어로 편하게 (이동중엔 음성입력) 입력할 수도 있다.

☑ 대화형 프로세스 실행

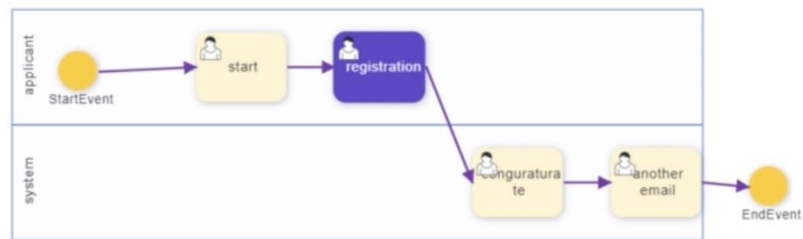
- 업무 단계별 담당자에게 업무 지시 및 독려
- 업무 체크포인트 준수 여부 확인 후 자동 반려
- 프로세스 진행 모니터링
- 이메일 발송 등 시스템 액티비티 수행

프로세스 정의 목록

id	name	description
<input checked="" type="checkbox"/> company_entrance	신규입사관리	신규 입사 관리 프로세스
<input type="checkbox"/> vacation_request	휴가 신청	Vacation processing process
<input type="checkbox"/> vacation_addition	휴가 추가 프로세스	Process for adding additional vacation days

Items per page: 10 1-3 of 3

Select



☑ 에이전트가 초안 생성

☑ 멀티에이전트를 통한 프로세스 자동화

☑ 멀티 에이전트 구성

- 멀티에이전트 (Crew) 구성을 생성
- Task, Goal, Tool 선택
- 프로세스 내 프로세스



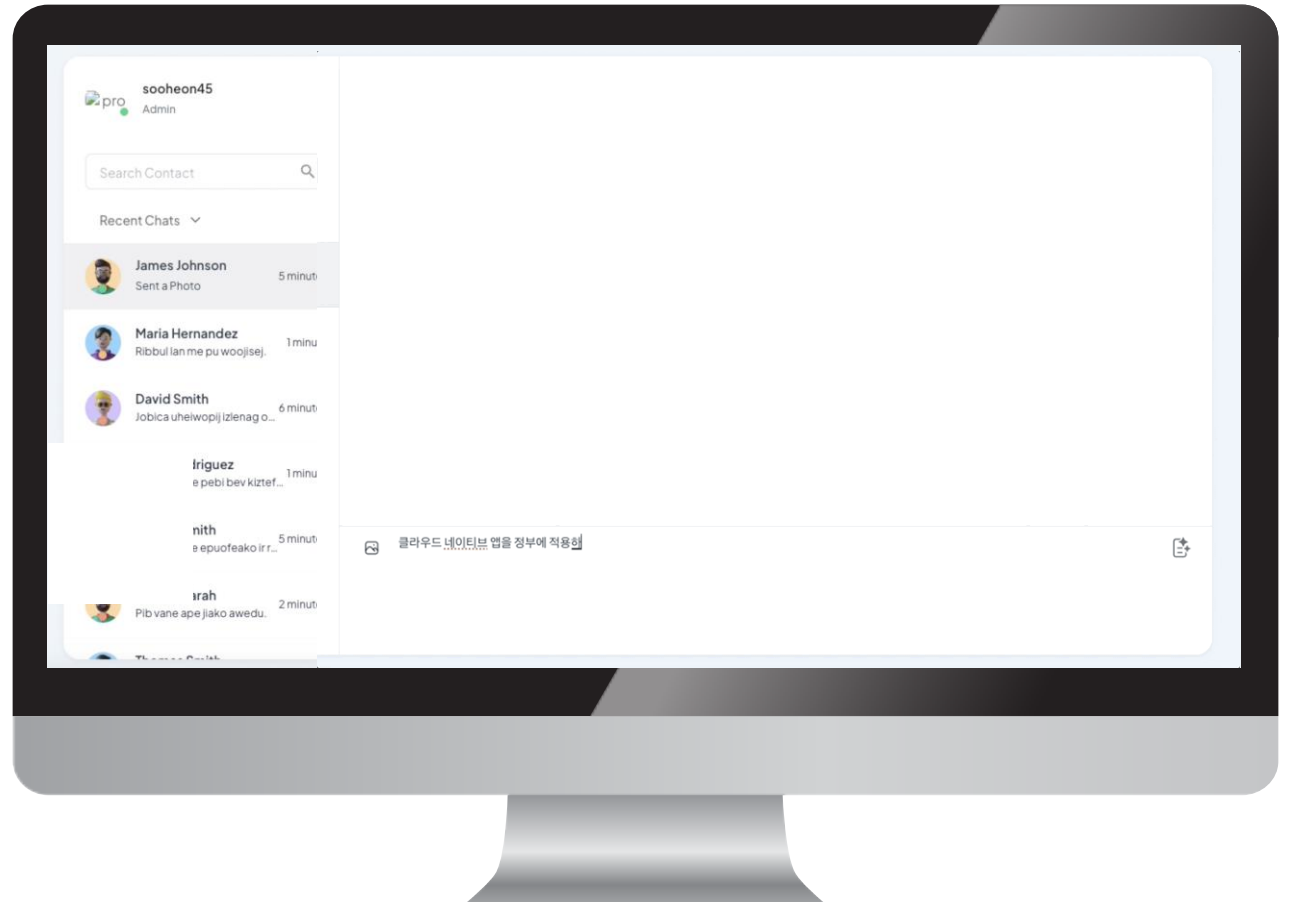
☑ 에이전트 실행

- 각 에이전트의 목표를 달성할 때까지 실행
- 에이전트 간의 위임, 질의 등으로 자율적이고 창의적인 프로세스 실행



☑ 다양한 도구 사용

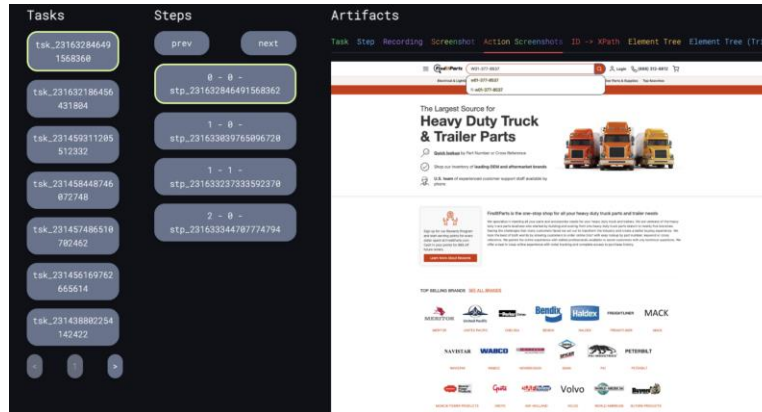
- 내부 문서 검색
- 오피스 도구 사용하여 문서 생성
- 다양한 Gen AI 이용한 생성



☑ 페르소나별 유저 인터페이스

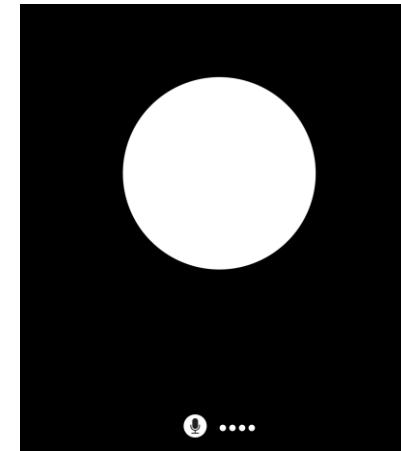
내근/사무직

- ☑ 데스크톱/브라우저
 - RPA / UI 자동화



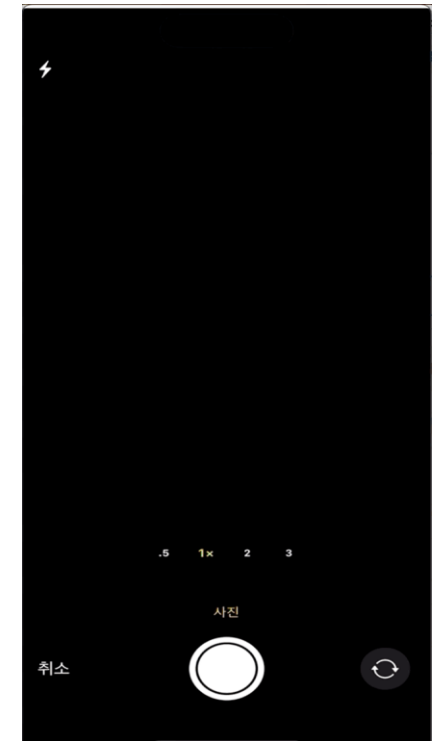
영업/임원

- ☑ 모바일 (운전중)
 - 음성인식 기반 대화

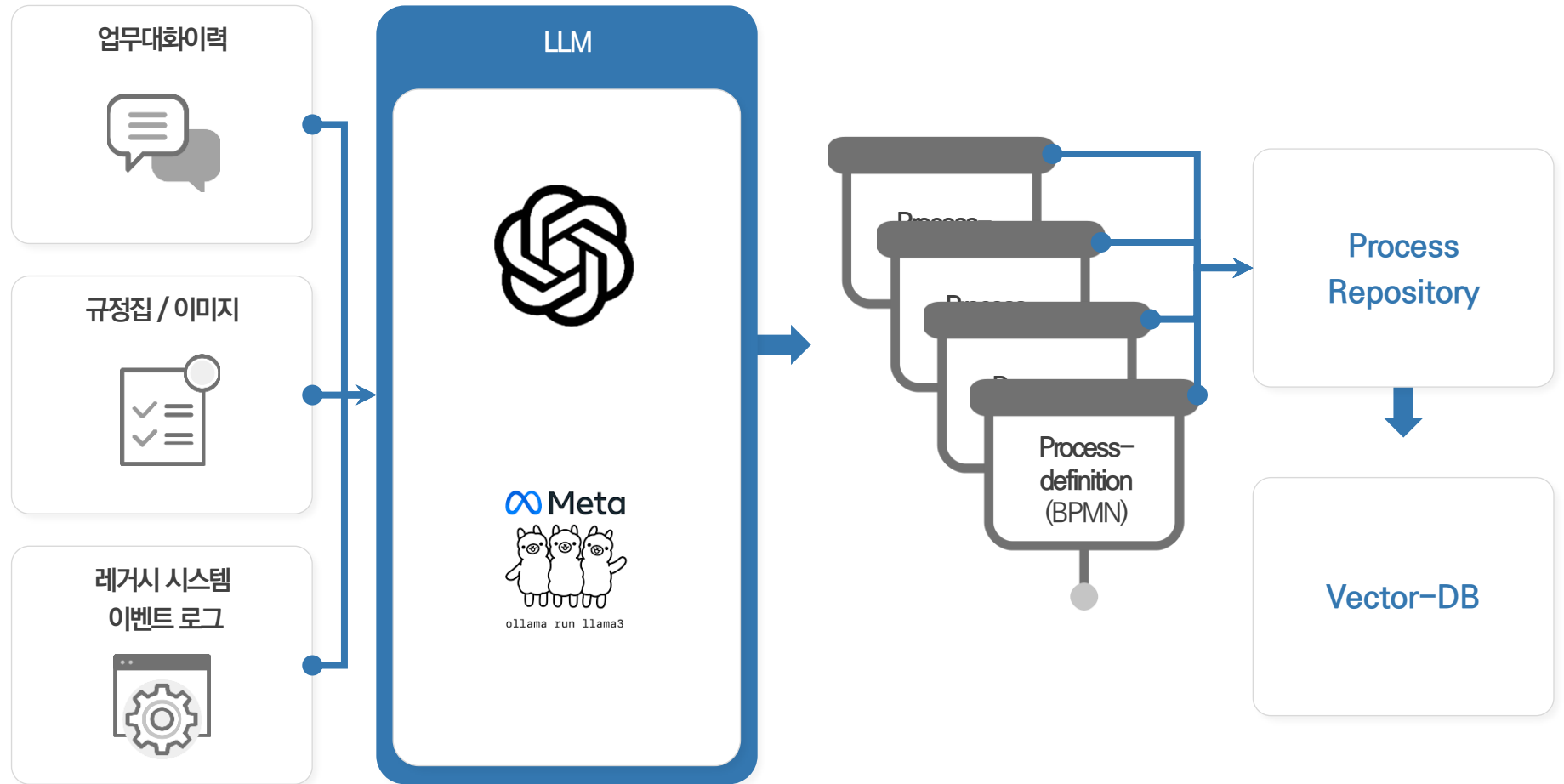


현장직

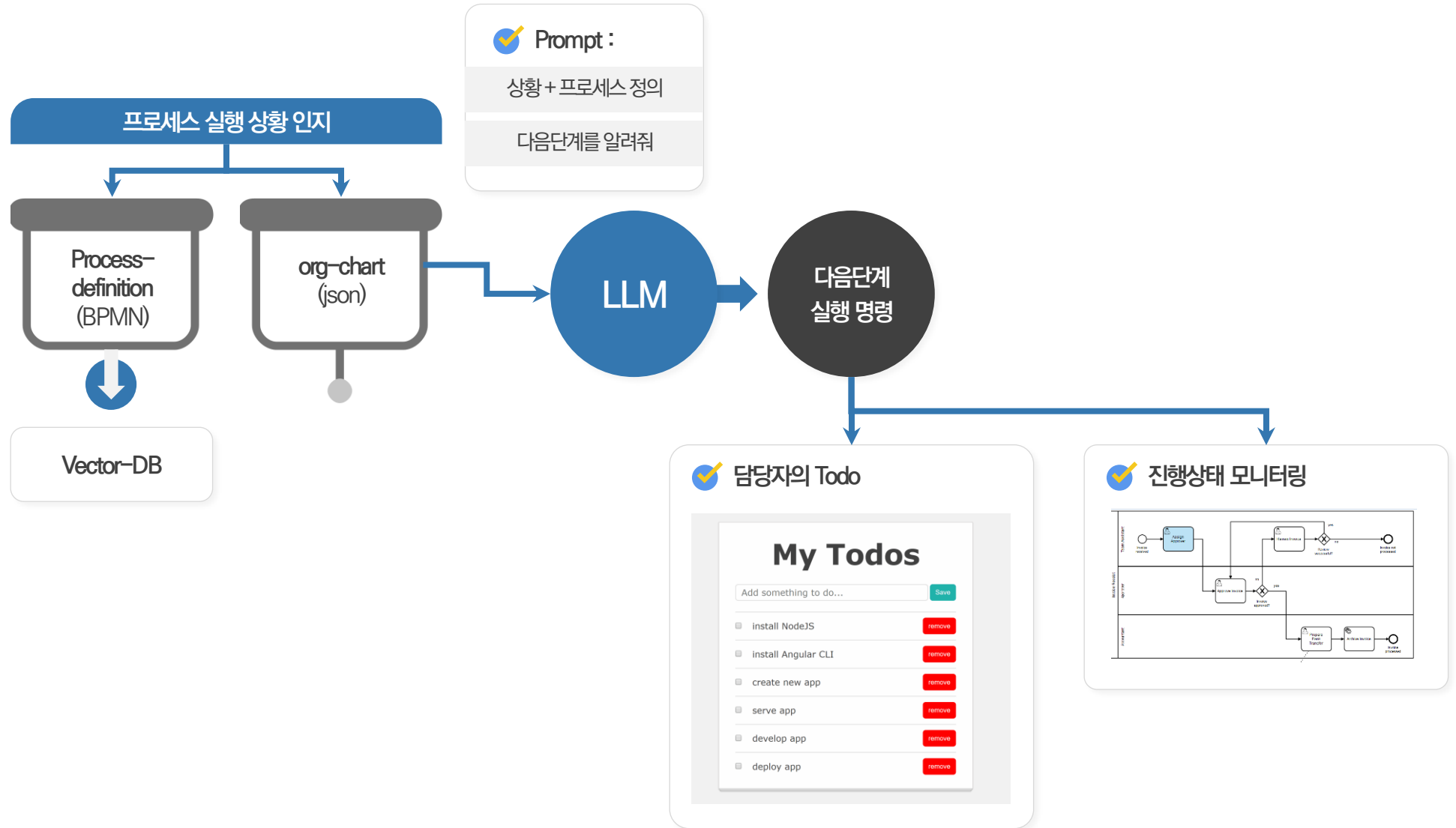
- ☑ 모바일/태블릿
 - 영상인식 / 음성인식



☑ Process Definition 단계



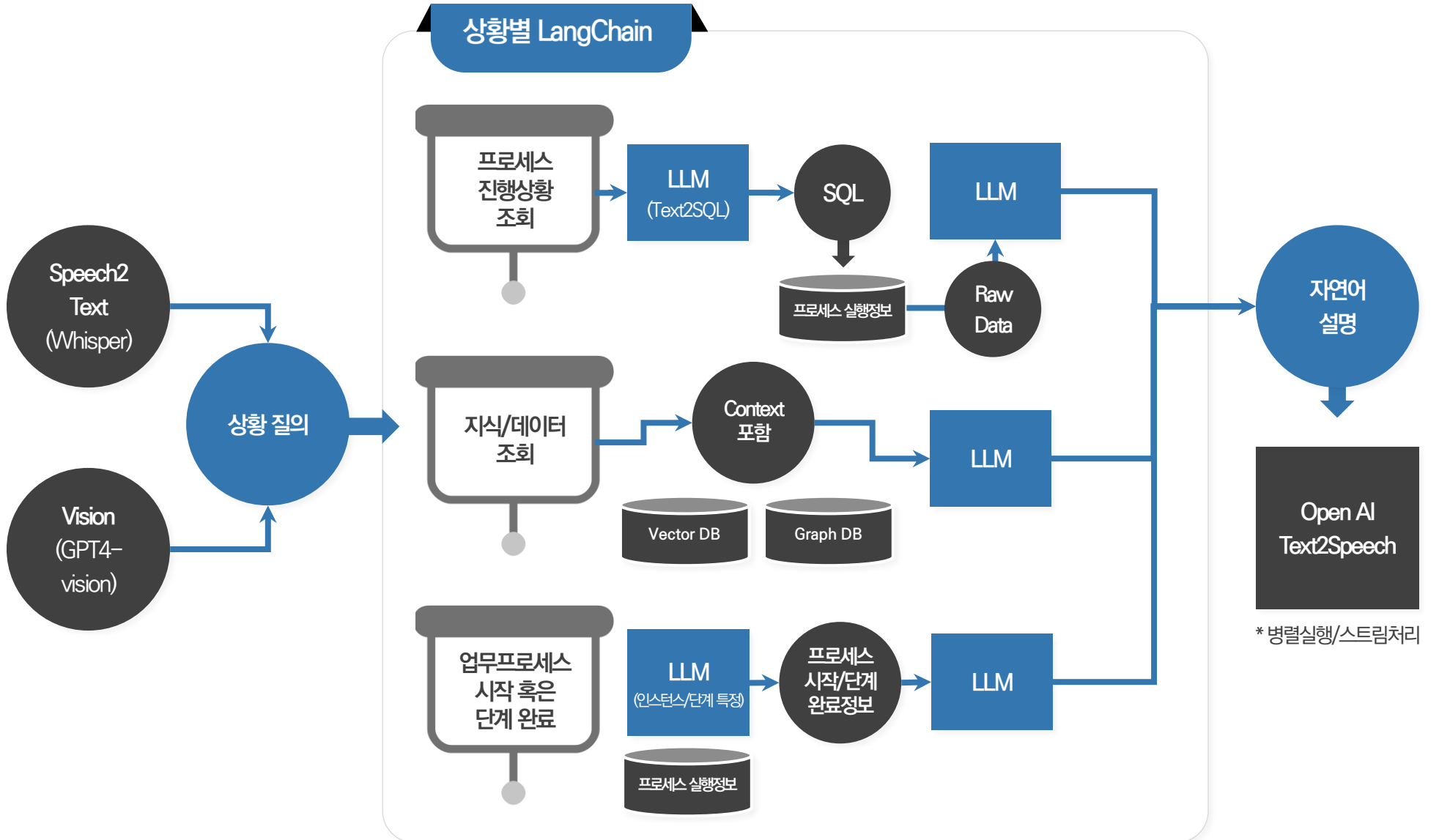
☑ Process Execution 단계



☑ Agent 기반 초안 생성



☑ 음성대화 기반 질의 응답



- ✓ AI를 기반한 정보 시스템 구축 방법은 **코드 기반과 코드가 없는 방법**이 있다
- ✓ 코드를 생성하는 전략은 **SW공학의 단계를 잘 나누어 구현하고 여러 서비스로 분해**하면 좋은 결과가 나온다
- ✓ **코드 없이 자연어를 기반한 구축 방법**으로 BPMN을 기반한 업무 프로세스의 실행과 시스템 연동이 가능하다
- ✓ 단순히 LLM 적용을 넘어 **사용자 경험에 어떻게 기능을 녹여내느냐**가 성공의 관건
- ✓ GPT4o와 같은 **생성형 AI 기술**은 사용자 인터페이스를 그 어느때보다 쉽게 만들어 준다.
- ✓ **MSA Easy 와 Process GPT**는 이러한 도전이고 오픈소스이다
 - <https://www.uengine.org/>
 - <https://github.com/msa-ez/platform>
 - <https://github.com/uengine-oss>



Process GPT

