도면에서 부품표까지
현업의 기준을 AI와 함께 코드로 옮기다

분야 : 업무 생산성 개선을 위한 AI 활용

활용 도구 : ChatGPT

한 줄 소개 : 현업의 부품표 작성 규칙과 예외를 ChatGPT와 함께 Python 프로그램으로 구현하고, 실행 결과와 기준 부품표를 비교하며 오류를 추적하고 개선한 업무 자동화 사례입니다.

1. AI를 활용하게 된 계기

가. 도면에서 반복해 읽어야 하는 부품 정보

- 품질 업무에서는 도면에 흩어진 부품 정보를 모아 부품표를 작성하고, 상위 조립체와 하위 부품의 관계, 개정 정보와 누락 여부를 확인해야 합니다. 도면마다 표의 위치와 문자 표현이 달라 같은 항목을 반복해서 읽고 정리해야 했습니다.

나. 현업의 판단 기준을 반복해서 사용할 도구

- 제가 업무에서 사용하는 규칙과 예외를 ChatGPT에 설명하고 Python 프로그램으로 구현했습니다. 부품 정보를 정리하면서 원문을 보존하고, 판단이 불확실한 부분을 사람이 다시 확인할 수 있게 하는 것이 목표였습니다.

2. AI를 활용한 방법

가. 사람이 정한 업무 기준

1) 읽어야 할 항목과 관계 정의

- 도면 번호와 개정 정보, 부품 항목, 상위 조립체와 하위 부품 관계, 분류 기준과 예외를 정리해 ChatGPT에 전달했습니다.

가) 원문과 공란 보존

- 정보를 임의로 채우지 않고 원문과 공란을 보존하도록 기준을 정했습니다.

나) 불확실한 결과의 검토 표시

- 최신 도면 후보가 충돌하거나 표 구조가 불확실하면 검토 사유를 남기도록 했습니다.

나. AI의 구현 제안과 사람의 결과 확인

- ChatGPT는 구현 방법과 Python 코드를 제안했습니다. 저는 기대한 결과와 실제 결과를 대조하고, 잘못 나온 부분의 입력 조건과 판단 기준을 다시 설명했습니다. 요구사항, 구현, 실행과 결과 확인을 반복하며 프로그램을 개선했습니다.
- 프로그램은 DXF의 문자, 속성, 표 객체와 표선 위치를 이용합니다. DWG는 외부 변환기를 거쳐 DXF로 처리합니다. AI는 개발 과정에 활용했고, 프로그램은 구현된 규칙에 따라 도면을 처리합니다.

3. 실행 결과와 검증

- 부품 정보 추출, 계층 정리, 분류와 검토 대상 표시를 하나의 흐름으로 연결했습니다. 결과는 「BOM」, 「검토필요」, 「도면목록」, 「미사용도면」, 「누락도면」의 다섯 시트로 출력합니다. 기준 결과와 새 결과를 비교해 추가·삭제 행, 바뀐 값과 순서도 확인하도록 구현했습니다.
- 제출 자료에는 공개용 가상 DXF 예제로 프로그램의 동작을 재현한 결과를 실었습니다. DXF 7개에 도면 번호 6개와 RA·RB 개정을 포함하고, 수정하지 않은 v9.3 원본으로 DXF 표 판독부터 계층 정리와 Excel 저장까지 실행했습니다.
- 실행 전에 예제의 설계 명세로 기준값을 별도로 정했습니다. 생성된 BOM 8행의 13개 항목(공란 포함), 총 104개 값을 대조해 모두 일치함을 확인했습니다. 최신 RB 선택과 부모가 다른 동일 부품 행의 개별 유지도 확인했습니다.

가. 공개용 예제 실행 기록

공개용 DXF 예제 실행 01 / 03

v9.3 프로그램 실행 기록

DWG BOM Builder v9.3 원본으로 공개용 예제를 실행한 기록

프로그램	DWG BOM Builder v9.3 · 원본 파일 수정 없음
실행 환경	Linux · Python 3.12.14 · OCR OFF
입력 도면	공개 DXF 7개 · 도번 6개 · 최상위 99000001
실행 결과	정상 종료(코드 0) · BOM 8행 · Excel 저장 완료

실제 실행 로그 발췌

```
전체 DWG/DXF 파일: 7개
인식 성공: 7개
인덱싱 완료: 도번 6개
DXF 판독 상태: 판독실패 도번 0개 / 판독실패 시트 0개
BOM 행: 8 / 검토필요: 0
사용 도면: 6 / 미사용 도면: 0 / 누락 참조: 0
```

실행 범위: DXF 표 판독 · 계층 전개 · Excel 저장 (DWG 변환 미실행)

실제 실행 로그의 관련 부분을 발췌해 정리했습니다.

원본 SHA-256: 702c3a8bc2eb0634… / 실행 UTC: 2026-10-15T14:03:33

자료 1. v9.3 원본의 실제 콘솔 실행 로그를 발췌해 정리했습니다. 공개용 DXF 7개에서 도면 번호 6개를 읽어 BOM 8행을 생성했습니다.

나. 같은 실행에서 생성된 Excel 결과

공개용 DXF 예제 실행 02 / 03

Excel BOM 생성 결과

원본 생성 Excel · BOM 시트 주요 열 발췌 (8행 전체)

99000001_BOM_v9.3_공개예제.xlsx

생성 시트 5개: BOM / 검토필요 / 도면목록 / 미사용도면 / 누락도면

행	BOM레벨 G열	품명(한글) K열	도면번호 M열	원생산자 부품번호 N열	수량 J열
2	1	책상 정리함 조립체	99000001		1
3	....ㄴ2	받침 조립체	99000002		2
4	........ㄴ3	지지대	99000005		2
5	........ㄴ3	간격재	99000006		4
6	........ㄴ3	나사		SCREW-M4X12	4
7	....ㄴ2	바닥판	99000003		1
8	....ㄴ2	덮개	99000004		1
9	....ㄴ2	나사		SCREW-M4X12	8

수량은 각 상위조립체 기준 원도면 수량입니다.

원본 XLSX 읽기 전용 발췌 · 빈 칸은 원본 출력의 빈 값 · 값과 서식 수정 없음

자료 2. 같은 실행에서 생성된 BOM 8행의 주요 열입니다. 계층·품명·도면 번호·부품 번호·원본 수량을 발췌했습니다.

다. 기준 부품표와 자동 생성 결과 대조

공개용 DXF 예제 실행 03 / 03

기준값과 실행 결과 대조

실행 전에 별도로 선언한 공개 예제 기준값과 실제 출력 비교

13개 필드 × 8행 = 104개 값 일치 / 불일치 0개

기준: expected.json · 실제: 원본 생성 XLSX · 행 순서도 일치

행	품명	차상위 조립체 품번	Level 기대 / 결과	수량 기대 / 결과	13개 필드 대조
2	책상 정리함 조립체		1 / 1	1 / 1	13 / 13 일치
3	받침 조립체	99000001	2 / 2	2 / 2	13 / 13 일치
4	지지대	99000002	3 / 3	2 / 2	13 / 13 일치
5	간격재	99000002	3 / 3	4 / 4	13 / 13 일치
6	나사	99000002	3 / 3	4 / 4	13 / 13 일치
7	바닥판	99000001	2 / 2	1 / 1	13 / 13 일치
8	덮개	99000001	2 / 2	1 / 1	13 / 13 일치
9	나사	99000001	2 / 2	8 / 8	13 / 13 일치

대조 항목: 부모 · 계층 · 품명 · 수량 · 도면번호 · 부품번호 등 13개 필드

수량은 각 상위조립체 기준 원도면 수량입니다.

이번 공개용 DXF 예제 8행·13개 항목의 대조 결과입니다.

자료 3. 실행 전 기준값과 8행의 13개 항목(공란 포함)을 대조해 104개 값이 모두 일치했습니다.

4. 오류를 발견하고 개선한 과정

가. 회로도 문자를 부품으로 잘못 읽은 문제

- 초기 프로그램을 다시 실행해 확인하던 중 부품 목록에 들어가면 안 되는 회로도 문자가 부품처럼 추출되는 오류를 발견했습니다. 숫자와 문자가 정렬된 모습만으로 판단하면 신호명이나 핀 번호가 나열된 영역도 잘못 포함할 수 있었습니다.

나. 오류 상황을 판정 규칙으로 구체화

1) 표 구조와 부품표 문맥 확인

- 오류 상황과 기대 결과를 ChatGPT에 설명했습니다. 현재 코드에는 표선과 셀 구조, 반복 신호명, 핀 번호와 배선 주석에 더해 도면 번호, 규격과 제조사 문맥을 함께 살피는 규칙을 담았습니다.

2) 제외 이유 기록과 정상 표 보호

- 정상 부품표를 과하게 제외하지 않도록 문맥을 함께 확인하고, 후보를 제외한 이유는 로그에 남기도록 했습니다. 오류 발견과 판단 기준 설명은 제가 맡고 구현과 수정에는 ChatGPT의 도움을 받았습니다.

5. 다른 사람이 활용하는 방법

가. 기준과 작은 예제부터 준비

- 반복해서 읽고 정리하는 항목을 정의하고, 공개 가능한 일반 예제와 기준 결과를 준비합니다. AI에 한 기능씩 구현하도록 요청한 뒤 실제 실행 결과를 직접 대조합니다.

나. 정상 사례와 예외 사례를 함께 확인

- 오류가 나오면 입력 조건, 기대 결과와 실제 결과를 구분해 설명합니다. 정상 예제와 예외 예제를 함께 확인하고 불확실한 결과는 사람이 검토하도록 표시합니다. 검증 결과를 공유할 때에는 시험 범위와 기준을 함께 적어 다른 사람이 같은 방법으로 확인할 수 있게 합니다.

6. 구현 근거와 자료 출처

가. 회로 신호표 판정 코드

회로·배선 신호표 제외와 사유 기록

실제 프로그램 소스코드 발췌 | DWG BOM Builder v9.3

```
if _headerless_looks_like_schematic_signal_table(items):
    dbg.append(
        f"v6.2 회로/배선 신호표 제외: ITEM {items[0].item_no}~{items[-1].item_no}, "
        f"{len(items)}행 / 반복 신호·넷 패턴"
    )
    return [], dbg, ""
```

원본 17019~17024행 | 판정 및 출력 구조의 구현 근거

자료 4. v9.3 원본 소스코드 중 회로·배선 신호표를 제외하고 이유를 기록하는 부분입니다. 4항의 개선 과정을 설명하는 구현 근거입니다.

나. 실행 자료의 범위와 출처

- 프로그램 구현은 직접 개발한 DWG BOM Builder v9.3 소스코드를 바탕으로 설명했습니다. 3항에는 이번에 별도로 구성한 공개용 가상 DXF 예제의 실제 실행 로그, 원본 생성 Excel과 실행 전 기준값 대조 결과를 제시했습니다.
- 이번 공개 예제는 OCR을 사용하지 않고 DXF의 문자와 표선을 판독했습니다. 검증 범위는 별도로 정한 기준값과 BOM 8행의 13개 항목(공란 포함) 대조입니다. 회로 신호표 제외 기능은 위 코드 발췌로 구현 근거를 제시했습니다.
- 예제 도면, 설계 명세와 기준값, 생성된 Excel, 실행 로그와 대조 결과는 별도 공개용 실행 자료 묶음에 함께 정리했습니다.