😀

기능 명세서

WPF Klarf Review Tool 개발 기능 명세서

1. 문서 개요

- **프로젝트명**: Klarf Viewer Tool
- **주요 기술**: `C#, WPF, .NET Framework, MVVM 디자인 패턴`
- **설명:** 반도체 웨이퍼 검사 결과인 **Klarf Format 파일**을 읽고 사용자에게 **결함 정보를 제공하는 프로그램을 제작**한다. 전체 웨이퍼 시각 정보, 개별 결함 이미지 정보, 파일 탐색기, 결함 상세 리스트 등을 제공해야 한다.(Wafer Map Viewer, Defect Image Viewer, File List Viewer, Defect List Viewer) 각 뷰어는 사용자 입력에 따라 **동기화 되어 더 나은 UX** 를 제공해야 한다.

2. 시스템 아키텍처

- **플랫폼**: Windows Desktop Application
- **아키텍처**: **MVVM (Model-View-ViewModel) 패턴**

3. 기능 요구사항 (Functional Requirements)

FR-1: 파일 관리 및 조회

- **FR-1.1 (폴더 열기)**
  - 사용자는 **[Open Folder]** 버튼을 통해 로컬 시스템의 특정 폴더를 선택할 수 있어야 한다.
- **FR-1.2 (폴더 목록 표시)**
  - 폴더 선택 시, 해당 폴더 내부에 폴더가 있다면 **File List Viewer**에 **Tree View 형태**로 표시되어야 한다.
- **FR-1.3 (파일 목록 표시)**
  - 특정 폴더 선택 시, 해당 **폴더 내의 파일이 있다면 폴더 Tree View** 우측에 **List Viewer** 표시되어야 한다.
- **FR-1.4 (파일 선택 및 로딩)**
  - 사용자가 파일 목록에에서 특정 Klarf 파일을 선택하면, 시스템은 해당 파일을 로드하여 파싱을 시작해야 한다.
- **FR-1.5 (파일 파싱 상태 바)**
  - 상태바를 통하여서 파싱 진행률을 사용자에게 제공해야 한다.
- **FR-1.6 (선택 파일 초기화)**
  - 사용자는 **[Refresh]** 버튼을 통해 파싱된 정보를 초기화 하여 다른 파일을 선택할 수 있어야 한다.
- **WPF Klarf Review Tool 개발 기능 명세서**

1. 문서 개요

- **프로젝트명**: Klarf Viewer Tool
- **주요 기술**: `C#, WPF, .NET Framework, MVVM 디자인 패턴`
- **설명:** 반도체 웨이퍼 검사 결과인 **Klarf Format 파일**을 읽고 사용자에게 **결함 정보를 제공하는 프로그램을 제작**한다. 전체 웨이퍼 시각 정보, 개별 결함 이미지 정보, 파일 탐색기, 결함 상세 리스트 등을 제공해야 한다.(Wafer Map Viewer, Defect Image Viewer, File List Viewer, Defect List Viewer) 각 뷰어는 사용자 입력에 따라 **동기화 되어 더 나은 UX** 를 제공해야 한다.

2. 시스템 아키텍처

- **플랫폼**: Windows Desktop Application
- **아키텍처**: **MVVM (Model-View-ViewModel) 패턴**

3. 기능 요구사항 (Functional Requirements)

FR-1: 파일 관리 및 조회

- **FR-1.1 (폴더 열기)**
  - 사용자는 **[Open Folder]** 버튼을 통해 로컬 시스템의 특정 폴더를 선택할 수 있어야 한다.
- **FR-1.2 (폴더 목록 표시)**
  - 폴더 선택 시, 해당 폴더 내부에 폴더가 있다면 **File List Viewer**에 **Tree View 형태**로 표시되어야 한다.
- **FR-1.3 (파일 목록 표시)**
  - 특정 폴더 선택 시, 해당 **폴더 내의 파일이 있다면 폴더 Tree View** 우측에 **List Viewer** 표시되어야 한다.
- **FR-1.4 (파일 선택 및 로딩)**
  - 사용자가 파일 목록에에서 특정 Klarf 파일을 선택하면, 시스템은 해당 파일을 로드하여 파싱을 시작해야 한다.
- **FR-1.5 (파일 파싱 상태 바)**
  - 상태바를 통하여서 파싱 진행률을 사용자에게 제공해야 한다.
- **FR-1.6 (선택 파일 초기화)**
  - 사용자는 **[Refresh]** 버튼을 통해 파싱된 정보를 초기화 하여 다른 파일을 선택할 수 있어야 한다.
- **FR-1.7 (이미지 저장)**
  - 사용자는 **[Save All Iamge]** 버튼을 통해 `Tiff 이미지 파일`을 다른이름으로 저장 할 수 있어야 한다..

FR-2: Wafer Map Viewer

- **FR-2.1 (Wafer Map 렌더링)**

    - **Klarf 파일**의 정보를 기반으로 **Wafer Map**을 시각적으로 **렌더링**해야 한다.
    - 각 **Die**는 사각형 형태로 그려지며, **Wafer 내의 상대적 위치**에 배치해야 한다.
- **FR-2.2 (불량 Die 강조)**
    - 하나 이상의 **Defect**를 포함하는 **Die**는 정상 **Die**와 명확히 구분되는 강조 표시를 해야 한다.
- **FR-2.3 (선택 동기화)**
- `Defect Info Viewer`에서 특정 **Defect** 선택 시, 해당 **Defect**가 속한 **Die**가 **Wafer Map** 상에서 시각적으로 강조(예: 테두리 색상 변경)되어야 한다.
- **FR-2.4 (Die 클릭 이벤트 - 동기화)**
    - 사용자가 **Wafer Map**의 특정 **Die**를 클릭하면, `Defect Image Viewer` 와 `Defect Info Viewer` 는 해당 **Die** 의 **Defect** 이미지를 제공해야 한다.

FR-3: Defect Info Viewer

- **FR-3.1 (불량 리스트 표시)**
    - **Klarf** 파일의 **Defect** 정보를 **리스트 형태**로 출력해야 한다.
- **FR-3.2 (불량 선택 기능)**
    - 사용자는 Defect 리스트에서 마우스 클릭으로 특정 **Defect** 항목을 선택할 수 있어야 한다.
    - 항목 선택 시, 선택된 Defect의 정보가 `Defect Image Viewer`와 `Wafer Map Viewer`에 즉시 반영되어야 한다.
        - `Defect Image Viewer`: 해당 **Defect의 TIF 이미지**로 변경
        - `Wafer Map Viewer`: 해당 **Defect의 Die 위치** 강조
- **FR-3.3(Wafer Info 제공)**
    - Wafer ID, Lot ID, Timestamp, Device ID을 제공해야 한다.
- **FR-3.4 (Defect Die 이동 버튼)**
    - [◀], [▶] 버튼을 통해 Defect 리스트의 `DEFECTID`를 따라 순차적으로 탐색할 수 있어야 한다.
    - 현재 선택된 항목의 인덱스와 전체 개수를 `(현재 인덱스) / (전체 DEFECT DIE 개수)` 형태로 표시해야 한다. (예: `1 / 453`)
- **FR-3.5 (Die 이동 버튼)**
    - [◀], [▶] 버튼을 통해 **불량이 있는 Die** 간에 이동할 수 있어야 한다.
    - 현재 선택된 Die의 인덱스와 전체 불량 Die 개수를 표시해야 한다. (예: `1 / 736`)
- **FR-3.6 (Defect in Die 이동 버튼)**
    - [◀], [▶] 버튼을 통해 **2개 이상 불량**을 가진 **Die** 내 **Defect** 이동이 가능해야 한다. (예: `1 / 2`)

FR-4: Defect Image Viewer

- **FR-4.1 (TIF 이미지 표시)**
    - 선택된 Defect에 해당하는 TIF(.tif) 이미지를 표시해야 한다.
- **FR-4.2 (이미지 확대/축소)**
    - 사용자는 **Bar UI(Slider)** 또는 버튼을 통해 **이미지의 배율**을 조절할 수 있어야 한다.
- **FR-4.3 (드래그를 이용한 사이즈 측정)**:
    - 사용자가 이미지 위에서 마우스를 클릭하고 드래그하면, **시작점과 끝점을 잇는 직선**이 그려져야 한다.
    - 드래그가 끝나면, 해당 직선의 길이(단위: pixel)를 계산하여 화면의 특정 위치에 표시해야 한다
- **FR-4.4**

FR-2: Wafer Map Viewer

- **FR-2.1 (Wafer Map 렌더링)**
  - **Klarf 파일**의 정보를 기반으로 **Wafer Map**을 시각적으로 **렌더링**해야 한다.
  - 각 **Die**는 사각형 형태로 그려지며, **Wafer 내의 상대적 위치**에 배치해야 한다.
- **FR-2.2 (불량 Die 강조)**
  - 하나 이상의 **Defect**를 포함하는 **Die**는 정상 **Die**와 명확히 구분되는 강조 표시를 해야 한다.
- **FR-2.3 (선택 동기화)**
- `Defect Info Viewer`에서 특정 **Defect** 선택 시, 해당 **Defect**가 속한 **Die**가 **Wafer Map** 상에서 시각적으로 강조(예: 테두리 색상 변경)되어야 한다.
- **FR-2.4 (Die 클릭 이벤트 - 동기화)**
  - 사용자가 **Wafer Map**의 특정 **Die**를 클릭하면, `Defect Image Viewer` 와 `Defect Info Viewer` 는 해당 **Die** 의 **Defect** 이미지를 제공해야 한다.

FR-3: Defect Info Viewer

- **FR-3.1 (불량 리스트 표시)**
  - **Klarf** 파일의 **Defect** 정보를 **리스트 형태**로 출력해야 한다.
- **FR-3.2 (불량 선택 기능)**
  - 사용자는 Defect 리스트에서 마우스 클릭으로 특정 **Defect** 항목을 선택할 수 있어야 한다.
  - 항목 선택 시, 선택된 Defect의 정보가 `Defect Image Viewer`와 `Wafer Map Viewer`에 즉시 반영되어야 한다.
    - `Defect Image Viewer`: 해당 **Defect의 TIF 이미지**로 변경
    - `Wafer Map Viewer`: 해당 **Defect의 Die 위치** 강조
- **FR-3.4 (Defect Die 이동 버튼)**
  - [◀], [▶] 버튼을 통해 Defect 리스트의 `DEFECTID`를 따라 순차적으로 탐색할 수 있어야 한다.
  - 현재 선택된 항목의 인덱스와 전체 개수를 `(현재 인덱스) / (전체 DEFECT DIE 개수)` 형태로 표시해야 한다. (예: `1 / 453`)
- **FR-3.5 (Die 이동 버튼)**
  - [◀], [▶] 버튼을 통해 **불량이 있는 Die** 간에 이동할 수 있어야 한다.
  - 현재 선택된 Die의 인덱스와 전체 불량 Die 개수를 표시해야 한다. (예: `1 / 736`)
- **FR-3.6 (Defect in Die 이동 버튼)**
  - [◀], [▶] 버튼을 통해 **2개 이상 불량**을 가진 **Die** 내 **Defect** 이동이 가능해야 한다. (예: `1 / 2`)

FR-4: Defect Image Viewer

- **FR-4.1 (TIF 이미지 표시)**
  - 선택된 Defect에 해당하는 TIF(.tif) 이미지를 표시해야 한다.
- **FR-4.2 (이미지 확대/축소)**
  - 사용자는 **Bar UI(Slider)** 또는 버튼을 통해 **이미지의 배율**을 조절할 수 있어야 한다.
- **FR-4.3 (드래그를 이용한 사이즈 측정)**:
  - 사용자가 이미지 위에서 마우스를 클릭하고 드래그하면, **시작점과 끝점을 잇는 직선**이 그려져야 한다.
  - 드래그가 끝나면, 해당 직선의 길이(단위: pixel)를 계산하여 화면의 특정 위치에 표시해야 한다.

4. 비기능 요구사항 (Non-Functional Requirements)

- **NFR-1 (성능)**: 10MB 미만의 일반적인 Klarf 파일은 3초 이내에 로딩 및 파싱이 완료되어야 한다. UI의 모든 상호작용(항목 선택, 버튼 클릭 등)은 200ms 이내에 응답하여 끊김 없는 사용자 경험을 제공해야 한다.
- **NFR-2 (안정성)**: 유효하지 않은 형식의 Klarf 파일을 열거나, 이미지 파일이 없는 경우 등의 예외 상황에서 프로그램이 강제 종료되지 않고, 사용자에게 적절한 오류 메시지를 안내해야 한다.
- **NFR-3 (유지보수성)**: ATI 코딩 표준 및 주석 형식을 철저히 준수해야 한다. MVVM 패턴을 통해 각 모듈의 의존성을 최소화하여 기능 수정 및 확장이 용이한 구조로 설계해야 한다.
- **NFR-4 (개발 프로세스)**:
  - **버전 관리**: 모든 소스 코드는 Git을 통해 관리되어야 한다.
  - **커밋 정책**: 작업 내용은 의미 있는 단위로 분할하여 **매일 최소 1회 이상 커밋(Daily Commit)**하는 것을 원칙으로 한다.
