You're right, I apologize for the confusion. Let's modify the solution to use the Semi-ATE/Metis library instead, which provides proper STDF handling capabilities.

I've updated the solution to use the proper Semi-ATE Metis library instead. This revised script will handle STDF files correctly according to your requirements.

**Updated Solution:**

The script now uses Semi\_ATE.STDF from the Metis project to read and write STDF files. It maintains the same functionality:

1. Processes sequential STDF files chronologically
2. Identifies the same part across multiple test runs
3. Keeps only the most recent test results for each part
4. Properly handles both parametric (PTR) and functional (FTR) test records
5. Preserves global records and maintains the correct STDF structure

**Installation:**

First, install the Semi-ATE Metis library:

pip install Semi-ATE-Metis

**Usage:**

python stdf\_merger.py -i file1.stdf file2.stdf file3.stdf -o merged.stdf -v

**Key Features:**

* **Part Identification**: Uses head number, site number, and part ID (or XY coordinates) to identify unique parts
* **Record Management**: Handles all STDF record types correctly, maintaining their relationships
* **Test Merging**: When a part appears in multiple files, keeps only the most recent test data
* **Timestamp Updates**: Updates timestamps in header and footer records
* **Part Counting**: Updates part count records to match the final merged data

This solution should properly handle your workflow where the first file contains all parts and subsequent files only contain retests of failed parts or tests, producing a clean STDF file with only the final test results.

Would you like me to explain any specific aspect of the implementation in more detail?