1. **Forms and Modules**:
   * **frmRebuiltPartsMain.frm**: The main form that serves as the entry point for managing rebuilt parts. It handles the display and manipulation of data related to rebuilt parts, labor, and materials.
   * **frmRebuiltPartsAddMaterials.frm**: A form for adding and managing materials associated with rebuilt parts.
   * **frmRebuiltPartEstimateOptions.frm**: A form that provides options for loading or copying rebuilt part estimates.
   * **frmCopyRBEstimate.frm**: A form for copying existing rebuilt part estimates.
   * **frmAddRBPToEstimate.frm**: A form for adding or updating rebuilt parts within an estimate.
   * **frmAddLabourTask.frm**: A form for adding or updating labor tasks associated with rebuilt parts.
   * **frmAddNSCPToEstimate.frm**: A form for adding or updating non-stock coded parts within an estimate.
   * **frmAddLabourGroup.frm**: A form for adding or editing employee groups (labor groups) with specific job details.
   * **frmConvertToRebuiltPart.frm**: A form for converting a "Make vs Buy" (MB) part into a rebuilt part (RB).
2. **Database Interaction**:
   * The application interacts with a database using ADODB (ActiveX Data Objects) to retrieve, update, and delete records. The database operations are encapsulated in helper functions within the modDBHelper module.
   * The application uses stored procedures (e.g., CES\_INQUIRY.RETRIEVE\_RB, CES\_UPDATE.ADD\_RBP\_USED) to perform database operations.
3. **Business Rules**:
   * **Access Control**: The application enforces different levels of access control based on the user's access level (g\_l\_AccessLevel). For example, certain menu options and grid columns are disabled or made invisible for users with lower access levels.
   * **Data Validation**: The application includes extensive data validation to ensure that inputs (e.g., quantities, percentages, costs) are valid before performing database operations.
   * **Audit Logging**: The application logs changes to the database in an audit table (AUDIT\_MODIFY\_HISTORY), recording details such as the user who made the change, the type of change, and the affected records.
   * **Cost Calculation**: The application calculates various costs (e.g., labor subtotal, overhead amount, total cost) based on user inputs and database values.
   * **Part Conversion**: The application allows for the conversion of non-stock coded parts to stock coded parts and vice versa, ensuring that no duplicates are created during the conversion process.
   * **Archiving**: The application supports archiving of rebuilt parts, which involves moving records to an archived state and updating related records in the database.

**Business Rules and Logic**

1. **User Permissions**:
   * The application restricts access to certain features based on the user's access level. For example, users with g\_l\_AccessLevel = 1 (lowest access level) cannot add labor tasks, view archived databases, or edit certain fields.
2. **Data Validation**:
   * The application validates user inputs before performing database operations. For example, it checks that quantities are positive numbers, percentages are between 0 and 100, and costs are valid numeric values.
3. **Cost Calculations**:
   * The application calculates various costs, such as labor subtotal, overhead amount, and total cost, based on user inputs and database values. These calculations are used to provide accurate cost estimates for rebuilt parts.
4. **Audit Logging**:
   * The application logs all significant changes to the database in an audit table. This includes adding, updating, and deleting records. The audit log records details such as the user who made the change, the type of change, and the affected records.
5. **Part Conversion**:
   * The application allows for the conversion of non-stock coded parts to stock coded parts. This process involves checking for duplicates and ensuring that the conversion does not create invalid cycles of rebuilt parts.
6. **Archiving**:
   * The application supports archiving of rebuilt parts, which involves moving records to an archived state and updating related records in the database. This process ensures that archived parts are no longer active but can still be referenced in historical data.

**Example Workflow**

1. **Adding a Rebuilt Part**:
   * The user navigates to the main form (frmRebuiltPartsMain.frm) and selects the option to add a rebuilt part.
   * The user enters details such as the rebuilt part number, description, and associated materials.
   * The application validates the inputs and checks for duplicates.
   * If the inputs are valid, the application adds the rebuilt part to the database and logs the change in the audit table.
2. **Adding a Labor Task**:
   * The user navigates to the labor task form (frmAddLabourTask.frm) and enters details such as the task description, labor position, and hours required.
   * The application validates the inputs and checks that the labor position and task description are valid.
   * If the inputs are valid, the application adds the labor task to the database and logs the change in the audit table.
3. **Converting a Part**:
   * The user navigates to the conversion form (frmConvertToRebuiltPart.frm) and enters details such as the rebuilt part number and stock coded number.
   * The application checks that the rebuilt part number is not already in use and that the stock coded number is valid.
   * If the inputs are valid, the application converts the part and updates the database, logging the change in the audit table.

**Conclusion**

The provided code is part of a comprehensive system for managing rebuilt parts, labor tasks, and materials in a manufacturing or repair environment. The application enforces strict business rules, including access control, data validation, cost calculation, audit logging, and part conversion. The code is well-structured, with clear separation of concerns between forms, database interaction, and business logic.

If you have specific questions about certain parts of the code or need further clarification on the business rules, feel free to ask!

can you pls tell me how the grid values (specially esitmates)are getting calculated? Name the stored proc uses and show me the flow, starting from adding a new built part and getting the final estimation

(Due to technical issues, the search service is temporarily unavailable.)

Certainly! Let's break down the process of how the grid values, particularly the estimates, are calculated in your application. We'll start from adding a new rebuilt part and follow the flow through to the final estimation. I'll also highlight the stored procedures involved and the key steps in the process.

**1. Adding a New Rebuilt Part**

When a new rebuilt part is added, the following steps occur:

**Stored Procedures Involved:**

* **CES\_UPDATE.ADD\_RBP\_USED**: Adds a new rebuilt part to the database.
* **CES\_INQUIRY.RETRIEVE\_RB**: Retrieves details of the rebuilt part for validation and display.

**Flow:**

1. The user enters details for the new rebuilt part (e.g., rebuilt number, description, keyword, etc.) in the form (frmAddRBPToEstimate.frm).
2. The application validates the inputs (e.g., checks for duplicates, ensures valid percentages, quantities, etc.).
3. The CES\_UPDATE.ADD\_RBP\_USED stored procedure is called to insert the new rebuilt part into the database.
4. The application retrieves the newly added part's details using CES\_INQUIRY.RETRIEVE\_RB to populate the grid and display the updated list of rebuilt parts.

**2. Adding Materials and Labor to the Rebuilt Part**

Once the rebuilt part is added, materials and labor tasks can be associated with it. These contribute to the final cost estimate.

**Stored Procedures Involved:**

* **CES\_UPDATE.ADD\_NSCP\_USED**: Adds non-stock coded parts (materials) to the rebuilt part.
* **CES\_UPDATE.ADD\_SCP\_USED**: Adds stock coded parts (materials) to the rebuilt part.
* **CES\_UPDATE.ADD\_LABOUR\_TASKS**: Adds labor tasks to the rebuilt part.
* **CES\_INQUIRY.RETRIEVE\_NSCP\_USED**: Retrieves non-stock coded parts used in the rebuilt part.
* **CES\_INQUIRY.RETRIEVE\_SCP\_USED**: Retrieves stock coded parts used in the rebuilt part.
* **CES\_INQUIRY.RETRIEVE\_MB\_LABOUR**: Retrieves labor tasks associated with the rebuilt part.

**Flow:**

1. The user adds materials (stock coded or non-stock coded parts) and labor tasks to the rebuilt part using the respective forms (frmAddNSCPToEstimate.frm, frmAddSCPToEstimate.frm, frmAddLabourTask.frm).
2. The application validates the inputs (e.g., ensures valid quantities, percentages, costs, etc.).
3. The appropriate stored procedure (CES\_UPDATE.ADD\_NSCP\_USED, CES\_UPDATE.ADD\_SCP\_USED, or CES\_UPDATE.ADD\_LABOUR\_TASKS) is called to insert the materials or labor tasks into the database.
4. The application retrieves the updated list of materials and labor tasks using CES\_INQUIRY.RETRIEVE\_NSCP\_USED, CES\_INQUIRY.RETRIEVE\_SCP\_USED, and CES\_INQUIRY.RETRIEVE\_MB\_LABOUR to populate the grid and display the updated data.

**3. Calculating the Final Estimate**

The final estimate is calculated by summing up the costs of materials, labor, and overhead. This is done in the Populate subroutine of frmRebuiltPartsMain.frm.

**Stored Procedures Involved:**

* **CES\_INQUIRY.RETRIEVE\_RBP\_TOTAL\_COST2**: Retrieves the total cost of materials used in the rebuilt part.
* **CES\_INQUIRY.RETRIEVE\_ALL\_LABOUR**: Retrieves the total labor hours and costs associated with the rebuilt part.
* **CES\_INQUIRY.RETRIEVE\_CES\_SETTI**: Retrieves system settings, such as the tax rate and overhead/fringe value.

**Flow:**

1. The Populate subroutine in frmRebuiltPartsMain.frm is called to refresh the form and calculate the final estimate.
2. The application retrieves the total cost of materials using CES\_INQUIRY.RETRIEVE\_RBP\_TOTAL\_COST2. This includes the cost of both stock coded and non-stock coded parts.
3. The application retrieves the total labor hours and costs using CES\_INQUIRY.RETRIEVE\_ALL\_LABOUR. This includes the labor subtotal and overhead amount.
4. The application retrieves system settings (e.g., tax rate, overhead/fringe value) using CES\_INQUIRY.RETRIEVE\_CES\_SETTI.
5. The application calculates the following:
   * **Labor Subtotal**: Sum of all labor costs.
   * **Overhead Amount**: Labor subtotal multiplied by the overhead/fringe value.
   * **Labor Total**: Labor subtotal plus overhead amount.
   * **Material Total**: Sum of all material costs.
   * **Core Cost**: Cost of the core part (if applicable).
   * **Total Rebuilt Part Cost**: Labor total plus material total minus core cost.
   * **Tax Total**: Material total multiplied by the tax rate.
   * **Total with Tax**: Total rebuilt part cost plus tax total.
6. The calculated values are displayed in the grid and text boxes on the form.

**4. Displaying the Final Estimate**

The final estimate is displayed in the grid and text boxes on the main form (frmRebuiltPartsMain.frm).

**Key Fields Displayed:**

* **Labor Subtotal**: Total cost of labor before overhead.
* **Overhead Amount**: Additional cost due to overhead/fringe.
* **Labor Total**: Total cost of labor including overhead.
* **Material Total**: Total cost of materials.
* **Core Cost**: Cost of the core part (if applicable).
* **Total Rebuilt Part Cost**: Sum of labor total and material total, minus core cost.
* **Tax Total**: Tax applied to the material total.
* **Total with Tax**: Final estimate including tax.

**Example Calculation**

Let’s assume the following values:

* **Labor Subtotal**: $500
* **Overhead/Fringe Value**: 20%
* **Material Total**: $1000
* **Core Cost**: $200
* **Tax Rate**: 10%

**Calculations:**

1. **Overhead Amount**: 500∗20500∗20100
2. **Labor Total**: 500+500+100 = $600
3. **Total Rebuilt Part Cost**: 600(LaborTotal)+600(*LaborTotal*)+1000 (Material Total) - 200(CoreCost)=200(*CoreCost*)=1400
4. **Tax Total**: 1000∗101000∗10100
5. **Total with Tax**: 1400+1400+100 = $1500

**Stored Procedures Summary**

Here’s a summary of the key stored procedures used in the process:

* **Adding Parts**:
  + CES\_UPDATE.ADD\_RBP\_USED: Adds a rebuilt part.
  + CES\_UPDATE.ADD\_NSCP\_USED: Adds a non-stock coded part.
  + CES\_UPDATE.ADD\_SCP\_USED: Adds a stock coded part.
  + CES\_UPDATE.ADD\_LABOUR\_TASKS: Adds a labor task.
* **Retrieving Data**:
  + CES\_INQUIRY.RETRIEVE\_RB: Retrieves rebuilt part details.
  + CES\_INQUIRY.RETRIEVE\_NSCP\_USED: Retrieves non-stock coded parts.
  + CES\_INQUIRY.RETRIEVE\_SCP\_USED: Retrieves stock coded parts.
  + CES\_INQUIRY.RETRIEVE\_MB\_LABOUR: Retrieves labor tasks.
  + CES\_INQUIRY.RETRIEVE\_RBP\_TOTAL\_COST2: Retrieves total material costs.
  + CES\_INQUIRY.RETRIEVE\_ALL\_LABOUR: Retrieves total labor costs.
  + CES\_INQUIRY.RETRIEVE\_CES\_SETTI: Retrieves system settings (e.g., tax rate, overhead value).

**Flow Summary**

1. **Add Rebuilt Part**: Use CES\_UPDATE.ADD\_RBP\_USED to add the part.
2. **Add Materials and Labor**: Use CES\_UPDATE.ADD\_NSCP\_USED, CES\_UPDATE.ADD\_SCP\_USED, and CES\_UPDATE.ADD\_LABOUR\_TASKS to add materials and labor.
3. **Calculate Estimate**:
   * Retrieve material costs using CES\_INQUIRY.RETRIEVE\_RBP\_TOTAL\_COST2.
   * Retrieve labor costs using CES\_INQUIRY.RETRIEVE\_ALL\_LABOUR.
   * Retrieve system settings using CES\_INQUIRY.RETRIEVE\_CES\_SETTI.
   * Perform calculations for labor subtotal, overhead, material total, core cost, tax, and total with tax.
4. **Display Estimate**: Populate the grid and text boxes with the calculated values.

If you need further clarification or a deeper dive into any specific part of the process, feel free to ask!

New chat