File Breakdown: `src/utils/simplified_parameter_collection.py`

File Location

`src/utils/simplified_parameter_collection.py`

Overview

The `simplified_parameter_collection.py` file implements a straightforward approach to collecting missing parameters from users. It provides a more direct parameter collection mechanism compared to the LLM-based approach used elsewhere in the system, making it more efficient for routine parameter gathering tasks while still maintaining clear descriptions and examples.

Key Responsibilities

- Collect missing parameters from users through console interaction
- Provide clear descriptions and examples for each parameter
- Support various parameter types for different functions
- Maintain a consistent user interface for parameter collection
- Log function calls for debugging and monitoring
- Enable extension with additional parameter descriptions

Core Functionality

Main Function Definition

```
@log_function_call
def get_missing_parameters_simple(function_name: str, missing_params: List[str],
                                  initial_args: Dict[str, Any] = None) -> Dict[str, Any]:
    """
    A simplified version of parameter collection that doesn't rely on LLM for
descriptions.

    Parameters:
        function_name (str): The name of the function requiring parameters
        missing_params (List[str]): List of parameter names that are missing
        initial_args (Dict[str, Any], optional): Initial arguments already collected

    Returns:
        Dict[str, Any]: Dictionary of collected parameters
    """
    collected_args = initial_args.copy() if initial_args else {}

    print(f"Nova needs to collect information for {function_name}...")

    # Parameter descriptions for common energy modeling parameters
    param_descriptions = {
        "location": "The geographic location for the energy model (e.g., UK, France,
Spain, etc.)",
        "generation": "The generation type for the model (e.g., solar, wind, hydro,
```

```
thermal, bio)",
        "energy_carrier": "The energy carrier to model (e.g., electricity, hydrogen,
methane)",
        "prompt": "The detailed prompt describing what you want to model",
        "scenario_name": "The name for this scenario",
        "analysis_type": "The type of analysis to perform (e.g., basic, detailed)",
        "style": "The report style to generate (e.g., executive_summary, detailed)"
    }

    # Examples for common parameters
    param_examples = {
        "location": "UK, France, Germany, or 'all' for all available locations",
        "generation": "solar, wind, hydro, thermal, bio, or 'all' for all types",
        "energy_carrier": "electricity (default), hydrogen, methane",
        "prompt": "A detailed description of what you want to model",
        "scenario_name": "baseline_2025, high_renewables_2030",
        "analysis_type": "basic, detailed, comprehensive",
        "style": "executive_summary, technical_report, presentation"
    }

    for param in missing_params:
        # Get description and examples from our predefined dictionaries, or use defaults
        description = param_descriptions.get(param, f"The {param} for {function_name}")
        examples = param_examples.get(param, "No examples available")

        # Create a simple prompt
        print(f"\nNova: I need the '{param}' for this task.")
        print(f"Description: {description}")
        print(f"Examples: {examples}")

        # Get user input
        user_response = input("> ").strip()

        # Store the response
        collected_args[param] = user_response

    return collected_args
```

Key Features

1. **Simplified Approach**: Direct parameter collection without relying on LLMs
2. **Parameter Descriptions**: Provides clear descriptions for common parameters
3. **Usage Examples**: Includes examples to guide the user's input
4. **Function Context**: Shows the function name for better context
5. **Default Handling**: Uses sensible defaults for unknown parameters
6. **Input Validation**: Trims whitespace from user input
7. **Clean Interface**: Presents a consistent, user-friendly interface

Integration

- Used as an alternative to LLM-based parameter collection
- Called by agents when they need to collect missing parameters
- Works alongside the knowledge base for storing collected values
- Compatible with both synchronous and asynchronous workflows

- Supports the function mapping system through consistent parameters

Workflow

1. Function receives a list of missing parameters and function context
2. For each parameter:
    - Retrieves the parameter description and examples
    - Displays a prompt with clear information
    - Collects the user's input
    - Stores the response in the result dictionary
3. Returns the complete collection of parameters

Implementation Notes

- Uses predefined dictionaries for parameter descriptions and examples
- Provides a more efficient alternative to LLM-based parameter collection
- Maintains a consistent interaction pattern for parameter gathering
- Focuses on energy modeling parameters but can be extended
- Preserves any initially provided arguments
- Uses plain console input for simplicity
- Could be extended with validation logic for parameter values