Milestone 7: Validation and Verification

Fatih Kurt

6 December, 2024

Contents

1 Introduction

Building upon the work completed in Milestone 6, Milestone 7 focuses on the validation and verification (V&V) of the simulation model. This step ensures the model accurately represents the real-world system and operates correctly. Validation checks whether the right model is being built, while verification ensures the model is built right.

Timeline Perspective: As this milestone precedes the final deliverable, it is essential to allocate sufficient time and effort to (V&V). Typically, validation and verification may require a significant but manageable amount of time compared to Milestone 6. Plan accordingly to ensure all aspects are thoroughly addressed.

Validation answers the question, "Are we building the right model?", while Verification answers, "Are we building the model right?" This guide provides a systematic approach to perform (V&V) effectively, ensuring your simulation is ready for the final submission.

2 Validation

2.1 Purpose of Validation

Validation ensures that the simulation model accurately represents the real-world system by comparing simulation outputs with real-world data or expert expectations.

2.2 Methods of Validation

2.2.1 Parameter Validation

Check whether the model parameters are within reasonable and acceptable ranges based on real-world data or literature.

2.2.2 Historical Data Validation

Compare the simulation outputs with historical data to validate the model's ability to replicate past behavior.

2.2.3 Cross-Model Validation

Compare your model's results with those from similar published models or simulations in the literature to assess consistency.

2.2.4 Face Validation

Compare the model's behavior with expert opinions or known behavior patterns of the real system.

2.2.5 Sensitivity Analysis Follow-up

Use insights from Milestone 6 to see if the model responds to parameter changes as expected.

2.3 Guidance on Validation

2.3.1 Documenting Validation Results

- Use tables, graphs, and statistical tests.
- Clearly label visuals and include units of measurement.
- Include explanations for each result, highlighting how they support the model's validity.

2.3.2 Addressing Discrepancies

- Identify and analyze discrepancies.
- Analyze the causes of discrepancies (e.g., model assumptions, data inaccuracies).
- Adjust the model if needed and re-validate.

2.3.3 Acceptable Margins of Error

- Define acceptable error margins based on industry standards.
- Example: $\pm 5\%$ for financial simulations or $\pm 1\%$ for engineering models.
- Justify the margins of error you consider acceptable.

3 Verification

3.1 Purpose of Verification

Verification ensures that the simulation model is implemented correctly without any logical or coding errors.

3.2 Methods of Verification

3.2.1 Unit Testing

Test individual components or functions of your simulation code to ensure they work correctly in isolation.

3.2.2 Integration Testing

Test the interaction between integrated components to identify issues in the interfaces and interaction between modules.

3.2.3 Regression Testing

Ensure that recent code changes have not adversely affected existing functionalities.

3.2.4 Edge Case Testing

Examine the model's behavior under extreme or boundary conditions to identify any unexpected results.

3.2.5 Code Review

Conduct a thorough review of your code to identify syntax errors, logical flaws, or inefficiencies.

3.2.6 Version Control Best Practices

Utilize version control systems (e.g., Git) to manage changes, track revisions, and collaborate effectively.

3.3 Guidance on Verification

3.3.1 Common Verification Pitfalls to Avoid

- Overlooking error messages or warnings.
- Insufficient test coverage.
- Not documenting failed tests.
- Over-reliance on a single verification method.

3.3.2 Documenting Testing Results

- Record inputs, outputs, and resolutions.
- Note any issues found and how they were resolved.
- Use test case templates for clarity.

3.3.3 Using Debugging Tools Effectively

- Utilize built-in debugging features of your programming environment.
- Employ logging to trace the execution flow and identify issues.
- Break down complex functions into smaller, testable units. clarity.

3.3.4 Example Code Snippet for Unit Testing

3.3.5 Version Control Best Practices

- Commit changes frequently with meaningful messages.
- Use branches to develop new features or test changes.
- Regularly merge and test branches to prevent conflicts.
- Keep a backup of your repository in a remote location.

4 Validation and Verification Checklist

Use the following checklist to ensure thorough validation and verification:

4.1 Validation Checks

- Have you compared simulation outputs with real-world data?
- Are all model parameters within reasonable and justified ranges
- Does the model's behavior align with expert expectations?
- Have you documented all validation results using appropriate visuals?
- Have you performed historical data validation?
- Did you conduct cross-model validation where applicable?

4.2 Verification Checks

- Have you tested the model with boundary and edge conditions?
- Did you verify that all input parameters are properly handled?
- Have you checked for mathematical and logical consistency in your code?
- Is the model's behavior reasonable across different time scales or scenarios?
- Have you systematically documented all testing results?
- Did you perform unit, integration, and regression testing?
- Are you following version control best practices?

4.3 Documentation Checks

- Is all documentation up to date and reflective of the current model
- Have you documented all assumptions and limitations?
- Are test cases and results properly recorded?

4.4 Version Control Checks

- Are all changes committed with descriptive messages?
- Have you tagged versions appropriately?
- Is the repository synchronized with the remote server?

4.5 Peer Review Checks (if applicable)

- Has the code been reviewed by peers or mentors?
- Have you incorporated feedback from reviews?

4.6 Data Validation Completeness Checks

- Is the data used for validation complete and reliable?
- Have you checked for data inconsistencies or anomalies?

5 Common Pitfalls

5.1 Validation Mistakes

- Ignoring data discrepancies.
- Over-relying on a single validation method.

5.2 Verification Oversights

- Not documenting failed tests.
- Insufficient test coverage.

5.3 Avoiding Overfitting in Validation

5.4 Tips for Efficient Testing Strategies

6 Documentation Requirements

- Clearly state assumptions.

- Report validation and verification results.
- Document model limitations.

6.1 Assumptions Documentation

6.2 Reporting Validation Results

6.3 Guidelines for Error Documentation

6.4 Documenting Model Limitations

6.5 Format for Reporting Test Cases

7 Conclusion

Validation and verification are essential to ensure the accuracy and reliability of the simulation model. By validating the model against real-world data and verifying the code implementation, a strong foundation is established for the final deliverable.