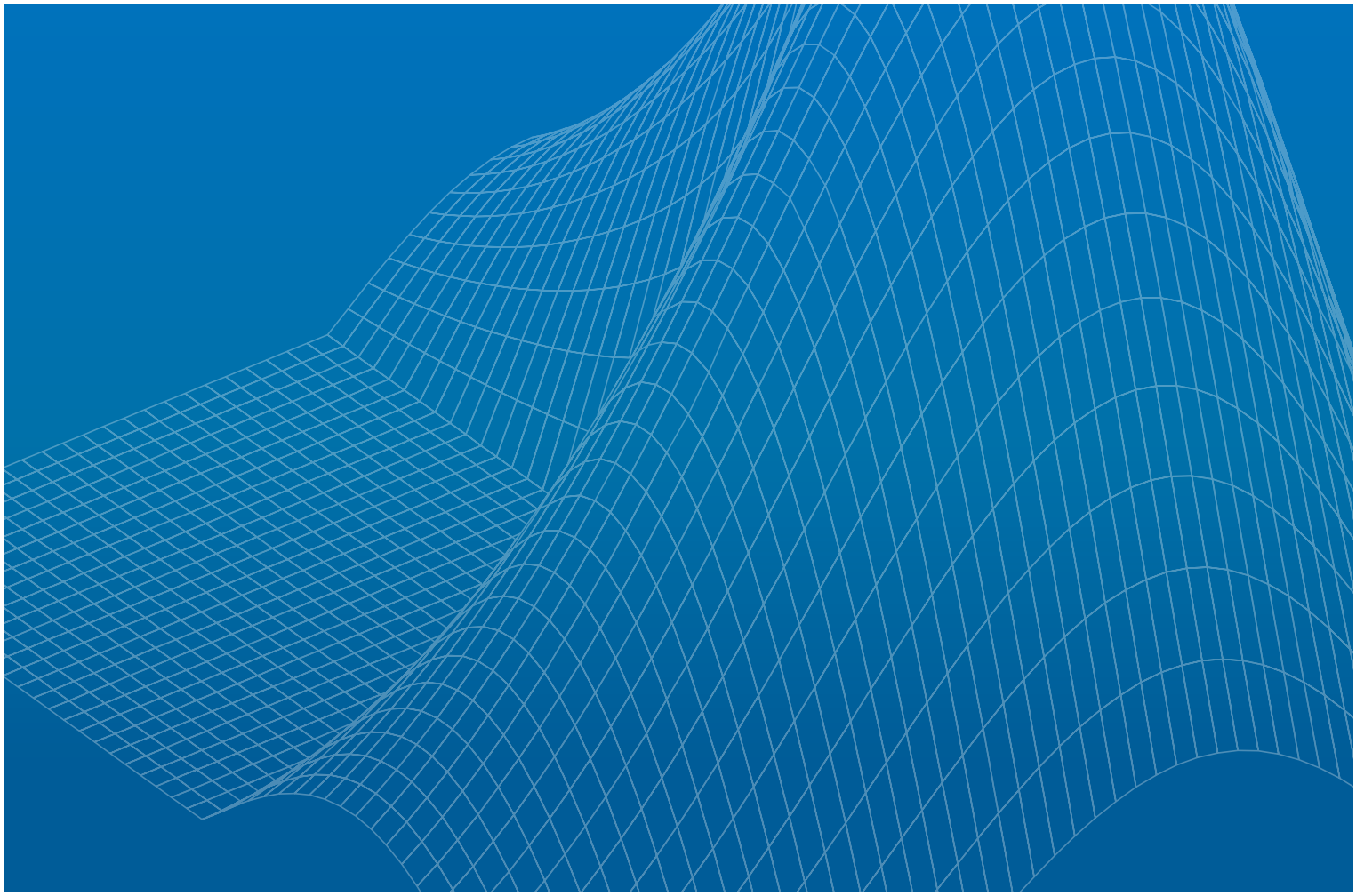

MATLAB and Simulink Version Upgrades for Large Organizations

By Judy Wohletz and Vinod Reddy

WHITE PAPER

MathWorks®

1 Introduction

Upgrading to a new version of MATLAB and Simulink will make the latest advances in Model-Based Design available to your organization, leading to improved productivity and better results. The upgrade's net benefit can be measured as a return on investment (ROI). The return calculation includes increases in productivity plus reductions in cost and risk. The investment is measured as the cost of evaluating and rolling out an upgrade, plus the cost of process and tooling changes required for the organization to make effective use of the new capabilities in the upgrade.

The success of any upgrade depends upon the strategy used to implement it. This paper presents MathWorks' recommended process for a systematic approach to enterprise-wide upgrades, designed to maximize access to benefits and minimize the costs, risks, and disruptions associated with achieving those benefits. These recommendations are based on practical experience in guiding enterprise customers in their upgrades to new versions of MATLAB and Simulink.

The upgrade process consists of six main phases:

1. Assess
2. Plan
3. Migrate
4. Test
5. Release
6. Support

Each phase of the process has triggers, inputs, activities, outputs, and exit criteria. The process itself is supported by a sustainment strategy.

2 Overview

To maximize ROI, it is imperative before pursuing an upgrade to understand the upgrade process, potential upgrade paths, and most importantly, potential benefits and related costs.

The tasks or activities required to upgrade to a new version can be grouped into logical phases: Assess, Plan, Migrate, Test, Release, and Support (Figure 1).

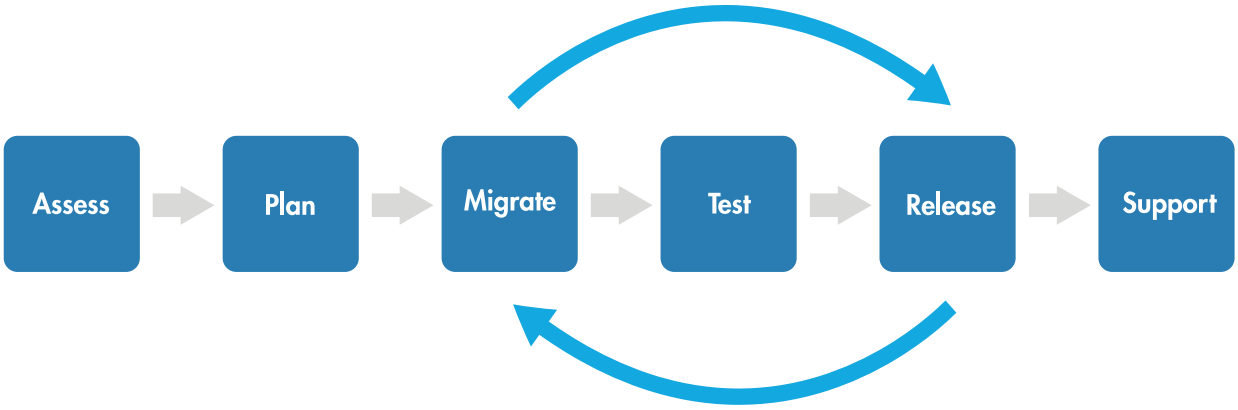

Figure 1. A workflow for phases in a typical upgrade project.

2.1 Assess

In the first phase of the upgrade process the goal is to understand the overall effect of the upgrade and to assess whether the benefits of the upgrade outweigh the cost, risk, and effort. Sufficient ROI is needed before proceeding to the next phase of the upgrade project.

- Triggers
  - Operating system obsolescence
  - New MATLAB or Simulink features
  - New hardware development platform
  - OEM/vendor/partner collaboration
  - New project
- Inputs
  - Current MathWorks software, third-party software and hardware, development or application software and hardware, development platform such as computers and operating systems
  - Timeline for upgrade
- Activities
  - Select initial target version

    - Compile a list of current issues and a list of requirements that you expect the new version to address
    - Assess whether the team has the capability to successfully execute the upgrade project or if additional technical training is required.
    - Assess whether the team has sufficient resources to handle the upgrade project.
    - Migrate a small, representative sample of models, code, and scripts to gain a better understanding of the effects of the version migration
    - Prepare an Upgrade Evaluation Report that includes upgrade procedures, results (including benefits as well as metrics such as simulation speed, code size, and so on), challenges, and error messages. The Upgrade Evaluation Report should also include an ROI estimate based on the current understanding of the potential benefits, estimated cost, and risks.
- Output(s)
    - Go/no-go decision
    - Initial target version
    - Upgrade Evaluation Report
- Exit Criteria
    - Good understanding of potential benefits, cost, and effort needed to execute the upgrade, documented in an Upgrade Evaluation Report, which is used in making the go/no-go decision

2.2 Plan

The purpose of the Plan phase is to define the overall scope and plan for the entire upgrade project.

- Trigger
    - Go decision
- Inputs
    - Upgrade Evaluation Report
- Activities
    - Create a business case
    - Establish the scope by identifying affected models, projects, and organizations, as well as key stakeholders
- Outputs
    - Business case for upgrade

    - An Upgrade Plan containing an upgrade timeline; list of affected projects, models, organizations, and stakeholders; estimates for required resources, training, and effort as well as cost and risks; dependencies; and target versions for hardware and software
- Exit Criteria
    - Approved business case

2.3 Migrate, Test, and Release

The Migrate, Test, and Release phases are closely related to one another and performed in an iterative manner to upgrade models to the new version.

2.3.1 Migrate

In the Migrate phase, you upgrade models and custom tools in an iterative manner to work with the new version. In this context, a custom tool is defined as any application or setting that has been customized in the MATLAB and Simulink environment. Examples of custom tools include:

- Custom Simulink block libraries
- MATLAB startup scripts
- Custom storage classes
- Custom system target files
- MATLAB Report Generator or Simulink Report Generator setup files
- Custom Model Advisor checks
- Modeling style guidelines
- Default configuration parameter settings
- MATLAB scripts used to customize your MATLAB and Simulink environment

The Migrate phase is executed in several iterations of three sub-phases: Initial Migration, Custom Tool Migration, and Automated Migration. In these iterations, a combination of automatic and manual approaches are typically required to upgrade models, scripts, user interfaces, templates, interfaces to third-party tools, and other applications.

The activities in this phase only result in migration of the models to the new version. The models will still need to be tested during the Test phase of the process, which is described in a later section.

2.3.1.1 Initial Migration

In the Initial Migration sub-phase, you migrate a small set of models that represent a cross section of the models developed within the organization with the goal of developing procedures, tools, and a deeper understanding of the new features. During this sub-phase, you use ***Upgrade Advisor***, a tool designed to help upgrade and improve models with the current release. This sub-phase supports the next sub-phase and helps reduce risk and effort.

- Trigger
  - Approved business case
- Input
  - Upgrade Evaluation Report
  - Upgrade Plan
  - Models and other related artifacts
- Activities
  - Run Upgrade Advisor on models
  - Successfully update the diagram, simulate and generate code for the models
  - Document any error or warning messages using the Diagnostic Viewer in Simulink
  - Resolve errors, if any
  - Compare the warning messages to determine if they are unique to the new version or they also exist in the current version
  - Evaluate the severity of the warning messages
  - Decide if you want to resolve the warning messages
  - Document all other issues in an Upgrade Issues List
- Output
  - Updated Upgrade Evaluation Report
  - New Upgrade Issues List
  - Updated Upgrade Plan
  - Subset of models migrated to the new version
- Exit Criteria
  - Subset of models are migrated

2.3.1.2 Custom Tool Migration

In addition to migrating your custom tools, it is also important to retire custom tool features by adopting built-in features during the Custom Tool Migration sub-phase. By reducing the customizations your organization uses, you will reduce the long-term costs of maintaining those tools.

- Trigger
  - Approved Upgrade plan

    - Subset of migrated models
- Input
    - Upgrade Evaluation Report
    - Upgrade Issues List
    - Models and other related artifacts
    - Custom tools
- Activities
    - Upgrade custom tools to the new version
    - Replace custom tools with built-in MATLAB and Simulink functionality when possible
    - Validate the custom tools in the new version
    - Develop automation tools that reduce effort and ensure consistency by automating the manual steps in the upgrade process
- Output
    - Updated Upgrade Evaluation Report
    - Updated Upgrade Issues List
    - Updated Upgrade Plan
    - Upgraded custom tools
    - Automation tools
- Exit Criteria
    - Models and custom tools are upgraded
    - Automation tools are developed

2.3.1.3 Automated Migration

In the Automated Migration sub-phase, you upgrade the remaining models using the procedure and automation tools developed in the previous sub-phase. Two principal goals of this phase are to resolve migration issues and improve the automation tools so that they are available for use by other teams.

- Trigger
    - Models and custom tools are upgraded
    - Automation tools are developed
- Input
    - Automation tools
    - Upgraded custom tools
    - Remaining models and other related artifacts
    - Upgrade Issues List
- Activities

    - Run automation tools in conjunction with the Upgrade Advisor (automation tools invoke the Upgrade Advisor) on all remaining models
    - Successfully update the diagram, simulate and generate code for the models
    - Document any error or warning messages using the Diagnostic Viewer in Simulink
    - Resolve errors, if any
    - Resolve warnings that hinder migration
    - Document all issues in the Upgrade Issues List
    - Update help documentation for custom tools and automation tools
    - Create release notes for custom tools and automation tools
    - Resolve issues by improving custom tools and automation tools
    - Create or update training materials, including relevant differences between old and new versions, custom tools, and automation tools
- Output
    - Improved automation tools
    - Updated Upgrade Evaluation Report
    - Updated Upgrade Issues list
    - Updated Upgrade Plan
    - Models upgraded to the new version
    - Release notes for custom tools and automation tools
    - Updated help documentation for custom tools and automation tools
    - New or updated training materials
- Exit Criteria
    - Known issues related to models, custom tools and automation tools are resolved
    - Models, custom tools and automation tools are upgraded

2.3.2 Test

The goal of the Test phase is to ensure that results produced by the model and code using the new version are functionally and numerically equivalent within acceptable limits to those produced using the previous version.

- Trigger
    - Models are upgraded
- Input
    - Upgraded custom tools

    - Models and other related artifacts
    - Input test data and expected outputs
- Activities
    - Conduct open-loop, model-in-the-loop (MIL), software-in-the-loop (SIL), and processor-in-the-loop (PIL) simulation on the desktop
    - Conduct hardware-in-the-loop (HIL) simulation, rapid prototyping, and on-target rapid prototyping in the lab
    - Perform any system testing or hardware testing that is normally performed
    - Review results and accept the validated model
    - Define additional test data and expected outputs as needed
- Output
    - Validated models in new version
    - Additional test data and expected outputs if needed
    - Test reports and related artifacts
- Exit Criteria
    - Validation of upgraded models is complete

2.3.3 Release

In the Release phase, you release the upgraded models, custom tools, and new versions of MATLAB and Simulink.

- Trigger
    - Test phase complete
- Input
    - Validated models
    - Input test data and expected outputs
    - Help documentation for custom tools
    - Release notes for custom tools
    - Upgrade Plan
- Activities
    - Update Upgrade Plan
    - Schedule and conduct training classes
    - Create a repository for all materials
    - Notify users
- Output
    - Updated Upgrade Plan

    - Released version of documents, tools, and training
- Exit Criteria
    - All models planned for update are upgraded to new version

2.4 Support

In the Support phase, the organization provides ongoing support for the users of the models, custom tools, and MathWorks software as issues arise.

- Trigger
    - User identifies an issue
- Input
    - Custom tools
    - Model with issues
    - Help documentation
    - Training materials
- Activities
    - Reproduce the issue
    - Identify the root cause
    - Fix the issue (in the model, custom tools, or elsewhere)
    - Notify users of the new versions
    - Update the repository
    - Update issue tracking list
    - Update release notes for custom tools if necessary
    - Release updated version with the fix if necessary
- Output
    - Updated release notes
    - Fixed version of the model or custom tools
    - Updated issue tracking list
- Exit Criteria
    - Issue is resolved

3 Detailed Upgrade Workflow

This chapter reviews the six process phases defined in Chapter 2 but includes the details for each step.

3.1 Assess

In the Assess phase the goal is to understand the overall effect of the upgrade and to determine if the benefits of the upgrade outweigh the cost, risk, and effort. If the ROI for the upgrade is insufficient, the organization may simply decide not to proceed with the upgrade. Upgrade processes can be triggered by a number of events, including a new project, collaboration with a new company, new hardware, or a new operating system.

3.1.1 Project Initiation Considerations

The first step of the upgrade process is to evaluate the current situation. This evaluation typically includes most or all of the following steps:

- Create an inventory of the software and hardware currently in use at your company.
- Document the current versions of MathWorks software in use at your company as well as at the companies that you are collaborating with.
- Document the third-party software and hardware tools that are being used with MathWorks products and identify which versions of MathWorks software the third-party vendors plan to support.
- Identify which computers and operating systems your company supports and plans to support in the future.
- Collaborate with the key stakeholders at your company who will be affected by this upgrade and agree on a rough estimate of a timeline for the upgrade.

3.1.2 Choose a Target MATLAB and Simulink Version

Prior to deciding which MATLAB and Simulink version that your organization should upgrade to, review the ***Release Notes*** for new features, ***Bug Reports*** for known issues, ***System Requirements***, and ***Supported Compilers***. Also review ***Platform Availability by Products***, ***Choosing a Computer to Run MATLAB and Simulink Products***, and ***Adopting 64-Bit Windows*** if applicable to your organization's situation. If you use third-party software, contact the vendors to verify which versions of MATLAB and Simulink they currently support and plan to support. It is also important to understand their timing for support of a new MATLAB and Simulink version after it is released, especially if you are targeting a MATLAB and Simulink version that hasn't been released yet.

Keep in mind that you may want to change the target MATLAB and Simulink version at some point in the future during the upgrade process. You may encounter unexpected issues going forward, or MathWorks may release new features that would benefit your organization. Instead of locking into

one specific version this early in the process, maintain the flexibility to select the version that will provide the highest ROI for your specific circumstances. If the frequency at which you upgrade is more than a year, you should be prepared to take advantage of these benefits by changing the target version instead of waiting years to use the new features.

3.1.3 Prepare Your Technical Environment

The next step is to make sure that you have access to the computers, operating systems, MathWorks software, development software and hardware, and third-party software and hardware that your company is using for the projects that will be affected by the upgrade.

3.1.4 Initial Testing

Once the technical environment is set up and the upgrade criteria are established, select one model to upgrade from the previous version to the target version. It is a good idea to choose a large model that is representative of a majority of the models in use at your company, contains common modeling patterns, and complies with your modeling style.

Once you select a model, perform the steps below to complete an initial test. While this process typically requires several manual steps, it is essential for identifying issues with upgrading to the target MATLAB and Simulink version. The models, custom libraries, and custom tools should be tested "as-is" with only the minimum modifications required to eliminate error messages that would prevent further testing.

Recommended activities for the assessment phase:

1. Test Current Version
    a. Open the current MATLAB and Simulink version
    b. Add custom libraries or custom tools to the MATLAB path
    c. Open the model in the current MATLAB and Simulink version
    d. Update the diagram
    e. Simulate the model
    f. Resolve error messages, if any
2. Test Target Version
    a. Open the target MATLAB and Simulink version
    b. Add custom libraries or custom tools to the MATLAB path
    c. Open the model in the target MATLAB and Simulink version
    d. Run Upgrade Advisor on the model
    e. Record warning messages and error messages

i. Correct error messages

ii. Correct warning messages only if they hinder further testing

iii. Document and track all issues that are encountered (you will likely encounter them with other models as well)

f. Resolve the issues prior to releasing the target MATLAB and Simulink version

If you encounter an issue that you can't resolve while upgrading the selected model to the target MATLAB and Simulink version, you may want to test the model in an intermediate MATLAB and Simulink version. If you are upgrading from a particularly old version, you may need to upgrade to an intermediate MATLAB and Simulink version. The reason that this extra step may be required is warnings and error messages about an incompatibility may be phased out after several releases. If the time between your upgrades is longer than the timeframe for the warning and error message phase-out, you can miss important messages that will help you debug the root cause of an issue. This approach can be helpful for debugging or trying to isolate an issue, but the intermediate version is typically only a temporary stop on the way to the new target version. A recommended practice is to select a MATLAB and Simulink version halfway between the current and target MATLAB and Simulink versions. Refer to *Initial Migration – Advanced Case* in this document for more information on debugging issues in an intermediate version.

3.1.5 Regression Testing

Run regression tests on tasks that users commonly perform, so that you have a rough idea how long these tasks will take in the new MATLAB and Simulink version. Whenever possible, run the same tasks in the previous version and compare the test results. It is recommended that you run these tests on some of the larger models that are currently in use.

This activity will help you discover potential issues before releasing the new version. It is much easier to delay upgrading to a new MATLAB and Simulink version than to retract the release of a new version or custom tools. Testing is essential for minimizing the upgrade's effect on production programs, because it enables you to identify issues and then find solutions or develop workarounds as needed before implementing the upgrade.

3.1.6 Decision to Upgrade

After completing the regression testing and evaluating the test results with the upgrade criteria, you can make a decision on whether to proceed to the next phase of the upgrade process.

3.2 Plan

The purpose of the Plan phase is to define the overall scope and plan for the upgrade project. This should encompass the remaining stages of the process, including Migrate, Test, Release, and Support phases.

3.2.1 Upgrade Project Management

Develop an overall plan for the upgrade that includes the target date for the upgrade and the target MATLAB and Simulink version. In the plan, clearly define phases and the activities required to meet the exit criteria for each phase. Also define the roles and expectations for each team member as well as the various teams involved in the upgrade process. Typical roles include program manager, manager, tools team, engineering teams, information technology (IT) team, third-party vendors, and MathWorks consultants.

Schedule milestone and status meetings to provide key stakeholders with regular updates and status reports on the upgrade. In these meetings, review the status for each team member. Any issues encountered during the upgrade process should be shared with the team in these meetings. Conduct technical reviews of the completed activities for each phase on a regular basis.

3.2.2 Create a Business Case

Creating a business case prior to upgrading to a new version will clarify the benefits of the upgrade and the associated costs. Benefits may include improved workflows made possible by new features in MathWorks products. The business case for an upgrade may also be driven by operating system or third-party software upgrades, if the current version of MATLAB and Simulink does not support the new operating system or third-party software.

3.2.3 Define Goals Upfront

It is important to limit the scope of the upgrade process. Trying to do too much at one time increases the risk of delays or even failure for the upgrade project.

Whenever possible, upgrade your models "as-is" without introducing new features. Introducing new features as you are introducing new custom tools and a new MATLAB and Simulink version complicates the upgrade process and makes it more difficult to validate the models. Wait until after the model has been completely upgraded and validated in the new MATLAB and Simulink version before introducing new features.

During the upgrade process, focus your upgrade testing on your organization's typical workflows; for example, updating, simulating, and generating code from models. After the models and custom tools are upgraded, the models will need to be validated in the new version by the engineers responsible for their development.

It is a good idea to include among your goals the replacement of custom tool features with built-in Simulink functionality when possible. For example, you may plan to replace custom library blocks

with new Simulink blocks that provide the same functionality. Set a goal to remove modeling style guidelines that no longer apply in the new version and add new guidelines for new Simulink features that you plan to use. You should consider adding the evaluation of custom tool documentation and training material as a goal.

3.3 Migrate

In the Migrate phase, you upgrade models and custom tools in an iterative manner to the new version. In each iteration, a combination of automatic and manual steps are typically required to upgrade models, scripts, user interfaces, templates, interfaces to third-party tools, and other applications. The iterative approach provides a solid foundation at each phase of the upgrade process to gain confidence in the new version with while reducing risk and effort.

In the Migrate phase, models are only upgraded to the new version, not yet tested. The models are tested during the Test phase.

3.3.1 Set Up the Environment

You will need to set up modeling environment before migrating models to the new version. Depending on your environment, this step may include accessing custom tools, setting the MATLAB path, setting up the compiler, and launching a Simulink Project. It is important to make as few changes as possible to set up the environment in the new MATLAB and Simulink version. Postpone changes to custom tools and MATLAB scripts until the Custom Tool Migration phase of the upgrade process.

3.3.2 Identify Models to Migrate

In the Assess phase of the process, you selected, upgraded, and tested one model in the new version to assess the process and identify potential issues. In the Migrate phase, you expand your testing to more models. If you can't test one version of all available models, then it is recommended that you select models with varying sizes and different modeling styles as well as models from different teams. For this phase of the process, you want to select the edge cases and uncommon modeling patterns. If you know of specific groups or individuals that are creating models outside the norm, include their models for this phase of the testing. Testing edge cases will help identify unexpected issues and resolve them when you are in the process of upgrading rather than after you release the new MATLAB and Simulink version, giving you more time to resolve any issues without affecting a production deadline. The following are typical complex cases:

- Models with nested libraries
- Models with Model Reference blocks
- Models with configurable subsystems or variants
- Models with both Model Reference blocks and nested libraries

3.3.3 Initial Migration – Simple Case

The purpose of the Initial Migration phase is to test your typical workflow; for example, by updating a diagram, running a simulation, and generating code. Use the following list of activities as a guide for completing this phase:

- In the old version,
  - Run the Simulink Manifest Tools to identify all files associated with your current project
  - Successfully update the diagram, simulate and generate code for the models
  - Document all the warning messages using the Diagnostic Viewer in Simulink
- In the new version,
  - Set up the MATLAB path and environment
  - Run the Simulink Manifest Tools to verify all required files are present
  - Run Upgrade Advisor on the selected models
    - Review the report
    - Apply the minimum required recommended fixes (any fixes that do not hinder the migration of selected models can be applied later in the process)
  - Successfully update the diagram, simulate and generate code for the models
    - Document error and warning messages using the Diagnostic Viewer
    - Resolve errors
    - Compare the warning messages to determine if they are unique to the new version or they existed in the old version
    - Evaluate the severity of the warning messages
    - Evaluate whether to resolve the warning messages by adopting new features
    - Document all other issues in an Upgrade Issues List (for example, document any custom tools that had issues undetected by Upgrade Advisor)

Review the following sections for advice on the minimum changes required for MATLAB files, data dictionaries, custom block libraries, and custom tools during the Initial Migration phase.

MATLAB Script and Files

This section covers dependent files such as MATLAB scripts and files that are required to update the model successfully. Some examples include startup MATLAB scripts, set up files, custom files used for automation, and data files in MAT format. Any files that the model is dependent on should be added to the MATLAB path.

Data Dictionary

For this phase, load the data associated with the models or data dictionary and resolve any issues with either loading the data or successfully updating the model after the data is loaded.

Custom Block Libraries

Upgrade Advisor automatically updates Simulink custom library blocks present in the model. Upgrade Advisor will:

- Update custom blocks created from built-in Simulink blocks if they are present in a custom block library and the library links are active
- Update custom S-Function blocks present in the model or a custom library
- Generate results for each Upgrade Advisor check

After you review the results, you can have Upgrade Advisor make the changes to the blocks automatically. Upgrade Advisor will not update masks; you will have to update them manually.

Add custom block libraries to the MATLAB path to resolve broken links in your model. Only make the minimum changes necessary to ensure that the model can be updated. Other changes (optimizations, new features, or S-Function API changes) should be made later.

Custom Tools

As with custom block libraries, make only the minimum changes to custom tools necessary to ensure that the model can be updated. Other changes and optimizations can be completed in the Custom Tool Migration sub-phase.

3.3.4 Initial Migration – Advanced Case

In rare cases, when upgrading from a version that is much older than the target version, Upgrade Advisor may fail to update your models and be unable to identify the root cause of the issue. Most of these cases are caused by customizations to MATLAB and Simulink or an edge case modeling pattern that wasn't intended or expected for Simulink. In these instances, you may have to upgrade to an intermediate MATLAB and Simulink version first to isolate and debug issues and later upgrade to the target version once these issues are resolved.

3.3.5 Custom Tool Migration

In addition to migrating custom tools, it is also important to look for opportunities to retire custom tool features by adopting built-in features.

3.3.5.1 Upgrade Custom tools

You may need to update custom storage classes, custom system target files, MATLAB Report and Simulink Report Generator setup files, custom Model Advisor checks, modeling style guidelines, and any custom MATLAB scripts when upgrading to a new MATLAB and Simulink version. If you have

template or demo models, consider modifying them by adopting new Simulink features that are available that you would benefit from using. This is also a good time to migrate your Configuration Parameter settings to MathWorks recommended settings (for example: ***Simplified Initialization Mode***). This may also be an opportunity to agree on common Configuration Parameter settings across the organization.

Modeling style guidelines will need to be updated for the new version. You should consider removing modeling style guidelines that no longer apply in the new version and add new guidelines for new Simulink features that you plan to use.

While developing and migrating your custom tools, it is recommended that you create a design document that documents the requirements and intended functionality for your custom tools. For your custom tools, follow a development process similar to your production software development process. Develop test plans and test cases, and use them to test your custom tools when you upgrade to a new MATLAB and Simulink version. When possible, automate the test plans and run automated tests on each prerelease as well as your target MATLAB and Simulink version. Once you have upgraded to a new MATLAB and Simulink version, update the design documents, test plans, and test cases as necessary.

Review the following sections and identify any files that need to be upgraded. Then, review new Simulink features, optimizations, and documented APIs to look for opportunities to retire custom tool features with built-in Simulink functionality.

MATLAB Script and Files

Upgrade MATLAB scripts and files that are required to successfully update, simulate, and generate code from the model without errors. Examples include:

MATLAB scripts that interact with MATLAB

- Project or MATLAB environment setup scripts
- Compiler setup scripts
- Data analysis scripts

MATLAB scripts that interact with Simulink

- Preload and postload scripts within model callbacks
- Scripts that load configuration sets
- Scripts that set configuration options
- Scripts for custom menus
- User interface scripts
- Scripts in block callbacks
- Scripts within masks

Simulink Customization and Configuration

Upgrade any Simulink files that may be required to successfully update, simulate, and generate code from the model without errors. Examples include:

Simulation

- Simulation data
- Logging customizations
- Visualization customizations
- Report generation setup files
- Data comparison tools

Code Generation

- System Target Files (STF)
- Template Make Files (TMF)
- Target Language Compiler (TLC) files
- Code Generation Template (CGT) files
- Code Replacement Library (CRL) files

Verification and Validation

- Unit testing customizations
- Test data
- Reports
- Signal and test authoring tools
- Custom Model Advisor checks
- Simulink Design Verifier customizations
- Polyspace static analysis tools configuration

Data Dictionary

Load the data dictionary and resolve any issues with loading or successfully updating the model. Upgrade MATLAB scripts and ASCII-based data dictionaries to Simulink Data Dictionary. Refer to the ***Simulink Data Dictionary*** documentation for more information.

Custom Block Libraries

When you upgrade custom block libraries to the new version, use the opportunity to migrate custom blocks to built-in Simulink blocks or features. This is also an opportunity to improve and optimize the custom blocks if new or enhanced features are available.

The newly available built-in blocks with the same functionality may have different dialog parameters values than the current custom block. In such cases, you may copy the block dialog parameters from the old parameter field to the new parameter field. This can be done manually or automated via a MATLAB script, depending on the effort involved, which will depend on the number of blocks that have to be replaced. If you are phasing out a block that you don't want teams to use anymore, you will need to create a legacy library that is on the MATLAB path but is clearly marked so that users no longer access it. You can remove this library after you update your models in the new MATLAB and Simulink version and run upgrade MATLAB scripts. Run Upgrade Advisor on your libraries. It may be necessary to recompile your S-functions if any changes were made to them and if you plan to support more platforms.

3.3.5.2 Custom Tool Testing

Test your custom tools using their test plans and test cases to verify that the custom tools in the new MATLAB and Simulink version perform as intended. Next, start testing the upgrade process for your models. It is a good idea to test the custom tools and the upgrade process on multiple PCs with all operating systems your organization supports.

3.3.5.3 Third-Party Tool Testing

It is recommended that you test all of your third-party tools with the new MATLAB and Simulink release prior to releasing the new MATLAB and Simulink version. In some cases, third-parties may not immediately release a new version of their software that is compatible with a new version of MATLAB and Simulink. Verify the timing with your suppliers, and plan accordingly.

Third-party products to consider when upgrading:

- Requirements management tools (for example, IBM Rational DOORS)
- Configuration management tools
- Change management tools
- Co-simulation tools
- IDEs
- Compilers
- Hardware

3.3.6 Automated Migration

Once you have performed the initial migration and custom tool migration, the next step is to automate as much of the upgrade process as possible.

3.3.6.1 Upgrade Automation Tools

It is recommended that you automate the model and custom tools upgrade process. A convenient approach is to create a single master upgrade MATLAB script that calls all other upgrade scripts, so that you need to run only one MATLAB script to upgrade any model. This MATLAB script should have a call to Upgrade Advisor, a custom library upgrade script, a configuration parameter settings script, a Simulink data object script, and any additional scripts needed to upgrade the custom tools. The automation tools should generate a report that provides details about errors and warnings encountered for each phase.

3.3.6.2 Test Automation Tools

You should run the upgrade script with the upgraded custom tools on all prior to releasing the new MATLAB and Simulink version. If you are not able to upgrade all of your models, you should select models with different modeling styles, with different model sizes, and from different groups. Once you upgrade the models, you should update the diagrams, simulate them, and generate code. Review the warning messages and document any errors that you encounter.

It is a good idea to automate this test procedure, so you can run these tests in batch mode on a group of models and automatically document the issues. You may need to develop workarounds or MATLAB scripts to resolve some of these issues prior to releasing the new version of MATLAB and Simulink.

3.3.6.3 Migrate Remaining Models

At this point, you have upgraded custom tools and custom block libraries and developed upgrade automation tools. The next step is to upgrade your remaining models.

As models are upgraded, document issues that arise. The issues could be in models, automation tools, custom block libraries, or custom tools. Resolve the issues following the recommended steps in the Migrate phase and then run the automation tools again. Repeat these steps iteratively until all known issues are resolved.

3.4 Test

The goal of the Test phase is to ensure that the model and code in the new version of MATLAB and Simulink are functionally and numerically equivalent within acceptable limits to the model and code in the previous version.

In the final phase of testing, a small group of selected users beta test the custom tools and the model upgrade process. The beta testers should include experienced MATLAB and Simulink users with deep knowledge of the relevant Simulink models and the ability to provide feedback to improve the process and tools. The group should start to use the new MATLAB and Simulink version and custom tools for their everyday work with any required third-party software tools.

3.5 Release

The purpose of the Release phase is to release the new MATLAB and Simulink version, custom tools, automation tools and to provide training to the users.

3.5.1 Training

It is recommended that you offer two types of training classes for users during an upgrade. The first type is a seminar or workshop that summarizes some of the major new features now available with the upgraded MATLAB and Simulink versions and highlights the features and benefits most pertinent to your organization. The second type focuses on the upgrade process, recommended workarounds for specific issues, new style guidelines, and new features for your custom tools.

3.5.2 Releasing

Once you have completed the testing, you are ready to release the custom tools to your users. It is a good practice to include release notes in each version of your custom tool software. It is also a good practice to obfuscate the MATLAB code by ***P-Coding*** the files prior to release, even if the tools will only be used internally. This practice will discourage engineers from fixing issues themselves without informing others, which can lead to teams using different versions of the custom tools and possible upgrade issues in the future. Place the custom tools or installer in a location that everyone in the organization can access. Include installation and upgrade instructions when notifying the users that new versions of the custom tools and MATLAB and Simulink are available.

It is recommended that engineers upgrade their own Simulink models instead of having a separate group perform the migration. The engineers who developed the models have the expertise needed to perform validation. They also are aware of the production deadlines that they are facing and what portions of the model will need to be modified for future versions of the production software. If they decide that they want to upgrade their model to the new release, they will need to reach an agreement to use the validated model from other key stakeholders, which may include other engineering groups downstream in the process that will be affected. The goal is to ensure that migrating to a new MATLAB and Simulink version doesn't introduce delays in a production schedule. All new modeling projects should be required to use the new MATLAB and Simulink version. It's a good idea to set a deadline for everyone to complete the migration of their model to the new MATLAB and Simulink release. Until they do, you may need to support multiple MATLAB and Simulink versions.

The purpose of extensive testing and the creation of MATLAB scripts is to automate the process as much as possible. If for some reason, it is not possible for engineers to upgrade their own models, then you would need test cases that produce the desired level (100% is recommended) of model test coverage. Test the model and the generated code in a software-in-the-loop (SIL) environment and verify that the simulation and code generation outputs match the outputs from the previous MATLAB and Simulink version, via automation if possible.

3.6 Support

The purpose of the Support phase is to continuously perform post-upgrade activities to minimize the work for the next upgrade and to provide continuous support to users.

3.6.1 Post-Upgrade Activities

After you complete the upgrade, you can perform a number of post-upgrade activities to make the next upgrade easier. If you didn't create design documents detailing the requirements and functionality of your custom tools during the Custom Tool Migration phase, you should do so in the Support phase.

Use this time to create test plans and test cases for the custom tools and typical workflows. You can then use the test plans and test cases to test your custom tools when you upgrade to a new MATLAB and Simulink version. The test cases should include the expected outputs for each test. When test plans are documented and test cases exist, it is easier to automate your tests the next time that you upgrade. You can run automated tests on each prerelease even if you don't intend to upgrade to that release and provide feedback to MathWorks. This feedback will enable MathWorks to improve the product or fix any issues that you encounter prior to you migrating to the next MATLAB and Simulink version. Of course, you can also run automated tests on the new MATLAB and Simulink version that you intend to target.

After the custom tool tests are automated, the next step is to automate the model migration and model validation processes so that you can run tests on your models in batch mode and automatically generate reports on the issues. The model and the generated code need to be tested in an SIL environment with the desired level of test coverage to verify that the simulation and generated code outputs match the outputs from the previous MATLAB and Simulink version. This type of SIL testing can also be automated to reduce the workload for the next upgrade. Some examples of tests that can be run in a batch mode with a reporting mechanism include:

- Validating that the custom tool outputs match between MATLAB and Simulink versions
- Migrating a group of models to the new MATLAB and Simulink version
- Running model updates, running simulations, and generating code on a group of models in the new MATLAB and Simulink version
- Differencing the generated code between MATLAB and Simulink versions
- Comparing the simulation results between MATLAB and Simulink versions
- Comparing the simulation results to the generated code results in the same MATLAB and Simulink version
- Comparing the generated code results between MATLAB and Simulink versions

4 Sustainment: A Continuous Upgrade Philosophy

MathWorks recommends the adoption of a continuous upgrade philosophy. Continuously performing upgrade activities ensures that the next upgrade is easier than the last upgrade. To assist with the adoption of this philosophy, consider taking advantage of prerelease testing and Industry Model Testing, as well as MathWorks seminars, webinars, and Conferences.

4.1 Prerelease Testing

When possible, test your models and custom tools for each prerelease and send feedback to MathWorks. It is best not to wait to test your models on only the release that you plan to upgrade to, because many new features are introduced between your current version and your future version. If you test the prereleases every six months, the early testing feedback you provide enables MathWorks to fix issues that you encounter by the time of your next upgrade. Testing each prerelease not only enables you to assess the release more thoroughly than reading Release Notes but it also minimizes the effort needed to upgrade to a new MATLAB and Simulink version. Furthermore, it facilitates learning of new features and provides guidance to enterprise decision-makers on the appropriate time to upgrade.

4.2 Industry Model Testing

Consider submitting your models to Industry Model Testing. The following is an excerpt from the Industry Model Testing information sheet:

Industry Model Testing (IMT) is part of a strategic quality initiative at The MathWorks. The IMT system is designed to identify, isolate, and prevent customer-facing regressions in MathWorks software. As part of this system, The MathWorks tests real customer models in a secure environment. This environment is integrated with the MathWorks software build, test, and release processes. Including your model in the IMT process dramatically reduces the chance of release incompatibilities.

4.3 Seminars, Webinars, and Conferences

Attending MathWorks seminars, webinars, and conferences will help you keep up-to-date with new features and enable you to make informed decisions on which MATLAB and Simulink version that your organization should target.

5 MathWorks Support

MathWorks offers a wide variety of support options for the upgrade process including online documentation, an active user community, and consulting services.

6 Appendix

6.1 Upgrade Tools

Upgrade tools are available to ease the transition from the old MATLAB and Simulink version to the new MATLAB and Simulink version.

6.1.1 Upgrade Advisor

Depending on which version that you are upgrading from and upgrading to, there are different upgrade tools available to assist you with your upgrade.

If your target version is R2012b or a later version, use ***Upgrade Advisor*** (from the Model Editor, select Analysis > Model Advisor > Upgrade Advisor) to upgrade your models. Upgrade Advisor includes the older upgrade tools (if they are still relevant) and additional checks.

Between R2006a and R2012b, run Model Advisor upgrade checks on your models. The Upgrading to the Current Simulink Version checks can be found in Model Advisor under the By Task folder.

In versions prior to R2006a, use ***slupdate*** to upgrade a model to a new version. Likewise, slreplace_mux, which was introduced in R14, is a useful MATLAB script to run on older models that used Mux blocks to create buses instead of Bus Creator blocks.

Even though Upgrade Advisor is recommended with all MATLAB and Simulink versions starting in R2012b, the other tools cited above maybe useful in the upgrade process. For example, if you are upgrading an older model, you may have to run slreplace_mux or another upgrade tool on the model in your current version before upgrading that model to the target version.

6.1.2 Performance Advisor

Performance Advisor checks your model for conditions and settings that can slow simulations. It can recommend modeling optimizations, implement them automatically, and run simulations in accelerator mode for you. This feature was introduced in R2012b.

6.1.3 Visual Comparison Tools

Visual comparison tools should also be used while upgrading to a new version. You can use Simulink Model XML Comparison, a feature of Simulink Report Generator, to compare models between two versions. You can also use Simulation Data Inspector to compare the expected outputs between multiple simulations. During upgrades, you can use it to compare the results for the same model in two different MATLAB and Simulink versions or the results from a model and its generated code in the same MATLAB and Simulink version. Refer to ***Simulation and Code Comparison*** in the Simulink documentation for more information.

92973v00 02/16