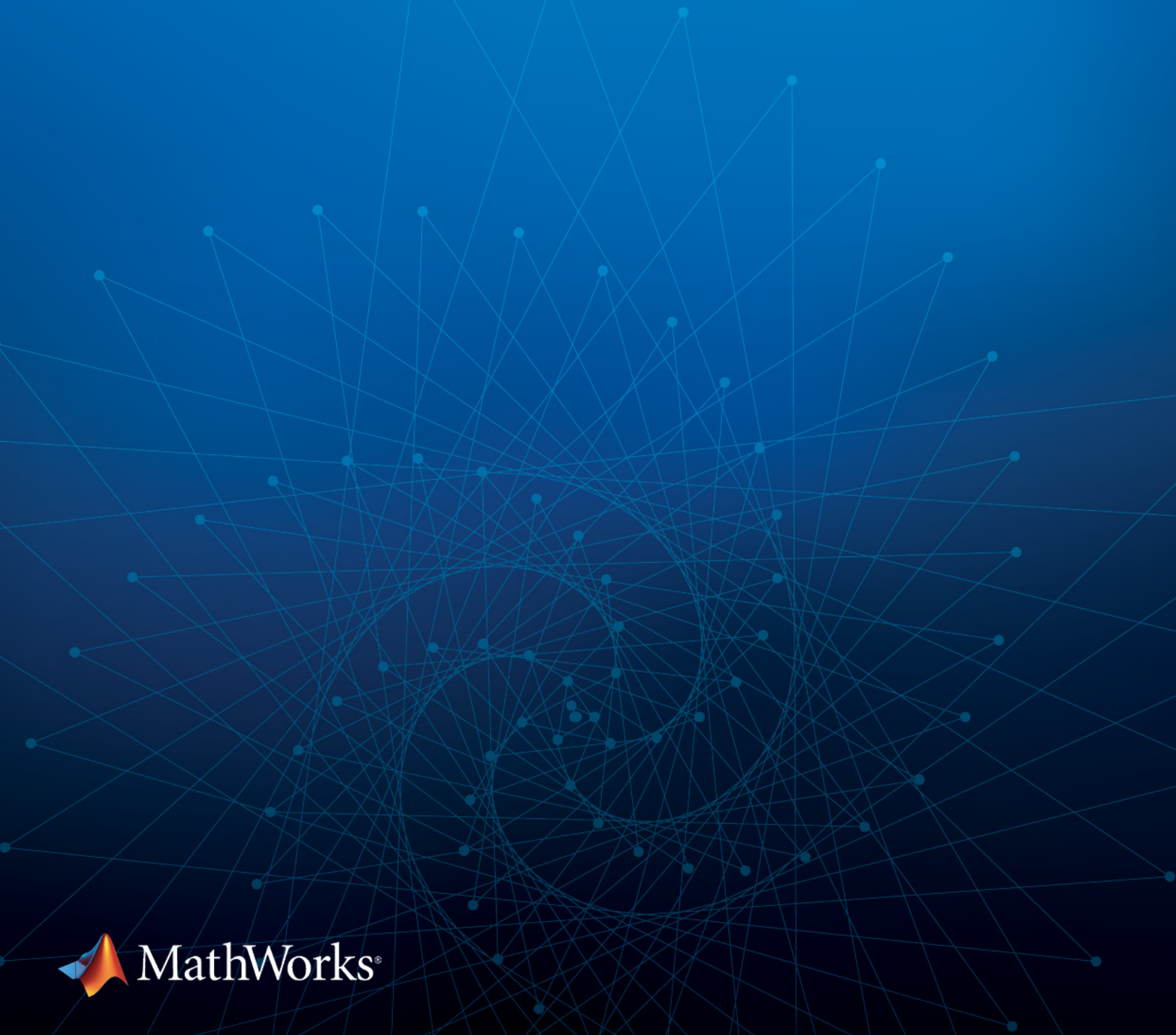
WHITE PAPER

MATLAB and Simulink Version Upgrades for Large Organizations

By Judy Wohletz, Vinod Reddy, and Jim Ross, MathWorks



MathWorks®

Table of Contents

1 Introduction

Historically, development teams have upgraded to new tool versions before beginning a new program and remained locked into those versions throughout the entire multi-year product development lifecycle. However, companies are introducing new technologies at a record pace, making it difficult to upgrade or even investigate new technology during active product development. A proactive upgrade strategy provides organizations with the opportunity to balance the risk of changing versions with the risk of being unable to incorporate new technology and methods.

Upgrading to a new release of MATLAB and Simulink will make the latest technology 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 and 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 recommendations from MathWorks for a systematic approach to enterprise-wide upgrades, designed to leverage advances in MATLAB and Simulink products to the maximum extent and minimize the costs, risks, and disruptions associated with achieving the 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

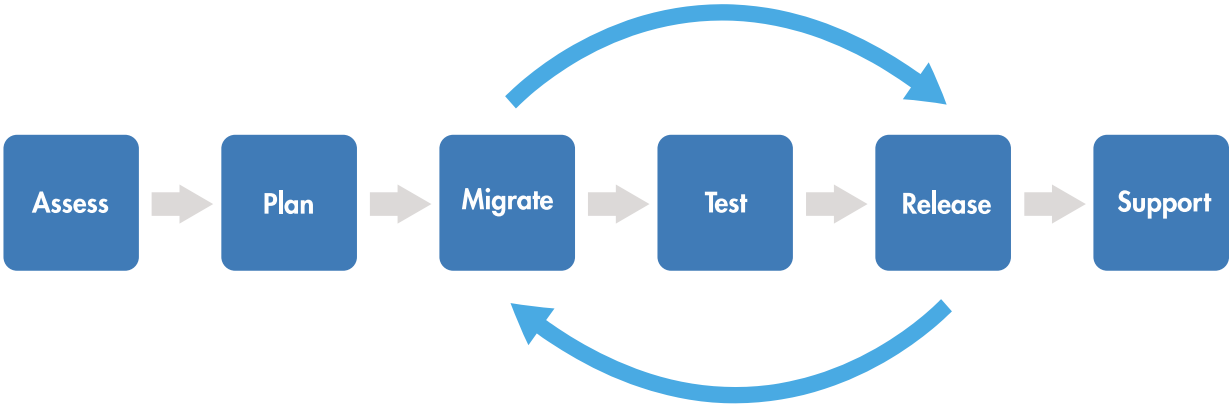
The most important step in every phase of the process is communication. It is critical for all stakeholders to be fully informed throughout a version upgrade. Early notification of the assessment, outcome, and key reasons behind the upgrade decision can help users prepare. Once a decision is made to upgrade, the organizational plan should be quickly communicated so local planning can occur. Communication of timelines, successes, and difficulties encountered along the way will help teams be successful when it is their turn to upgrade.

A final consideration for an efficient and robust upgrade process is the use of Continuous Integration (CI). Using a CI environment for development workflows improves repeatability and reduces the effort required to execute tests. Also, using a CI environment pays off during day-to-day work and dramatically reduces the effort required to evaluate upgrades. If CI is already part of your development workflows, only minor changes may be needed to provide flexibility to test against specific MATLAB versions. If CI is not a part of your current development workflows, this may be a good activity to pursue.

These activities combine to minimize risk and maximize the chance for a successful upgrade.

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.



A workflow for phases in a typical upgrade project.

Each phase of the process has triggers, inputs, activities, outputs, and exit criteria. The process itself is supported by a sustainment strategy. This workflow should be supported throughout by a targeted communication plan that keeps all stakeholders informed about intent, status, decisions, and timelines as well as significant issues identified during the upgrade.

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 outweigh the cost, risk, and effort. The intent is to determine early if a show-stopping problem exists and avoid investing additional effort in evaluating that specific version. Sufficient ROI is needed before proceeding to the next phase of the upgrade project.

Assess Objective: Understand the impact of an upgrade	
Triggers	1. Operating system obsolescence 2. New MATLAB and Simulink features, including entirely new products and capabilities 3. New hardware development platform 4. OEM/vendor/partner collaboration 5. New project/program
Inputs	1. Current state of the project to be upgraded 2. How many releases of MATLAB and Simulink are between your current version and the upgraded version? 3. Current version of MATLAB and Simulink, third-party software and hardware, development or application software and hardware, and development platform such as computers and operating systems 4. Timeline for upgrade
Activities	1. Select initial target version

	2. Compile a list of current issues and a list of requirements that you expect the new version to address 3. Identify a team to migrate a small, representative sample of models, code, and scripts using Upgrade Advisor 4. Migrate a small, representative sample of models, code, and scripts to gain a better understanding of the effects of the version migration 5. Run regression tests on tasks that users commonly perform to gather metrics on how long the tasks will take in the new version 6. Assess whether this team has the capability to successfully execute the upgrade project or if additional technical training is required 7. Assess whether the team has sufficient resources to handle the upgrade project 8. Prepare an Upgrade Evaluation Report that includes: • Upgrade procedures • Results that include benefits and metrics such as simulation speed, code size, and more • Challenges • Error messages • ROI estimate based on the current understanding of the potential benefits, estimated cost, and risks
Outputs	1. Go/no-go decision 2. Initial target version 3. Upgrade Evaluation Report

2.2 Plan

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

Plan Objective: Define the overall scope of the upgrade project	
Triggers	Go decision
Inputs	Upgrade Evaluation Report
Activities	1. Create a business case 2. Establish the scope by identifying affected models, projects, organizations, and key stakeholders
Outputs	1. Business case for upgrade 2. 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

2.3 Migrate, Test, and Release

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

2.3.1 Migrate

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

- Custom Simulink block libraries
- S-Function blocks
- MATLAB startup scripts
- Custom storage classes in Embedded Coder Dictionary
- 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 subphases: Initial Migration, Custom Tool Migration, and Automated Migration. These iterations typically require a combination of automatic and manual approaches 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. You will still need to test the models during the Test phase of the process, which is described in a later section.

2.3.1.1 Initial Migration

In the Initial Migration subphase, you migrate a small set of Simulink models and projects 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. This is done by using *Upgrade Advisor*, a tool designed to help upgrade and improve models with the current version. This subphase supports the next subphase and helps reduce risk and effort.

Initial Migration Objective: Migrate a subset of models	
Triggers	Approved business case
Inputs	1. Upgrade Evaluation Report 2. Upgrade Plan 3. Models and other related artifacts
Activities	1. Run Upgrade Advisor on models 2. Successfully update the diagram, simulate, and generate code for the models 3. Document any error or warning messages using the Diagnostic Viewer in Simulink 4. Resolve errors if any 5. Compare the warning messages to determine if they are unique to the new version or exist in the current version 6. Evaluate the severity of the warning messages 7. Decide if you want to resolve the warning messages 8. Document all other issues in an Upgrade Issues List

Outputs	1. Updated Upgrade Evaluation Report 2. New Upgrade Issues List 3. Updated Upgrade Plan 4. Subset of models migrated to the new version

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 subphase. The long-term goal is to eliminate the custom tools by adopting built-in features. This will make future upgrades easier and more automatic. It will also substantially reduce future upgrade effort and cost.

Custom Tool Migration Objective: Upgrade models and custom tools and develop automation tools	
Triggers	1. Approved Upgrade Plan 2. Subset of migrated models
Inputs	1. Upgrade Evaluation Report 2. Upgrade Issues List 3. Models and other related artifacts 4. Custom tools
Activities	1. Upgrade custom tools to the new version 2. Replace custom tools with built-in MATLAB and Simulink functionality when possible 3. Validate custom tools in the new version 4. Develop automation tools that reduce effort and ensure consistency by automating the manual steps in the upgrade process
Outputs	1. Updated Upgrade Evaluation Report 2. Updated Upgrade Issues List 3. Updated Upgrade Plan 4. Upgraded custom tools 5. Automation tools

2.3.1.3 Automated Migration

In the Automated Migration subphase, you upgrade the remaining models using the procedure and automation tools developed in the previous subphase. 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.

Automated Migration Objective: Resolve known issues related to models, custom tools, and automation tools; upgrade models, custom tools, and automation tools	
Triggers	1. Models and custom tools are upgraded 2. Automation tools are developed
Inputs	1. Automation tools 2. Upgraded custom tools

	3. Remaining models and other related artifacts 4. Upgrade Issues List
Activities	1. Run automation tools in conjunction with Upgrade Advisor (automation tools should invoke Upgrade Advisor) on all remaining models 2. Successfully update the diagram, simulate, and generate code for the models 3. Document any error or warning messages using the Diagnostic Viewer in Simulink 4. Resolve errors if any 5. Resolve warnings that hinder migration 6. Document all issues in the Upgrade Issues List 7. Update help documentation for custom tools and automation tools 8. Create release notes for custom tools and automation tools 9. Resolve issues by improving custom tools and automation tools 10. Create or update training materials, including relevant differences between old and new versions, custom tools, and automation tools
Outputs	1. Improved automation tools 2. Updated Upgrade Evaluation Report 3. Updated Upgrade Issues List 4. Updated Upgrade Plan 5. Models upgraded to the new version 6. Release notes for custom tools and automation tools 7. Updated help documentation for custom tools and automation tools 8. New or updated training materials

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.

Test Objective: Validate upgraded models	
Triggers	Models are upgraded
Inputs	1. Upgraded custom tools 2. Models and other related artifacts 3. Input test data and expected outputs
Activities	1. Run regression tests on tasks that users commonly perform and compare the results to the previous version 2. Conduct open-loop, model-in-the-loop (MIL), software-in-the-loop (SIL), and processor-in-the-loop (PIL) simulations on the desktop 3. Conduct hardware-in-the-loop (HIL) simulation, rapid prototyping, and on-target rapid prototyping in the lab 4. Perform any system testing or hardware testing that is normally performed 5. Review results and accept the validated model

	6. Define additional test data and expected outputs as needed
Outputs	1. Validated models in new version 2. Additional test data and expected outputs if needed 3. Test reports and related artifacts

2.3.3 Release

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

Release Objective: Release the new MATLAB version and custom tools	
Triggers	Test phase complete
Inputs	1. Validated models 2. Input test data and expected outputs 3. Help documentation for custom tools 4. Release notes for custom tools 5. Upgrade Plan
Activities	1. Update Upgrade Plan 2. Schedule and conduct training classes 3. Create a repository for all materials 4. Notify users
Outputs	1. Updated Upgrade Plan 2. Released version of documents, tools, and training

2.4 Support

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

Support Objective: Resolve reported issues	
Triggers	User identifies an issue
Inputs	1. Custom tools 2. Model with issues 3. Help documentation 4. Training materials
Activities	1. Reproduce the issue 2. Identify the root cause 3. Fix the issue (in the model, custom tools, or elsewhere) 4. Notify users of the new versions 5. Update the repository 6. Update issue tracking list

	7. Update release notes for custom tools if necessary 8. Release updated version with the fix if necessary
Outputs	1. Updated release notes 2. Fixed version of the model or custom tools 3. Updated issue tracking list

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. The intent is to determine early if a show-stopping problem exists and avoid investing additional effort in evaluating that specific version. If the ROI for the upgrade is insufficient, the organization may decide to delay the upgrade to the next version. Each time an upgrade is delayed creates an incremental barrier to upgrading to future versions. Upgrade processes can be triggered by numerous events, including new projects, collaboration with new companies, new hardware, or new operating systems.

3.1.1 Project Initiation Considerations

The first step of the upgrade process is to evaluate the current situation. This evaluation 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 MATLAB and Simulink in use at your company and companies that you are collaborating with
- Document the third-party software and hardware tools that you are using with MATLAB and Simulink products
- Identify which versions of MATLAB and Simulink products 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 an estimate of a timeline for the upgrade
- Review issues from previous upgrades

3.1.2 Choose a Target MATLAB and Simulink Version

Prior to deciding which MATLAB and Simulink version that your organization should upgrade to, filter and search the *Release Notes* for new features in specific versions, *Bug Reports* for known issues, *System Requirements*, and *Supported Compilers*. Also, review *Platform Availability by Products* and *Choosing a Computer to Run MATLAB and Simulink Products* if applicable to your organization's situation. Another thing to consider when selecting your target MATLAB and Simulink version is the Update. You can access Updates by clicking on the bell icon on the top right corner of the MATLAB window. MathWorks recommends that you use the latest Update whenever it is available and as soon as it is feasible for your organization or programs. If you use third-party software, contact the vendors to verify which versions and Updates 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 and Updates after it is released, especially if you are targeting a MATLAB and Simulink version that has not 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, 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, 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 ensure that you have access to computers, operating systems, MATLAB and Simulink, development software and hardware, and third-party software and hardware that your company is using for the projects affected by the upgrade. Set aside a dedicated sandbox for the evaluation and migration efforts because saving a model in the new version makes it difficult to open in the original version. Finally, new versions mean you need to consider additional settings to ensure proper results.

A sandbox means one or more new, dedicated branches in a repository to enable:

- Collaboration between multiple users or teams
- Sharing updates to tools or libraries
- Use existing CI workflows easily
- Simplified reuse of new or enhanced CI workflows developed as part of the upgrade

It is important to be able to scale the sandbox for use in the initial testing within the Assess phase with the ability to apply the same approach on a larger scale for the later Migrate, Test, and Release phases. Testing in the Assess phase is intended to be minimal and should emphasize checking for issues that cannot easily be resolved, understanding warnings, or identifying performance issues. More complete testing is recommended for later phases such as Migration and Testing phases.

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 represents 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. Test the models, custom libraries, and custom tools "as-is" with only the minimum modifications required to eliminate error messages that would prevent further testing. Keep in mind that the intent of this Assess phase is to rapidly gain insight into the feasibility of the upgrade while identifying issues that need to be resolved before a successful migration can take place on a larger scale.

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 Simulink model or projects in the current MATLAB and Simulink version. Accept any recommended Simulink model configuration settings.
    d. Update the diagram
    e. Simulate the model
    f. Resolve all error messages and determine appropriate action (if any) for warnings
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 Simulink model or project 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 cannot resolve while upgrading the selected model to the target MATLAB and Simulink version, test the model in an intermediate version. If you are upgrading from an old version, you may need to upgrade to an intermediate MATLAB and Simulink version. This extra step may be required to ensure warnings and error messages about incompatibilities are identified. 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 help debug or 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.

3.1.5 Regression Testing

Run regression tests on tasks that users commonly perform so that you can estimate 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.

In the Assess phase, regression testing is a quick test to see if the specific MATLAB and Simulink version is a viable candidate to upgrade to. Ideally, this regression testing is performed quickly and with every prerelease regardless of what target version you plan to upgrade to. The following table provides a key list of items that should be checked to evaluate a specific MATLAB and Simulink version in the Assess phase. This is not an exhaustive list but provides a representative list of important characteristics. In the Assess phase, the most important aspect to evaluate is the performance and whether there are show-stopping issues that cannot be resolved. Additional information about Regression Testing is covered in the Testing phase.

Result	Ideal	Notes
Simulation	Simulates without errors	The model simulates with errors, but the errors can be resolved.
Code Generation	Generate code without errors	The model generates code with errors, but the errors can be resolved.
Simulation Time	Faster than the previous version	A modest increase in simulation time may be acceptable, but the lower time should be maintained as the standard.

Code Generation Time	Faster than the previous version	A modest increase in code generation time may be acceptable, but the lower time should be maintained as the standard.

3.1.6 Decision to Upgrade

After completing the regression testing and evaluating the test results with the upgrade criteria, you can decide on whether to proceed to the next phase of the upgrade process or defer the upgrade to a later date.

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 phases of the process, including Migrate, Test, Release, and Support phases.

3.2.1 Upgrade Project Management

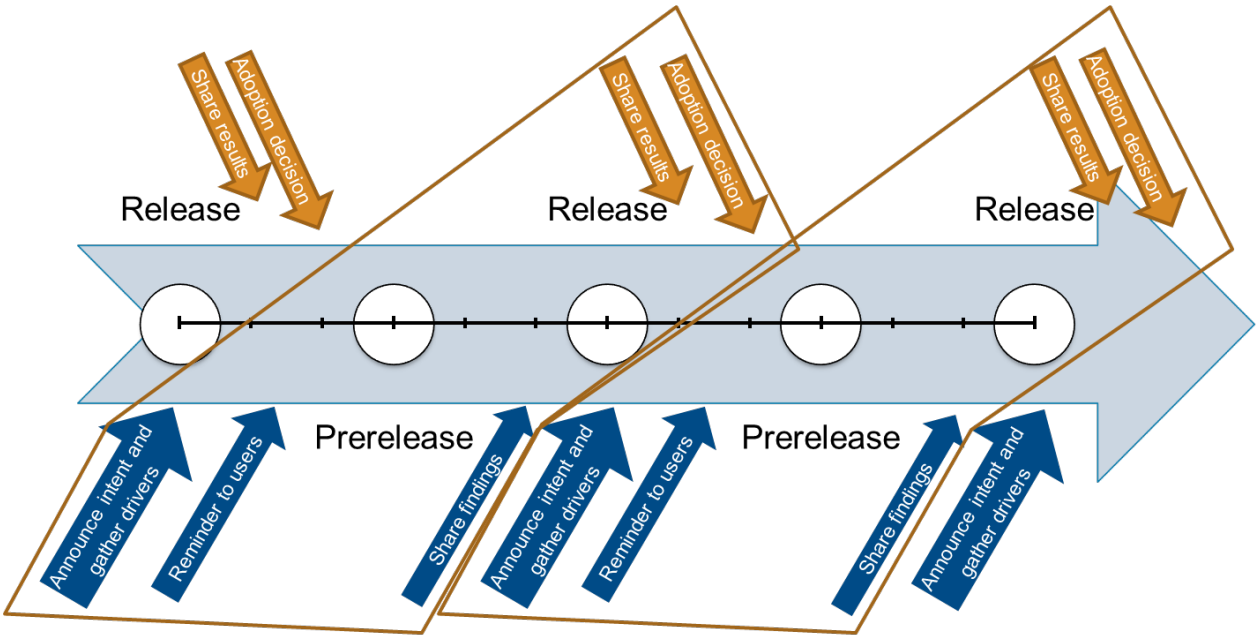
Develop an overall plan for the upgrade that includes the target date for the upgrade and the target MATLAB and Simulink version. In the plan, clearly define phases and the activities required to meet the exit criteria for each phase. Also define the roles and expectations for each team member as well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

Phase	Upgrade Lead	Engineering Teams	Management Sponsor	IT	Program Manager	Tools Team
Assess	Lead	Inform	Inform	Support	Inform	Contribute
Plan	Lead	Contribute	Contribute	Contribute	Contribute	Contribute
Migrate	Lead	Contribute	Inform	Support	Inform	Contribute
Test	Lead	Contribute	Inform	Inform	Inform	Contribute
Release	Contribute	Lead	Inform	Inform	Inform	Lead
Support	Lead	Contribute	Inform	Inform	Inform	Contribute

Communication is critical throughout the entire upgrade process: plans, status, decisions, and next steps should be shared regularly with the stakeholders. It is never too early to communicate about a potential upgrade. Notify the user community of new features that would drive an upgrade and include a timeframe for the Assess and Migrate phases. Share the results of each phase, including what went well and any difficulties identified by the tools team or the users. Continue to provide updates on status and decisions as well as any changes in timeframe. Note that communication should be an ongoing effort throughout an upgrade. Finally, for maximum impact, consider the needs of each stakeholder group and tailor the communications appropriately.

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 regularly. The following table and figure are examples of a communication timeline in relation to MATLAB and Simulink prerelease and release schedules.

Communication Timeline		
Date	**Audience**	**Topics**
3 months prior to the prerelease of MATLAB and Simulink	User base and management	Reminder of upcoming prerelease testing Intention(s) for upgrading, including: • List of known drivers for the upgrade • List of known barriers for the upgrade Request volunteers to test the prerelease
1 month prior to the prerelease of MATLAB and Simulink	User base	Reminder of upcoming prerelease testing Schedule of upcoming prerelease testing Changes in known drivers/barriers
Prerelease of MATLAB and Simulink	Active prerelease testing participants	Share prerelease testing progress and schedule
1 month after the prerelease of MATLAB and Simulink	User base	Share prerelease testing results, including successes and difficulties Share remaining prerelease testing plans, if any
1 month before the release of MATLAB and Simulink	User base and management	Share prerelease testing results Share upgrade assessment plan
Release of MATLAB and Simulink	Active upgrade assessment participants	Share upgrade assessment progress and schedule
1 month after the release of MATLAB and Simulink	User base	Share upgrade assessment results, including successes and difficulties Share remaining upgrade assessment plans, if any
2 months after the release of MATLAB and Simulink	User base and management	Share upgrade decision when ready, including plans and schedule



Communication timeline related to MathWorks releases

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 MATLAB and Simulink products. The business case for an upgrade may also be driven by operating systems 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 features in Simulink that you plan to use. Consider adding the evaluation of custom tool documentation and training material as a goal.

3.3 Migrate

In the Migrate phase, upgrade models and custom tools iteratively to the new version. In each iteration, 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 while reducing risk and effort for the entire organization and the end users.

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 a 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 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 starting the Custom Tool Migration phase of the upgrade process. Scale up the sandbox approach used in the Assess phase for use on multiple projects.

It is often advantageous to create standard configuration settings early in the migration phase. This can minimize duplication of effort and reduce later integration issues. The recommended settings from the initial testing provide a good starting point, but review any new or modified settings for each code target to ensure that these are appropriate. Apply these reviewed settings to models at the start of the migration 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 cannot test all the available models, then it is recommended that you select models with varying sizes and different modeling styles from various 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 create models outside the norm, include their models for this phase of the testing. Testing edge cases helps 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 typical advanced cases are of models with:

- Nested libraries
- Model Reference blocks
- Configurable subsystems or variants
- Both Model Reference blocks and nested libraries

3.3.3 Initial Migration

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:

- Open the projects
- 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 path and environment in MATLAB using projects
- 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 making updates to the model or 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)

During the Initial Migration phase, make only the minimum changes required for MATLAB files, data dictionaries, custom block libraries, and custom tools necessary to ensure that the model can be updated. Any files that the model is dependent on should be added to the MATLAB path. New features, optimizations, configuration sets, and S-Function API changes should be completed later in the Custom Tool Migration subphase.

For the data dictionary, 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.

For 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, and 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.

In rare cases, when upgrading from a release that is much older than the target release, 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 was not intended or expected for Simulink. In these instances, you may have to upgrade to an intermediate release of MATLAB and Simulink first to isolate and debug issues and later upgrade to the target release once these issues are resolved. Refer to Initial Testing in this document for more information on debugging issues in an intermediate version.

3.3.4 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.

3.3.4.1 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, available features in Simulink that you would benefit from using. This is also a good time to migrate your Configuration Parameter settings to recommended settings from MathWorks. 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. Consider removing modeling style guidelines that no longer apply and add updated guidelines for new features in Simulink that you plan to use.

While developing and migrating your custom tools, it is recommended that you create a design document that details the requirements and intended functionality for those 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 features in Simulink, 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	MATLAB Scripts that Interact with Simulink
• MATLAB Project • Environment setup MATLAB scripts • Compiler setup scripts • Data analysis scripts • User-built GUIs	• Scripts that load parameters • Scripts that load configuration sets • Scripts that set configuration options • Scripts for custom menus • User interface scripts • Scripts in block callbacks • Scripts within masks

If you have developed GUIs using GUIDE, convert them to App Designer. MathWorks recommends that you use projects instead of using model callbacks. If you still need to use model callbacks (for example, preload and postload scripts) make sure that you turn on callback tracing to assist with debugging issues caused by the callbacks.

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	Code Generation	Verification and Validation
• Simulation data • Logging customizations • Visualization customizations • Report generation setup files • Data comparison tools	• System Target Files (STF) • Template Make Files (TMF) • Target Language Compiler (TLC) files • Code Generation Template (CGT) files • Code Replacement Library (CRL) files • SIL/PIL environments	• Unit testing customizations • Test data • Reports • Signal and test authoring tools • Custom Model Advisor checks • Simulink Design Verifier customizations • Polyspace static analysis tools configuration • SIL/PIL environments

Configuration Settings

An upgrade invariably brings changes in configuration settings—new settings, modified settings, and deprecated settings. These changes need to be identified, understood, documented, and applied. In more recent versions, the configuration settings for a model can be exported from the Model Explorer. See Save a Configuration Set in the Simulink documentation for more details. For older versions without this feature, it is possible to write a custom script to export these settings.

Updating the configuration settings should be done early in the Migration phase to ensure the settings are well tested, that everyone benefits from the effort, and that models are compatible for integration. It is recommended that a process for adopting configuration setting changes is in place.

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 and Embedded Coder Dictionary. Refer to the *Simulink Data Dictionary* and *Embedded Coder Dictionary* documentation for more information.

Custom Block Libraries

When upgrading custom block libraries to the new version, use the opportunity to migrate custom blocks to built-in Simulink blocks or features. You can use the Release Notes to filter and search for new blocks and 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 parameter 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 step can be done manually or automated via a MATLAB script. If you are phasing out a block that you do not 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.

Custom Model Advisor Settings

Model Advisor Checks and settings tend to change with every release. An upgrade offers an opportunity to review your current Model Advisor settings while investigating new checks. Any custom checks should also be tested and, if necessary, updated. A set of version-specific checks that meet your organization's needs and guidelines should be developed up-front to reduce the time spent by individual teams.

3.3.4.2 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. If you have not developed automated regression tests for your custom tools, create them now so they can be reused any time that you update and upgrade the custom tools.

3.3.4.3 Third-Party Tool Testing

It is recommended that you test all your third-party tools with the new release of MATLAB and Simulink 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. Examples include:

Third-Party Products to Consider when Upgrading

- Requirements management tools (for example, IBM Rational DOORS)
- Calibration support tools
- Configuration management tools integrated with MATLAB and Simulink products
- Change management tools integrated with MATLAB and Simulink products
- Authoring/system engineering tools

- Co-simulation tools
- IDEs
- Compilers
- Hardware

3.3.5 Automated Migration

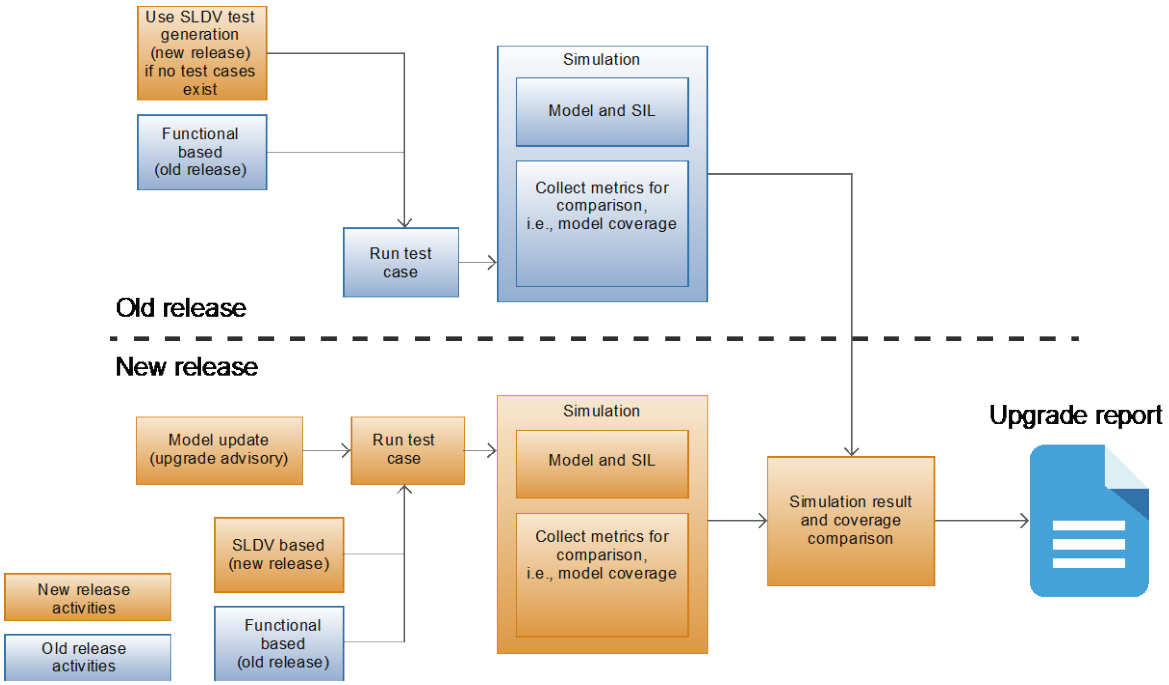
Once you have performed the initial migration and custom tool migration, the next step is to automate as much of the upgrade process as possible.

3.3.5.1 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 process should be as automated as possible. This MATLAB script should call various functions to automate the following:

- Upgrade Advisor
- Custom library content
- Configuration parameters
- Workspace objects
- Regression tests after upgrade
- Custom tools upgrade process

These automation tools generate a report that provides details about errors and warnings encountered for each phase. Here is an example workflow that can accomplish this using Simulink Design Verifier (SLDV) and SIL simulations. This workflow can be implemented with a purely script-based approach or a UI that automates the workflow and guides users through the upgrade process. The following figure is an example of an upgrade process that can be used to automatically upgrade Simulink models from one release to another.



Example Upgrade Process

3.3.5.2 Test Automation Tools

Run the upgrade script with the custom tools on all models prior to releasing the new MATLAB and Simulink version internally within your organization. If you cannot upgrade all of your models, select models with different modeling styles and model sizes from various 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 practice 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. This batch mode can also be implemented on a continuous integration server. By using a CI server, the computing resources can be pooled, and the activities can automatically occur once the organization is ready to begin an upgrade process.

3.3.5.3 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. This upgrade activity can be carried out on a local machine or scaled up to a CI server to complete the upgrade process automatically.

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. This iterative loop can also occur on a CI server. In this scenario, the CI server will execute the upgrade activities.

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.

3.4.1 Regression Testing

Run regression tests on tasks that users commonly perform so that you have an estimate how long these tasks will take in the new MATLAB and Simulink version. When 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 currently in use.

This activity will help you discover potential issues before releasing the new version. It is 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.

Regression testing should be thorough with a variety of checks against a baseline of current results. Consistency checks will help identify changes that must be understood and will eventually determine the success (or failure) of any upgrade. The following table provides a key list of items that should be checked to evaluate an upgrade. This is not an exhaustive list but provides a representative list of important characteristics. Note that both functionality and performance should be considered. Obviously, having (preferably automated) tests that cover these aspects makes this process more accessible. Any improvements to testing performed as part of the upgrade process can and should be integrated into the development workflow. Finally, the test plan should be scaled accordingly for the different phases.

Result	Ideal	Acceptable	Notes
Simulation Results	Identical results	Within tolerance	Logged signals should include all model outputs. Additional signals can be logged if they are deemed important intermediate results.
Coverage Metrics	Identical Results	Explained by changes in the coverage tools	Simulation results from tests that achieve near 100% coverage can give greater confidence. NOTE: Changes to the test coverage tools may change the coverage metrics across versions. It is important to understand these situations and be sure that you are confident in your tests.
Code Generation	Equivalent	No errors	SIL testing can make Acceptable results here OK.
SIL Testing	Identical results	Within tolerance	Logged signals should include all model outputs. Additional signals can be logged if they are deemed important intermediate results.
Simulation Time	Faster than the previous release	Within tolerance	A modest increase in simulation time between the previous release and the new release may be acceptable.
Code Generation Time	Faster than the previous release	Within tolerance	A modest increase in code generation time between the previous release and the new release may be acceptable.
Third-Party Compatibility	Equivalent	Equivalent	Third-party software, tools, and possibly hardware must work with new versions of MATLAB.
Errors and Warnings	Similar Results		Any new errors should be resolved. Any new warnings should be investigated and understood.

3.4.2 Equivalence Testing

It is recommended that you run equivalence tests to compare the simulation results for each model and code in the previous version to the model and code in the new version. The process referenced in section 3.3.5.1 has an equivalence test phase in the example process. It is important that you only add any new MATLAB and Simulink features to the model after this equivalence testing is completed. Otherwise, the models and generated code may not be functionally equivalent between the versions. Simulink Test supports running tests in multiple releases of MATLAB, equivalence testing, and cross-release code testing. Simulink Design Verifier can be used to generate the test cases that results in 100% test coverage for the equivalence testing.

3.4.3 Continuous Integration Testing

A continuous integration environment is recommended to automatically upgrade, integrate, simulate, generate code, test equivalence, and generate test reports. This should be the long-term goal for large organizations, which will make the upgrade process more seamless than manually converting and testing each model and cost effective and with minimal impact on the end users. Using a CI server significantly reduces the amount of manual engineering effort needed during upgrade. The same workflow that is listed in section 3.3.5.2 can be used for all model and tool upgrades. The CI server should implement the custom process that has been put in place to upgrade models and MATLAB files. These tasks are repeated for each component on the CI server until an attempt has been made to upgrade each model and MATLAB file.

3.4.4 Beta Testing

In the final phase of testing, have a small group of selected users beta test the custom tools and the model upgrade process. Include beta testers that are 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.

3.4.5 Testing Updates

Test and incorporate MATLAB and Simulink Updates into your organizations test plans. MathWorks recommends that you use the latest Update whenever it is available and as soon as it is feasible for your organization or programs. Prior to releasing the target MATLAB and Simulink version within an organization, confirm which Update is available for that version and repeat the regression testing previously performed with the latest Update. Ideally, your organization has automated this type of testing using a CI server and this is a quick test to evaluate the Update. If not, it is recommended that you follow the regression testing documented in the Assess phase of this paper to identify issues that introducing the Update this late in the upgrade process may or may not cause. Updates contain bug fixes, so migrating to the latest Update should not introduce any new issues. If your organization does encounter an issue, report the issue to MathWorks Support by creating a Service Request on MathWorks website so a fix can be considered for a future Update. After you have upgraded to the new version, this regression testing can be quickly repeated on a CI server any time that MathWorks releases a new Update to assess the impact of releasing the new Update to your organization.

3.5 Release

The purpose of the Release phase is to release the new MATLAB and Simulink version, custom tools, and 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 does not 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 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 several post-upgrade activities to make the next upgrade easier. If you did not 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 used in your organization. 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 regression tests on each prerelease and provide feedback to MathWorks even if you do not intend to upgrade to that release. 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 regression tests on the new MATLAB and Simulink version that you intend to target as well as any Update that MathWorks releases.

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 in a CI environment 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. The following is an incomplete list of tests that can be run in a CI environment:

- 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

A retrospective look at upgrades can reveal some lessons worth documenting. It takes some effort and experience to determine which architecture, modeling style, and customization lessons apply to an upgrade in general and which lessons are related to a particular version. Create a repository or wiki to document these lessons with links to version-specific lessons. Document any new lesson on the appropriate version-specific page and regularly review those pages to find (and promote) the more general items.

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 seminars, webinars, and Conferences from MathWorks.

4.1 Prerelease Testing

When possible, test your models and custom tools for each prerelease and report issues to MathWorks Support by creating a Service Request on MathWorks website. 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.

If IMT is not appropriate, you may want to devise a similar approach internally. Implementing test automation in a CI environment is a first step. Adding flexibility for targeting a specific version of MATLAB and Simulink allows you to emulate MathWorks Industry Model Testing—at least for prerelease and general release versions. This can provide early notification of potential issues or provide early confidence in a new release.

4.3 Seminars, Webinars, and Conferences

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

5 MathWorks Support

MathWorks offers a wide variety of self-serve support options for the upgrade process. Some of these self-serve options include online product documentation, release notes that can be filtered to search for new features and incompatibilities, examples on how to use the new features, MATLAB Answers, bug reports, and an active user community (MATLAB Central). If additional support is required, contact support for specific questions about the products, training for product training, or consulting services for upgrade assistance.

6 Appendix

Checklist Content
ASSESS: UNDERSTAND THE IMPACT OF AN UPGRADE
☐ Identify the trigger for upgrading
☐ Evaluate the current situation, including possible incompatibilities
☐ Choose a target version of MATLAB and Simulink
☐ Test a few models in the new version with typical workflows
☐ Run regression tests on tasks that users commonly perform
☐ Create an Upgrade Evaluation Report
☐ Go/no go decision
PLAN: DEFINE THE OVERALL SCOPE OF THE UPGRADE PROJECT
☐ Create a business case stating the ROI
☐ Establish the scope by identifying affected models, projects, and organizations
☐ Create a detailed plan with timing, resources, key stakeholders, and more
MIGRATE: CONVERT A PROJECT IN AN ITERATIVE MANNER TO THE NEW VERSION
☐ Initial migration: Migrate a small set of models and test your typical workflows
☐ Custom tool migration: Replace custom tool functionality with built-in MATLAB and Simulink functionality
☐ Automated migration: Expand testing to models with different modeling styles and use automation tools to upgrade the rest of the models
TEST: VALIDATE UPGRADED MODELS
☐ Run regression tests on tasks that users commonly perform
☐ Conduct open-loop, MIL, SIL, and PIL simulations on desktops using Simulink products for verification and validation
☐ Conduct HIL, rapid-prototyping, and on-target rapid prototyping tests in the lab
☐ Test third-party tools
☐ Validate functional equivalence of models in the new version
☐ Complete beta testing of the new version with selected users
RELEASE: RELEASE THE NEW MATLAB VERSION AND CUSTOM TOOLS
☐ Schedule and conduct training classes
☐ Release new MATLAB version and custom tools to users
☐ Recommend that the model experts convert their models
☐ Set a deadline for everyone to complete the migration
SUPPORT: AUTOMATE THE UPGRADE PROCESS AND REPORT ISSUES
☐ Consider automating your upgrade process using a CI server
☐ Complete a retrospective and document lessons learned from the upgrade
☐ Test every prerelease and report issues to MathWorks Support