

Bringing CI/CD to Digital Engineering

Unit Testing, Model Assessments, & Build Automation

Robert Peters

Principal Engineer

Catherine Haggerty

Software Engineer

Mark Petrotta

Principal Engineer

Agenda

- About Us
- What We Can Learn From Software
- Applying CI/CD to MBSE
- "Hello World" Example
- Limitations & Future Considerations

About Us



- 40 years combined experience in software, 20 years in MBSE.
- Have collectively seen the benefits of Continuous Integration Continuous Delivery (CI/CD) in the software world. When we were each brought to the MBSE world, we saw that these things were missing.
- We value quality and efficiency. Early detection, yay!

What We Can Learn From Software

Software engineering nearly broke 20 years ago

- Team Structure
- Version Control Systems
- Review Process
- Merge / Integration Issues
- Build & Test Challenges
- Traceability & Requirements
- Tooling Lock-in

1991-2005



*decentralized growth
without tools*

2001-2010s



*heavyweight,
centralized, contractor-
driven processes*



**Modern era of distributed version
control, modular architectures, CI/CD,
and DevSecOps**

What We Can Learn From Software

What challenges do engineers face and how does software address them?

Common Challenges

- Manual validation leads to inefficiencies and missed errors.
- Slow feedback loops hinder agility and collaboration.
- Testing processes lack standardization and repeatability.
- Development process needs traceability across lifecycle.

The Software CI/CD Solution

- Automated pipelines catch common errors, inefficiencies, and security risks.
- Automation enables quicker feedback loops fostering agility and collaboration.
- Automated pipelines ensure the same tests run on a standardized environment.
- Entire development process is documented, allowing for strong traceability.

Reduced Errors

Faster Feedback

Standardized Process

Enhanced Traceability

The Software CI/CD Process

Software has developed efficient processes for addressing these engineering challenges.

- Feature requests, bugs, change requests, etc. are recorded in an issue tracking tool
- Source code version history is recorded, and branches are created for each issue
- Source code and test code are developed in conjunction, known as test-driven development
- Source code is automatically tested in a common environment using automation tools
- Releases are published to an integrated artifact repository
- Releases are deployed to active environment

Applying CI/CD to MBSE

Applying CI/CD to MBSE

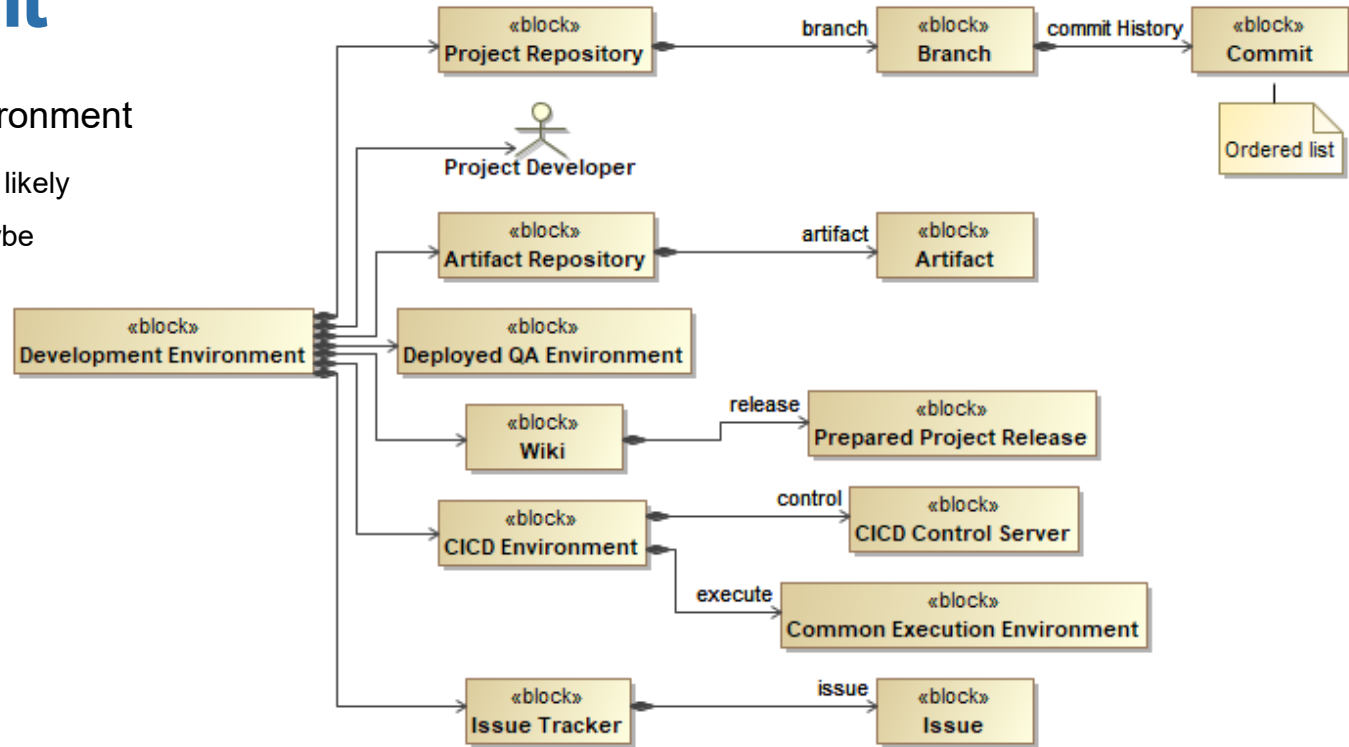
Comparing current software CI/CD solutions with current and future MBSE CI/CD solutions.

Topic	Current Software	Current MBSE	Future MBSE
Issue Tracking	Well-managed	Less formal	Improved traceability
Version Control	Feature-based branching	Minimal branching	Issue-based branching
Test-Driven Development	Unit tests for each method	Validation suites and simulation	Create model with testing in mind
Automated Testing	Runs for every push or pull request, blocks merge	Not enforced	Expanded testing, blocks merge
Integrated Artifact Repository	Build images are stored with strong traceability	Models are stored with limited traceability	MBSE artifacts are stored with strong traceability

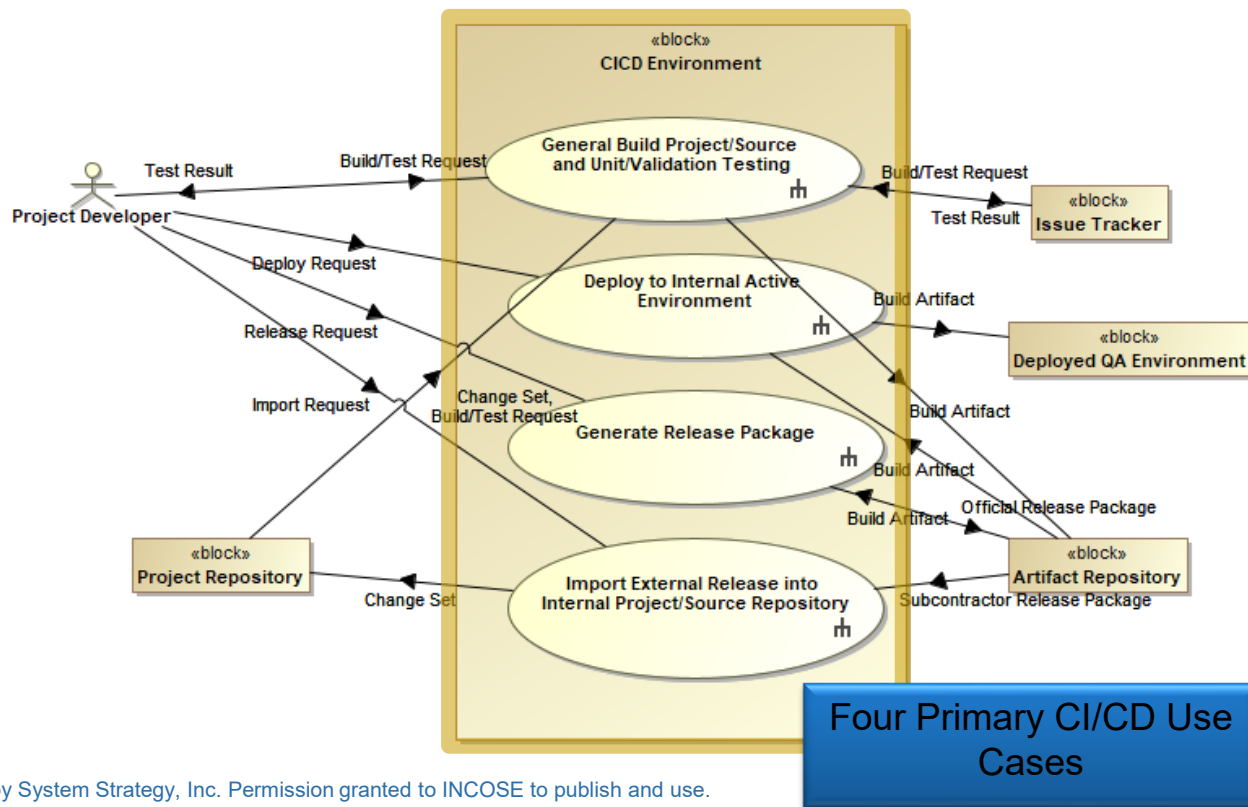
Common Development Environment

Focus on the CI/CD Environment

- Every software or MBSE team likely has each of these, except maybe the CICD Environment

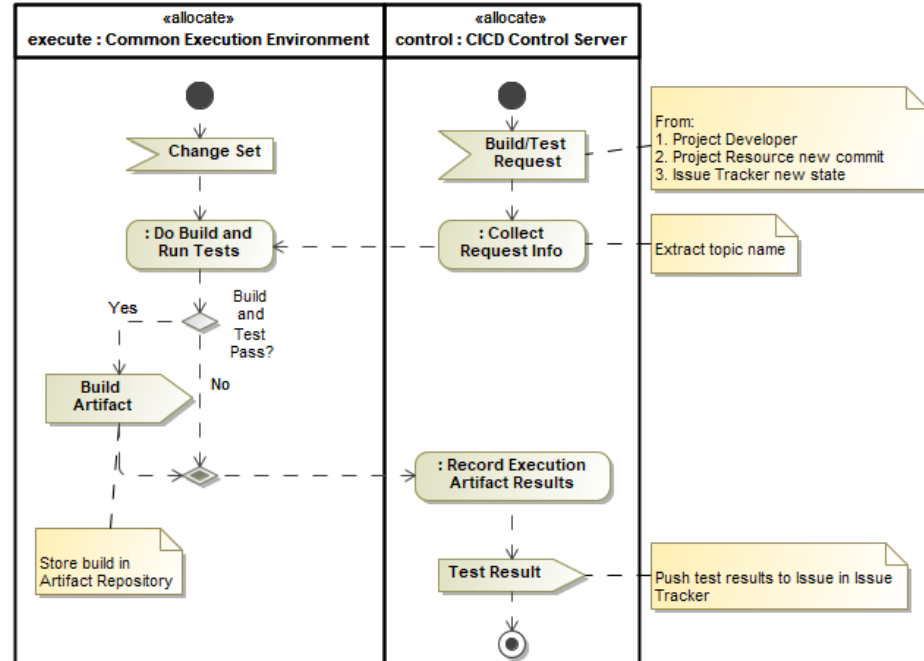


Four Common CI/CD Use Cases for Software and MBSE



General Build Project/Source and Unit/Validation Testing

- Software Examples
 - Build SW, check for build errors
 - Run Unit Test, check for logic errors
- MBSE Examples
 - Run simulation
 - Run automatic validation checks
- Hybrid Example (Model + Code)
 - Build code that interacts with model
 - Check against test model



```

graph TD
    A[General Build Project Source and Intermediation Testing] --> B[Deploy to Internal Action Environment]
    B --> C[Generate Release Package]
    C --> D[Import External Release into Internal Project/Source Repository]
  
```

-
- ```

graph TD
 subgraph execute ["execute : Common Execution Environment"]
 Start1(()) --> BuildArtifact1[/Build Artifact/]
 BuildArtifact1 --> SendToEnv[": Send to Deployed Environment and Execute"]
 SendToEnv --> BuildArtifact2[/Build Artifact/]
 end

 subgraph control ["control : CICD Control Server"]
 Start2(()) --> DeployRequest[/Deploy Request/]
 DeployRequest --> CollectInfo[": Collect Request Info"]
 CollectInfo --> RecordResults[": Record Execution Artifact Results"]
 RecordResults --> End2((()))
 end

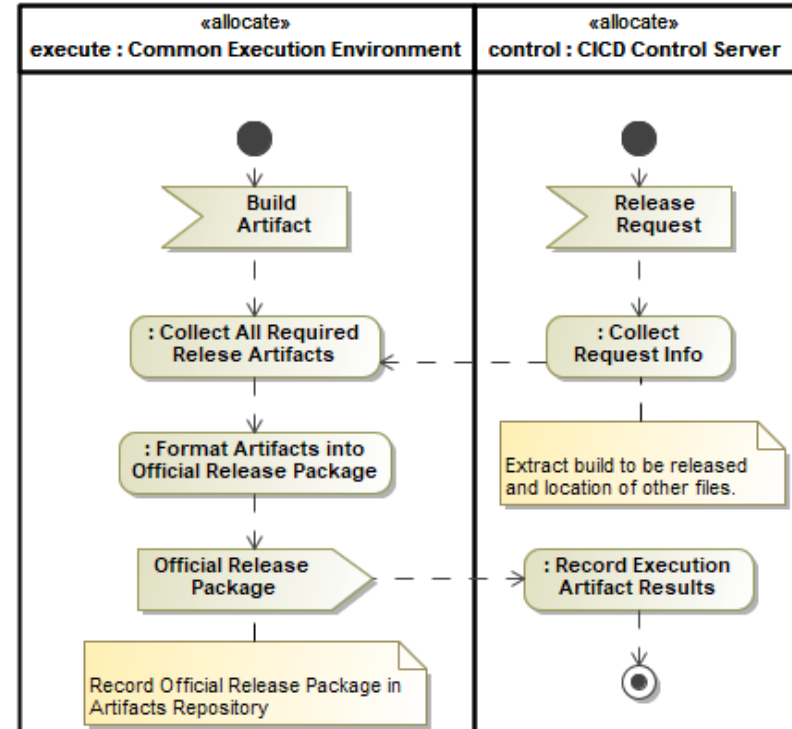
 BuildArtifact2 -.-> RecordResults

```

Extract build to be deployed and environment to deploy to.

# Generate Release Package

- Collect from project repository and other artifacts to create a consistent release package that is traceable to the commit history, issue history, tests, etc. and accessible via standard artifact repository
- Generate documentation for release or review
  - Extract diagrams and documentation from the model
  - Create list of change sets
  - Create list of closed issues
  - Generate the diff between two versions
- Alternate:
  - Generate metrics
  - Extract detailed custom information via scripts

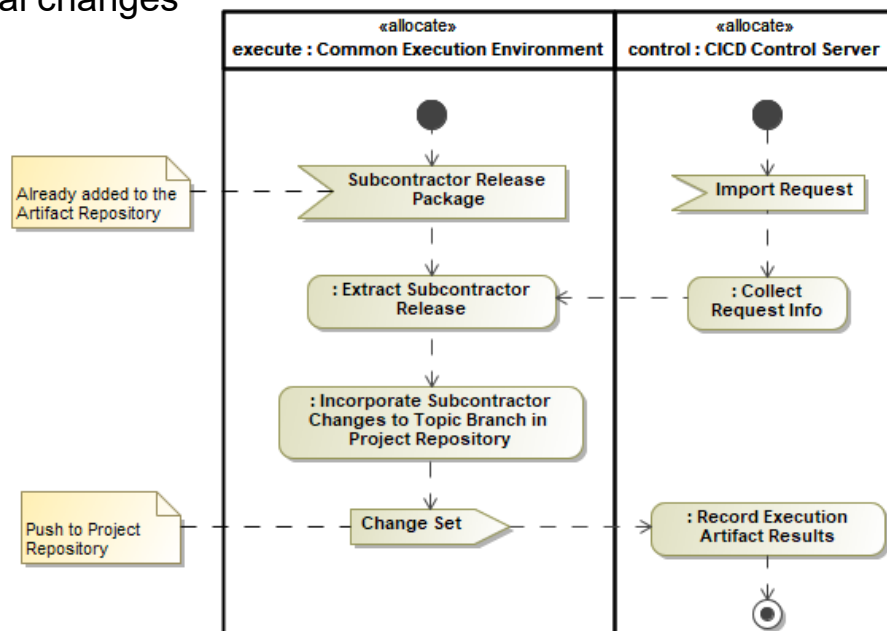


# Import External Release into Internal Project/Source Repository



Process for receiving and accepting external changes

- External suppliers may not have direct access to the Project Repository.
- This process is to integrate the release into a change set that is part of the Project Repository that can be handled like any other change.
- Process may also run automated acceptance tests including RoC.



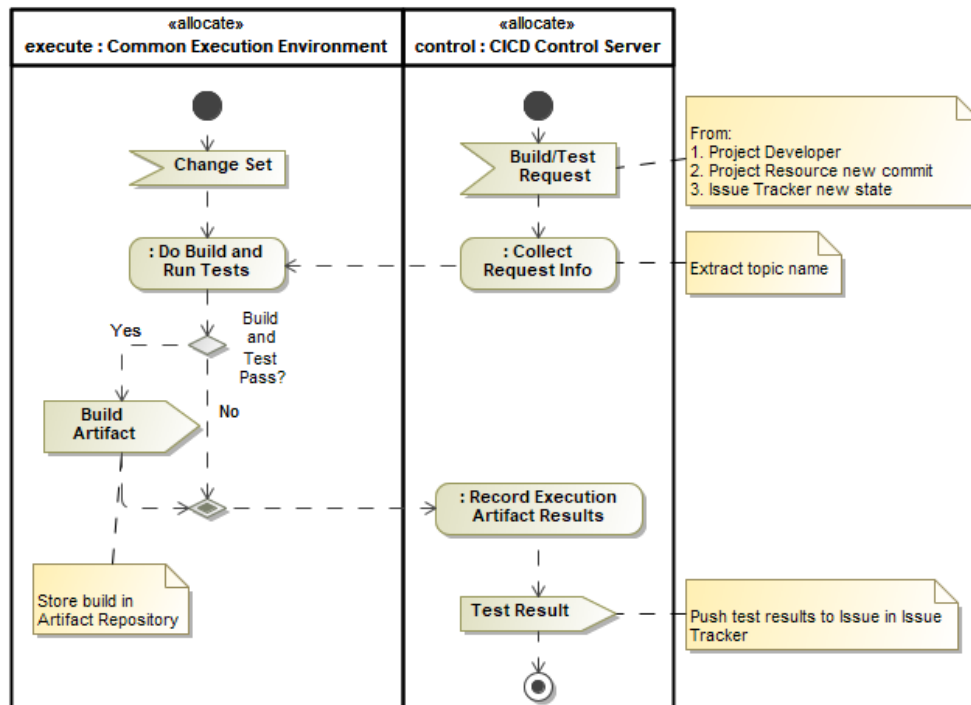
# **“Hello World” Example of CI/CD for MBSE**



# “Hello World” Example Overview

Doing a basic test of a model

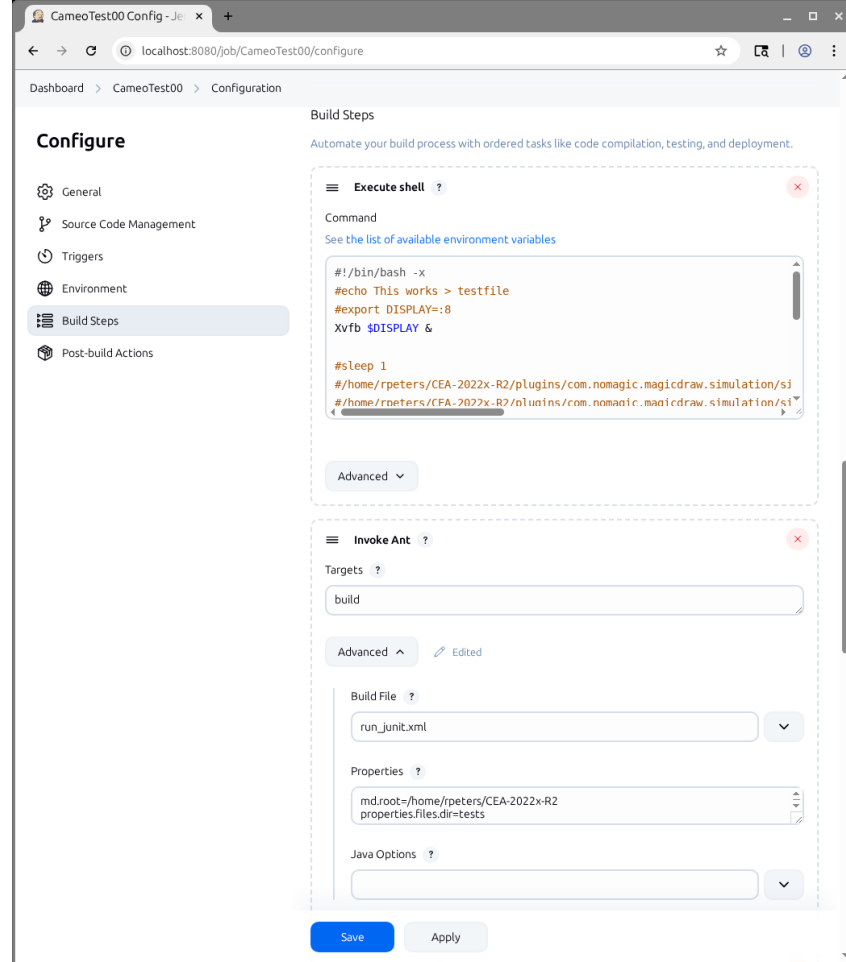
- Follows the behavior for use case “General Build Project/Source and Unit/Validation Testing”
- This test will use Jenkins as the CI/CD control server to kick off a Cameo simulation test from the command line and return the results to Jenkins



# Jenkins Project Configuration 1/2

Shows the unique configuration

- Xvfb is needed to run Cameo headless (in the background)
  - Xvfb: X virtual framebuffer
- The DISPLAY environment variable is configured on the Node
- The Ant scripts are on the local filesystem, but they would eventually be in git (or similar text-based project repository)



The screenshot shows the Jenkins 'Configure' page for a job named 'CameoTest00'. The 'Build Steps' tab is selected in the left sidebar. The main content area shows two build steps:

- Execute shell**: A step with a command block containing:
 

```
#!/bin/bash -x
#echo This works > testfile
#export DISPLAY=:8
Xvfb $DISPLAY &

#sleep 1
#/home/rpeters/CEA-2022x-R2/plugins/com.nomagic.magicdraw.simulation/si
#/home/rpeters/CEA-2022x-R2/plugins/com.nomagic.magicdraw.simulation/si
```
- Invoke Ant**: A step with the following configuration:
  - Targets**: build
  - Build File**: run\_junit.xml
  - Properties**: md.root=/home/rpeters/CEA-2022x-R2, properties.files.dir=tests
  - Java Options**: (empty)

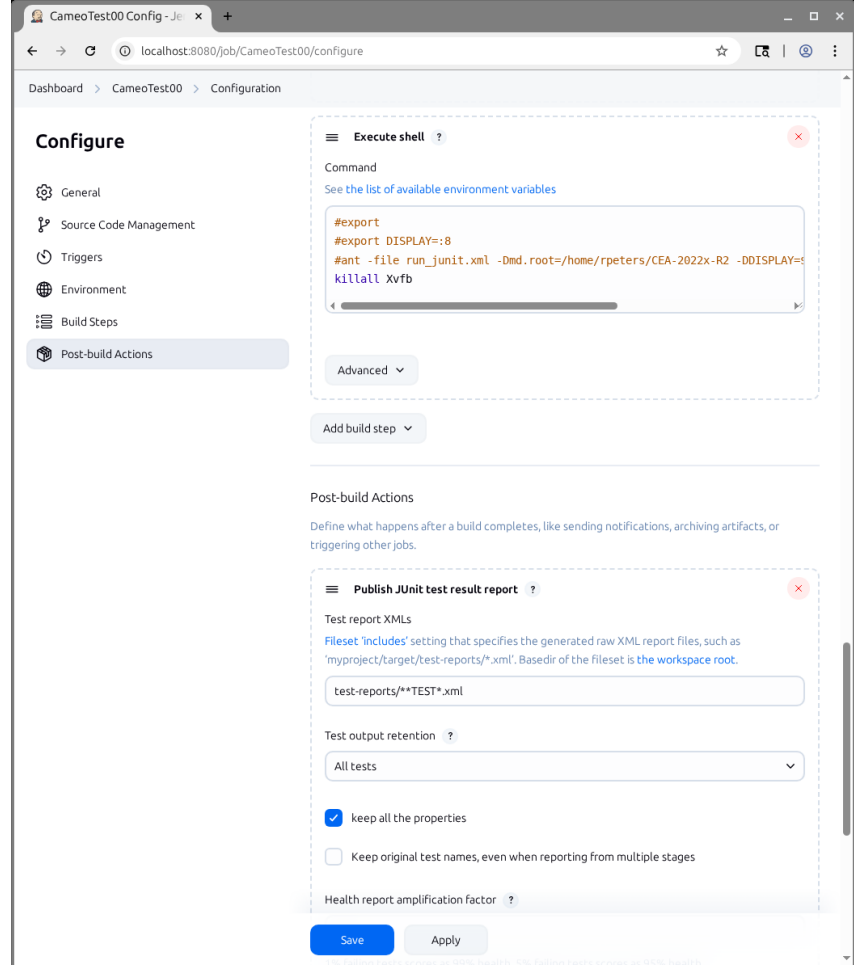
At the bottom of the configuration page are 'Save' and 'Apply' buttons.



# Jenkins Project Configuration 2/2

Shows the unique configuration

- Kill Xvfb to cleanup for next time
- Collect the test results as build artifacts



The screenshot shows the Jenkins configuration page for a job named 'CameoTest00'. The left sidebar contains a 'Configure' section with a list of tabs: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The 'Post-build Actions' tab is selected. The main content area shows two configuration sections:

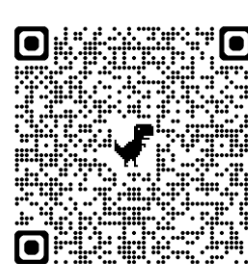
- Execute shell**: A section for running shell commands. The 'Command' field contains the following script:
 

```
#export
#export DISPLAY=:8
#ant -file run_junit.xml -Dmd.root=/home/rpeters/CEA-2022x-R2 -DDISPLAY=:8
killall Xvfb
```
- Publish JUnit test result report**: A section for publishing test results. It includes a 'Test report XMLs' field with the value 'test-reports/\*\*TEST\*.xml', a 'Test output retention' dropdown set to 'All tests', and checkboxes for 'keep all the properties' (checked) and 'Keep original test names, even when reporting from multiple stages' (unchecked). There is also a 'Health report amplification factor' field.

At the bottom of the configuration page are 'Save' and 'Apply' buttons.

# Example Cameo Project

The Cameo Simulation Toolkit example model.



CATIA | Cameo Enterprise Architecture 2022x - TestVerdictKind.mdzip [C:\Users\usofr\Downloads\TestSimulationCommandLine\tests\]

File Edit View Layout Diagrams Options Tools Analyze Collaborate 3DEXPERIENCE Window Help

Preview: - no preview - Full Model Create Diagram

Model [Read-Only] x

package Model [ Model ]

«SimulationConfig»  
**Run GeneratePassResult**  
«SimulationConfig»  
addControlPanel = false  
animationSpeed = 95  
autoStart = true  
autoStartActiveObjects = true  
cloneReferences = false  
constraintFailureAsBreakpoint = false  
executionTarget = GeneratePassResult  
fireValueChangeEvent = true  
initializeReferences = false  
numberOfRuns = 1  
runForksInParallel = true  
silent = false  
solveAfterInitialization = true  
startWebServer = false  
timeVariableName = "simtime"  
treatAllClassifiersAsActive = true

«SimulationConfig»  
**Run GenerateFailResult**  
«SimulationConfig»  
addControlPanel = false  
animationSpeed = 95  
autoStart = true  
autoStartActiveObjects = true  
cloneReferences = false  
constraintFailureAsBreakpoint = false  
executionTarget = GenerateFailResult  
fireValueChangeEvent = true  
initializeReferences = false  
numberOfRuns = 1  
runForksInParallel = true  
silent = true  
solveAfterInitialization = true  
startWebServer = false  
timeVariableName = "simtime"  
treatAllClassifiersAsActive = true

act [Test Case] GeneratePassResult [ Ger

«valueSpecification»  
**pass**

result

«valueType»  
**verdict : VerdictKind**

Case] GenerateFailResult [ GenerateFa

«valueSpecification»  
**fail**

result

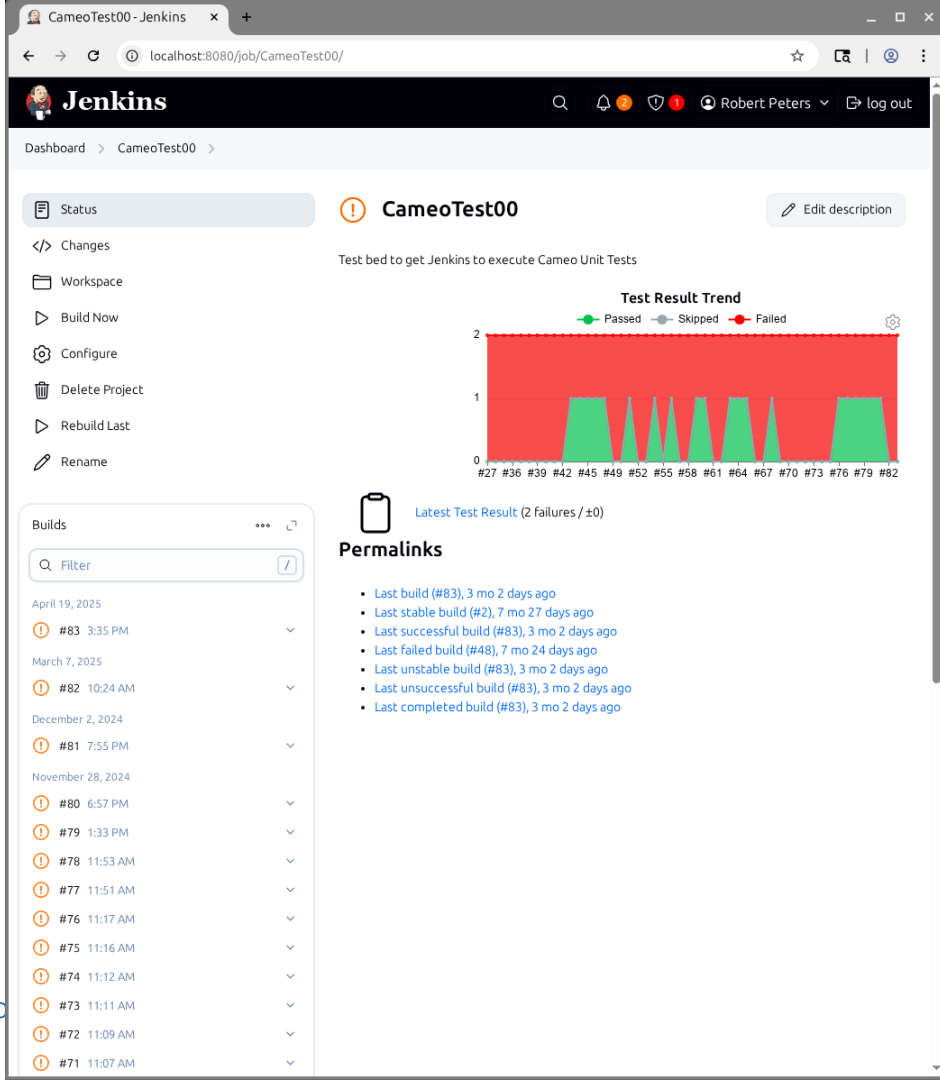
«valueType»  
**verdict : VerdictKind**

Ready

# Jenkins Project Results 1/5

## Test Result History

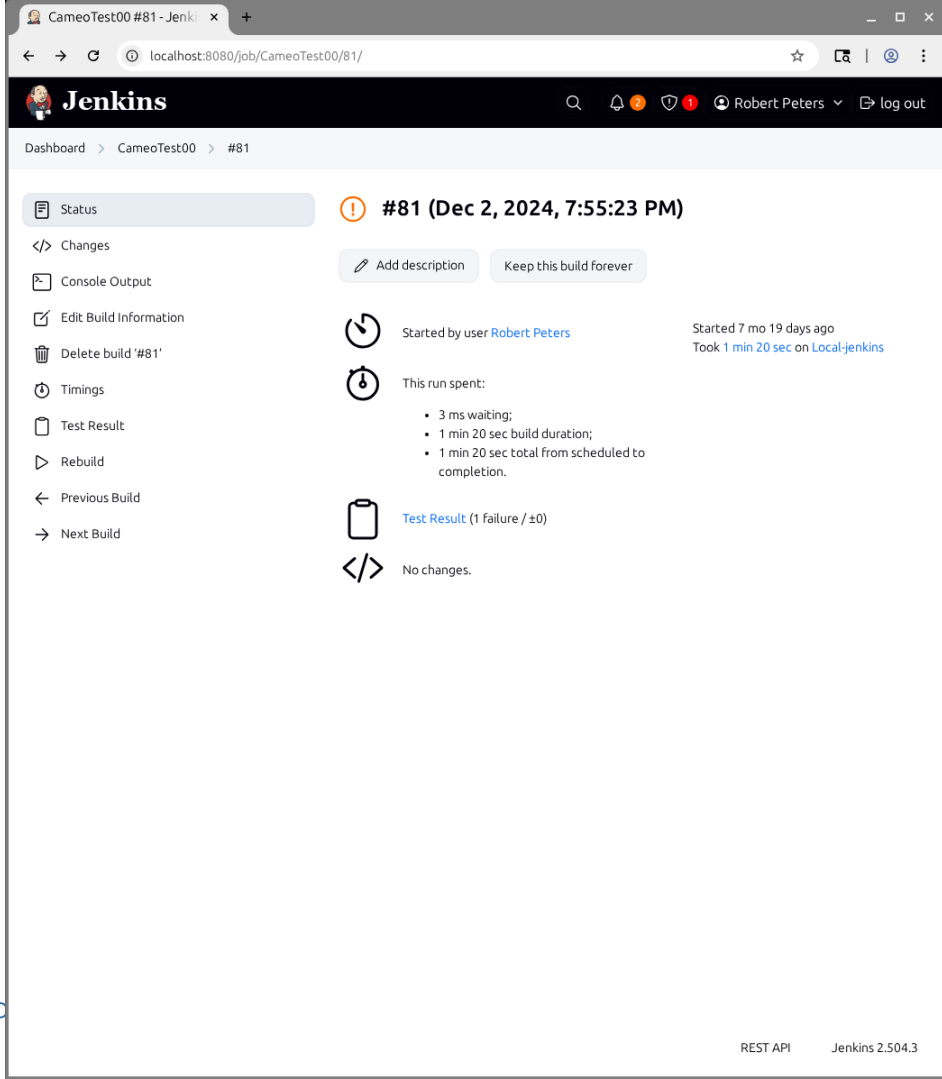
- When the build is running correctly, 1 will pass and 1 will fail.
- 2 fails is when the build isn't running correctly



# Jenkins Project Results 2/5

## Single build result

- Shows build time
- Has link to Test Result



The screenshot shows the Jenkins web interface for build #81 of the 'CameoTest00' job. The browser address bar shows 'localhost:8080/job/CameoTest00/81/'. The Jenkins header includes the logo, search, notifications, and user 'Robert Peters'. The breadcrumb trail is 'Dashboard > CameoTest00 > #81'.

On the left sidebar, the 'Status' tab is selected. Other options include 'Changes', 'Console Output', 'Edit Build Information', 'Delete build '#81'', 'Timings', 'Test Result', 'Rebuild', 'Previous Build', and 'Next Build'.

The main content area displays the build status as '#81 (Dec 2, 2024, 7:55:23 PM)' with a warning icon. It includes buttons for 'Add description' and 'Keep this build forever'. The build was started by user 'Robert Peters' 7 months and 19 days ago, taking 1 min 20 sec on 'Local-jenkins'.

The 'This run spent:' section lists the following timings:

- 3 ms waiting;
- 1 min 20 sec build duration;
- 1 min 20 sec total from scheduled to completion.

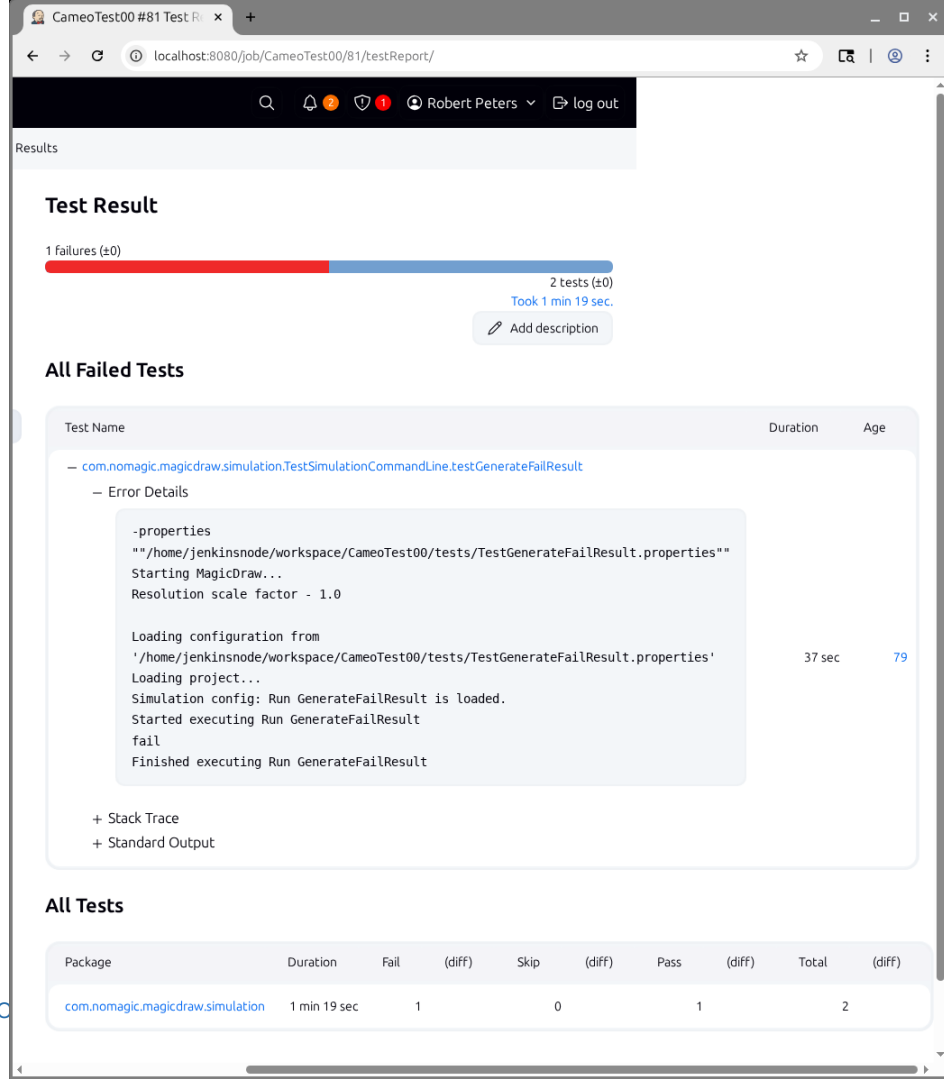
Below this, there is a link to the 'Test Result (1 failure / ±0)' and a note 'No changes.' at the bottom.



# Jenkins Project Results 3/5

## Test Result Details

- Expandable details about
  - Error Details
  - Stack Trace
  - Standard Output



The screenshot shows the Jenkins Test Results page for a job named 'CameoTest00 #81 Test Report'. The page displays a summary of test results, including a bar chart showing 1 failure and 2 tests. The duration of the test is 1 min 19 sec. Below the summary, there is a section for 'All Failed Tests' which shows a table with columns for Test Name, Duration, and Age. The table contains one entry for 'com.nomagic.magicdraw.simulation.TestSimulationCommandLine.testGenerateFailResult' with a duration of 37 sec and an age of 79. Below the table, there is a section for 'All Tests' which shows a table with columns for Package, Duration, Fail, (diff), Skip, (diff), Pass, (diff), Total, and (diff). The table contains one entry for 'com.nomagic.magicdraw.simulation' with a duration of 1 min 19 sec, 1 failure, 0 skips, 1 pass, and a total of 2 tests.

Results

### Test Result

1 failures (±0)

2 tests (±0)  
Took 1 min 19 sec.

[Add description](#)

### All Failed Tests

| Test Name                                                                         | Duration | Age |
|-----------------------------------------------------------------------------------|----------|-----|
| com.nomagic.magicdraw.simulation.TestSimulationCommandLine.testGenerateFailResult | 37 sec   | 79  |

— Error Details

```
-properties
"/home/jenkinsnode/workspace/CameoTest00/tests/TestGenerateFailResult.properties"
Starting MagicDraw...
Resolution scale factor - 1.0

Loading configuration from
'/home/jenkinsnode/workspace/CameoTest00/tests/TestGenerateFailResult.properties'
Loading project...
Simulation config: Run GenerateFailResult is loaded.
Started executing Run GenerateFailResult
fail
Finished executing Run GenerateFailResult
```

+ Stack Trace  
+ Standard Output

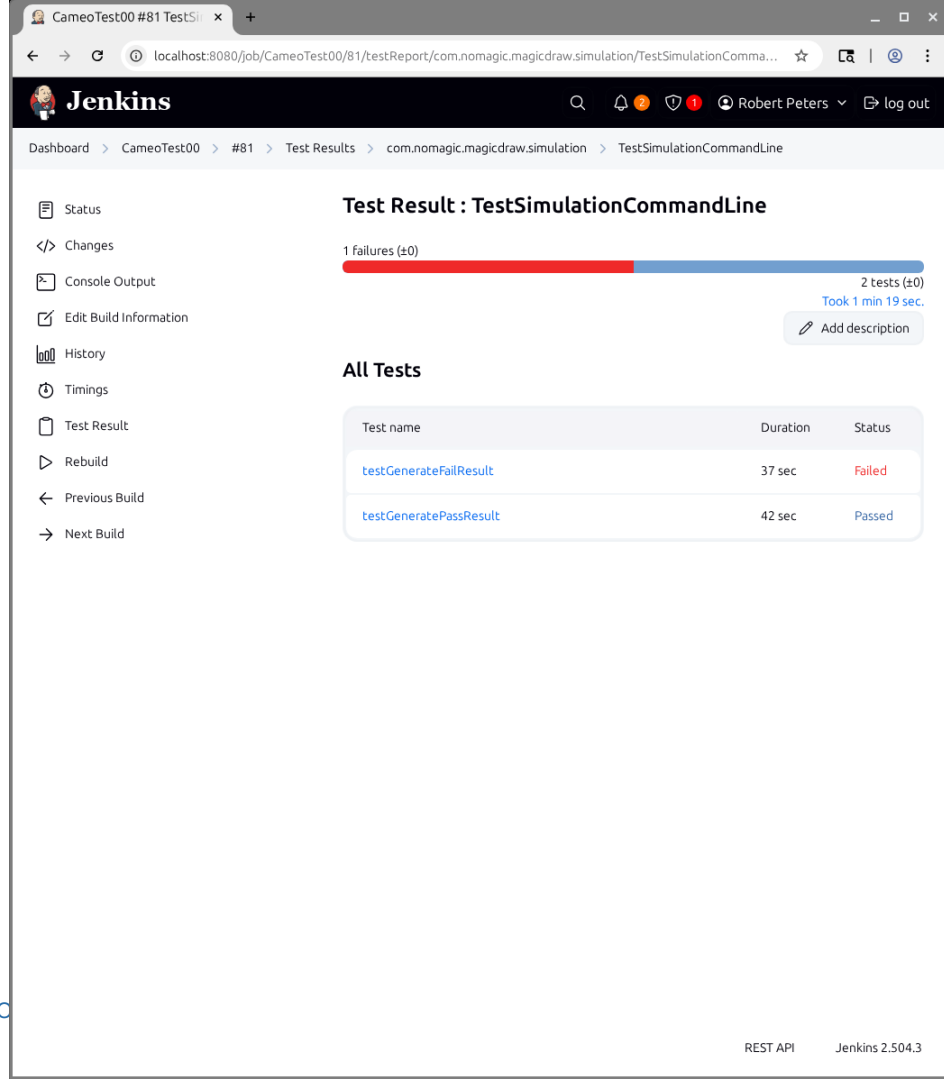
### All Tests

| Package                          | Duration     | Fail | (diff) | Skip | (diff) | Pass | (diff) | Total | (diff) |
|----------------------------------|--------------|------|--------|------|--------|------|--------|-------|--------|
| com.nomagic.magicdraw.simulation | 1 min 19 sec | 1    |        | 0    |        | 1    |        | 2     |        |

# Jenkins Project Results 4/5

## All Tests

- See links for the two tests
- Time Duration
- Pass/Fail Status



The screenshot shows the Jenkins web interface for a test run. The breadcrumb trail is: Dashboard > CameoTest00 > #81 > Test Results > com.nomagic.magicdraw.simulation > TestSimulationCommandLine. The left sidebar contains links for Status, Changes, Console Output, Edit Build Information, History, Timings, Test Result, Rebuild, Previous Build, and Next Build. The main content area is titled "Test Result : TestSimulationCommandLine" and shows "1 failures (±0)" with a progress bar. It also indicates "2 tests (±0)" and "Took 1 min 19 sec." with an "Add description" button. Below this is an "All Tests" table:

| Test name                              | Duration | Status |
|----------------------------------------|----------|--------|
| <a href="#">testGenerateFailResult</a> | 37 sec   | Failed |
| <a href="#">testGeneratePassResult</a> | 42 sec   | Passed |

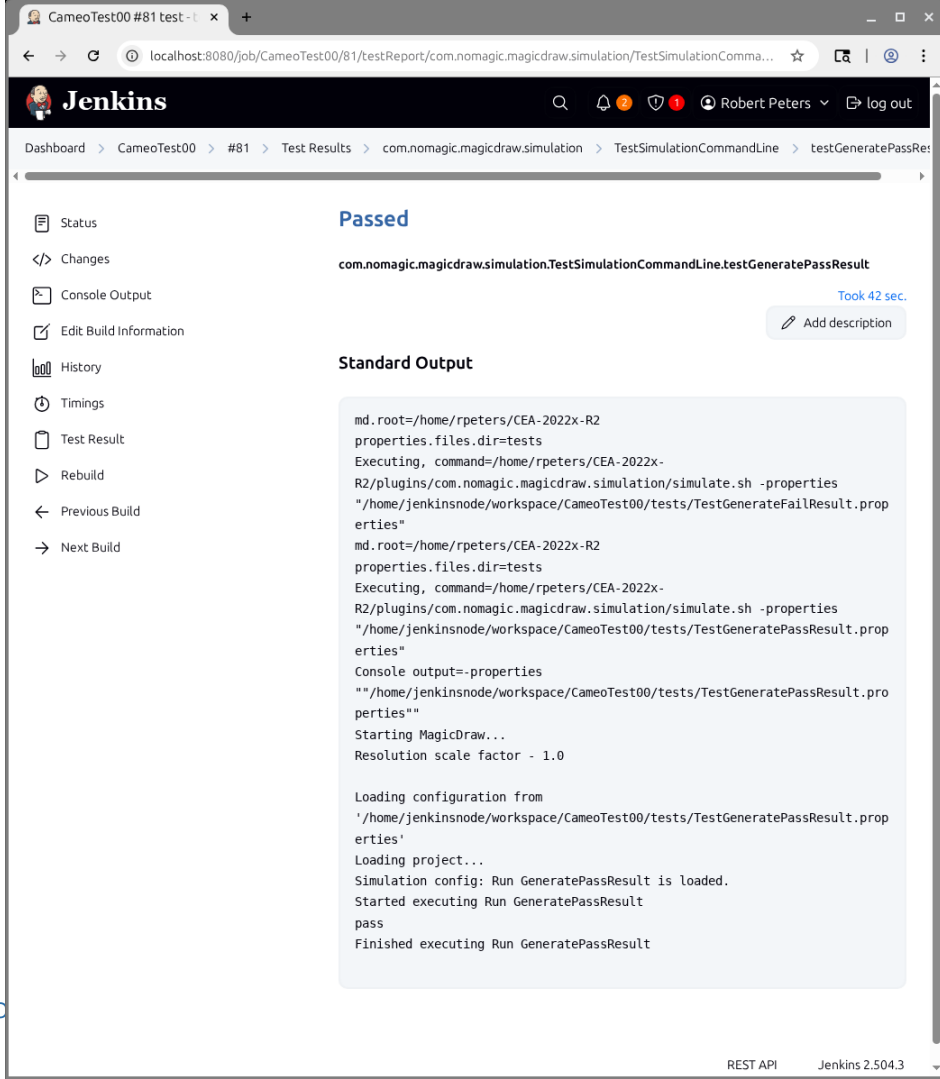
At the bottom right, it says "ose.org | 24". The footer contains "Copyright © 2025 by System Strategy, Inc. Permission granted to INCOSE" and "Jenkins 2.504.3".



# Jenkins Project Results 5/5

## Pass Test

- Standard Output
- Note the “pass” on the line second from the bottom



The screenshot shows the Jenkins web interface for a job named 'CameoTest00 #81 test'. The browser address bar shows 'localhost:8080/job/CameoTest00/81/testReport/com.nomagic.magicdraw.simulation/TestSimulationCommandLine/testGeneratePassResult'. The Jenkins header shows the user 'Robert Peters' and a 'log out' link. The breadcrumb trail is 'Dashboard > CameoTest00 > #81 > Test Results > com.nomagic.magicdraw.simulation > TestSimulationCommandLine > testGeneratePassResult'. The left sidebar contains links for Status, Changes, Console Output, Edit Build Information, History, Timings, Test Result, Rebuild, Previous Build, and Next Build. The main content area shows the test result 'Passed' for 'com.nomagic.magicdraw.simulation.TestSimulationCommandLine.testGeneratePassResult', which took 42 seconds. Below this is the 'Standard Output' section, which contains the following text:

```
md.root=/home/rpeters/CEA-2022x-R2
properties.files.dir=tests
Executing, command=/home/rpeters/CEA-2022x-R2/plugins/com.nomagic.magicdraw.simulation/simulate.sh -properties
"/home/jenkinsnode/workspace/CameoTest00/tests/TestGenerateFailResult.properties"
md.root=/home/rpeters/CEA-2022x-R2
properties.files.dir=tests
Executing, command=/home/rpeters/CEA-2022x-R2/plugins/com.nomagic.magicdraw.simulation/simulate.sh -properties
"/home/jenkinsnode/workspace/CameoTest00/tests/TestGeneratePassResult.properties"
Console output=-properties
""/home/jenkinsnode/workspace/CameoTest00/tests/TestGeneratePassResult.properties""
Starting MagicDraw...
Resolution scale factor - 1.0

Loading configuration from
'/home/jenkinsnode/workspace/CameoTest00/tests/TestGeneratePassResult.properties'
Loading project...
Simulation config: Run GeneratePassResult is loaded.
Started executing Run GeneratePassResult
pass
Finished executing Run GeneratePassResult
```

At the bottom right of the Jenkins interface, it says 'ose.org | 25'.

# Conclusion

# Conclusion - CI/CD to MBSE with Example

Comparing future MBSE goals to our “Hello World” example.

| Topic                          | Future MBSE                                        | What We Showed    |
|--------------------------------|----------------------------------------------------|-------------------|
| Issue Tracking                 | Improved traceability                              | NA                |
| Version Control                | Issue-based branching                              | NA                |
| Test-Driven Development        | Create model with testing in mind                  | Check             |
| Automated Testing              | Expanded testing, blocks merge                     | Check             |
| Integrated Artifact Repository | MBSE artifacts are stored with strong traceability | Partial (Jenkins) |

# Conclusion

There are a lot of opportunities, but still some challenges to implementing this in your environment.

- Implementing CI/CD offers the ability to manage multiple concurrent development lifecycles across a team of developers/modelers.
- Test cases are a powerful tool for enabling quality and conformance in software and are now being realized for MBSE.
- MBSE has broader use cases. The solutions for each program using models may be different and hard to reuse as a community but should be reusable within a single company.
- The current supported mechanisms for executing MagicDraw from the command line are limited, particularly when using a CAC for login.
- SysML v2 textual notation will likely bring CI/CD for MBSE closer to software.

Scalability

Quality & Conformance

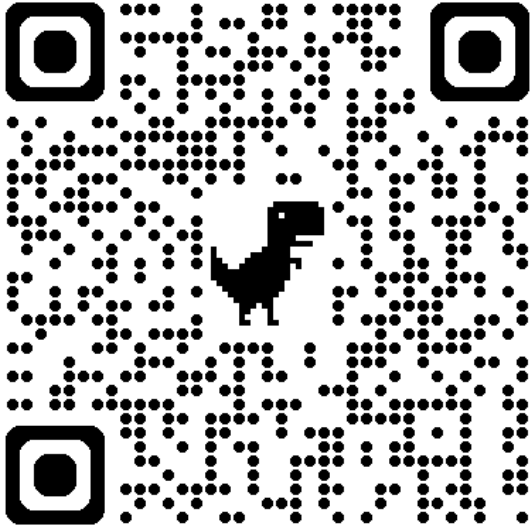
Highly Configurable

Technical Challenges

Interdisciplinary Convergence

# Summary

CI/CD is great! Let's all start using it to automate repeated quality checks.



← QR Code to this presentation and additional materials (Will be available next week.)