29th Annual INCOSE international symposium

Orlando, FL, USA

July 20 - 25, 2019

A Model-Based V&V Test Strategy Based on Emerging System Modeling Techniques

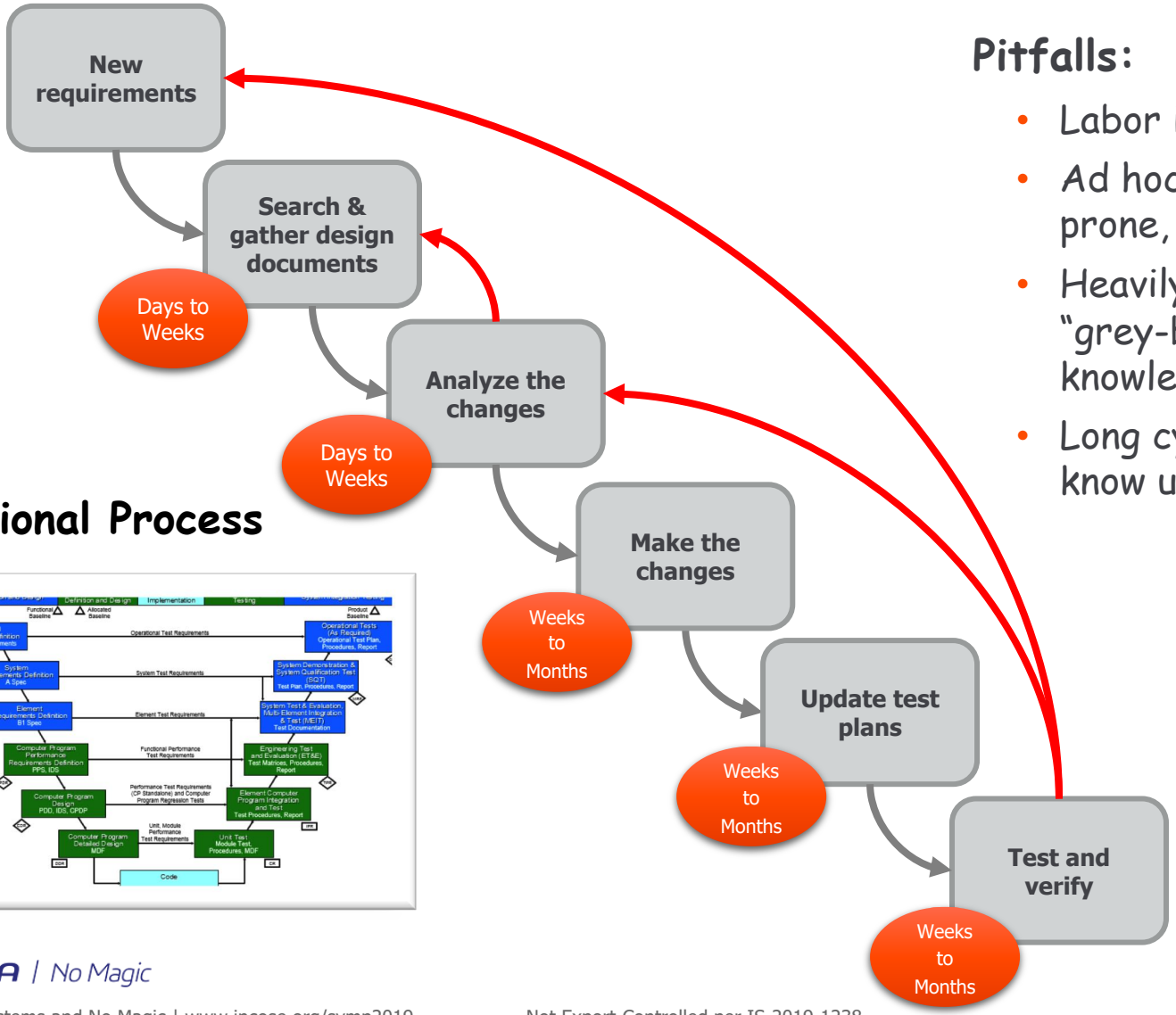
Dr. Gan Wang, BAE Systems Intelligence & Security

Dr. Saulius Pavalkis, Dassault Systems, CATIA | No Magic

Outline

- Problem Statement and Motivations
- The Two-Stage V&V Testing Concept
- Detailed Method Description with a Use Case
- Anticipated Benefits
- Conclusions

Challenges In V&V Testing Today

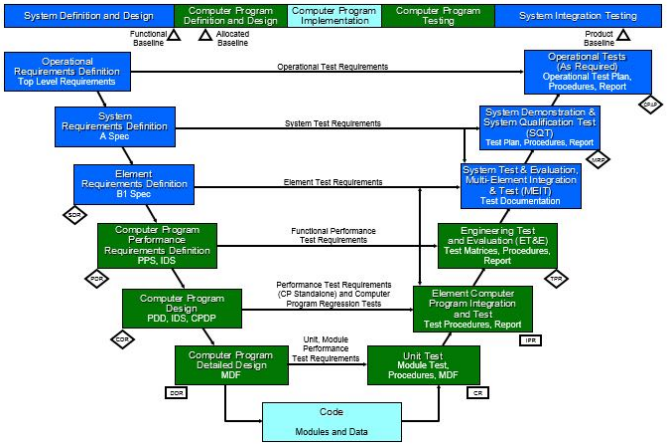


Traditional Process

Pitfalls:

- Labor intensive
- Ad hoc, disjoint, error prone, and many "churns"
- Heavily reliance on "grey-beard" tribal knowledge
- Long cycle time – won't know until the end

Motivation: Change of System Test CONOPS

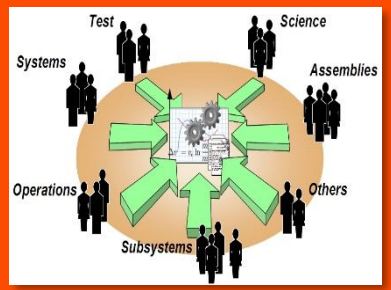


Continuous Integration and Testing into Technical Baseline

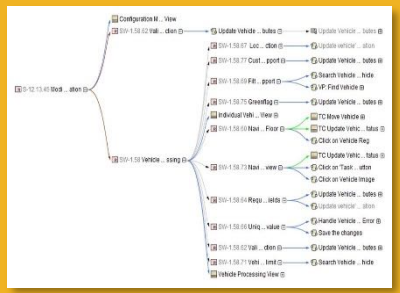
Enabled by Model-based V&V Testing:

- ✓ Test integration in models – go "paperless"
- ✓ Test automation leveraging architectural traceability
- ✓ Early understanding of change impacts and early detection of defects
- ✓ Reducing "churns," and collapsing cycle time

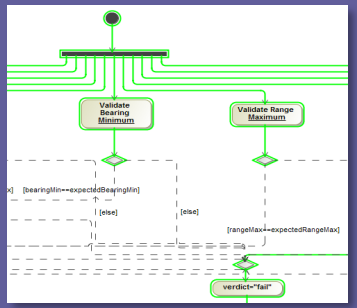
Key Capability Requirements



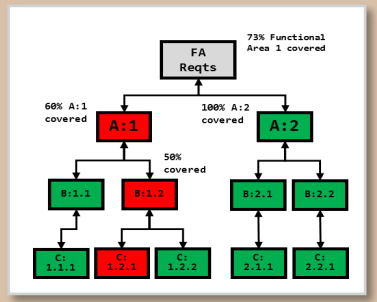
Single Source of Truth



Architecture Traceability



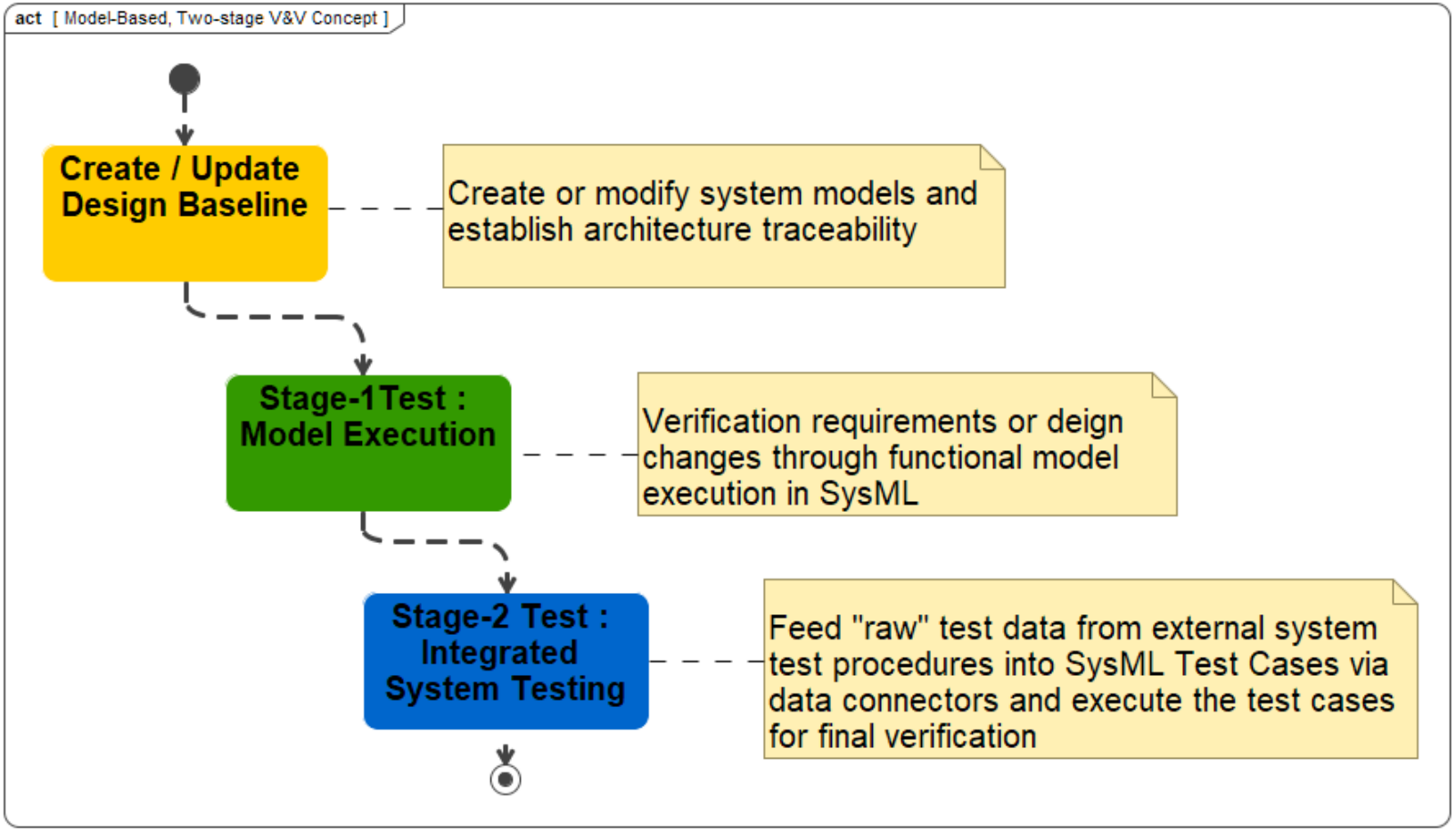
Early Verification



Requirement Coverage

Continuous Integration and Testing into Technical Baseline

Model-Based, Two-Stage V&V Concept



Formalization of Requirement Verification Based on Continuation of Digital Thread

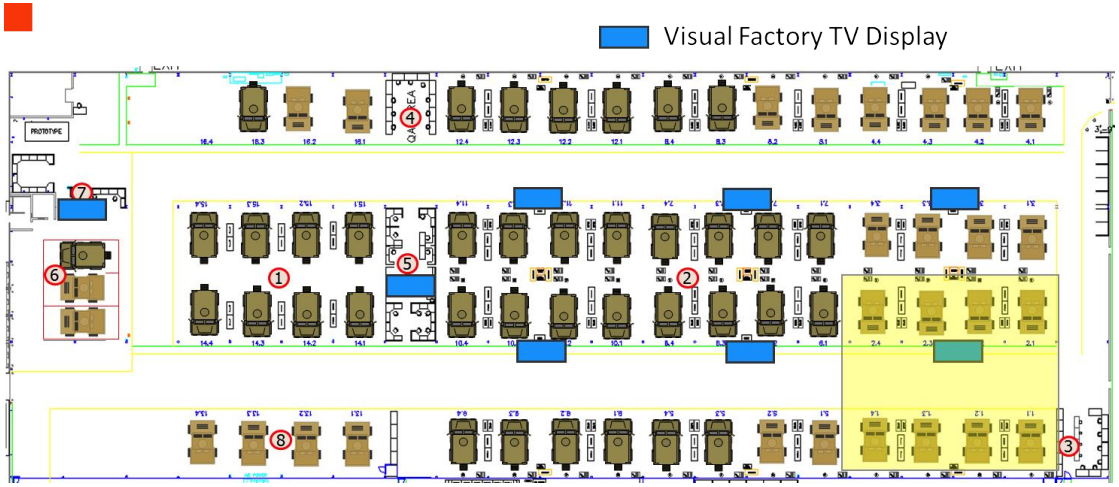
Basic premises:

- *Logical* space: requirements can be satisfied by SysML objects
- *Physical* space: can be connected to the *Logical* space with architectural traceability through continuity of digital thread

Requirement Type	Logical Space: Satisfied By SysML	Physical Space: Extended Digital Thread
Functional	Operation or Behavior (Activity, State Machine or Interaction)	System test plans/procedures
Interface	Port, Connector, Item Flow and/or Constraint (Port)	System test plans/procedures
Performance	Value Property (BDD)	System test plans/procedures
Physical	Structural Element (BDD, IBD)	Parts/BOMs, SW Objects System test plans/procedures and inspections
Design Constraint	Block or Part Property (BDD, IBD)	mCAD, eCAD System test plans/procedures and inspections

A Pilot Use Case: Visual Factory Development

- Visual Factory: a lean, scaled version of *Manufacturing Execution System (MES)*
  - *Office*: desktops
  - *Factory floor*: data kios, large screen displays
  - Digital shop floor
- Real-time, collaborative production execution management
  - Planning, scheduling & configuration
  - Real-time data entry, data collection, & production status
  - Multi-team, shop-floor management

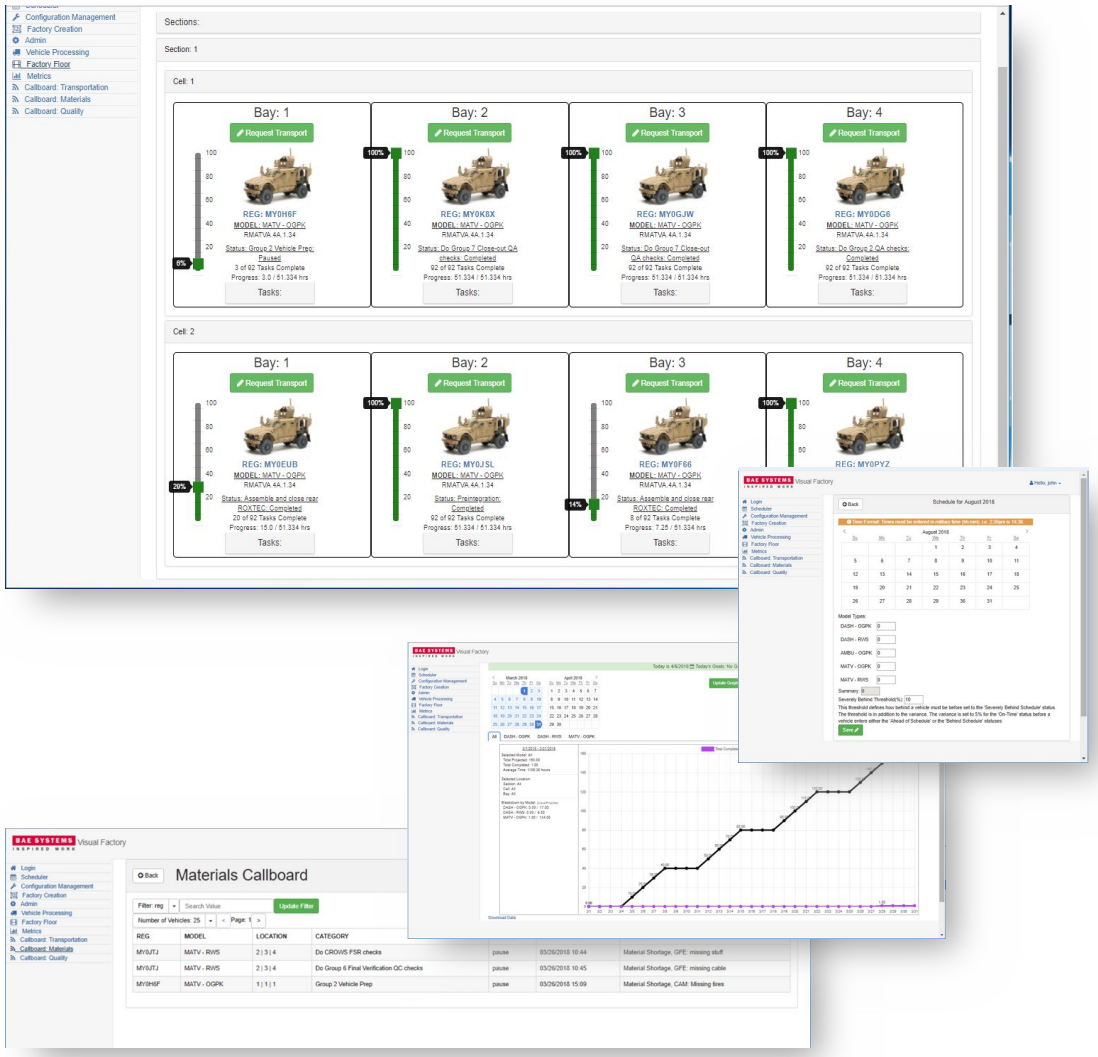


Production Teams:

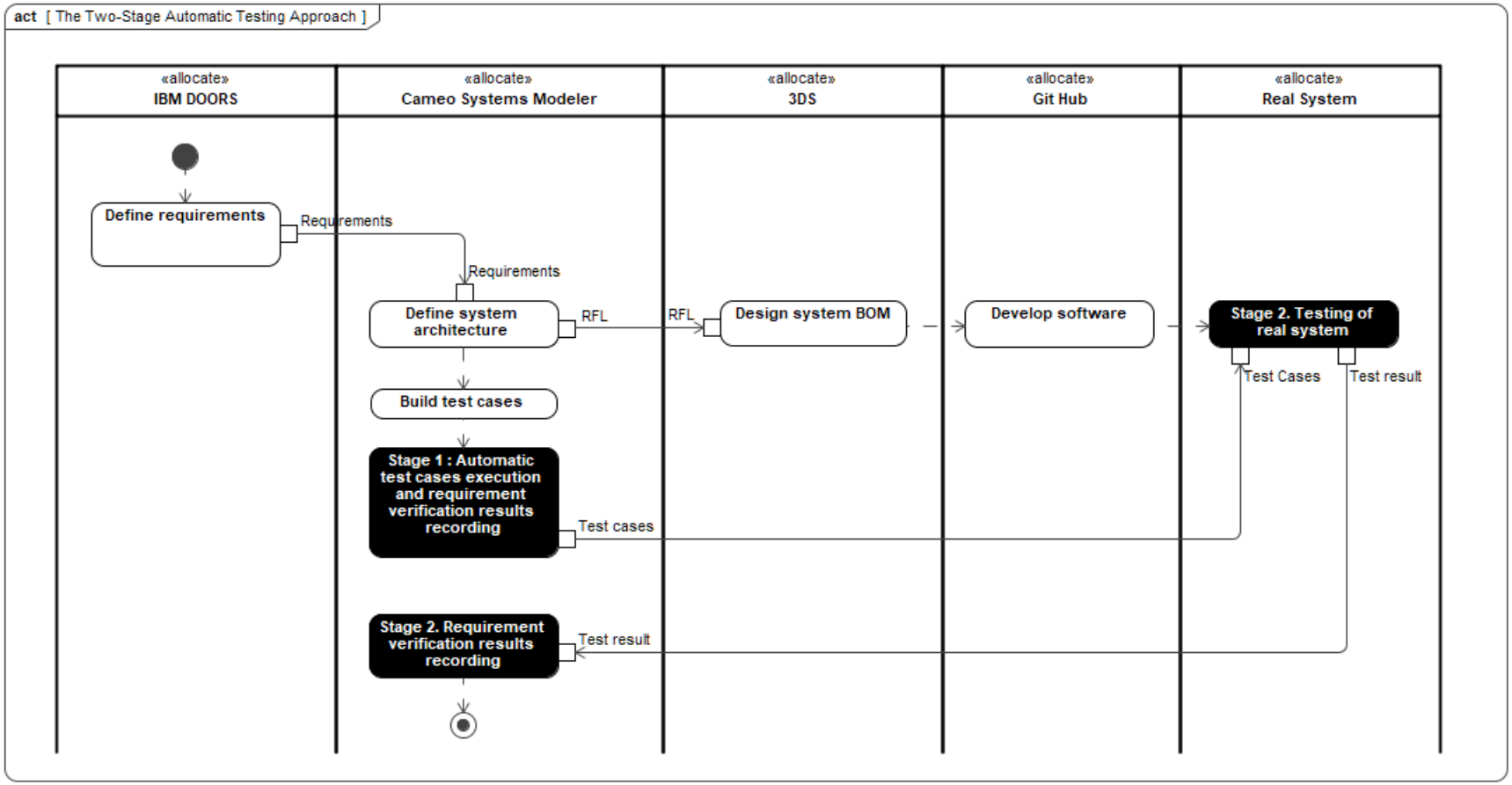
1. Overhead Work
2. Vehicle Integration
3. Quality Control
4. Quality Assurance
5. Bullpen Area
6. Heat Shrink Wrapping
7. Transportation Coordination
8. APO QA Area

Visual Factory System & Application

- Hardware:
  - Server
  - Thin & thick clients
  - TV displays
  - Network
- Application Software:
  - A web application
  - Factory layout
  - Production scheduling
  - Vehicle/system configuration
  - Work initiation, work-in-progress tracking
  - Production floor status
  - Call-boarding and issue resolution (material, transport, quality, etc.)
  - User, team management
  - Data collection and analysis

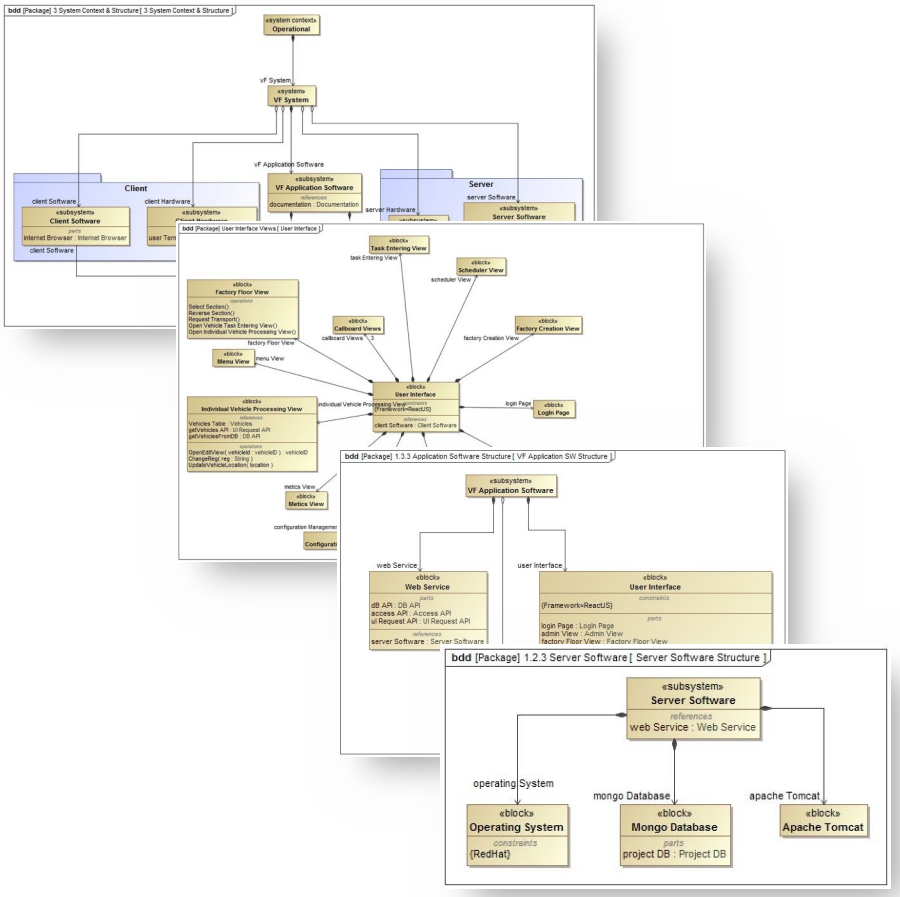


High-level Design Process: Piloting the Two-Stage V&V Testing in an Integrated Modeling Environment

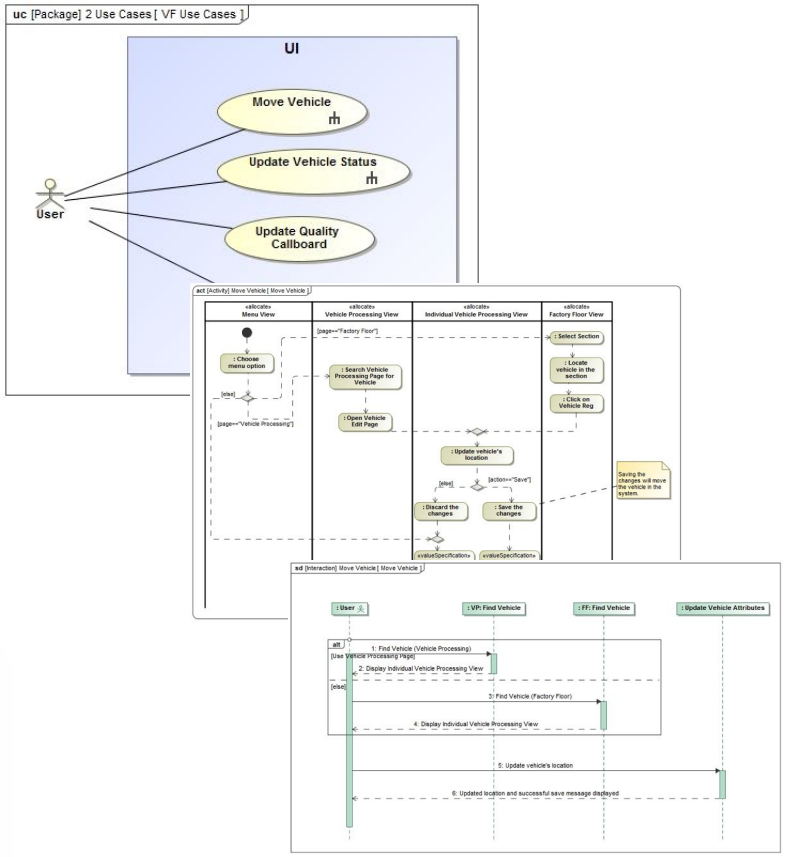


Step 1: Modeling the System and Capture the Design Baseline

System, HW & SW Components:

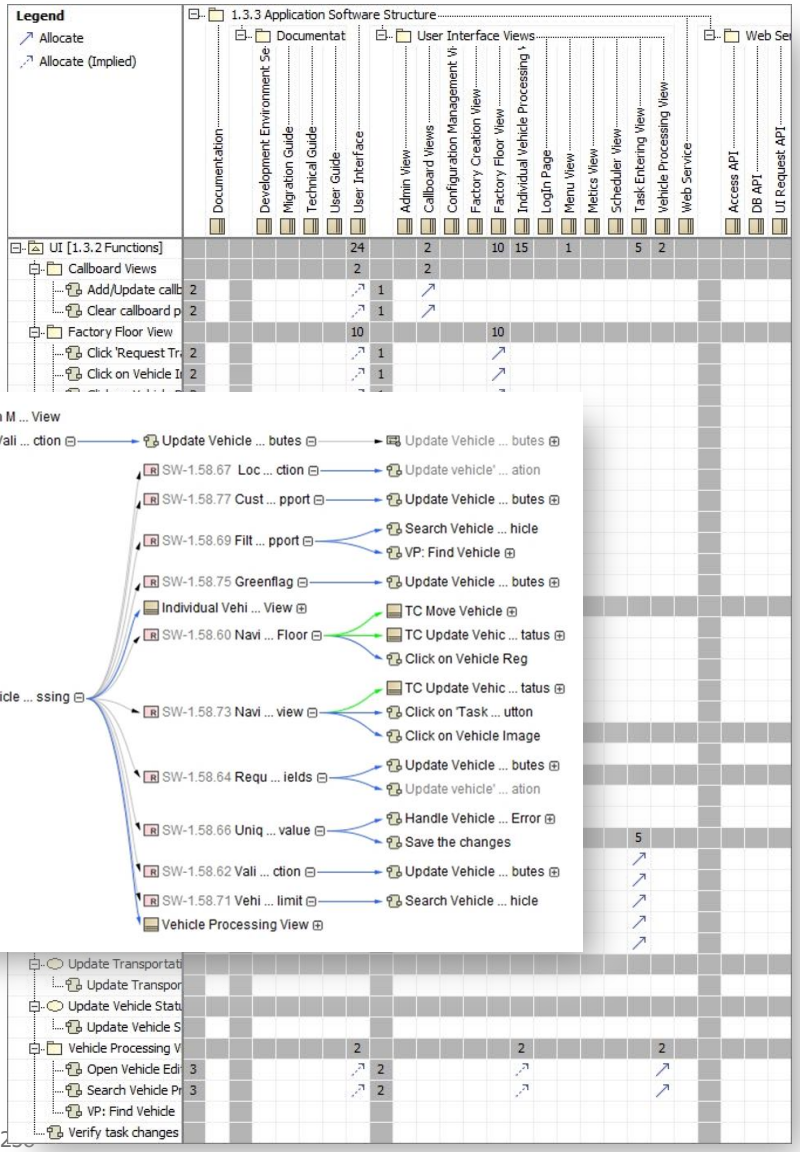


Functions, Workflows:

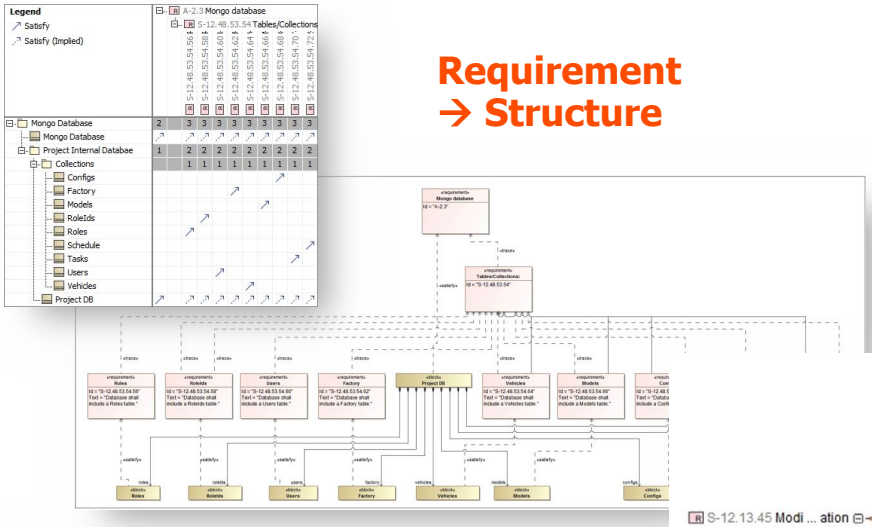


System Architecture Traceability

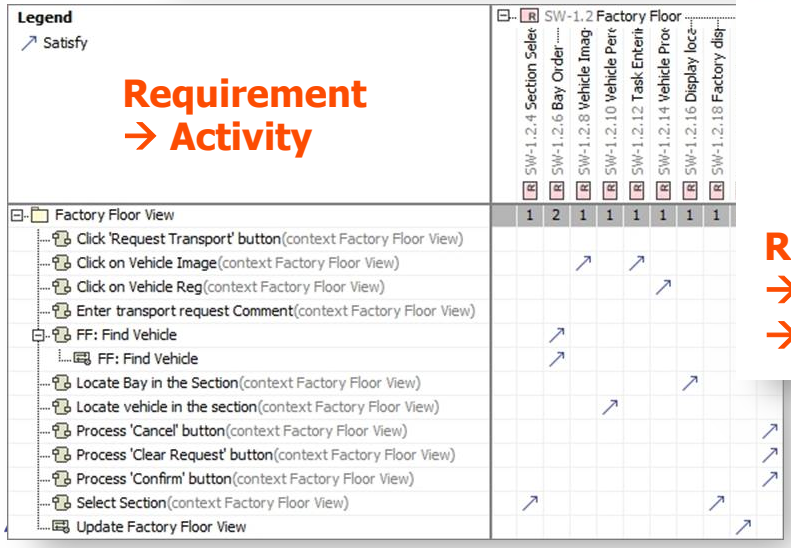
Activity → Structure



Requirement → Structure

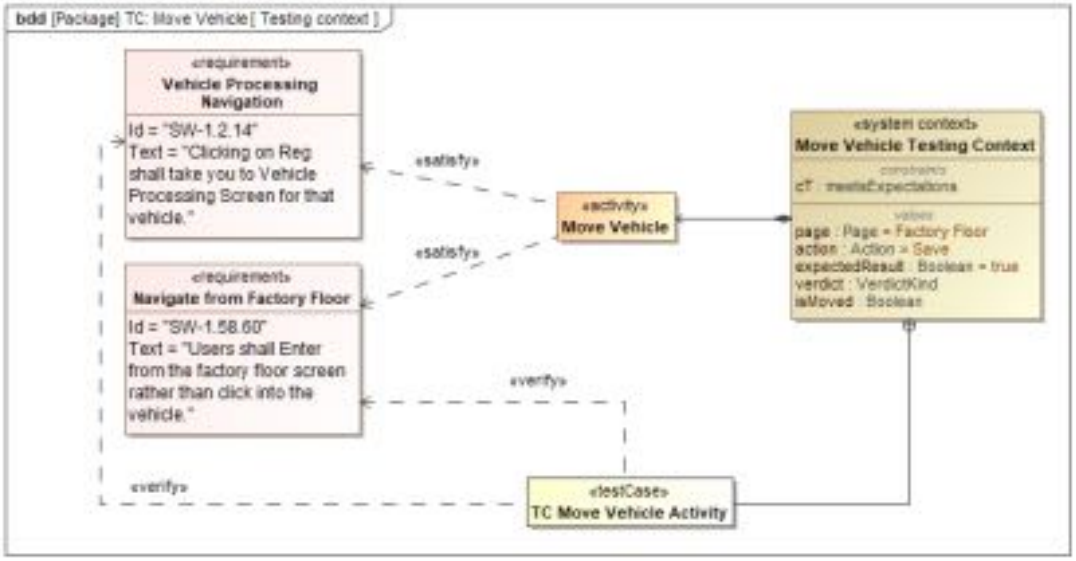


Requirement → Activity

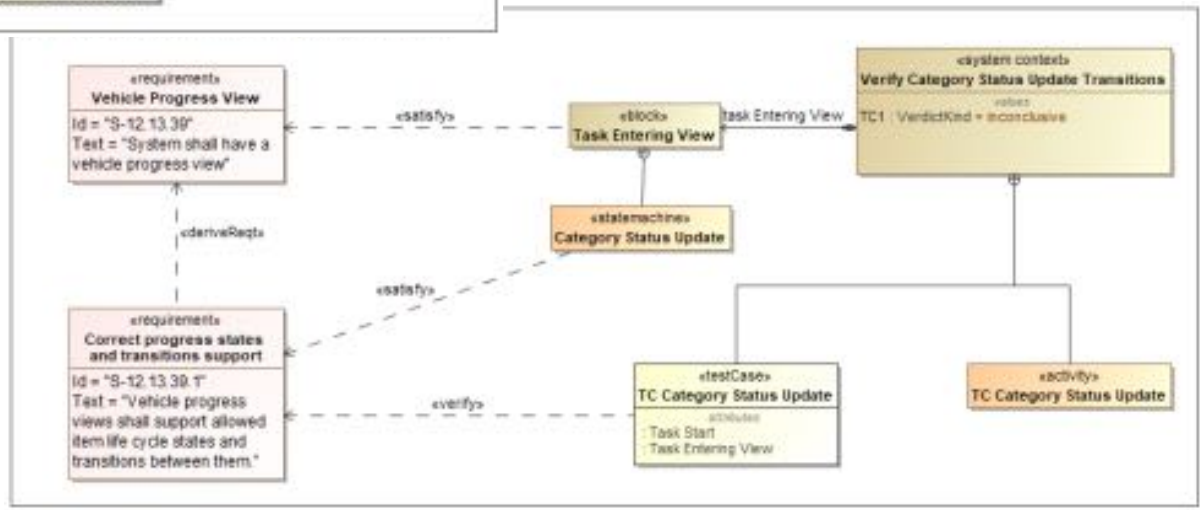


Requirement → Structure → Activity

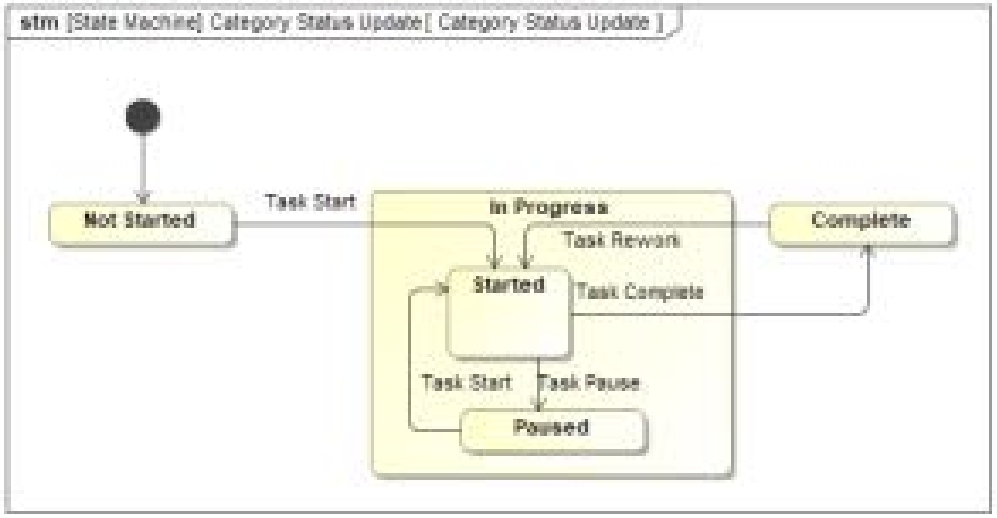
Tests Cases and Test Configurations Capture



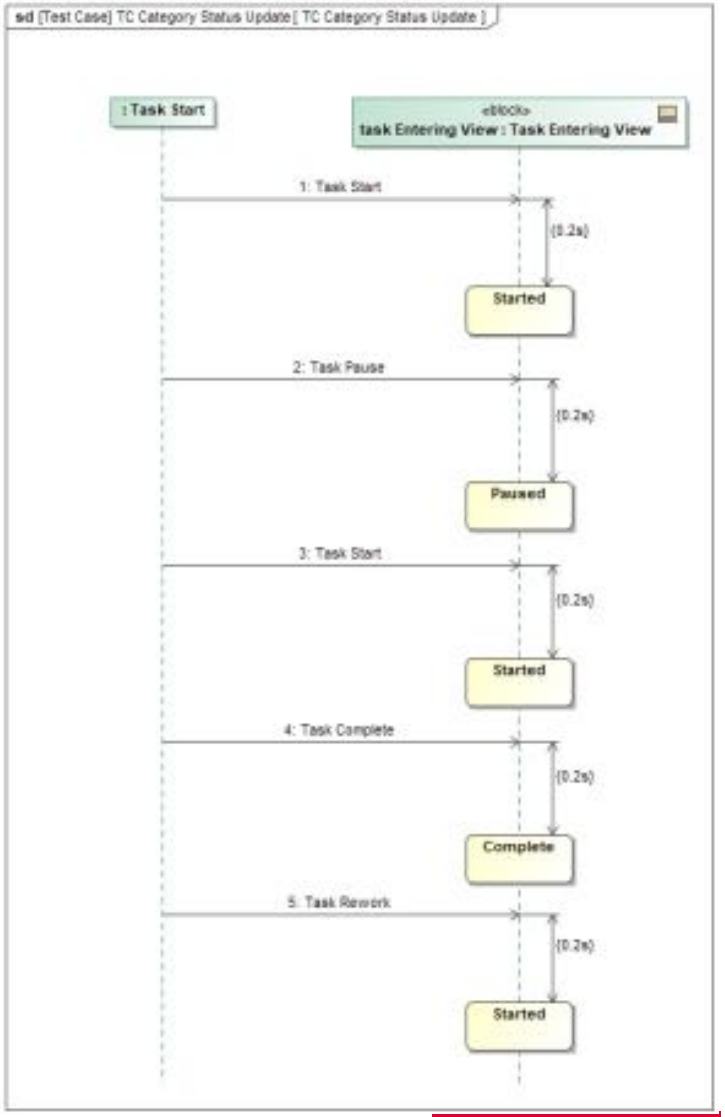
- Model relationships between text based requirement, system model, and test case



Test Case as Sequence Diagram



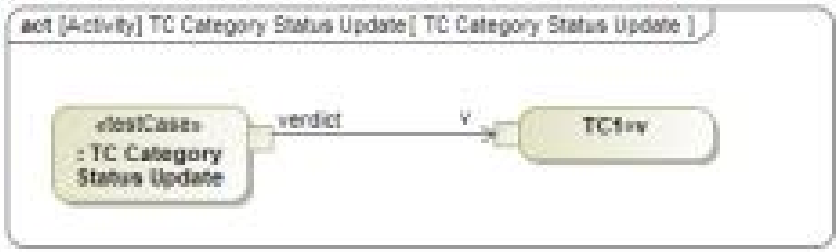
- State machine representing factory item / product states



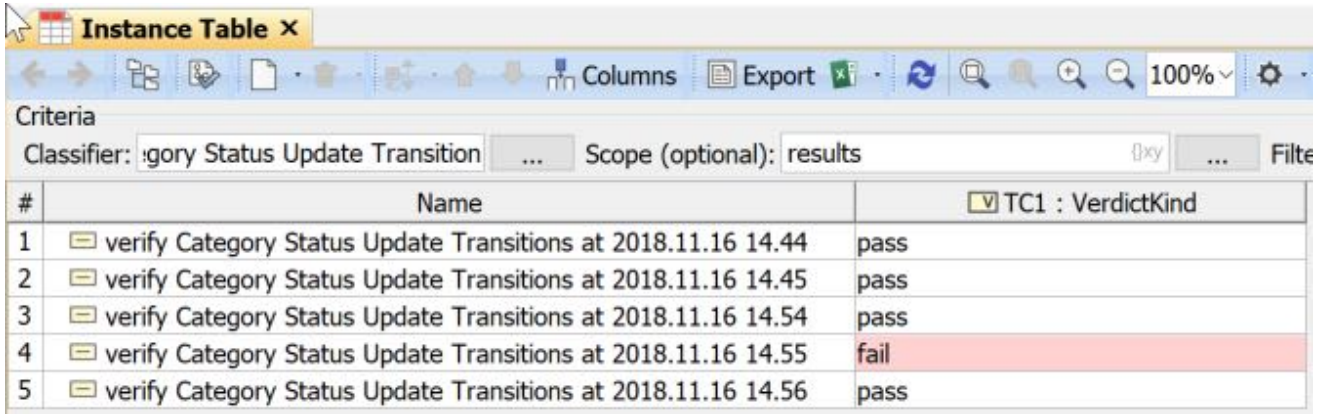
- Sequence diagram specifying Status Update test cases scenario

Stage 1: Simulated Test Case Execution and Recording of Results

- Activity diagram used to orchestrate sequence diagrams test cases execution



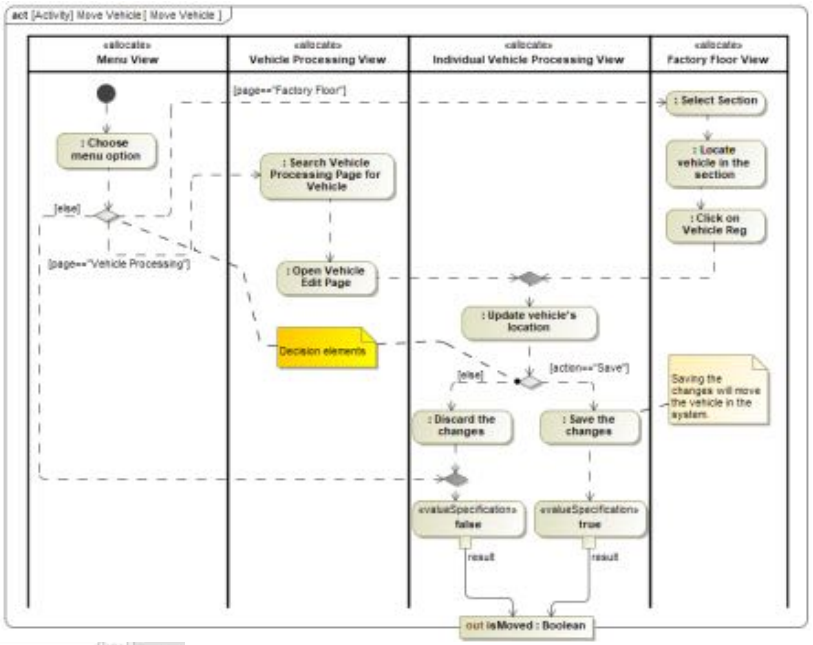
- Single test case execution captured as instance with time stamp



#	Name	TC1 : VerdictKind
1	verify Category Status Update Transitions at 2018.11.16 14.44	pass
2	verify Category Status Update Transitions at 2018.11.16 14.45	pass
3	verify Category Status Update Transitions at 2018.11.16 14.54	pass
4	verify Category Status Update Transitions at 2018.11.16 14.55	fail
5	verify Category Status Update Transitions at 2018.11.16 14.56	pass

Test Case As Activity

- System function with decisions, automatically selected during Execution



TC Move Vehicle Results ×

Columns Export 100%

Criteria

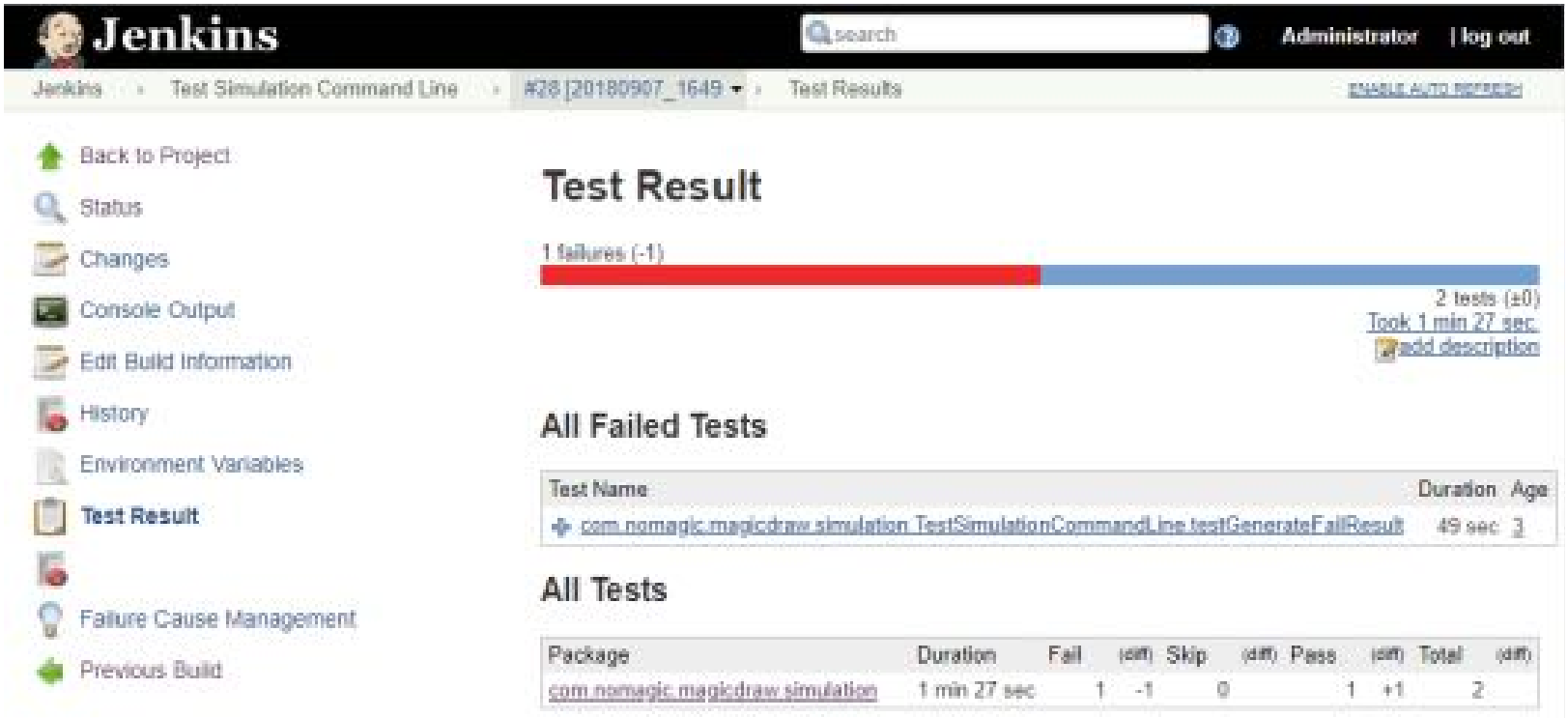
Classifier: TC Move Vehicle ... Scope (optional): results ...

#	page : Page	action : Action	expectedResult : Boolean	isMoved : Boolean	verdict : VerdictKind
1	Factory Floor	Cancel	false	false	pass
2	Log In	Cancel	false	false	pass
3	Factory Creation	Cancel	false	false	pass
4	Factory Floor	Cancel	false	false	pass
5	Configuration Manag...	Cancel	false	false	pass
6	Metrics	Cancel	false	false	pass
7	Factory Floor	Save	false	true	fail
8	Vehicle Processing	Save	true	true	pass
9	Callboard: Transport...	Save	false	false	pass
10	Metrics	Save	false	false	pass
11	Log In	Save	false	false	pass

- System function testing configuration and final testing verdict

Best Practice on Execution Automation

:> simulate -project "D:\\Simulation\\API\\TestVerdictKind.mdzip" -config "Run GeneratePassResult"

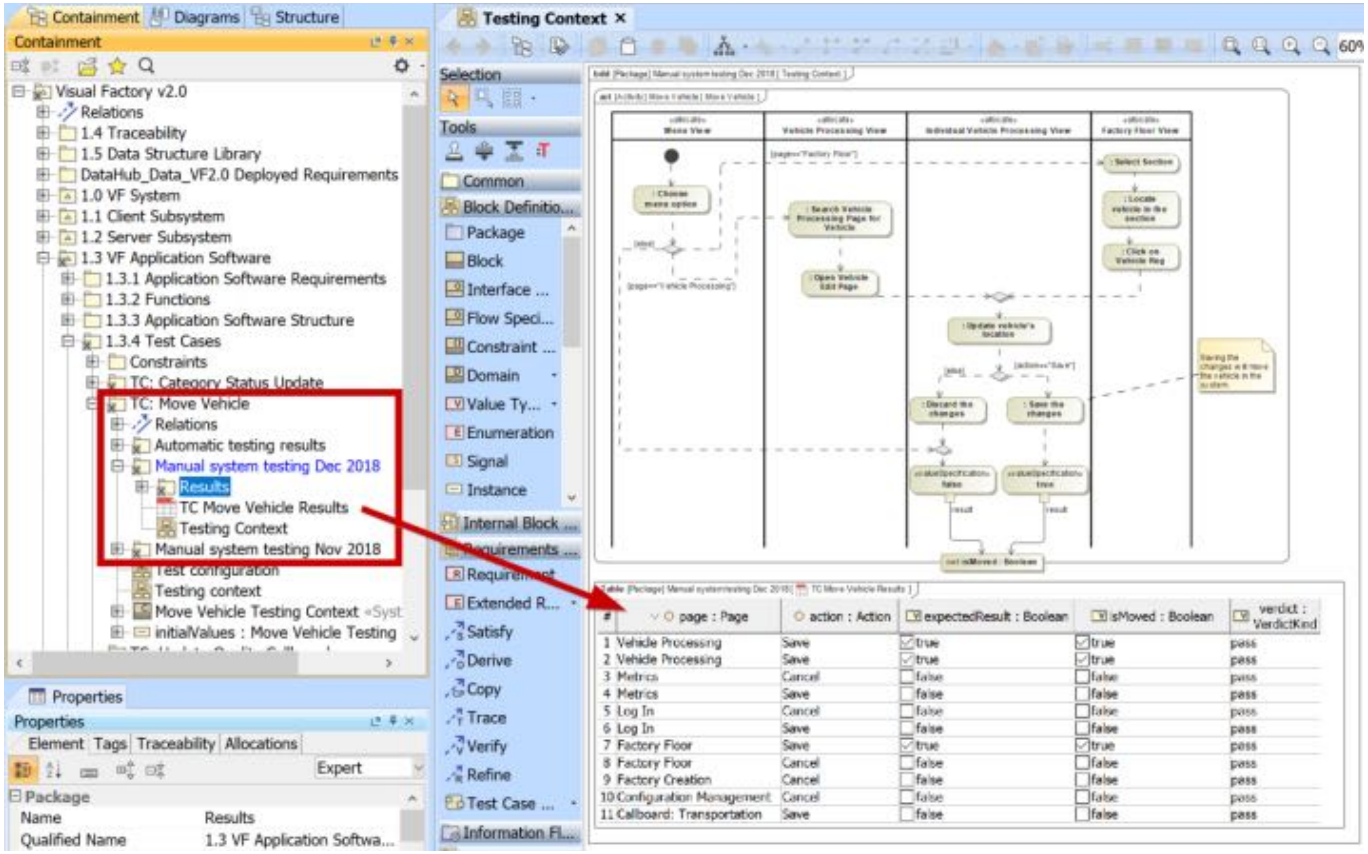


Now, implement the changes …

Stage 2. Final System Testing and Recording of Results

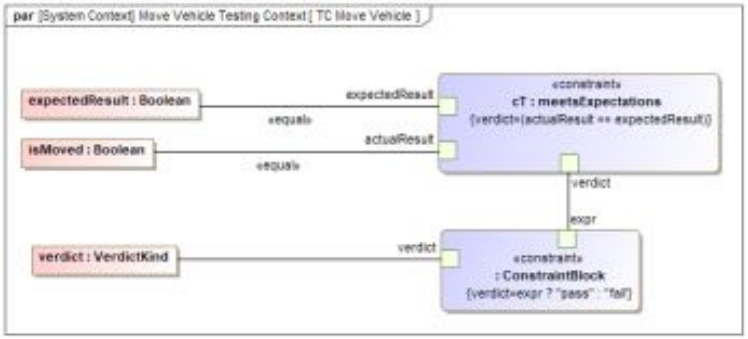
1. Reuse test cases and tests configuration table from 1st stage
2. Run manual or automatic tests on real system
3. Record test results into configuration table

Note: Each test iteration is held in a separate package



Stage 2. Final System Testing and Recording of Results (cont.)

- The raw test result is compared with expected/simulated result
- The final test verdict – whether a requirement is satisfied – is generated automatically, as system function test results are recorded



TC Move Vehicle Results

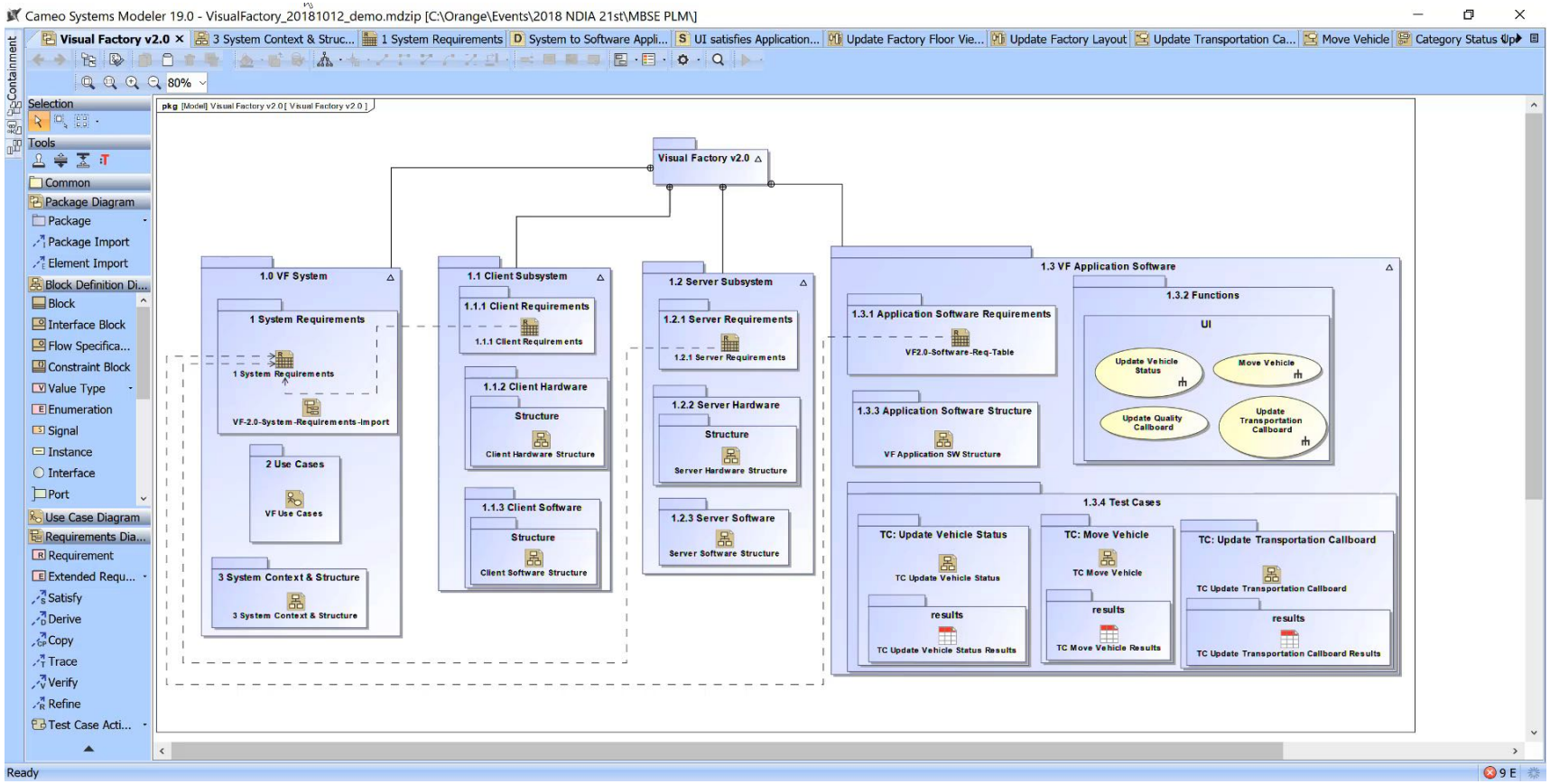
Columns | Export | 100%

Criteria

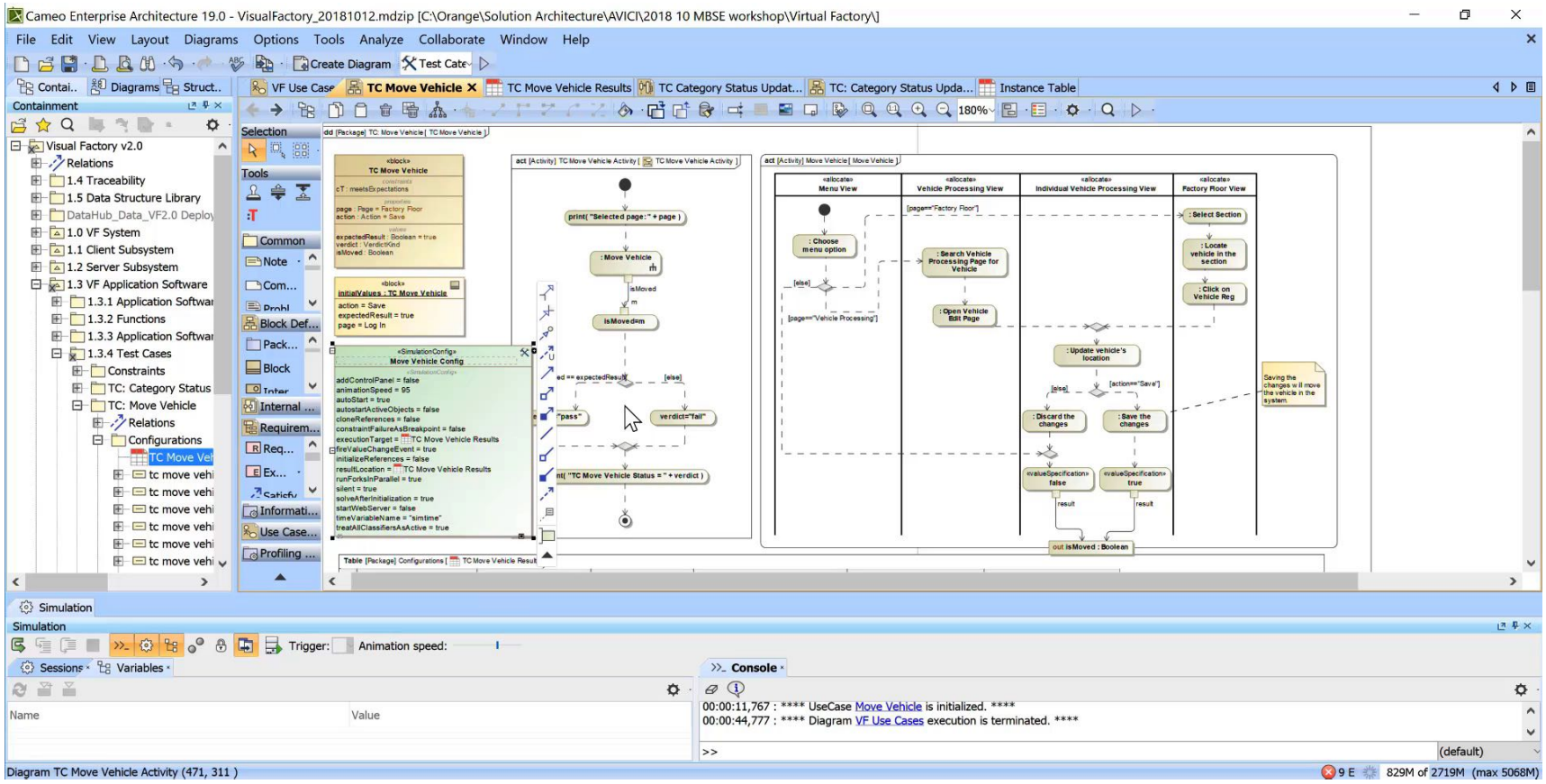
Classifier: TC Move Vehicle ... Scope (optional): results ...

#	page : Page	action : Action	expectedResult : Boolean	isMoved : Boolean	verdict : VerdictKind
1	Factory Floor	Cancel	false	false	pass
2	Log In	Cancel	false	false	pass
3	Factory Creation	Cancel	false	false	pass
4	Factory Floor	Cancel	false	false	pass
5	Configuration Manag...	Cancel	false	false	pass
6	Metrics	Cancel	false	false	pass
7	Factory Floor	Save	false	true	fail
8	Vehicle Processing	Save	true	true	pass
9	Callboard: Transport...	Save	false	false	pass
10	Metrics	Save	false	false	pass
11	Log In	Save	false	false	pass

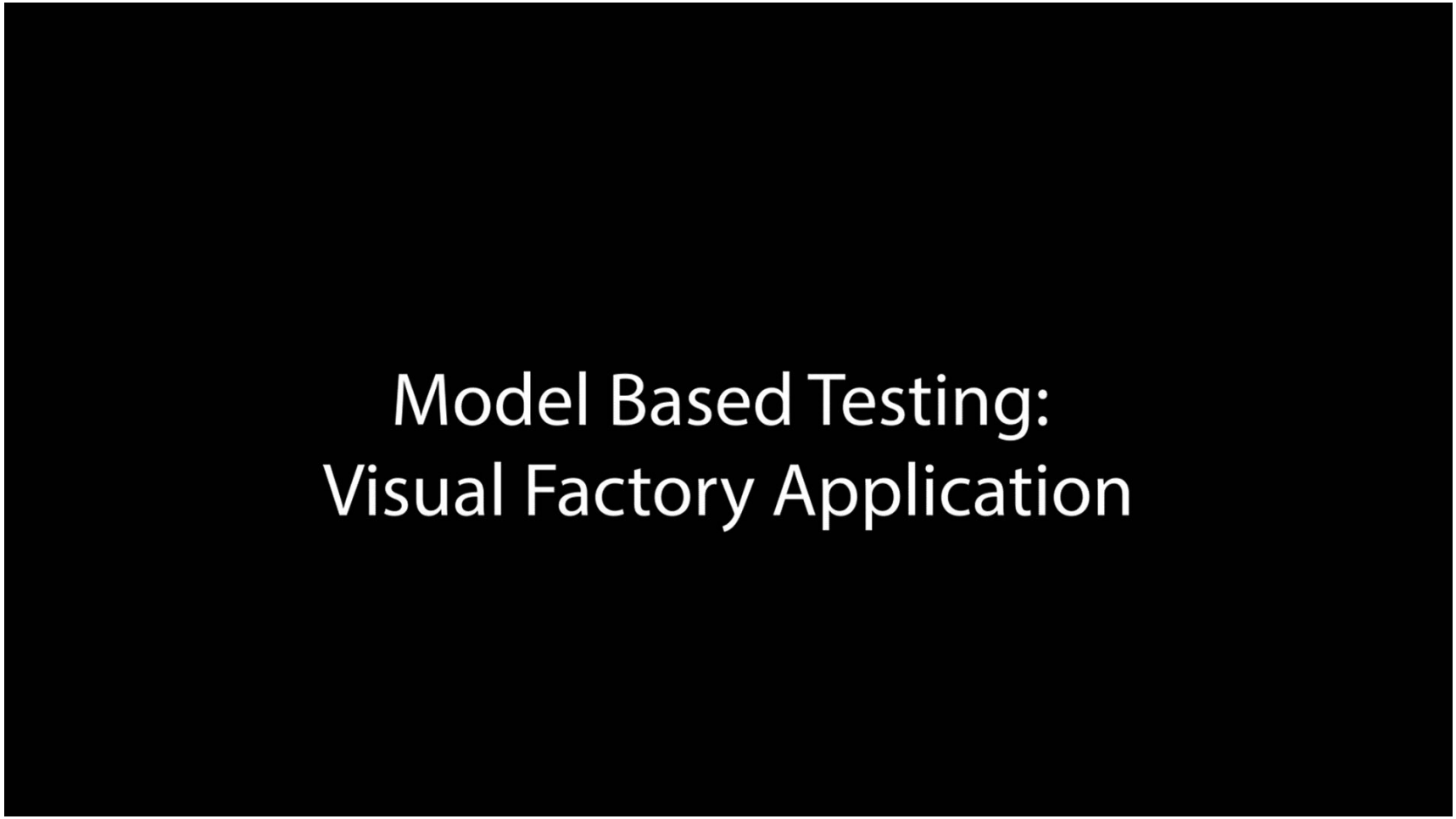
Visual Factory System Overview



Conducting Model-Based Testing Stage 1 (Video)



Conducting Model-Based Testing Stage 2 (Video)



Okay, Got It. But So What?

Implications:

- Early verification, especially critical/hard requirements
- Functional analysis, design trades, "what-if" scenarios – through configuration of multiple test scenarios and leverage of analytical capability of SysML
- Reusable test cases throughout life cycle
- Integration of external / standalone test procedures into the model space – automated, regression test suite

Anticipated Benefits:

- Improved system test rigor and system understanding
  - Change impacts
  - Detection of potential defects
- Reduced reliance on "grey-beards" and tribal knowledge
- Sped-up test cycle and minimized "churns"
- Reduced transactional cost and chaos from mounts of design documents

Conclusions

- Two-stage, model-based system V&V test strategy
  - Stage 1: verification through model execution
  - Stage-2: verification through model integration of external test procedures
- Extension of system modeling to system testing
  - Leveraging SysML capabilities
  - Enabling test automation
- Future work:
  - Automated data connectors for external test procedures
  - Automated regression test suites

Thank You

Dr. Gan Wang, BAE Systems
gan.wang@baesystems.com

Dr. Saulius Pavalkis, CATIA | No Magic
saulius.pavalkis@nomagic.com