28th Annual INCOSE international symposium

Washington, DC, USA
July 7 - 12, 2018

Apoorv Maheshwari, Purdue University

Minimum SysML Representations to Enable Rapid Evaluation using Agent-Based Simulation

July 9, 2018

Outline

- Introduction
- Contribution
- Methodology
- Case Study on National Airspace System
- Results & Discussion
- Conclusion

Introduction

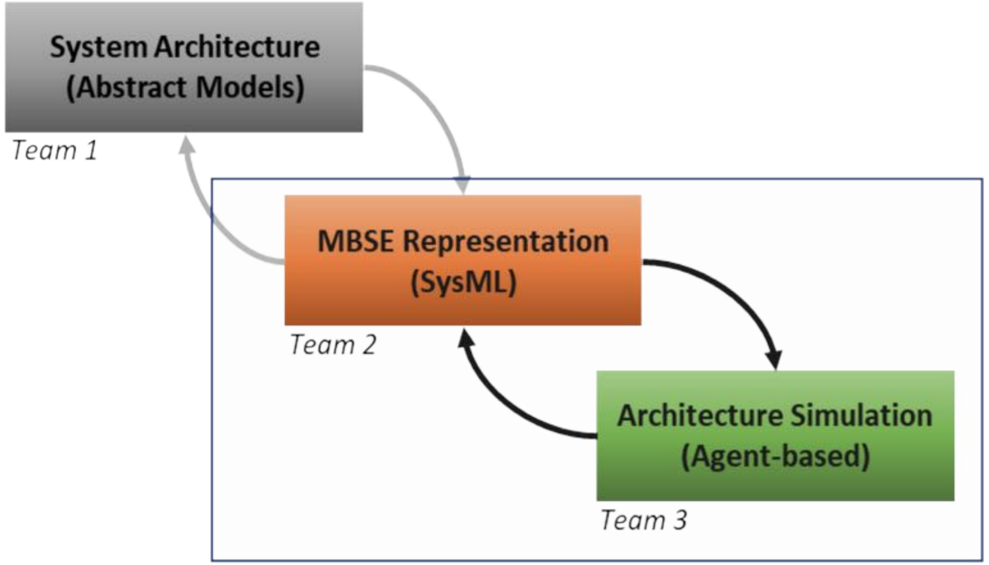
- Design process of complex systems
- Requires diverse engineering skill set
  - System Architects
  - Systems Engineers
  - Simulation Programmers
- MBSE fits well as an intermediary
- Traceability through MBSE

Contribution

“What is the minimum set of information required to perform *rapid* agent-based simulation of an *evolving* system architecture?”

- Demonstrate the use of a SysML-driven architecting and analysis of a complex system to evaluate system performance throughout the design process
- Tandem development of MBSE artifacts with an agent-based sim model
- Framework demonstrated through a NASA-supported study for rapidly evaluating architectural changes in the National Airspace System (NAS)

Methodology (1/2)



Architecture Evaluation Process Model
Connecting system architecting, systems engineering, and simulation programming

Goals

1. Perform rapid evaluation of evolving architecture by specifying minimum SysML representation
2. Demonstrate the end-to-end process application

Methodology (2/2)

Team	Roles & Expertise	Goals & Deliverables
Team 1 – Purdue University Industrial and Systems Engineering	System Architecting and domain knowledge of NAS/ATS	Provide state models of the NAS architectures
Team 2 – AT Corp Inc.	MBSE artifact development, SE Expertise	Convert NAS state model into SysML to facilitate SE and simulation
Team 3 – Purdue University Aeronautics and Astronautics Engineering	Agent-based Simulation Development, System-of-Systems M&S expertise	Simulation of NAS architectures

- The complete work was divided among three different teams – each with unique expertise and part of different organizations
- Teams interacted at regular, but large intervals

Mapping between SysML and ABS

Agent-based simulation	SysML Diagrams	Remarks
Agent Attributes	Block Definition Diagram (BDD)	BDD lists all the agents that are required in the simulation and provide a black-box representation of the agent
Agent Methods	Internal Block Diagram (IBD)	IBD provides the white-box view of the agent, describing functional and physical decomposition of the agent
	Parametric Diagram (PD)	PD describes the working of the methods required for agent functionality
Interaction Rules	Activity Diagram (AD)	AD represent the state-transition logic(s) for each agent. These rules can be defined for both agent-agent and agent-environment interaction.
Recorded Variables	Requirements Diagram (RD)	RD describes what performance metrics should be calculated from the simulation to perform the analysis.

Why National Airspace System (NAS)?

- System-of-systems
- Unmanned Aerial Systems in NAS integration
  - 6500 commercial, 210,000 general aviation aircraft currently
  - 450,000 registered UAS users in May 2016; expected to grow to 2.7 million by 2024
- FAA asking for integrated approach to understand system-wide impacts of architectural changes
- Traditional document-based approach lack traceability and evaluation capabilities

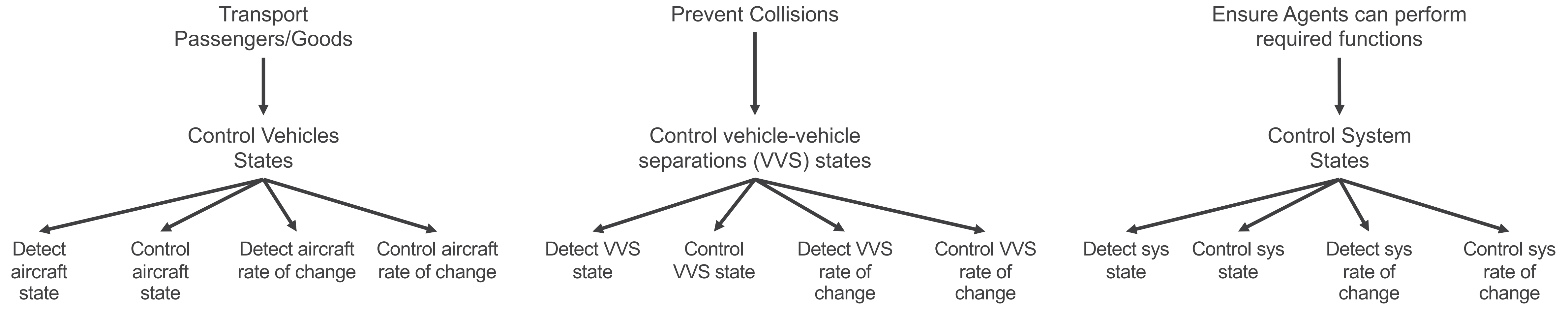
Scope of current study

- NASA-sponsored study to develop *ab initio* NAS architectures

The method developed is model-based, starting with a high-level abstraction and progresses through an iterative approach in which an architecture is evolved and refined through progressive evaluation using an integrated simulation modeling and analysis capability.

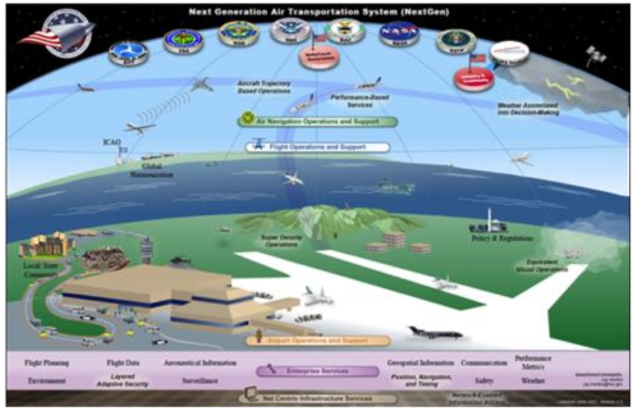
  - Develop system architecture via system model
  - Evaluate using standard evaluation tool
  - Revise the model based upon the simulation results

Generating the SysML diagrams


Transport Passengers/Goods
Control Vehicles States
Detect aircraft state
Control aircraft state
Detect aircraft rate of change
Control aircraft rate of change
Prevent Collisions
Control vehicle-vehicle separations (VVS) states
Detect VVS state
Control VVS state
Detect VVS rate of change
Control VVS rate of change
Ensure Agents can perform required functions
Control System States
Detect sys state
Control sys state
Detect sys rate of change
Control sys rate of change

- Team 1 provided physical decomposition of the NAS (agents)
- Team 2 created SysML diagrams and added information about the performance metrics

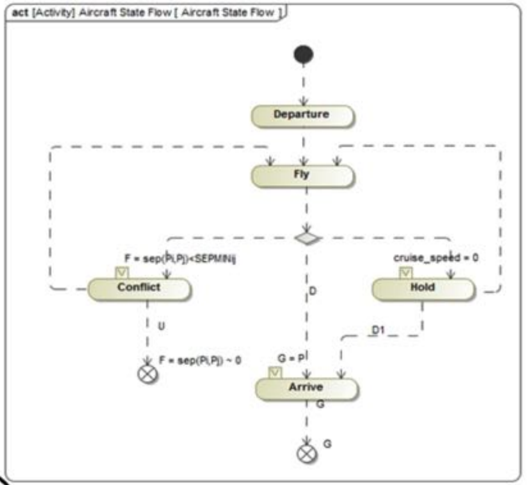
Traceability in Architecture Evaluation


Next Generation Air Transportation System (NextGen)
Aircraft Trajectory Based Operations
Performance-Based Services
Weather Assimilated into Decision-Making
Air Navigation Operations and Support
Flight Operations and Support
ICAO
Global Harmonization
Super Density Operations
Policy & Regulations
Local Sim Community
Equivalent Visual Operations
Airport Operations and Support
Flight Planning
Flight Data
Aeronautical Information
Environment
Layered Adaptive Security
Surveillance
Enterprise Services
Geospatial Information
Communication
Performance Metrics
Position, Navigation, and Timing
Safety
Weather
Net Centric Infrastructure Services

https://www.nasa.gov/topics/aeronautics/features/8q_nextgen.html

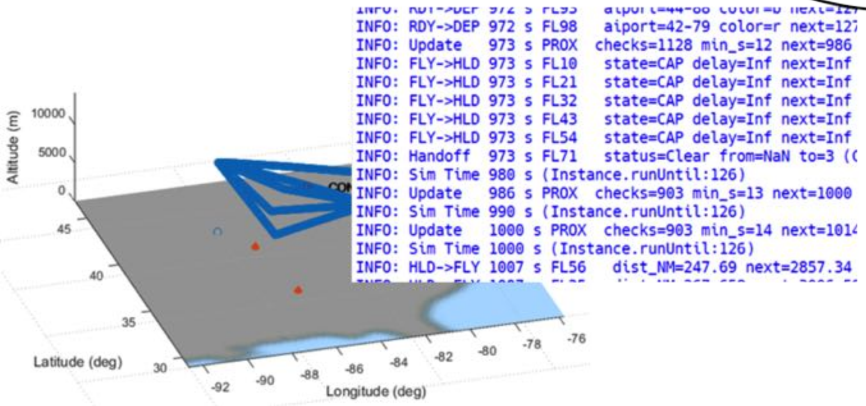
Operational View

SysML

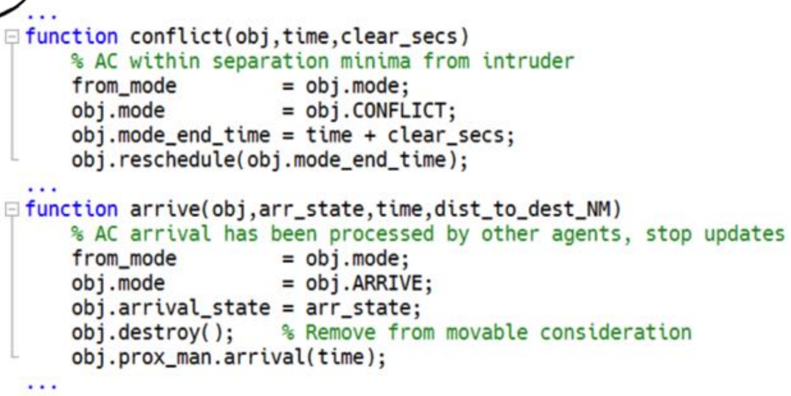


ABS Results

ABS Code



```
INFO: RDY->DEP 972 s FL98   aiport=42-79 color=r next=127
INFO: Update   973 s PROX  checks=1128 min_s=12 next=986
INFO: FLY->HLD 973 s FL10   state=CAP delay=Inf next=Inf
INFO: FLY->HLD 973 s FL21   state=CAP delay=Inf next=Inf
INFO: FLY->HLD 973 s FL32   state=CAP delay=Inf next=Inf
INFO: FLY->HLD 973 s FL43   state=CAP delay=Inf next=Inf
INFO: FLY->HLD 973 s FL54   state=CAP delay=Inf next=Inf
INFO: Handoff  973 s FL71   status=Clear from=NaN to=3 ((
INFO: Sim Time 980 s (Instance.runUntil:126)
INFO: Update   986 s PROX  checks=903 min_s=13 next=1000
INFO: Sim Time 990 s (Instance.runUntil:126)
INFO: Update   1000 s PROX  checks=903 min_s=14 next=1014
INFO: Sim Time 1000 s (Instance.runUntil:126)
INFO: HLD->FLY 1007 s FL56   dist_NM=247.69 next=2857.34
```



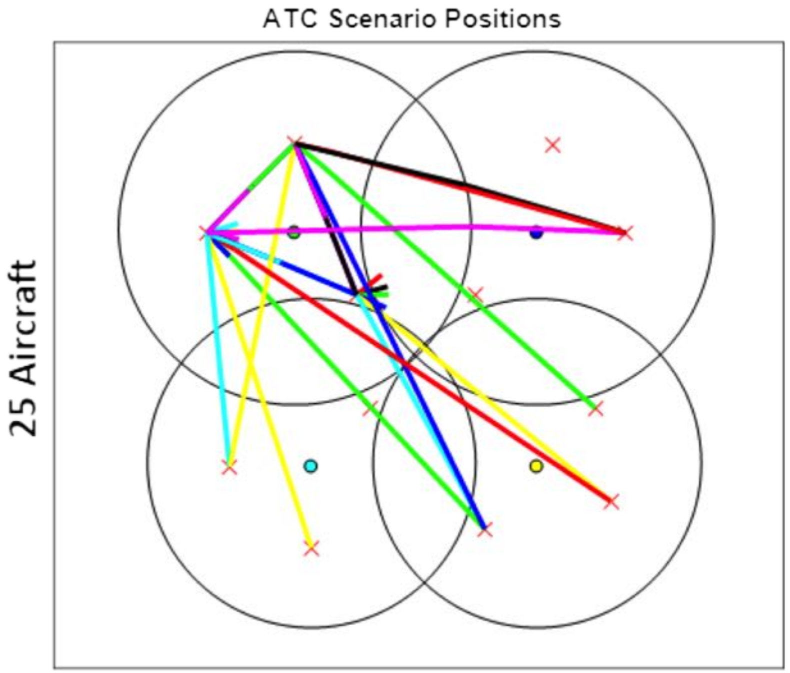
```
...
function conflict(obj,time,clear_secs)
    % AC within separation minima from intruder
    from_mode          = obj.mode;
    obj.mode           = obj.CONFLICT;
    obj.mode_end_time = time + clear_secs;
    obj.reschedule(obj.mode_end_time);
...
function arrive(obj,arr_state,time,dist_to_dest_NM)
    % AC arrival has been processed by other agents, stop updates
    from_mode          = obj.mode;
    obj.mode           = obj.ARRIVE;
    obj.arrival_state = arr_state;
    obj.destroy();    % Remove from movable consideration
    obj.prox_man.arrival(time);
...
```

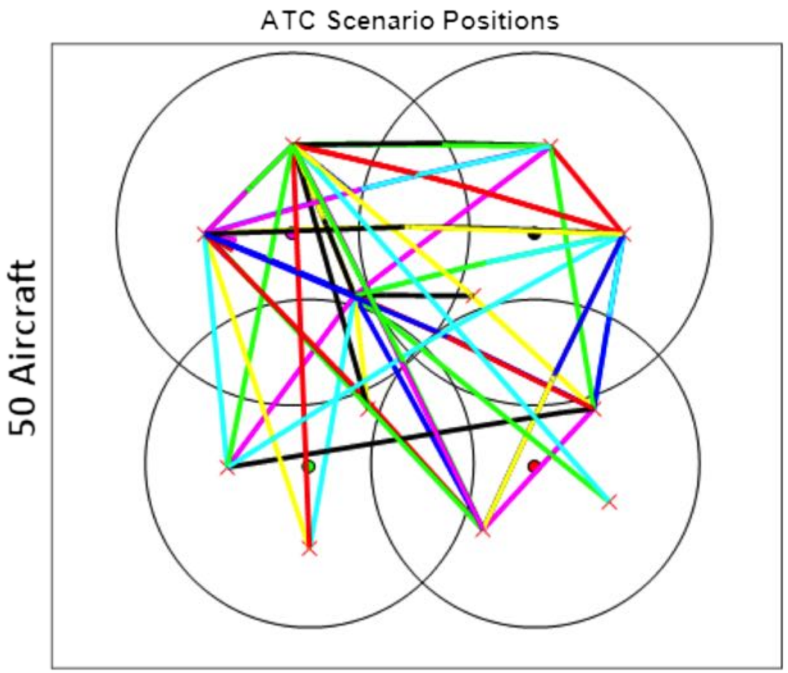
Architecture Evolution

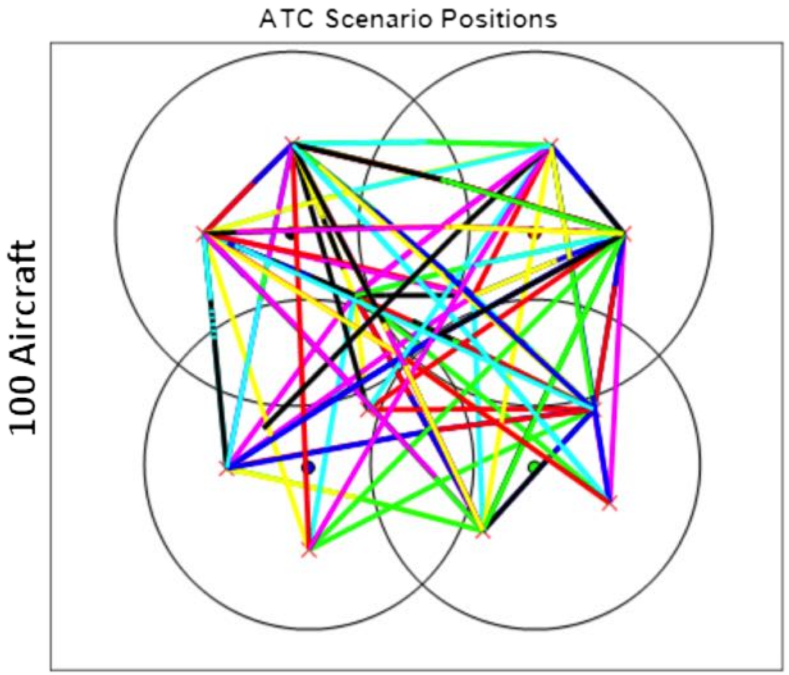
- Initial ‘as-is’ architecture evolved to ‘to-be’ architecture
- **“As-is”:** Human air traffic controllers directing all aircraft within a geographic area
- **“To-be”:** Human air traffic controllers are assisted by advanced automation to monitor and direct *streams** of aircraft instead of individual aircraft
- Observed significant reduction in time and effort to create the “to-be” architecture simulation

*A stream is defined by a controller supervising all aircraft sharing the same destination, regardless of origin or current position and much of functionality for resolving conflicts with a stream is now allocated to automated systems.

Scenario Description







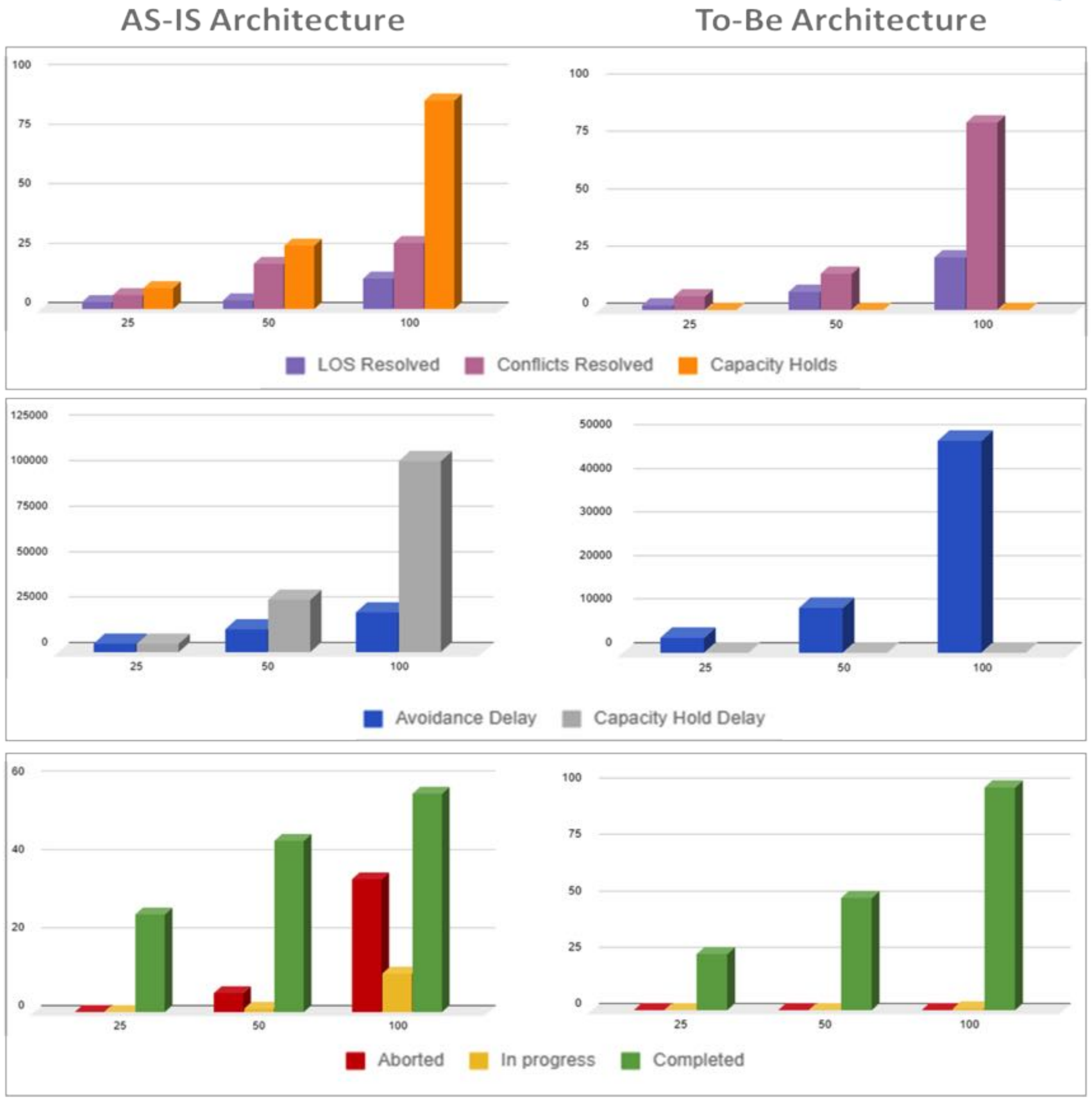
Aircraft Distribution b/w Airports

- 45% of the aircraft are flying from hub to hub
- 25% of the aircraft are flying from regional to hub airports
- 20% of the aircraft are flying from regional to regional across ATC
- 10% of the aircraft are flying from regional to regional within ATC

X Airport Location
O ATC Boundary
Flight Path b/w airports

Simulation Results

- *Capacity Holds*, *Capacity Hold Delay,* and *Resolved Conflicts* significantly improve due to the automation introduced in the “to-be” architecture
- Efficient operations as almost all the flights were successfully completed in “to-be” whereas in “as-is”, controllers were unable to handle the high load
- Difference in architectural performance can be clearly attributed to the architectural changes introduced in “to-be”



Conclusion

- Focused on identifying the minimum set of SysML representations required to perform a rapid evaluation of an evolving architecture
- Described a three-step simulation development process starting from abstract architecture and ending with the agent-based simulation while using SysML representations as a bridge
- Demonstrated the process through a NASA-sponsored case study and showed evidence that the proposed process significantly improves the simulation development time

Thank You!

Questions?

Apoorv Maheshwari
apoorv@purdue.edu

This publication was developed under work supported by NASA contract NNA16BD85C. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Purdue University, the National Aeronautics and Space Administration, or the US Federal Aviation Administration, nor do these organizations endorse any products or commercial services mentioned in this publication.

28th Annual INCOSE
international symposium

Washington, DC, USA
July 7 - 12, 2018

www.incose.org/symp2018