2019-12

Mission Scenario Generation and Characterization to Support Acquisition Decisions for Long Range Precision Fires-Maritime (LRPF-M)

Giammarco, Kristin M.

Monterey, California: Naval Postgraduate School

Mission Scenario Generation and Characterization to Support Acquisition Decisions

NPS-19-N248-A

KRISTIN GIAMMARCO, PHD

JANUARY 2020

SPONSORED BY NRP

Outline

- Project Objectives & Deliverables
- Research Methodology
- Models
- Findings & Conclusions
- Limitations
- Recommendations for Further Research
- Appendix: Overview of Monterey Phoenix (MP)

Project Objectives & Deliverables

Objectives

- Formally capture communication and decision flows among operations and control entities of a Navy/Marine Expeditionary Ship Interdiction System (NMESIS) Platoon operating in a littoral environment
- Answer the following research questions:
  - What are the alternative possible flows for a baseline mission, given events that can occur in the system's environment?
  - Can the mission scenarios be characterized with durations, probabilities and/or costs to support acquisition decisions?

Deliverables

- Sponsor Briefing
- NRP Executive Summary
- Project Poster

Research Methodology

- Collected source data*
- Created draft Monterey Phoenix (MP) model for a baseline scenario from source data
- Validated the operational content of the model internally
- Modeled alternatives to the baseline scenario, including both nominal and off-nominal scenarios
- Used the MP-Firebird tool for scope-complete scenario generation
- Assigned notional durations to the key events to estimate whole scenario durations
- Assigned notional probabilities for alternatives to calculate whole scenario probabilities

* Department of the Navy. (2017.) Littoral operations in a contested environment (Unclassified edition). Retrieved from https://mca-marines.org/wp-content/uploads/Littoral-Operations-in-a-Contested-Environment.pdf

Model Demonstration

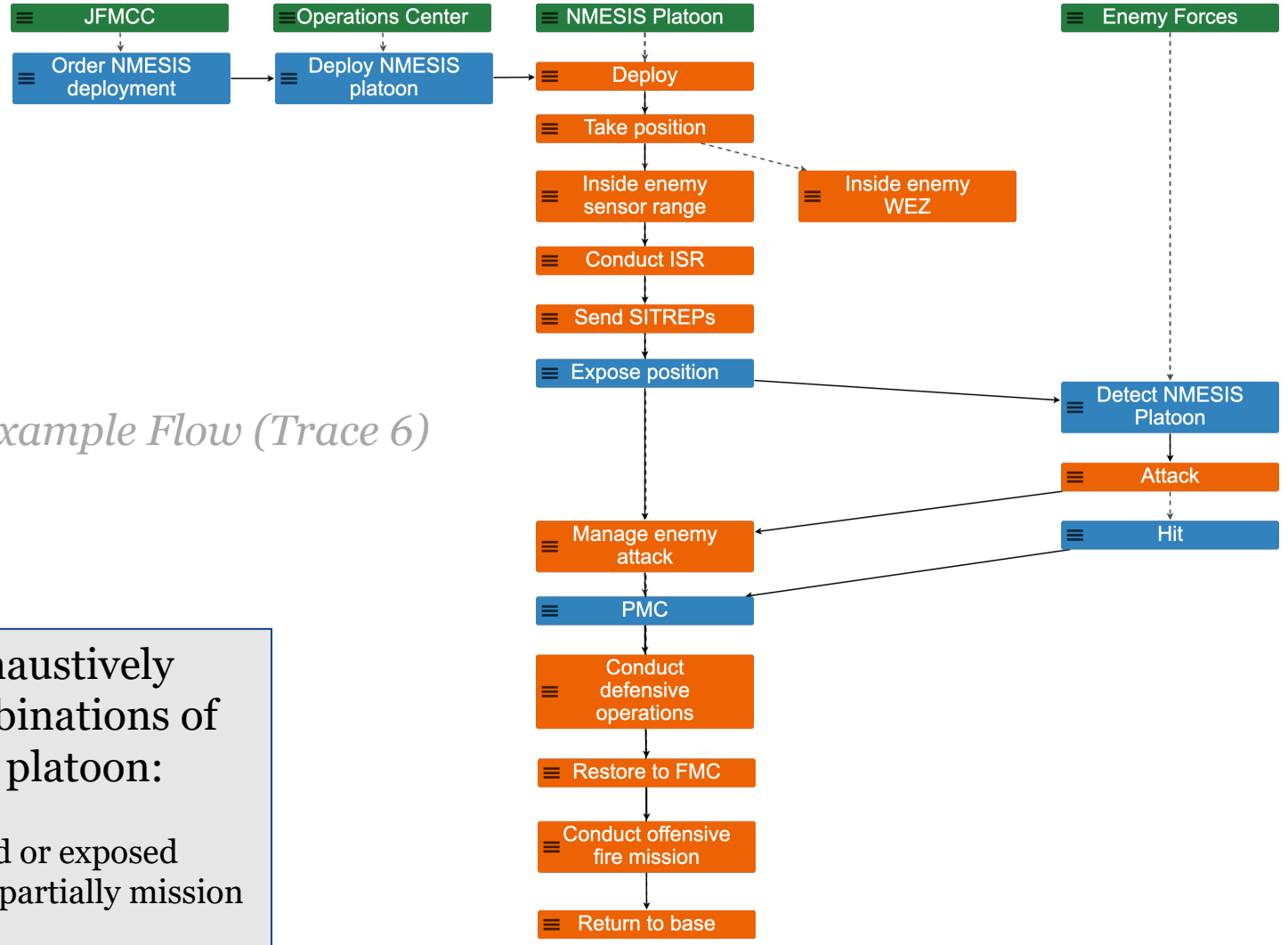
(Littoral Operation v13.mp)

NMESIS Platoon was deployed

Managed enemy attack!

NMESIS Platoon total time in action 121

Returns to base at 121 minutes



Example Flow (Trace 6)

An MP behavior model exhaustively generated all possible combinations of alternative flows, e.g., how platoon:

- deploys or fails to deploy
- position remains concealed or exposed
- is fully mission capable or partially mission capable

Findings & Conclusions

- Logical and simplifying constraints reduced the number of valid scenarios while keeping a formal record of important assumptions
- Implicit assumptions can be made explicit for all to understand
- Constraints can be toggled on or off to admit or reject different combinations of events during validation of the scenarios
- Notional durations and probabilities enabled whole scenario estimation of characteristics
  - MP-Firebird calculations exclude zero-probability scenarios rejected by constraints*
  - Notional values were used to test the approach instead of experiential or historical data

- MP modeling supports requirements discovery and analysis by providing scenario combinations that are unavailable in such numbers in a manual scenario generation process.

* Quartuccio, J. (2019.) Identification of behavior patterns in system of systems architectures (Doctoral dissertation). Pending publication on https://calhoun.nps.edu/handle/10945/16

Limitations

- Availability of subject matter experts (SMEs) willing to volunteer their time in order to validate the model and its constraints
  - the sample constraints applied to this model are only examples of constraints one might apply
- Notional probabilities were assigned to events in place of probabilities based on SME experiential or historical data

Recommendations for Future Research

- Further testing of the MP modeling approach on a real system at MCSC to see if it can expose real requirements
- Quantify the value (e.g., in time or dollars) of having exposed the assumptions, constraints and need for requirements to an actual program office
- A Naval Postgraduate School student working at MCSC would be an ideal candidate for involvement in future work
- The completion of a standalone installation of MP software would facilitate running models of real systems in the MCSC work environment

Questions?

Monterey Phoenix and Related Work:
https://wiki.nps.edu/display/mp
https://4.firebird.nps.edu

MONTEREY PHOENIX
BEHAVIOR MODELING

Kristin Giammarco: kmgiamma (at) nps.edu

Appendix: Overview of Monterey Phoenix

PURPOSE OF SYSTEM BEHAVIOR MODELS

BENEFITS OF MP

OVERVIEW OF THE MP-FIREBIRD TOOL

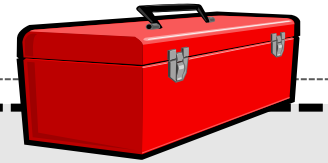
SMALL SCOPE HYPOTHESIS

GOOD MODELING PRACTICES

What is Monterey Phoenix?

- Monterey Phoenix (MP) is an NPS-developed approach and language for modeling behaviors and interactions for:
  - Systems, including Systems of Systems (SoS)
  - Software
  - Hardware
  - People
  - Organizations
  - Operational and Business Processes
- MP-Firebird is an easy-to-use behavior modeling tool that implements MP.
  - https://4.firebird.nps.edu

We Build Models to Answer Questions.



Analysis Questions

Source Data

collected from customers, subject matter experts, end users, etc.

CORE®

Enterprise Architect

Innoslate

CAMEO SYSTEMS MODELER

extendSim

OPCAT

Object-Process Methodology ISO 19450

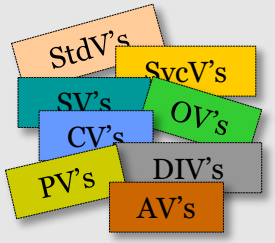
MONTEREY PHOENIX BEHAVIOR MODELING

MATLAB

SIMULINK

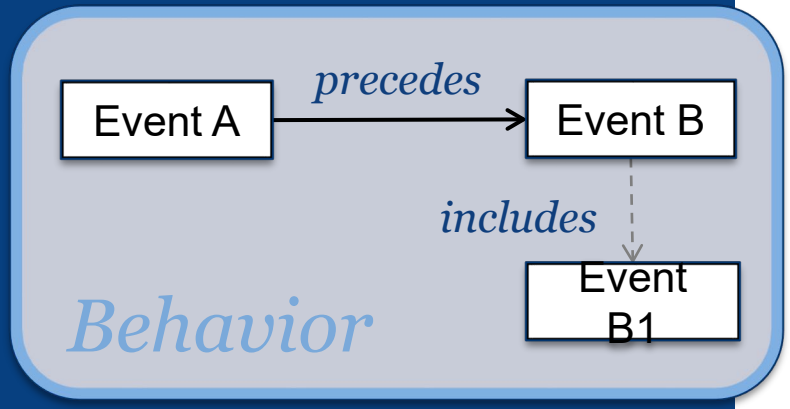
Results to Inform Decisions

Documentation



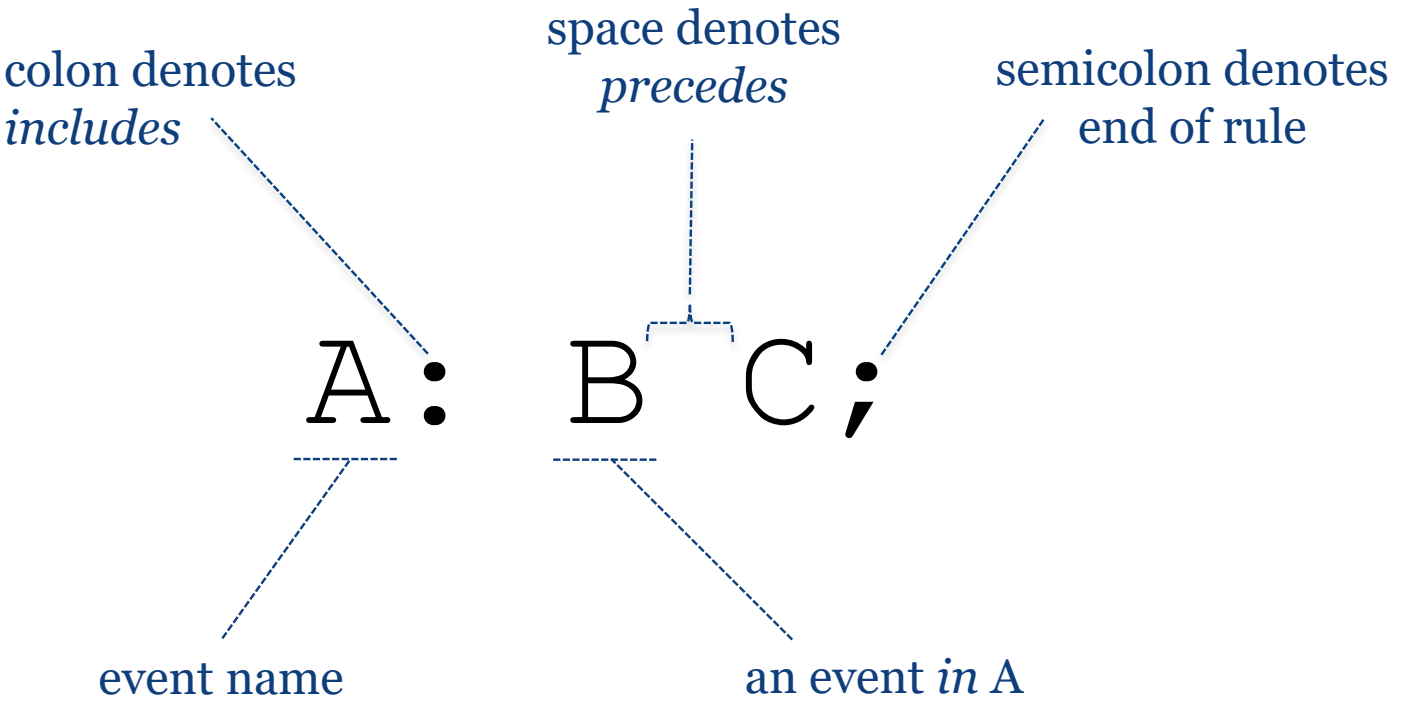
Executable modeling environments are used to answer questions and enable reasoning about the design of potential solutions.

Terms Defined



- ***Behavior*** is the way in which something conducts activity; a set of events.
  - The behavior of something can change based on interactions with other behaviors.
- An ***event*** is the fundamental building block for behavior.
  - An event can be an action with a beginning and end time and/or a duration, a condition, a state, an occurrence, or an outcome.
  - Events generally have verb-oriented names; but when an event represents an object's behavior, it may be named after the object (noun-oriented).
- Events have two basic relations:
  - ***precedence*** is a causal dependency; some event is required to precede another.
  - ***inclusion*** is a hierarchical dependency; some events are nested within other events.

Anatomy of an MP Grammar Rule

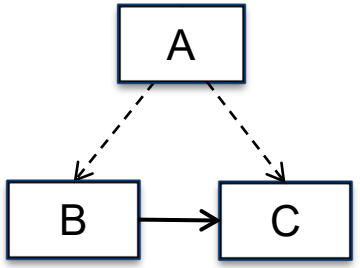


Example Patterns for Event Grammar Rules

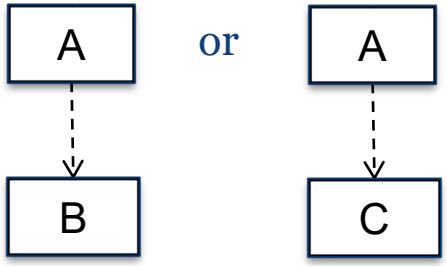
and

Example Trace Instances

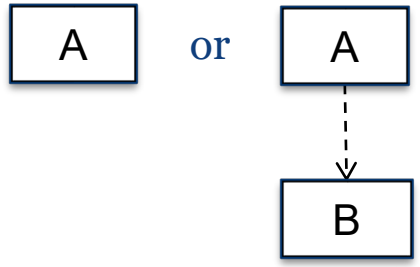
```
A: B C;
```



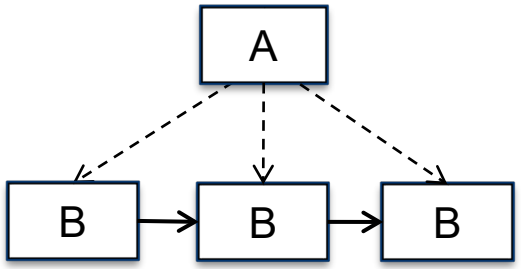

```
A: ( B | C );
```



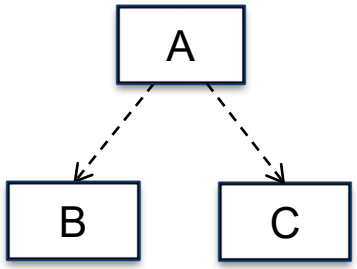

```
A: [ B ];
```



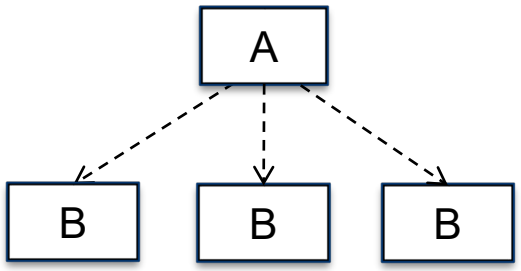

```
A: (* B *);
```




```
A: { B, C };
```




```
A: {* B *};
```



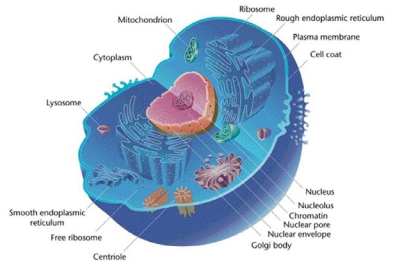
MP models behavior of

- Systems and SoS
- Software
- Hardware
- People
- Organizations
- Operational and Business Processes

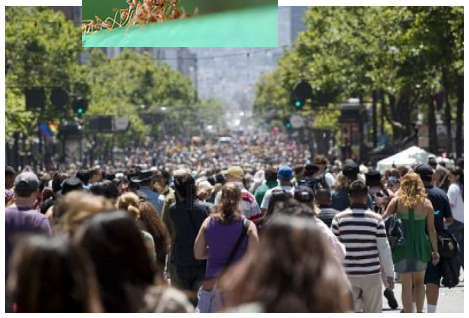
https://news.usni.org/2017/10/10/navy-releases-final-mq-25-stingray-rfp-general-atomics-bid-revealed

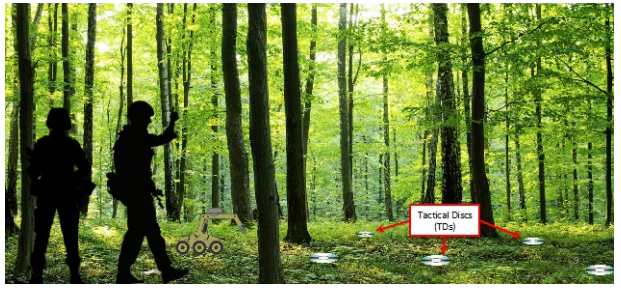
https://www.livescience.com/37009-human-body.html

https://geekszine.com/wp-content/uploads/2012/11/Social-Media-for-Business.jpg



https://biologydictionary.net/eukaryotic-cell/





Why should I learn MP?

(For starters…)

- Good process and product design has good **behavior** at its foundation.
  - Most of the challenges and risks associated with emergent behaviors are found where separate behaviors interact and produce a new behavior.
  - MP models behaviors and interactions separately, then combines them in simulation to generate *emergent behavior scenarios*.
- MP helps the modeler **clarify and fix ambiguity** in less formal representations of systems and processes.
  - Other behavior modeling tools do not enforce the same degree of formality.
- MP is unlike any other approach when it comes to **scenario completeness and consistency**.
  - Event trace generation is **automatic and exhaustive** up to a user-defined scope, making quick work of a task that would be impractical for a human to perform manually and repeatedly and freeing the human to do what humans are good at.
- MP has the necessary features to **expose unexpected behaviors** and steer the behaviors in system and system of systems (SoS).
  - MP has surprised its users in showing unwanted behaviors permitted by the design.

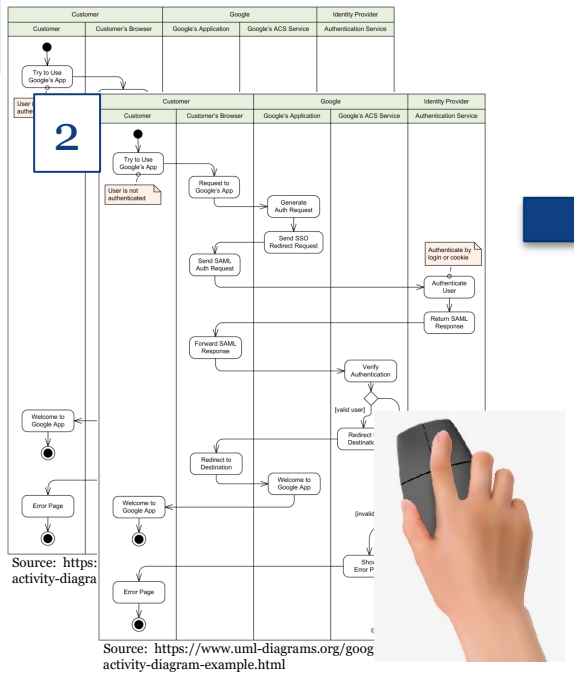
What sort of questions can I answer using MP?

- How many scenarios contain unwanted behaviors? What could they have cost, had they not been exposed?
  - What constraints / requirements will prevent the unwanted behaviors?
- How many valid possible expressions of system behavior are there?
  - Total?
  - If a given event were to occur or to not occur?
  - How many are failure modes?
- What is the probability of some event occurring?
  - In a given trace?
  - In any trace?
  - Given some other event occurs or does not occur?
- What is the probability of success/failure?
- Which behaviors are the most severe (highest consequence) and how likely are they to occur?
  - How should the risk be managed?

What sort of questions can I answer using MP?

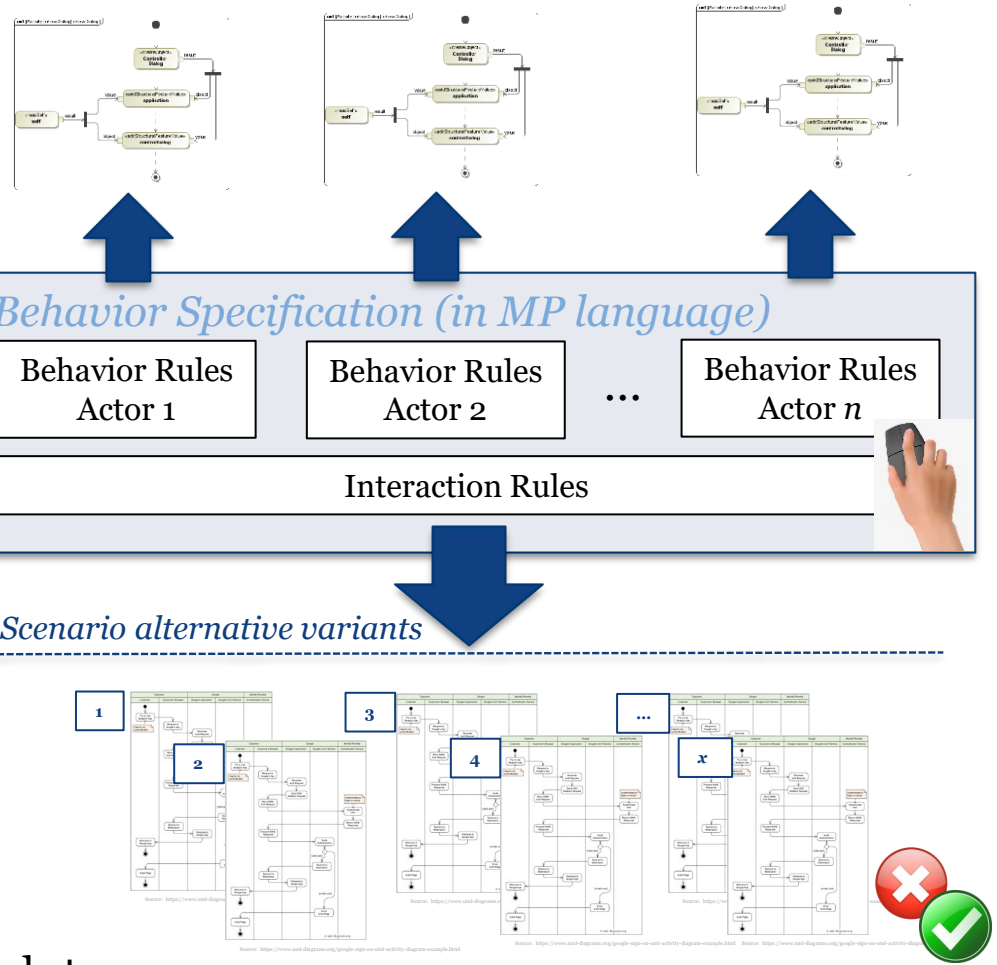
- What is the expected duration of a behavior sequence (scenario)?
  - For a given event trace? average / minimum / maximum for all traces at a given scope?
  - How many traces take longer (or shorter) than a given amount of time?
- How much of a resource does the behavior produce or consume over time?
  - For a given event trace? average / minimum / maximum for all traces at a given scope?
  - units, dollars, fuel, water, food, projectiles, waste, etc.
  - How many traces show a resource exceeding (or falling short of) a certain amount?
- How does the behavior capture and release shared resources over time?
  - Manpower, bandwidth, service stations, etc.
- How is the overall behavior impacted if there are not enough resources available for a given event to start or finish?
  - How many traces contain a resource contention issue?

A shift in the way we create models



Source: https:
activity-diagra

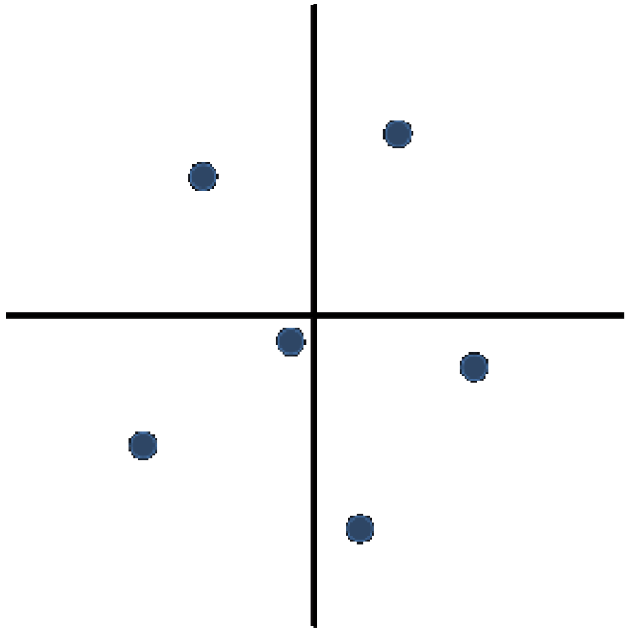
Source: https://www.uml-diagrams.org/goog
activity-diagram-example.html



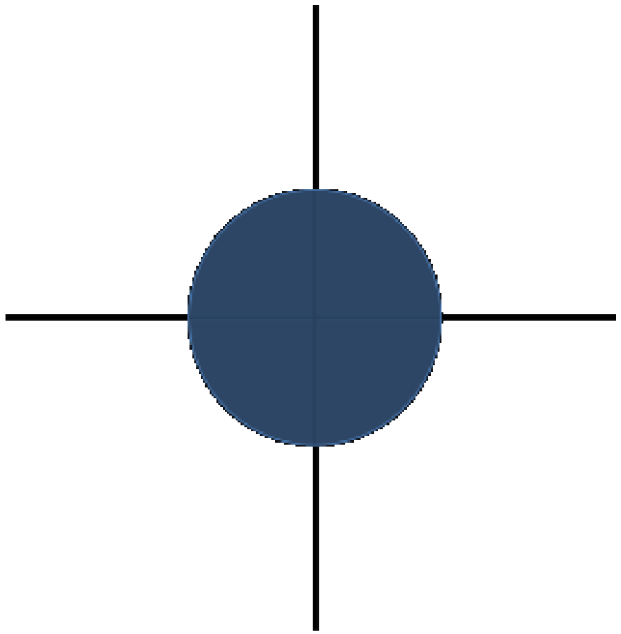
MBSE tools are often used as a digital step up from pencil-and-paper drawn diagrams. Some but not all tools guarantee consistency among some diagrams.

MP diagrams are generated automatically from a single behavior specification in the MP language, guaranteeing consistency among all diagrams.

What is “Scope” in MP?



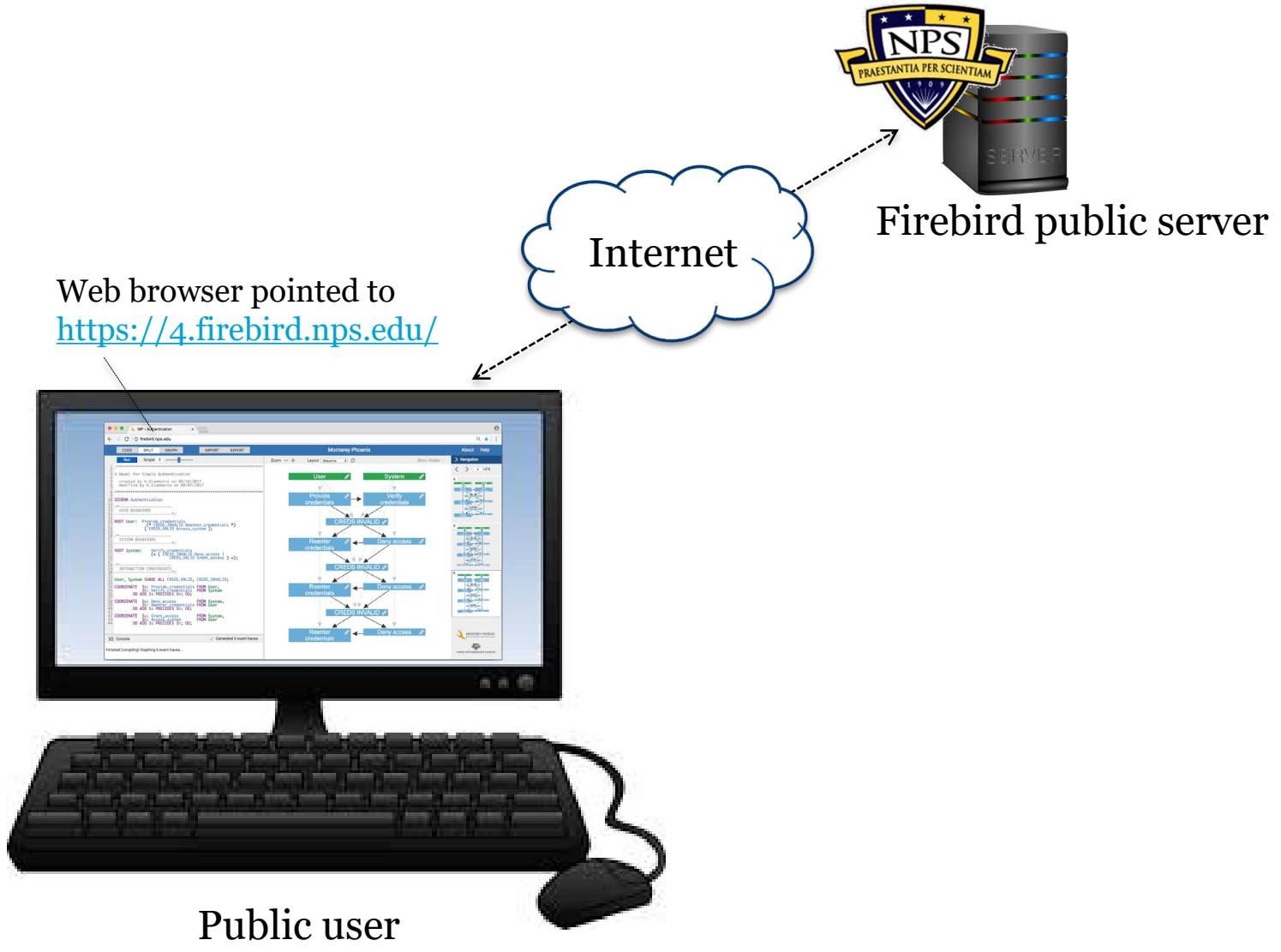
Testing: A few cases of arbitrary size



Scope-complete: All cases within a small bound

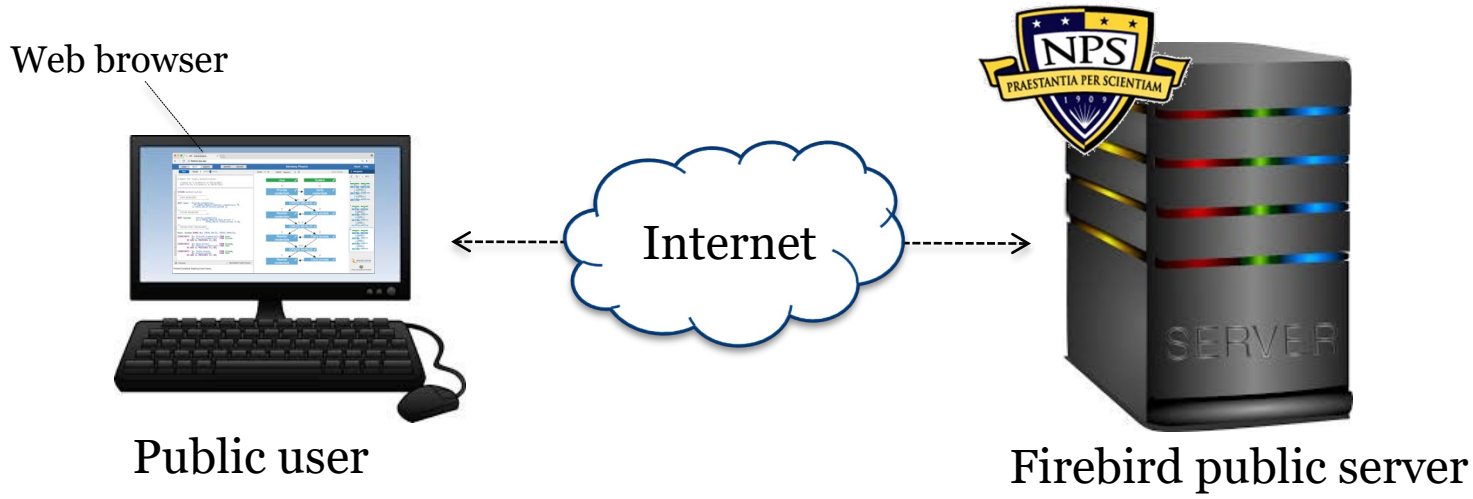
The Small Scope Hypothesis: most flaws in models can be demonstrated on small counterexamples

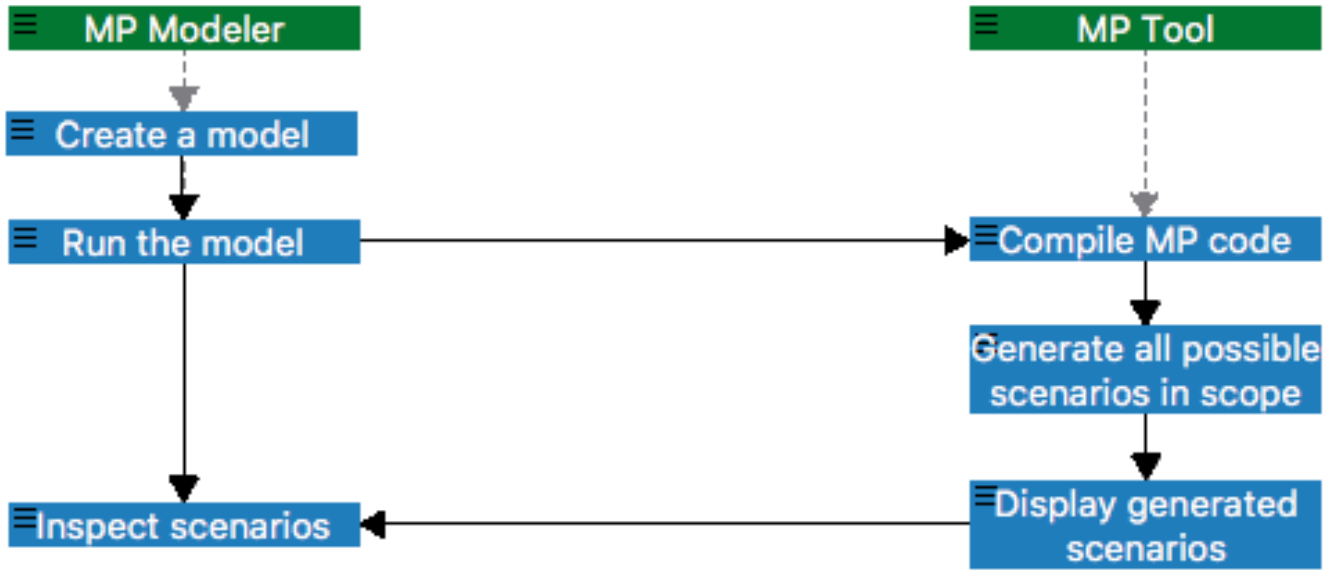
MP-Firebird Tool



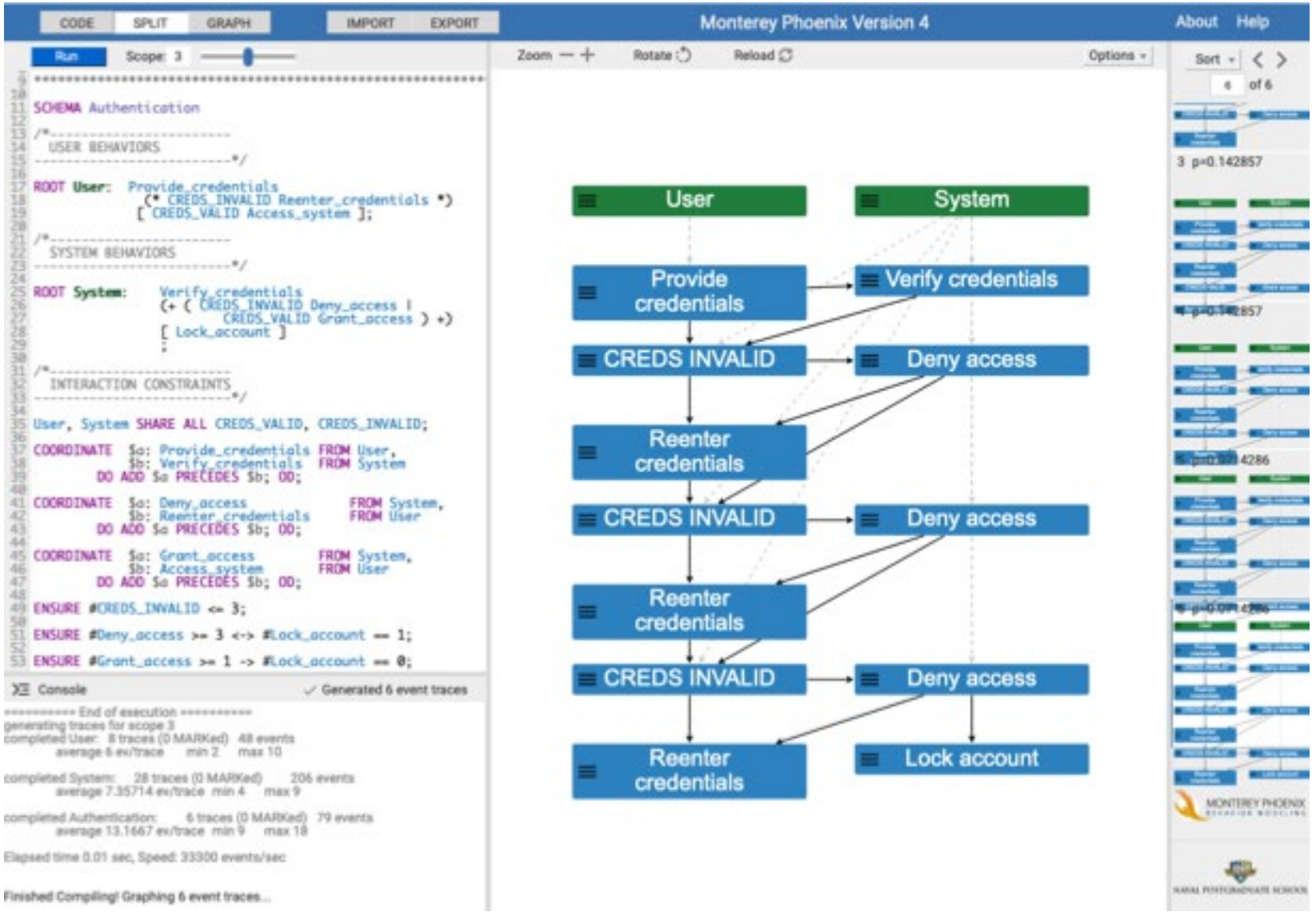
MP-Firebird Tool

https://4.firebird.nps.edu/





Authentication Model Demonstration



Some Use Cases for MP

- To verify and validate activity models developed in notations such as SysML [25]
- To generate comprehensive use case scenario variants for activity models [19]
- To count function points and estimate cost [10]
- To discover and document templates for behavior patterns [22]
- To detect, classify, predict and control emergent behaviors [29][30]

Example Emergent Behaviors Found using MP Modeling

- **An order processing system** enters a waiting state after a transaction is cancelled. [15]
- **A first responder** administers rescue medication to an unconscious patient, unaware that the medication was already administered. (Bryant 2016)
- The **International Space Station** is unaware of a hazardous condition within a supply spacecraft as that spacecraft approaches to dock. (Nelson 2015)
- A **UAV** on a search and track mission reaches a return-to-base condition, then finds and begins to track a new target. [18]
- A **UAV** on a small package delivery mission is commanded to drop its payload before reaching the target vessel. [20]

Inspect for incorrect or unintended behaviors

Far left: Baseline scenario; vessel located and payload on target.

Middle left: Vessel located but payload missed target.

Middle right: AV needs to return before vessel is located.

Far right: Vessel not found but AV drops payload.

AV_Temp.mp, debugging model for Av7f_phase3.mp developed by D. Shifflett 8/21/2018

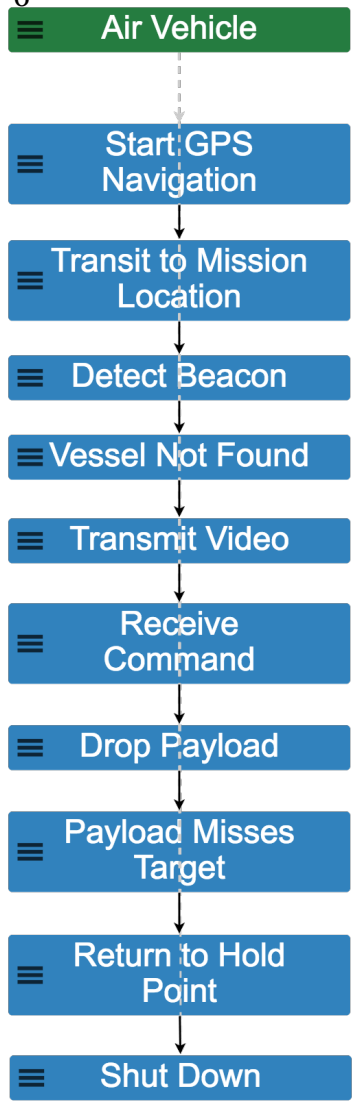
Inspect for incorrect or unintended behaviors

Could this scenario really happen?

Under what circumstances might this be negative behavior or positive behavior?

Though unintended, does trace 6 contain an idea for handling out of range vessels or AVs experiencing a return to base condition?

6

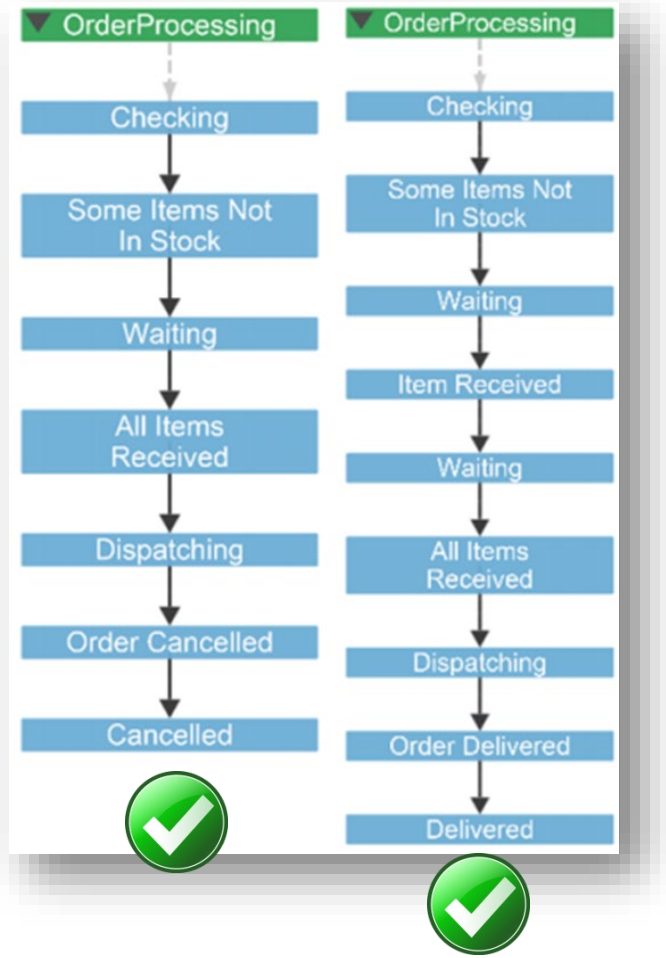


Vessel not found but AV drops payload.

AV_Temp.mp, debugging model for Av7f_phase3.mp developed by D. Shifflett 8/21/2018

An order processing system enters a waiting state after a transaction is cancelled.

Valid Scenarios: Orders conclude normally.

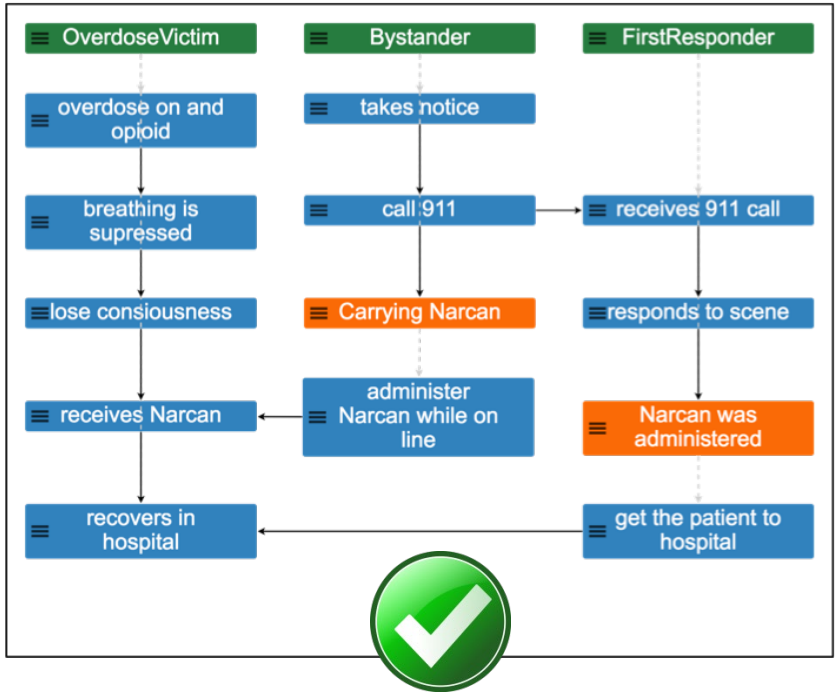


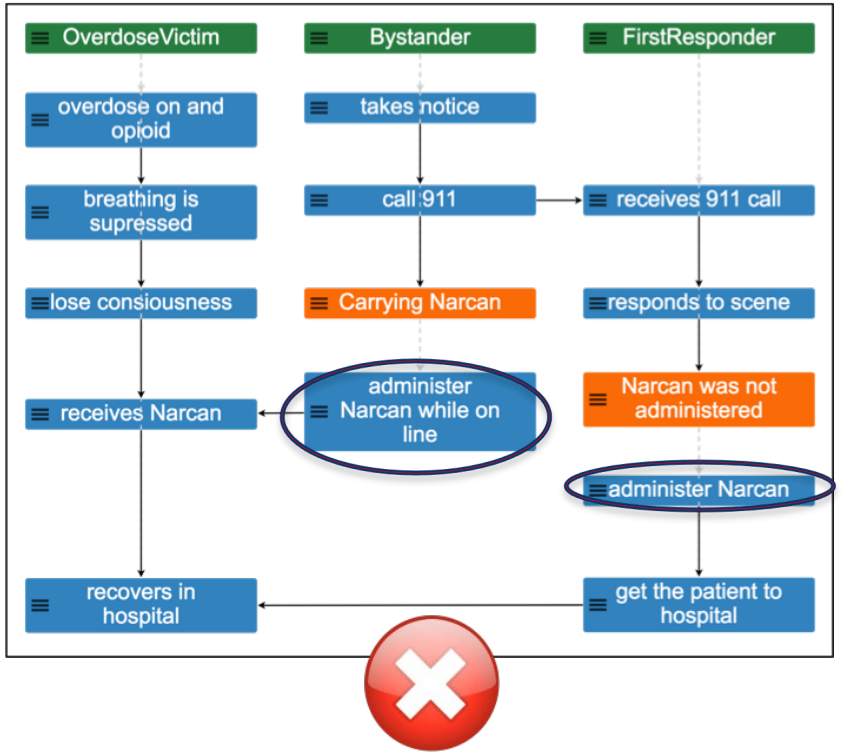
Invalid Scenario: This order hangs in a waiting state.

Example Found Requirement: The Order Processing System shall end all started transactions in either the Cancelled or Delivered state.

Pilcher, Joanne D. "Generation of department of defense architecture framework (DODAF) models using the Monterey Phoenix behavior modeling approach." Master's Thesis, Naval Postgraduate School, Monterey, CA. September 2015.

A first responder administers rescue medication to an unconscious patient, unaware that the medication was already administered.

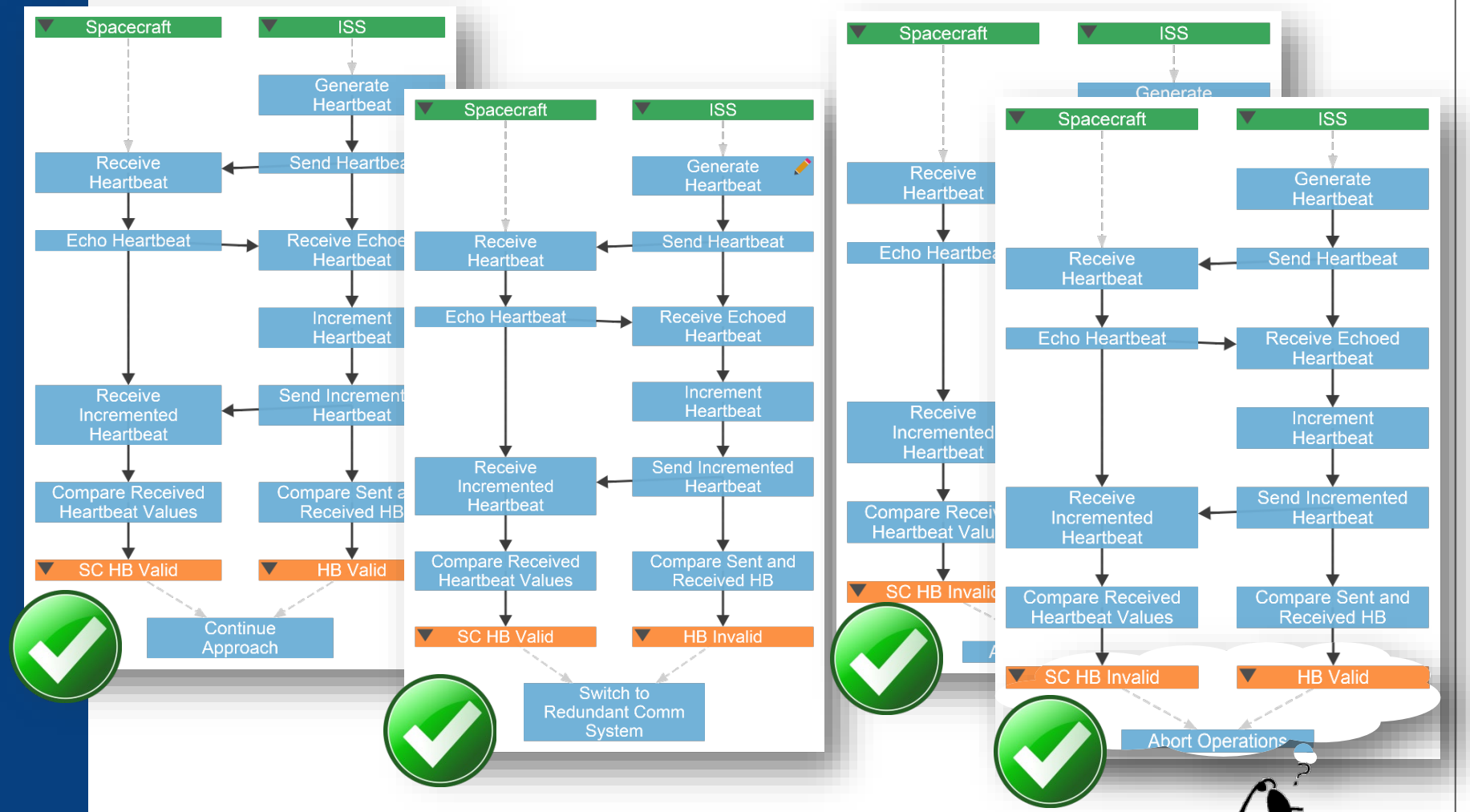




Example Found Requirement: Any Bystander who administers Narcan to an Overdose Victim shall place a band around the Overdose Victim's wrist that indicates the amount and time of the Narcan dose administered.

Bryant, Jordan. "Using Monterey Phoenix to analyze an alternative process for administering Naloxone." Capstone Research Project, Science and Math Academy, Aberdeen, MD. June 2016.

The International Space Station is unaware of a hazardous condition within a supply spacecraft as that spacecraft approaches to dock.

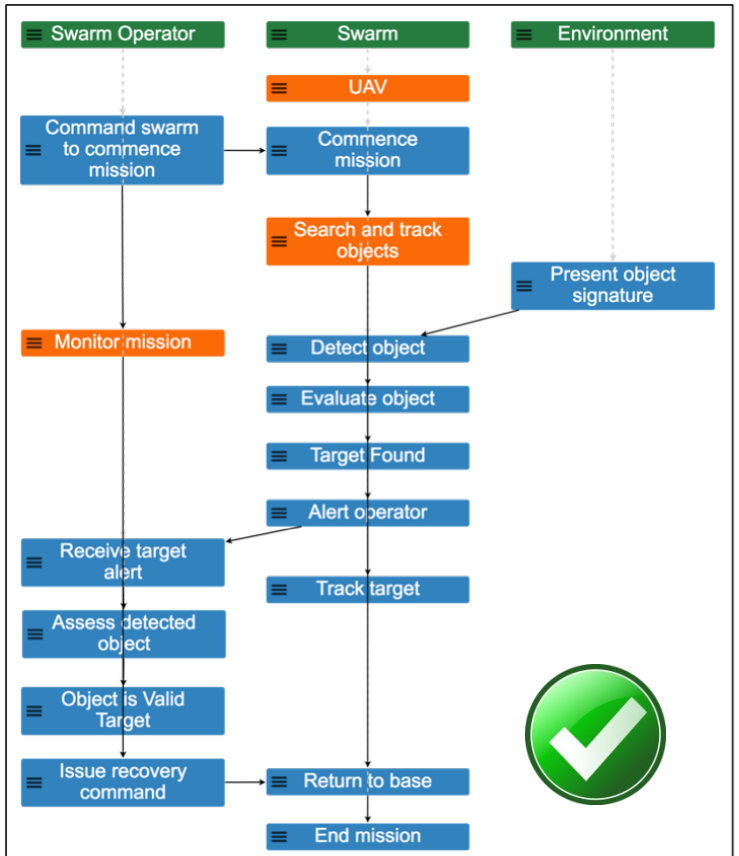


Example Found Requirement: The ISS shall abort docking operations with a spacecraft that has an invalid heartbeat comparison, even if the ISS heartbeat comparison is valid.

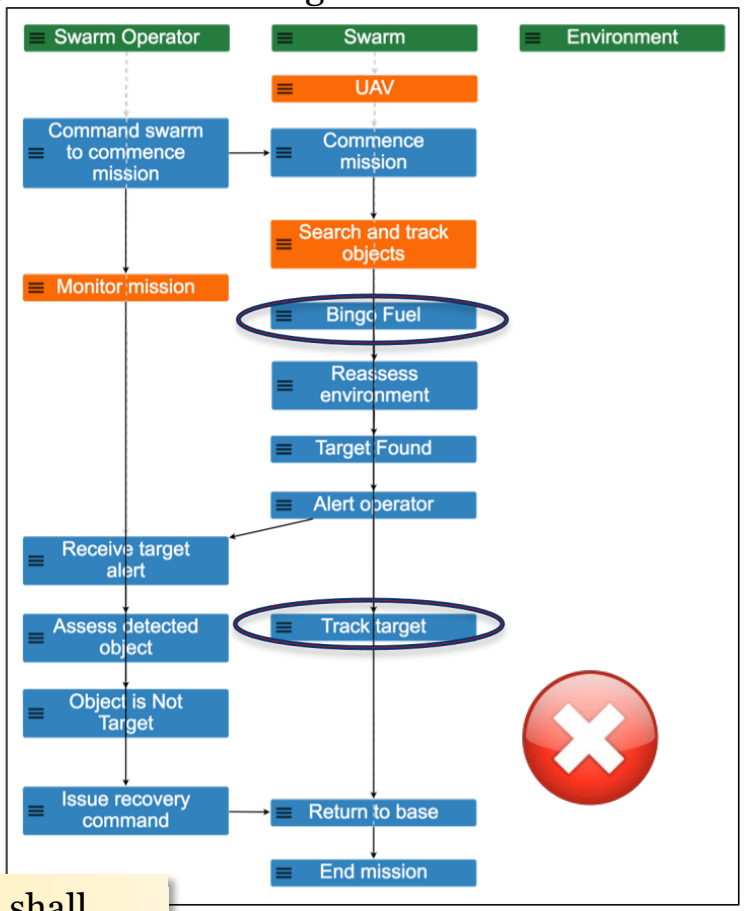
Nelson, Cassie. “Modeling a spacecraft communication system using Monterey Phoenix: a systems engineering case study.” Master's Project, Stevens Institute of Technology, Hoboken, NJ. November 2015.

A UAV on a search and track mission reaches a return-to-base condition, then finds and begins to track a new target.

Valid Scenario: Object detected, tracked, and determined by Swarm Operator to be a valid target



Invalid Scenario: Target tracked after bingo fuel condition



Example Found Requirement: A UAV that has reached a bingo fuel condition shall request permission from the Swarm Operator to track any new targets found.

or

Example Found Requirement: A UAV that has found a possible target after reaching bingo fuel shall relay the LKL of the target to the Swarm Operator, then continue to return to base.

or

Example Found Requirement: A UAV shall only track targets found before reaching bingo fuel conditions.

Revill, Michael B. "UAV swarm behavior modeling for early exposure of failure modes." Master's Thesis, Naval Postgraduate School, Monterey, CA. September 2016.

General Analysis of Emergent Behaviors

- **Detection**: Initial discovery of emergent behavior.

- **Classification**:
- **Simple:** derived from element properties and relationships in non-complex or 'ordered' systems [5].
- **Weak:** desired (or at least allowed) emergence produced by a complex system [5].
- **Strong:** unexpected emergence not observed until simulation, testing, or operations [6].

- **Prediction**: Postulation of potential future states of emergence based on detected behaviors.

- **Control**: Management of positive or negative emergent behaviors through M&S or other analysis.

Definition set paraphrased from [4-1]

Example Analysis of Emergent Behaviors with MP

Trace	Detected Behavior	Predicted Behavior	Classification	Control Strategy
2	Vessel located and payload on target.	Mission success - The payload meets the target and the patient is able to use the medication.	Weak Positive Emergence	Valid possible outcome (baseline scenario). Clarify the assumed outcome that the patient is able to use the medication.
3	Vessel located but payload missed target.	Mission failure - The payload misses the target and the patient falls into a diabetic coma.	Weak Negative Emergence	Valid possible outcome. Clarify the assumed outcome that the patient falls into a diabetic coma.
4	AV needs to return before vessel is located.	Mission failure - The AV detects the emergency beacon, but has to return before it can locate the vessel.	Weak Negative Emergence	Valid possible outcome. No further control recommended.
6	Vessel not found but AV drops payload anyway.	Mission failure - The payload is dropped into the ocean without knowing the location of the vessel. Either the system experienced a malfunction, or the command to drop the payload was sent too soon.	Strong Negative Emergence	Add new event `System_malfunction` as alternative to `Receive_command` in Air Vehicle root event. Downgrade to Weak Negative Emergence.
		Mission success - The payload is intentionally dropped without video on the vessel and it is ultimately received by the vessel. The AV Operator may know from another source (such as the beacon) that the vessel is close by, or the payload may be equipped to close the remaining distance so that the AV has the range necessary for its return trip.	Strong Positive Emergence	Add new events to the model to clarify the specifics, assumed outcome, and associated new requirements. Downgrade to Weak Positive Emergence.

Good practices:

Comments & Indentation

3. Sequential events are vertically stacked and left-aligned.

5. The concluding semicolon for root event grammar rules are lined up beneath its corresponding ROOT keyword to make the end of the root grammar rule easily visible.

6. The left hand part of grammar rule is on its own line (applies to both roots and composites).

```
/*********************************************************

My Day Off

This model describes typical routine activities on a day
off.

Source:  Day.mp by M. Auguston, 03-12-2019

    created by:     K. Giammarco    05-01-2019
       (created root event grammar rules and coordination
        constraints)

    modified by:    K. Giammarco    05-02-2019
       (added optional, concurrent, and iterating
        concurrent events)

*********************************************************/

SCHEMA Day

ROOT Me:
        (*
            check_weather

            ( ( pick_up_umbrella    |
              pick_up_sunglasses  )
          go_for_a_walk
          return_home                 |

          stay_home                   )
        *)
;

check_weather:
        ( Sunny | Rainy );

stay_home:
        { clean_garage, listen_to_music };

go_for_a_walk: /* local scope to always bring two dogs */
        {+ <2..2> walk_dog +};

COORDINATE $a: Sunny, $b: pick_up_sunglasses DO
    ADD $a PRECEDES $b;
OD;

COORDINATE $a: Rainy, $b: (pick_up_umbrella | stay_home) DO
    ADD $a PRECEDES $b;
OD;

ROOT Weather:
        (* ( Sunny | Rainy ) *)
;

Me, Weather SHARE ALL Sunny, Rainy;
```

1. It is a good practice to include a comment section at the top of your model that has details about what the model is for (including what question(s) it answers, when mature), sources or references used to build the model, who created or modified the model, what changes they made, and when.

2. Opening and closing parentheses for iterations containing many events are aligned for easy comprehension of what is inside the iteration.

4. Closing parentheses following alternative events are right-aligned to the pertinent OR operator to make it easy to see where each OR operation ends.

7. Comments should be inserted throughout the model where necessary to explain an intention, assumption, or rationale.

8. COORDINATE statements broken up across multiple lines as needed.

Summary

- System behavior models are generally built to answer questions, inform decisions, and produce design descriptions that can be used for early verification and validation.
- MP helps its users remove ambiguity, generate consistent and scope-complete scenario sets, expose unexpected behaviors latent in a design, and uncover hard-to-see requirements.
- Leveraging the Small Scope Hypothesis [24], most behaviors of concern in an MP model can show up after just a few loop iterations.
- Exposed behaviors can be categorized as positive, negative, simple, weak, or strong.
  - Emergent behavior term definitions still undergoing discussion in the research community
  - MP is a tool for providing examples of what these terms mean in different systems

References

1. Auguston, M. Software Architecture Built from Behavior Models. *ACM SIGSOFT Softw. Eng. Notes* **2009**, *34*, 1–15.
2. Auguston, M. Monterey Phoenix, or How to Make Software Architecture Executable. In Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA'09), Orlando, FL, USA, 25–29 October 2009; pp. 1031–1038.
3. Auguston, M.; Whitcomb, W. System Architecture Specification Based on Behavior Models. In Proceedings of the 15th ICCRTS Conference (International Command and Control Research and Technology Symposium), Santa Monica, CA, USA, 22–24 June 2010.
4. Auguston, M.; Whitcomb, C.; Giammarco, K. A New Approach to System and Software Architecture Specification Based on Behavior Models. In Proceedings of the 3rd International Conference on Model-Based Systems Engineering (IC-MBSE 2010), Fairfax, VA, USA, 27–28 September 2010.
5. Rivera, J.; Auguston, M.; Finkbine, R. Applying Architecture Modeling Methodology to the Naval Gunship Software Safety Domain. In Proceedings of the 23rd Annual Systems & Software Technology Conference (SSTC 2011), Salt Lake City, UT, USA, 16–19 May 2011.
6. Giammarco, K.; Auguston, M. Well, You didn't Say not to! A Formal Systems Engineering Approach to Teaching an Unruly Architecture Good Behavior. In Proceedings of the Complex Adaptive Systems Conference, Baltimore, MD, USA, 13–15 November 2013.
7. Giammarco, K.; Auguston, M.; Baldwin, W.C.; Crump, J.; Farah-Stapleton, M. Controlling Design Complexity with the Monterey Phoenix Approach. In Proceedings of the Complex Adaptive Systems Conference, Philadelphia, PA, USA, 3–5 November 2014; pp. 204–209.
8. Auguston, M.; Giammarco, K.; Baldwin, W.C.; Crump, J.; Farah-Stapleton, M. Modeling and Verifying Business Processes with Monterey Phoenix, *Procedia Comput. Sci.* **2015**, *44*, 48–57.
9. Farah-Stapleton, M.; Auguston, M. Behavioral Modeling of Software Intensive System Architectures. In Proceedings of the Complex Adaptive Systems Conference, Baltimore, MD, USA, 13–15 November 2013.
10. Farah-Stapleton, M.; Auguston, M.; Giammarco, K. Executable Behavioral Modeling of System and Software Architecture Specifications to Inform Resourcing Decisions. *Procedia Comput. Sci.* **2016**, *95*, 345–353.
11. Farah-Stapleton, M. Executable Behavioral Modeling of System and Software Architecture Specifications to Inform Resourcing Decisions. Ph.D. Dissertation, Naval Postgraduate School, Monterey, CA, USA, September 2016.
12. Song, S.; Zhang, J.; Liu, Y.; Auguston, M.; Sun, J.; Song Dong, J.; Chen, T. Formalizing and verifying stochastic system architectures using Monterey Phoenix (SoSyM abstract) . In Proceedings of the 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), Ottawa, ON, Canada, 30 September–2 October 2015.
13. Song, S.; Zhang, J.; Liu, Y.; Auguston, M.; Sun, J.; Dong, J.S.; Chen, T. Formalizing and verifying stochastic system architectures using Monterey Phoenix. *Softw. Syst. Model.* **2016**, *15*, 453–471.
14. Hunt, S.S. Model Based Systems Engineering in the Execution of Search and Rescue Operations. Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, September 2015.
15. Pilcher, J.D. Generation of Department of Defense Architecture Framework (DODAF) Models Using the Monterey Phoenix Behavior Modeling Approach. Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, September 2015.

References (Continued)

16. Steward, V. Functional Flow and Event-Driven Methods for Predicting System Performance. Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, September 2015.
17. Ruppel, S. System Behavior Models: A Survey of Approaches. Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, June 2016.
18. Revill, M.B. UAV Swarm Behavior Modeling for Early Exposure of Failure Modes. Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, September 2016.
19. Giammarco, K.; Giles, K.; Whitcomb, C.A. Comprehensive use case scenario generation: An approach and template for modeling system of systems behaviors. In Proceeding of the 12th Annual System of Systems Engineering Conference, Waikoloa, HI, USA, 18–21 June 2017.
20. Giammarco, K. *Verification and Validation (V&V) of System Behavior Specifications*; Technical Report; Systems Engineering Research Center: Hoboken, NJ, USA, 2018.
21. Quartuccio, J.; Giammarco, K.; Auguston, M. Identifying Decision Patterns Using Monterey Phoenix. In Proceeding of the 12th Annual System of Systems Engineering Conference, Waikoloa, HI, USA, 18–21 June 2017.
22. Quartuccio, John and Kristin Giammarco. "A model-based approach to investigate emergent behaviors in systems of systems," Chapter 18 in Engineering Emergence: A Modeling and Simulation Approach, edited by Larry Rainey and Mo Jamshidi. Boca Raton, FL: CRC Press Taylor & Francis Group.
23. Auguston, M. System and Software Architecture and Workflow Modeling Language Manual (Version 4). 2018. Available online: https://wiki.nps.edu/display/MP/Documentation.
24. Jackson, D. Software Abstractions: Logic, Language, and Analysis; MIT Press: Cambridge, MA, USA, 2012.
25. Giammarco, Kristin. "Practical Modeling Concepts for Engineering Emergence in Systems of Systems." Proceeding of the 12th Annual System of Systems Engineering Conference, Waikoloa, HI, June 18-21, 2017.
26. Rainey, Larry and Mo Jamshidi. "Introduction and Overview for Engineering Emergence: A Modeling and Simulation Approach," Chapter 1 in *Engineering Emergence: A Modeling and Simulation Approach*, edited by Larry Rainey and Mo Jamshidi. Boca Raton, FL: CRC Press Taylor & Francis Group.
27. Page, S.E. 2009. *Understanding Complexity*. The Great Courses. Chantilly, VA, USA: The Teaching Company.
28. SEBoK authors. 2017. "System of Systems (SoS)," in BKCASE Editorial Board. 2016. *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 1.8. R.D. Adcock (EIC). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. *Released 27 March 2017, http://sebokwiki.org/wiki/Systems_of_Systems_(SoS)#Definition_and_Characteristics_of_Systems_of_Systems (accessed 12 July 2017)*.
29. Giammarco, Kristin and Mikhail Auguston. "Behavior modeling approach for the early verification and validation of system of systems emergent behaviors," Chapter 17 in *Engineering Emergence: A Modeling and Simulation Approach*, edited by Larry Rainey and Mo Jamshidi. Boca Raton, FL: CRC Press Taylor & Francis Group.
30. Giammarco, Kristin and Kathleen Giles. "Verification and validation of behavior models using lightweight formal methods." Proceedings of the 15th Annual Conference on Systems Engineering Research. Redondo Beach, CA. March 23-25, 2017. Best paper award.