A Formal Task-based Approach for Ensuring Trustworthy Human-Automation Interaction

Matthew L. Bolton, Assistant Professor

University at Buffalo
Department of Industrial and Systems Engineering
School of Engineering and Applied Sciences

Human Task Analytic Behavior Models

- Product of a task analysis
- Describe how humans achieve goals when interacting with a system
- Hierarchy (network) of goal-directed activities and actions

Trust in Automation / Autonomy

- A system can be trusted if it facilitates the human operators' tasks
- This is important because:
  - Systems that do not facilitate tasks produce unexpected interactions
  - Humans will change their task to adapt to system behavior, producing unexpected interactions
  - Unexpected interactions are dangerous

Unexpected Human Interactions:

A major contributor to failures in safety critical systems

Medicine

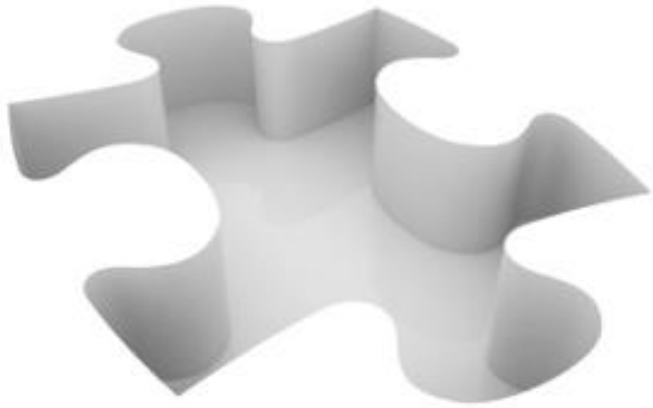
44,000 and 98,000 deaths and
1,000,000 injuries a year

Aviation

75.5% accidents in general aviation and
~ 50% in commercial aviation

Highway Safety

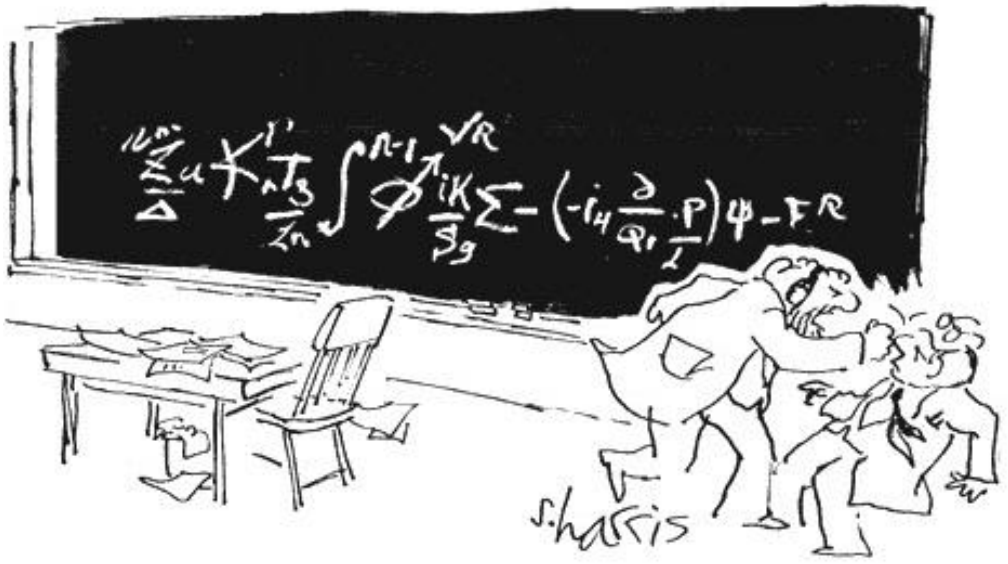
90% of all roadway crashes



Human factors analysis techniques can miss human-system interactions that could lead to system failures

Formal Methods:

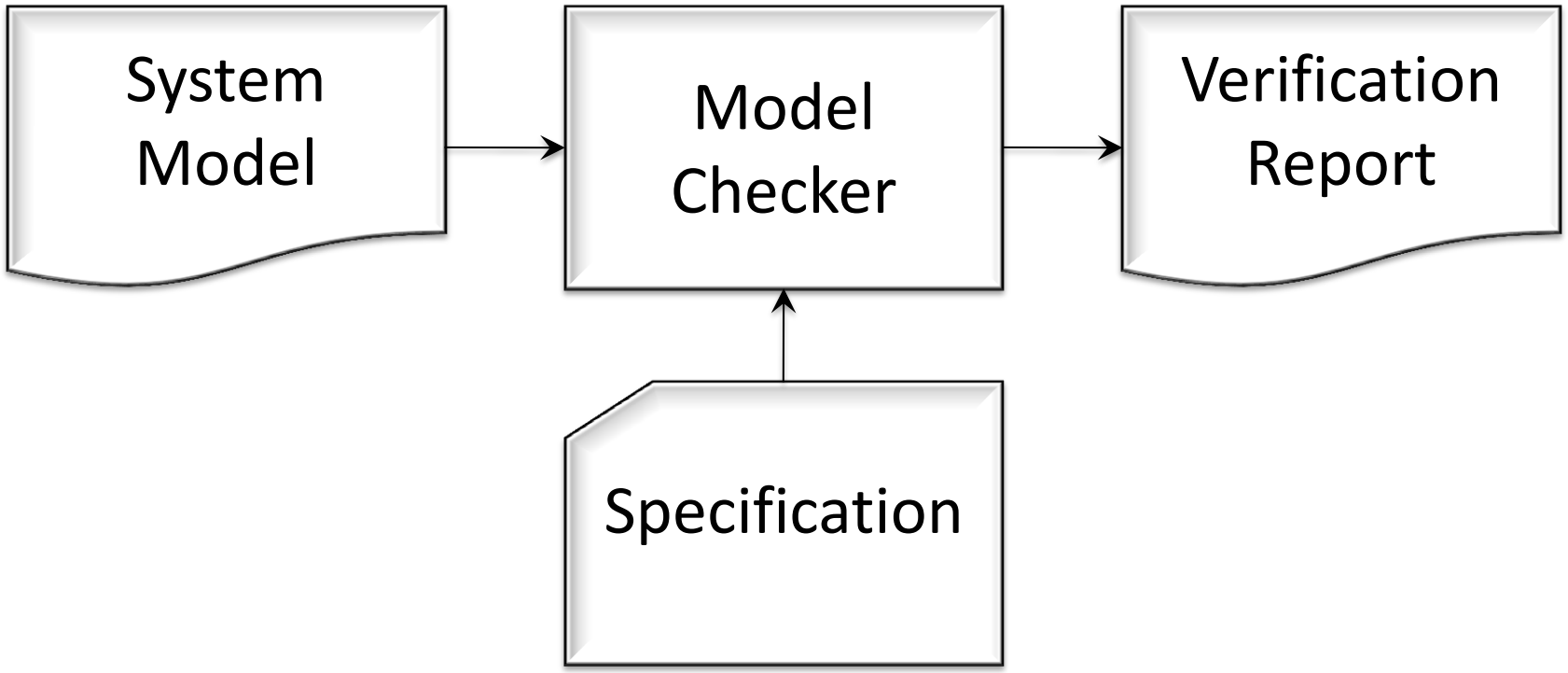
Tools and techniques for **proving** that a system will always perform as intended



"You want proof? I'll give you proof!"

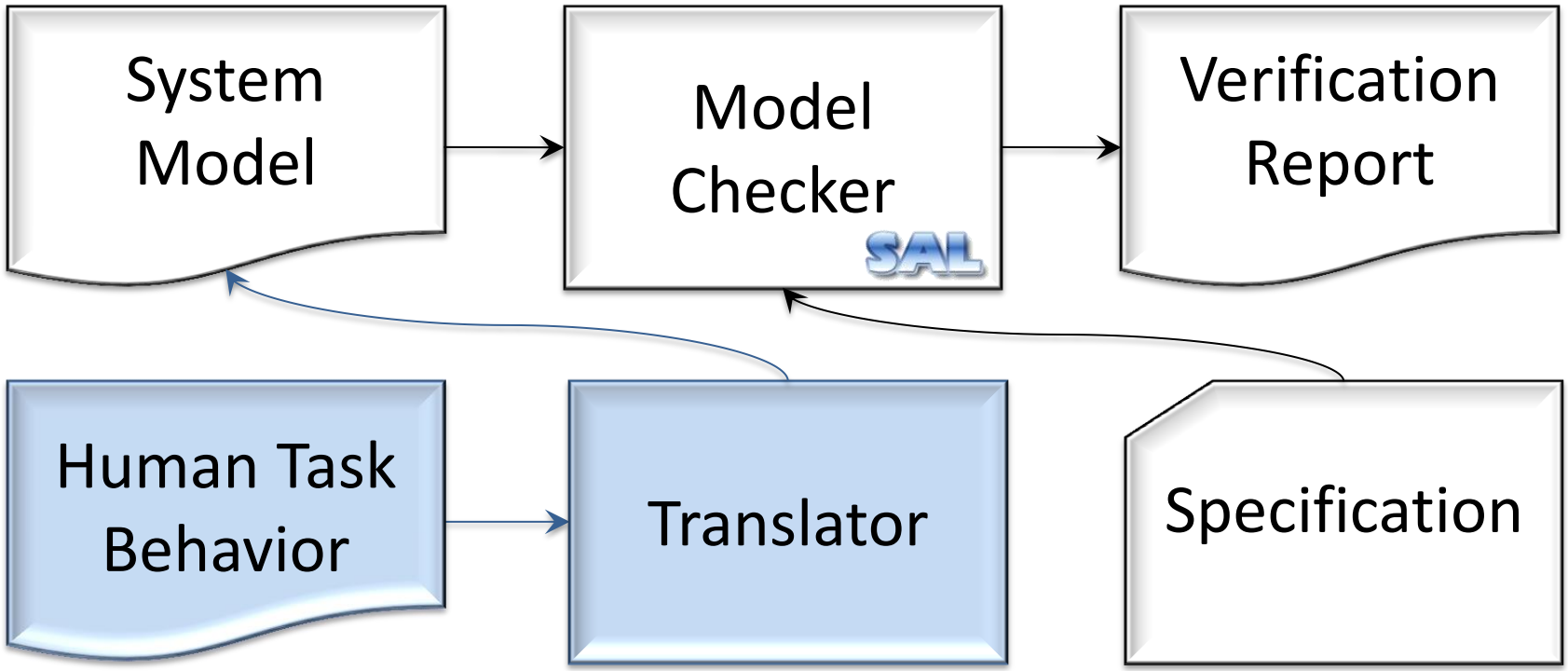
Model checking:

An automatic means of performing formal verification

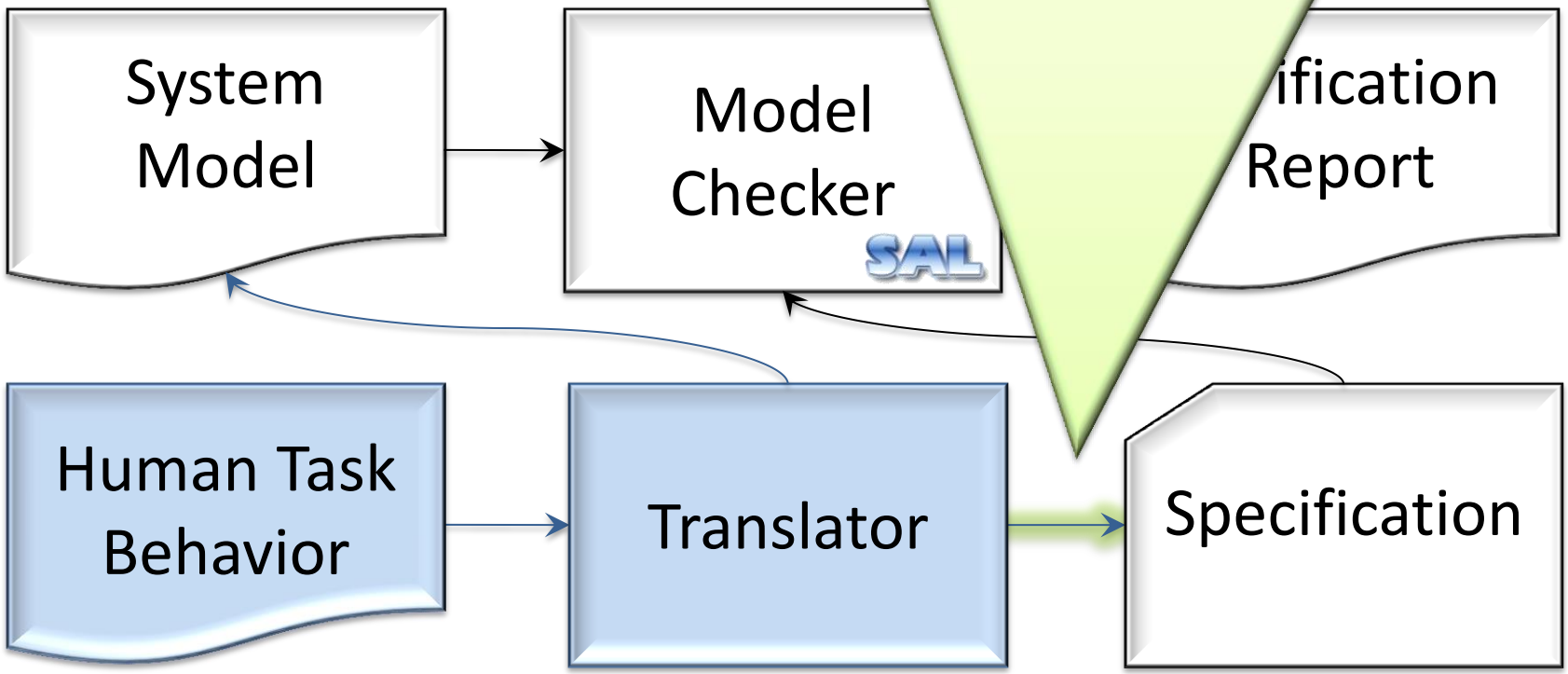


Including human behavior:

Human task behavior is incorporated into a formal system model and the entire model is checked.



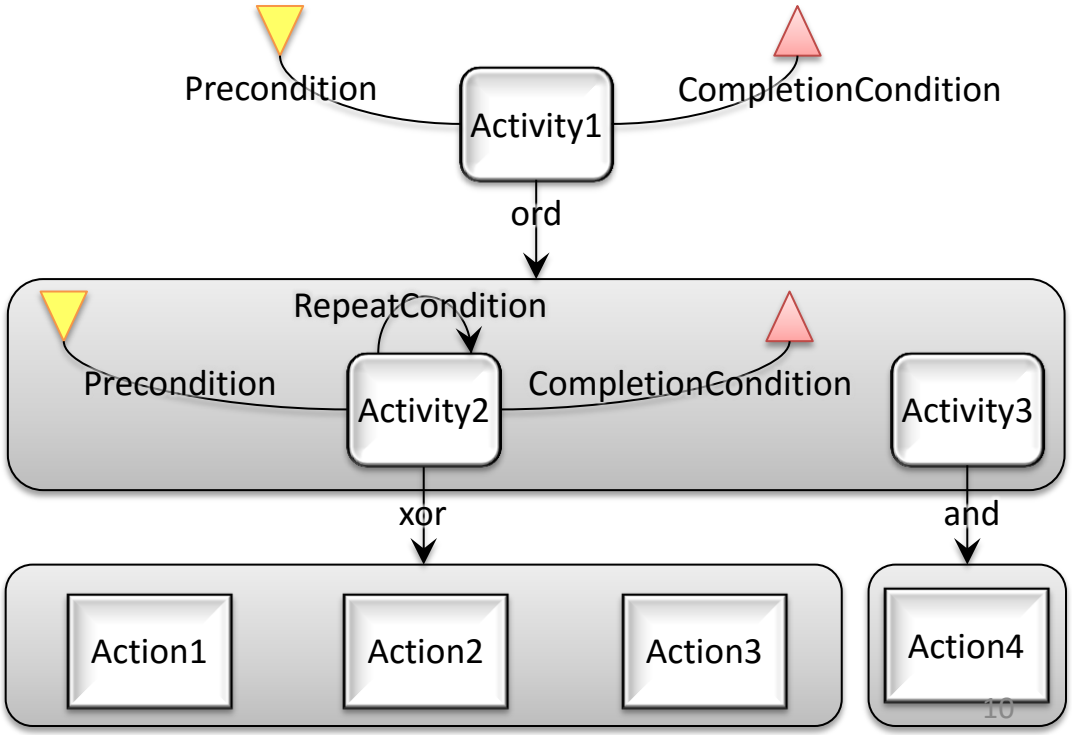
Focus: Generating specifications to check that humans will always be able to accomplish their goals without unanticipated human-automation interaction issues



Enhanced **O**perator **F**unction **M**odel (EOFM)

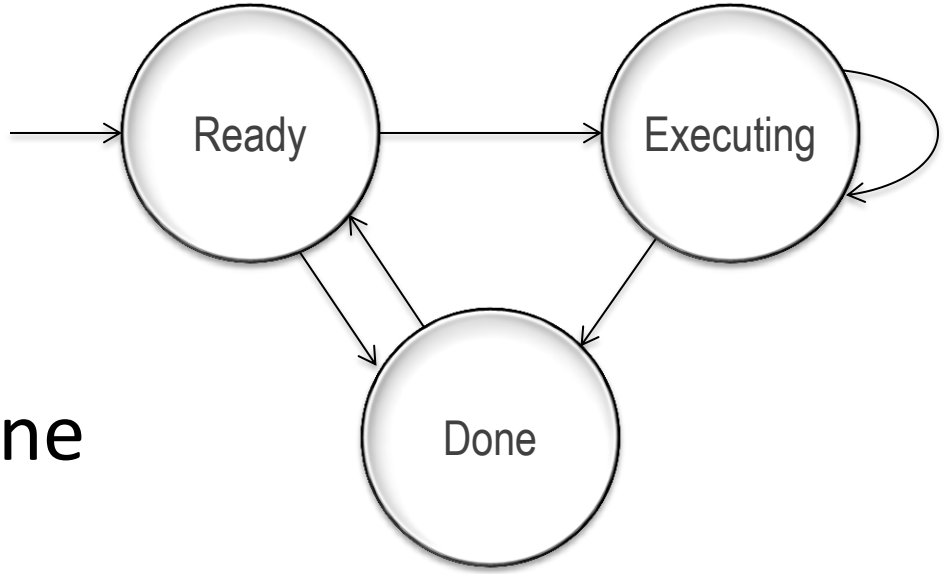
A generic task analytic modeling formalism

- Input/output model
- Hierarchical
- Platform-independent
- XML notation
- Visual notation
- Formal semantics



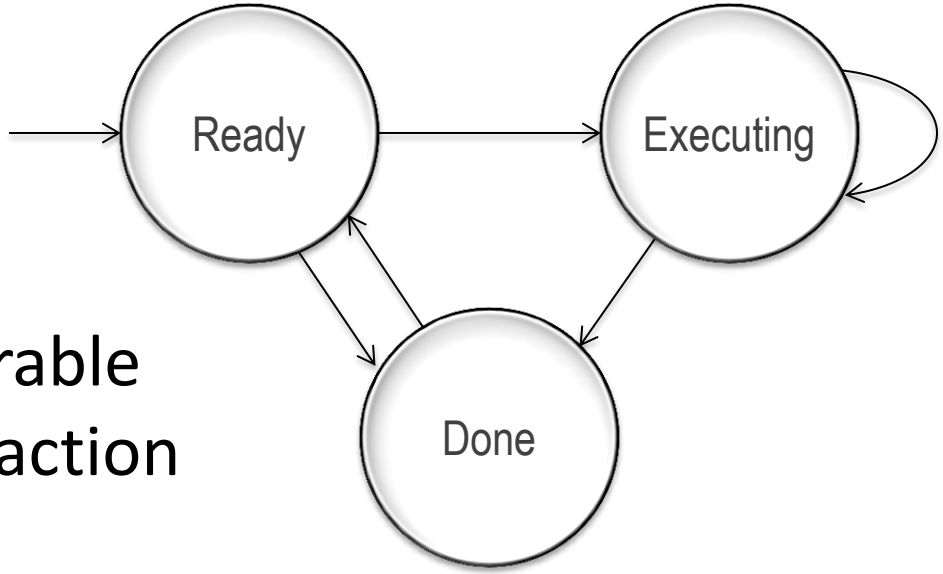
EOFM Formal Semantics

Each activity's and action's execution state is represented as a finite state machine



Specification Generation

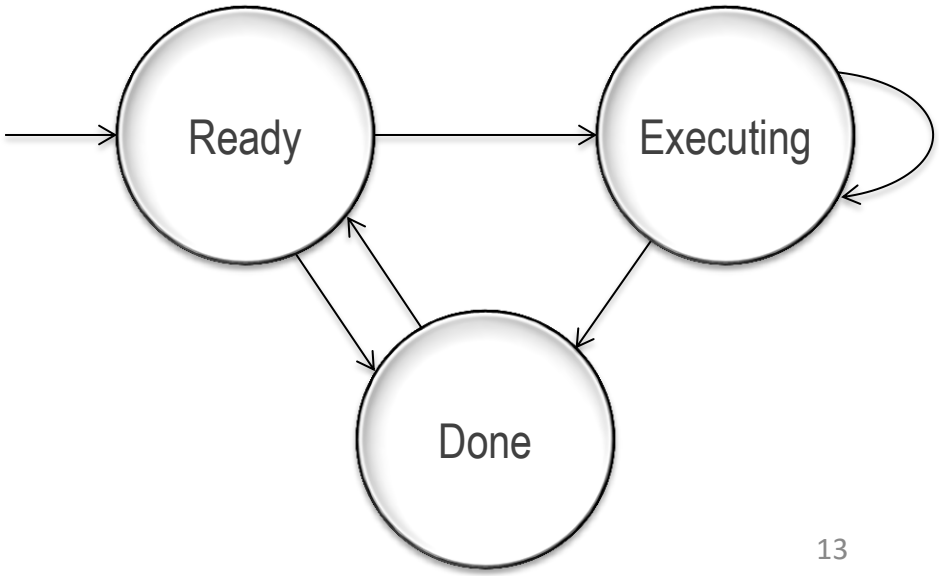
Use computation concepts to automatically generate properties asserting desirable human-automation interaction conditions



… so what can we check for?

Every element of the task should be applicable at some time in the use of the system

State Coverage: Every execution state of every activity and action should be reachable

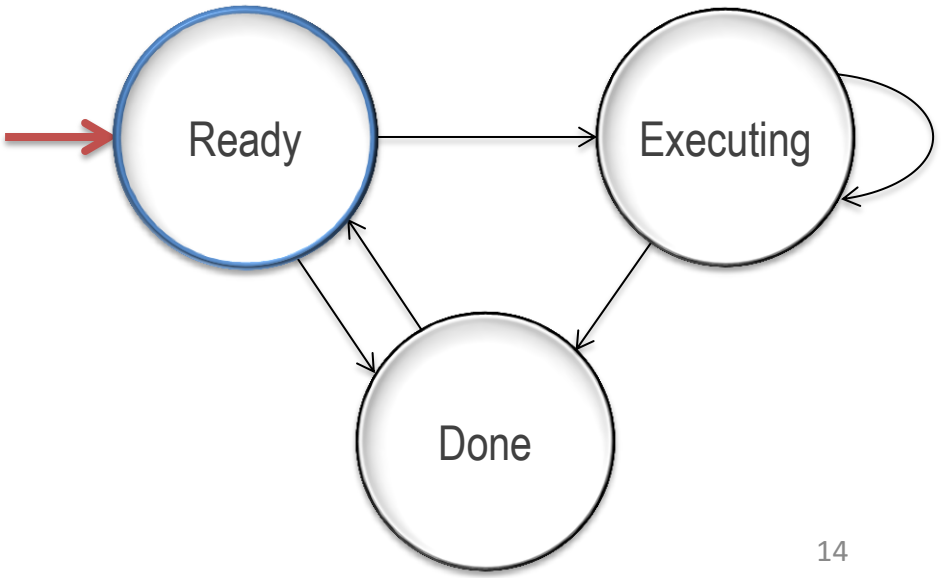


… so what can we check for?

Every element of the task should be applicable at some time in the use of the system

State Coverage: Every execution state of every activity and action should be reachable

Ready is always reachable, so no checking is necessary



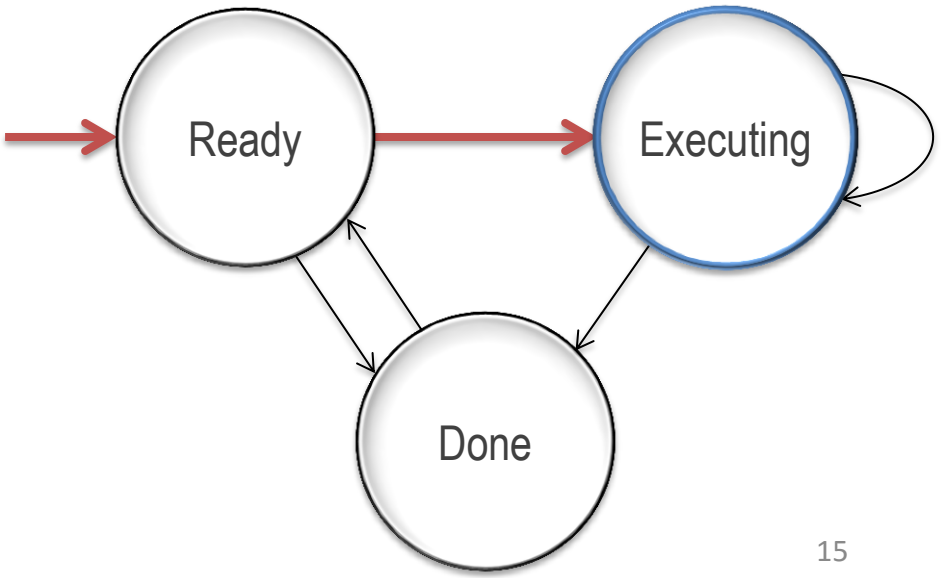
… so what can we check for?

Every element of the task should be applicable at some time in the use of the system

State Coverage: Every execution state of every activity and action should be reachable

Act Executability:

$$\mathbf{G}\neg(Act = Executing)$$



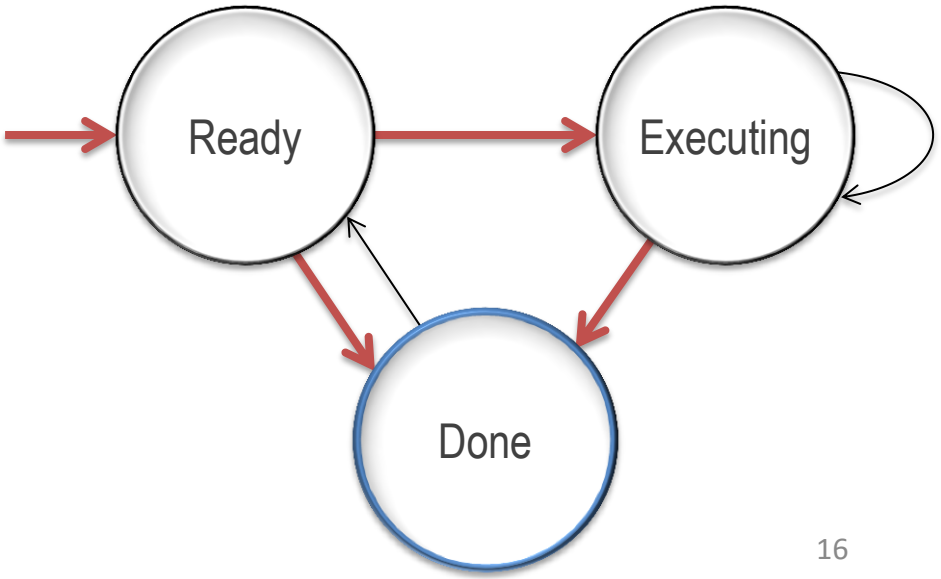
… so what can we check for?

Every element of the task should be applicable at some time in the use of the system

State Coverage: Every execution state of every activity and action should be reachable

Act Completability:

$$\mathbf{G}\neg(Act = Done)$$



… so what can we check for?

Every task that a human operator attempts should always be finishable

Starvation: No part of a task should ever be unable to obtain the resources it needs to finish

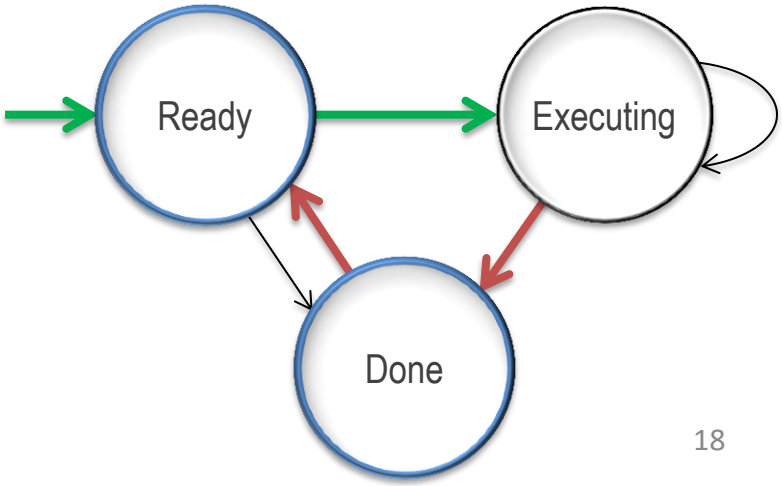
… so what can we check for?

Every task that a human operator attempts should always be finishable

Starvation: No part of a task should ever be unable to obtain the resources it needs to finish

Act Inevitable Completability:

$$\mathbf{G}\begin{pmatrix}(Act = Executing)\\ \Rightarrow \mathbf{F}(Act \neq Executing)\end{pmatrix}$$



… so what can we check for?

There should never be a situation where the human operator can never perform any task

Liveness: The human operator should always eventually be able to perform a task

… so what can we check for?

There should never be a situation where the human operator can never perform any task

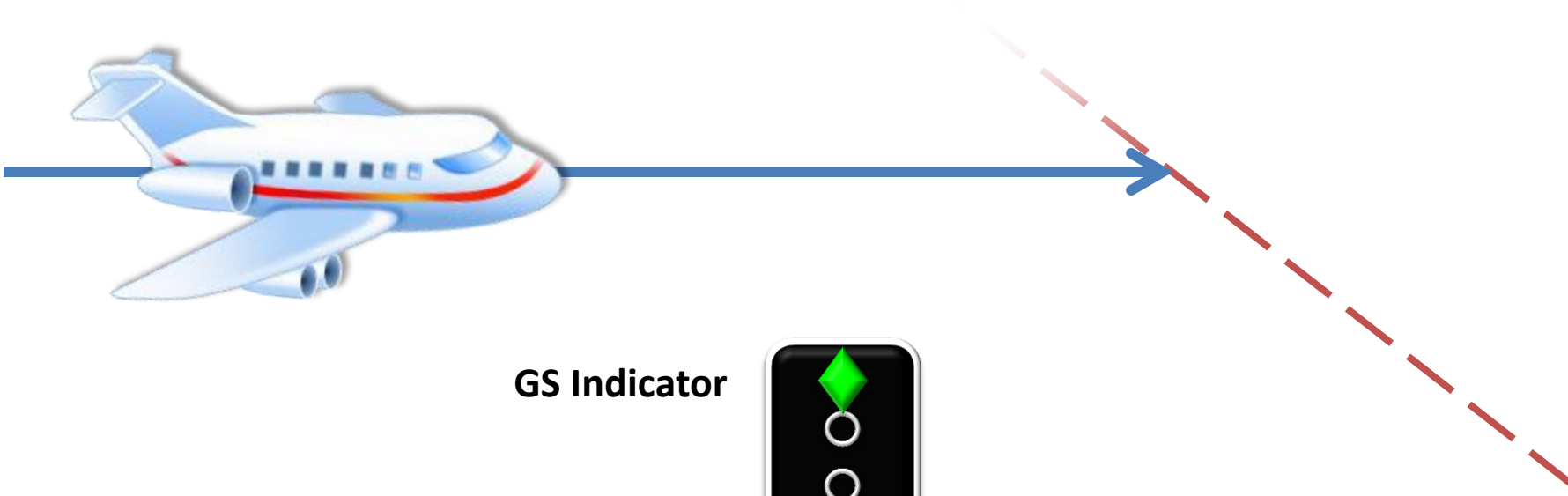
Liveness: The human operator should always eventually be able to perform a task

Task Liveness:

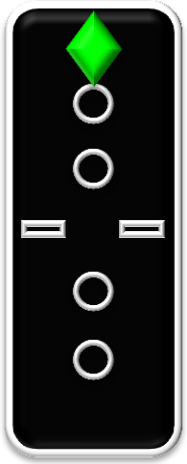
$$\mathbf{G}\neg\left(\mathbf{F}\left(\mathbf{G}\left(\bigwedge_{Act\in RootActivities}^{\forall RootActivities} Act \neq Executing\right)\right)\right)$$

Application

A pilot performing the before landing checklist of an aircraft

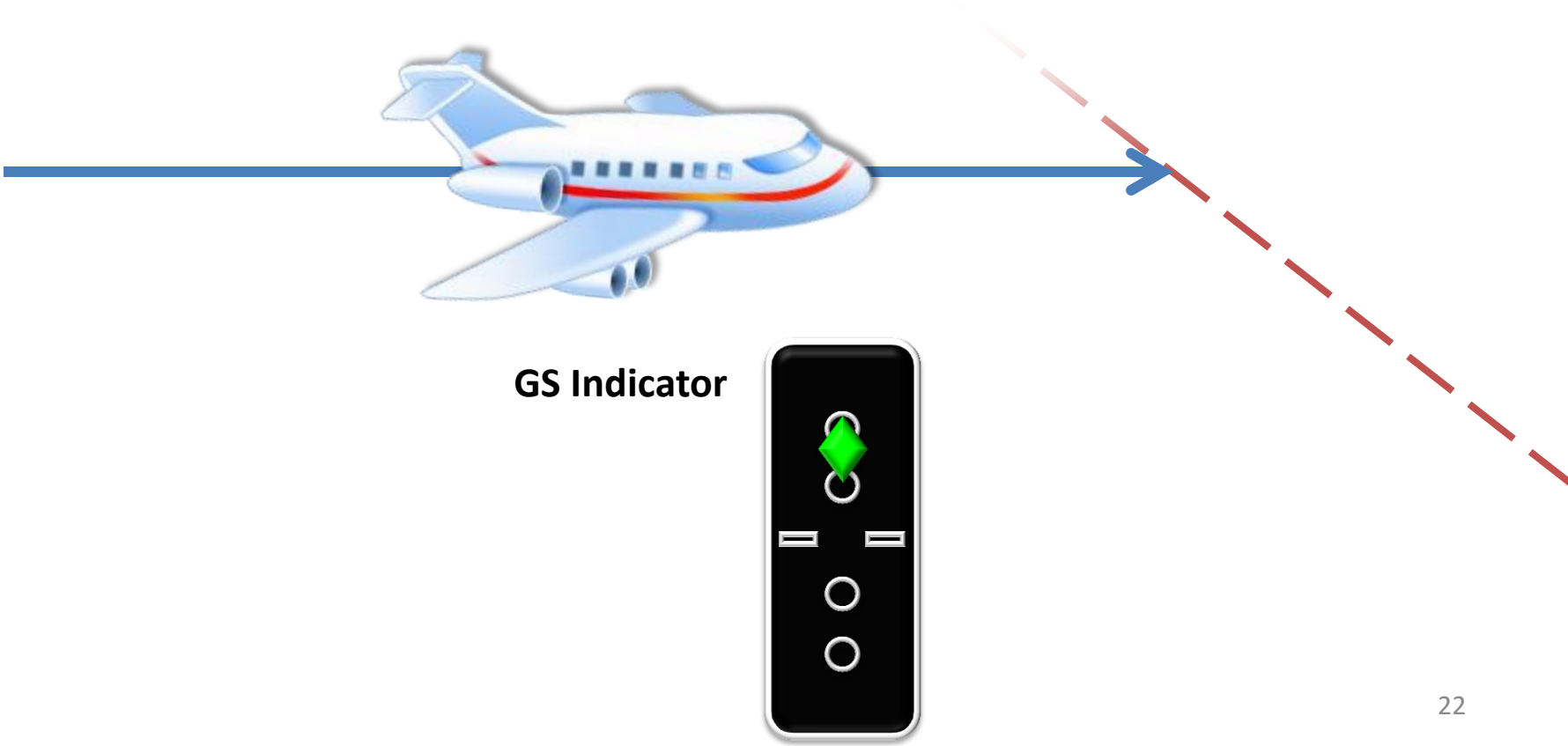


GS Indicator



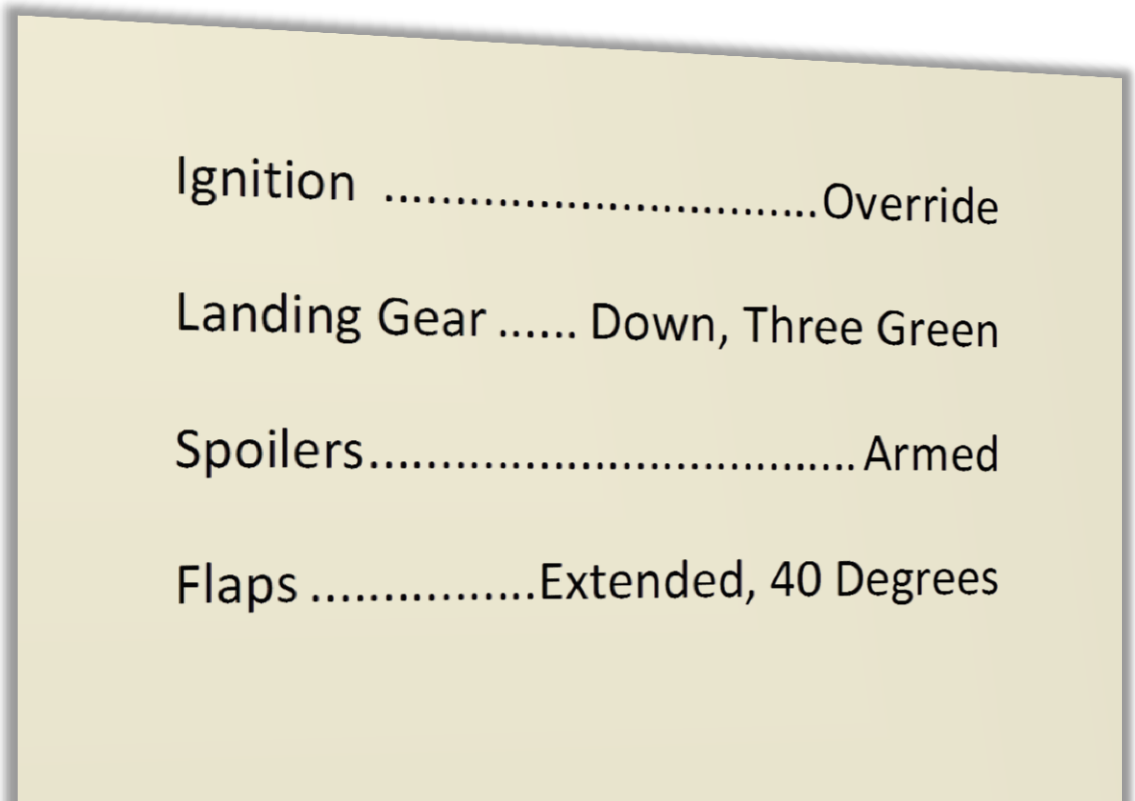
Application

A pilot performing the before landing checklist of an aircraft

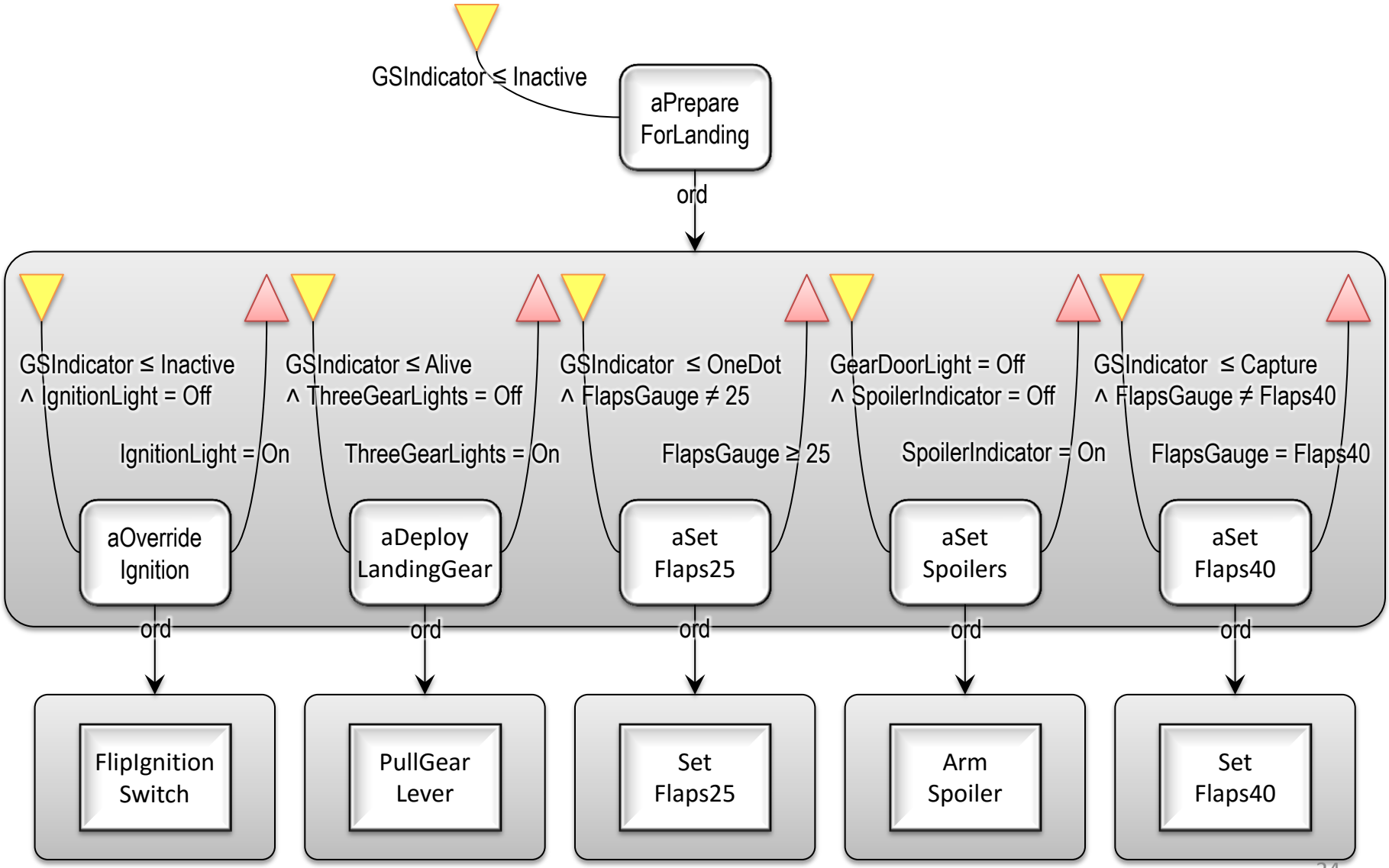


Application

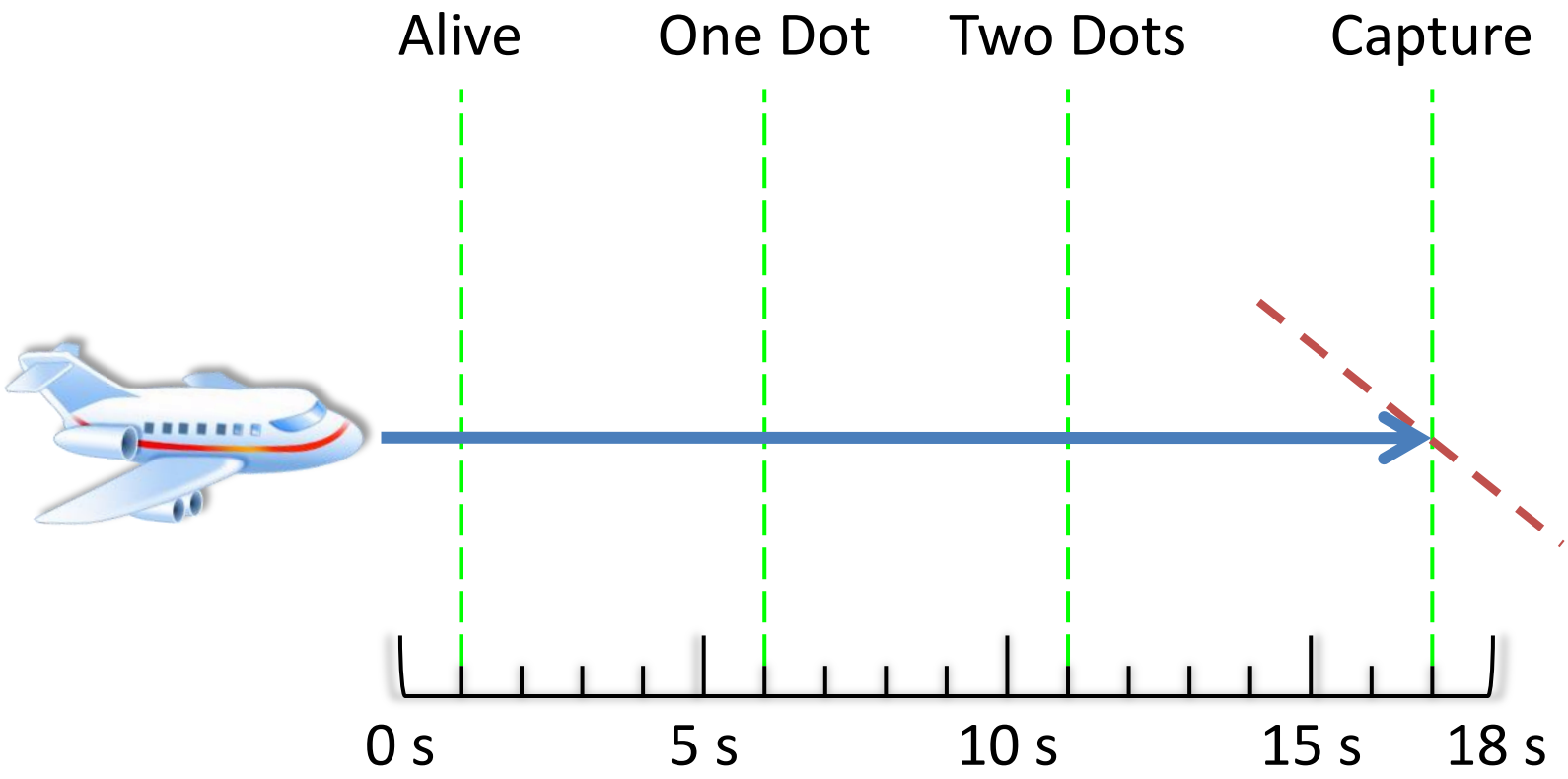
A pilot performing the before landing checklist of an aircraft



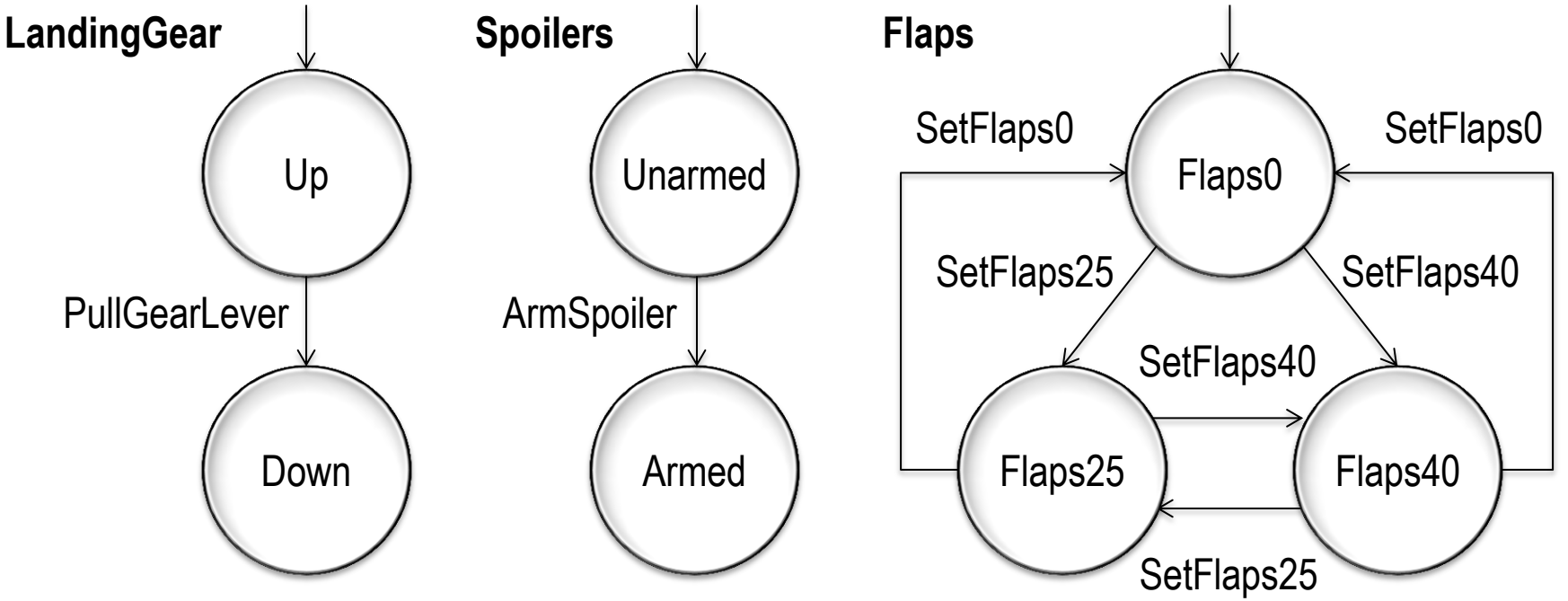
Human Task Behavior

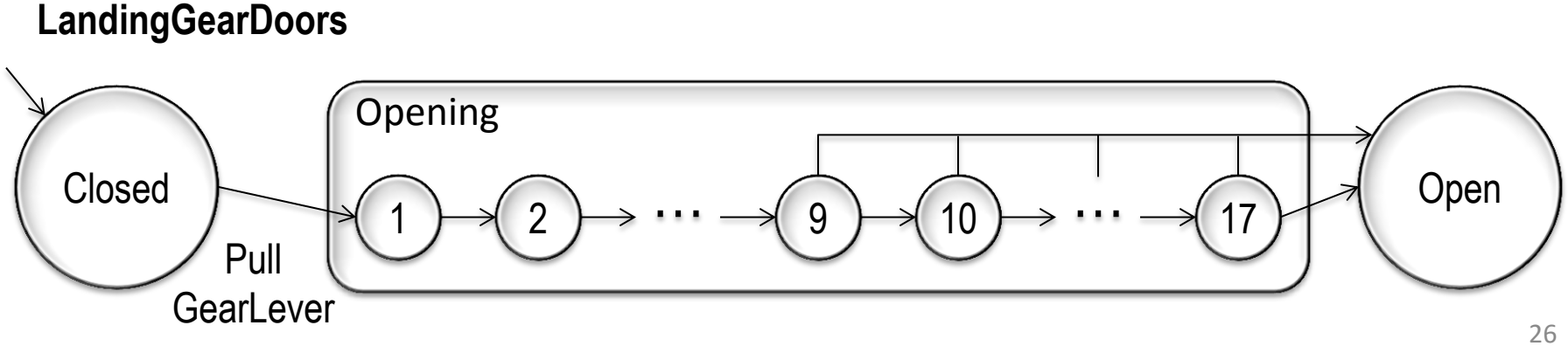


Environment



Automation


LandingGear
Up
PullGearLever
Down
Spoilers
Unarmed
ArmSpoiler
Armed
Flaps
SetFlaps0
Flaps0
SetFlaps0
SetFlaps25
SetFlaps40
SetFlaps40
Flaps25
Flaps40
SetFlaps25


LandingGearDoors
Closed
Pull
GearLever
Opening
1
2
9
10
17
Open

Human Interface

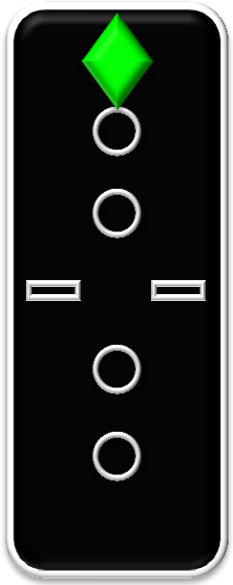
Ignition Switch

Gear Lever

Spoiler Lever

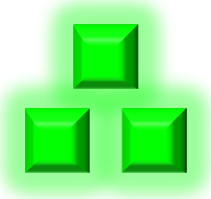
Flaps Selector

GS Indicator



Ignition Light

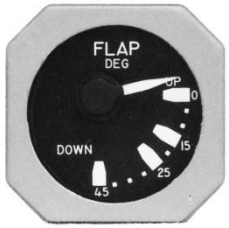
Three Gear Lights



Gear Door Light

Spoiler Indicator

Flaps Indicator



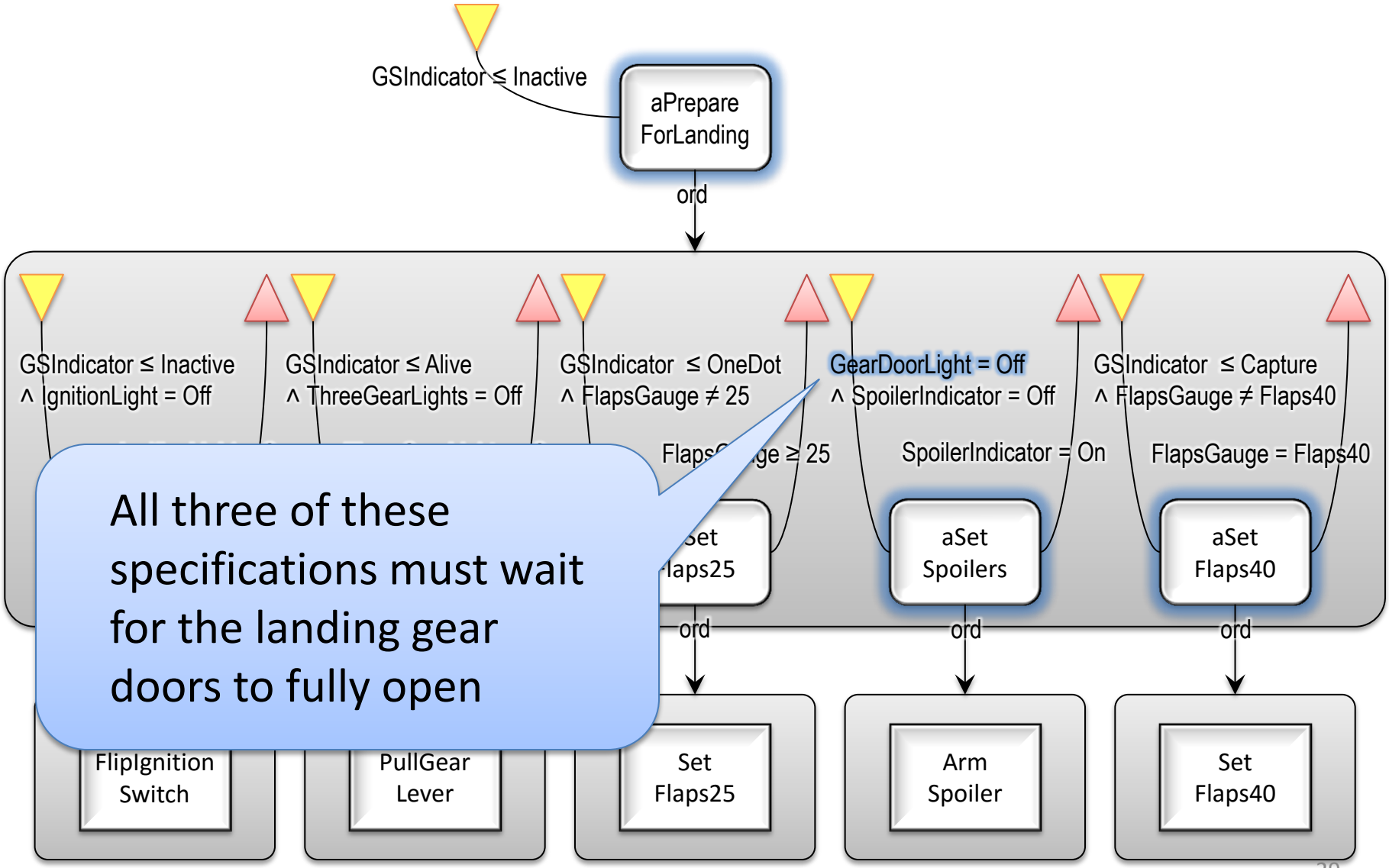
Verification Results

31 of the 34 properties generated the desirable result (total execution time = 14.6 seconds)

Inevitable Completability failed for three activities:

aPrepareForLanding aSetFlaps40 aSetSpoilers

Verification Results



This situation is dangerous…

- A failure to arm spoilers could result in an aircraft not staying on the runway
- Due to flaps settings, the airplane may be going to fast and overrun the runway

- The pilot may go off task
  - Attempt to deploy spoilers manually
  - Attempt to arm spoilers earlier

Contributions

- A novel method for discovering human-automation interaction issues that need not be anticipated by analysts
- Helps analysts ensure humans will trust the system because it will always support their task goals

Part of a Larger Infrastructure

Accounting for human cognitive and sensory limitations

Generating erroneous human behavior

Modeling human team behavior with communication and coordination

Generating miscommunications

Automatically creating functional interface designs from task models

Questions?

References

1. Ait-Ameur, Y., Baron, M., and Girard, P. (2003). Formal validation of HCI user tasks. In Proceedings of the International Conference on Software Engineering Research and Practice, pages 732–738, Las Vegas. CSREA Press.
2. Bolton, M. L. (ND). Automatic validation and failure diagnosis of human-device interfaces using task analytic models and model checking. Computational and Mathematical Organization Theory. DOI 10.1007/s10588-012-9138-6.
3. Bolton, M. L. and Bass, E. J. (2009). A method for the formal verification of human interactive systems. In Proceedings of the 53rd Annual Meeting of the Human Factors and Ergonomics Society, pages 764–768, Santa Monica. HFES.
4. Bolton, M. L. and Bass, E. J. (2010a). Formally verifying human-automation interaction as part of a system model: Limitations and tradeoffs. Innovations in Systems and Software Engineering: A NASA Journal, 6(3):219–231.
5. Bolton, M. L. and Bass, E. J. (2010b). Using task analytic models to visualize model checker counterexamples. In Proceedings of the 2010 IEEE International Conference on Systems, Man, and Cybernetics, pages 2069–2074, Piscataway. IEEE.
6. Bolton, M. L. and Bass, E. J. (2012). Using model checking to explore checklist-guided pilot behavior. International Journal of Aviation Psychology, 22(4):343–366.
7. Bolton, M. L., Bass, E. J., and Siminiceanu, R. I. (2012). Using phenotypical erroneous human behavior generation to evaluate human-automation interaction using model checking. International Journal of Human-Computer Studies, 70(11):888–906.
8. Bolton, M. L., Bass, E. J., and Siminiceanu, R. I. (2013). Using formal verification to evaluate humanautomation interaction in safety critical systems, a review. IEEE Transactions on Systems, Man and Cybernetics: Systems, 43(3):488–503.
9. Bolton, M. L., Siminiceanu, R. I., and Bass, E. J. (2011). A systematic approach to model checking human-automation interaction using task-analytic models. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 41(5):961–976.
10. Clarke, E. M., Grumberg, O., and Peled, D. A. (1999). Model checking. MIT Press, Cambridge.
11. De Moura, L., Owre, S., and Shankar, N. (2003). The SAL language manual. Technical Report CSL-01-01, Computer Science Laboratory, SRI International, Menlo Park.
12. Degani, A. (2004). Taming HAL: Designing interfaces beyond 2001. Macmillan, New York.
13. Emerson, E. A. (1990). Temporal and modal logic. In van Leeuwen, J., Meyer, A. R., Nivat, M., Paterson, M., and Perrin, D., editors, Handbook of Theoretical Computer Science, chapter 16, pages 995–1072. MIT Press, Cambridge.
14. Fields, R. E. (2001). Analysis of Erroneous Actions in the Design of Critical Systems. PhD thesis, University of York, York.
15. Kebabjian, R. (2012). Accident statistics. http://www.planecrashinfo.com/cause.htm. Accessed 3/25/2013.
16. Kenny, D. J. (2011). 22nd joseph t. nall report: General aviation accidents in 2010. Technical report, AOPA Air Safety Institute.
17. Kirwan, B. and Ainsworth, L. K. (1992). A Guide to Task Analysis. Taylor and Francis, London.
18. Maluf, D. A., Gawdiak, Y. O., and Bell, D. G. (2005). On space exploration and human error: A paper on reliability and safety. In Proceedings of the 38th Annual Hawaii International Conference on System Sciences, pages 79–84, Piscataway. IEEE.
19. Mansouri-Samani, M., Pasareanu, C. S., Penix, J. J., Mehlitz, P. C., OMalley, O., Visser, W. C., Brat, G. P., Markosian, L. Z., and Pressburger, T. T. (2007). Program model checking: A practitioners guide. Technical report, Intelligent Systems Division, NASA Ames Research Center, Moffett Field.
20. Mitchell, C. M. and Miller, R. A. (1986). A discrete control model of operator function: A methodology for information display design. IEEE Transactions on Systems Man Cybernetics Part A: Systems and Humans, 16(3):343–357.
21. Palanque, P. A., Bastide, R., and Senges, V. (1996). Validating interactive system design through the verification of formal task and system models. In Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, pages 189–212, London. Chapman and Hall, Ltd.
22. Paterno, F., Santoro, C., and Tahmassebi, S. (1998). Formal model for cooperative tasks: Concepts and an application for en-route air traffic control. In Proceedings of the 5th International Conference on the Design, Specification, and Verification of Interactive Systems, pages 71–86, Vienna. Springer.
23. Reason, J. (1990). Human Error. Cambridge University Press, New York.
24. Sandler, C., Badgett, T., and Thomas, T. M. (2004). The Art of Software Testing. John Wiley & Sons.
25. Schraagen, J. M., Chipman, S. F., and Shalin, V. L. (2000). Cognitive Task Analysis. Lawrence Erlbaum Associates, Inc., Philadelphia.
26. Sheridan, T. B. and Parasuraman, R. (2005). Humanautomation interaction. Reviews of human factors and ergonomics, 1(1):89–129.
27. Silberschatz, A., Galvin, P. B., and Gagne, G. (2009). Operating system concepts. J. Wiley & Sons.
28. Wing, J. M. (1990). A specifier's introduction to formal methods. Computer, 23(9):8, 10–22, 24.