Verification Models for Advanced Human-Automation Interaction in Safety Critical Flight Operations

TU Delft Delft University of Technology

University at Buffalo The State University of New York

IXION industry and aerospace

Agenda

- Project Introduction
- HAI Formal Verification Methodology Proposed
- Formal Methodology Application to Test Cases
  - ESTRACK G/S Control System
  - UAV G/S Control System
- Project Conclusions
- Future Work Identified

Project Introduction

ES6967 - Verification Models for AdvancedHuman-Automation Interaction inSafety Critical Flight Operations

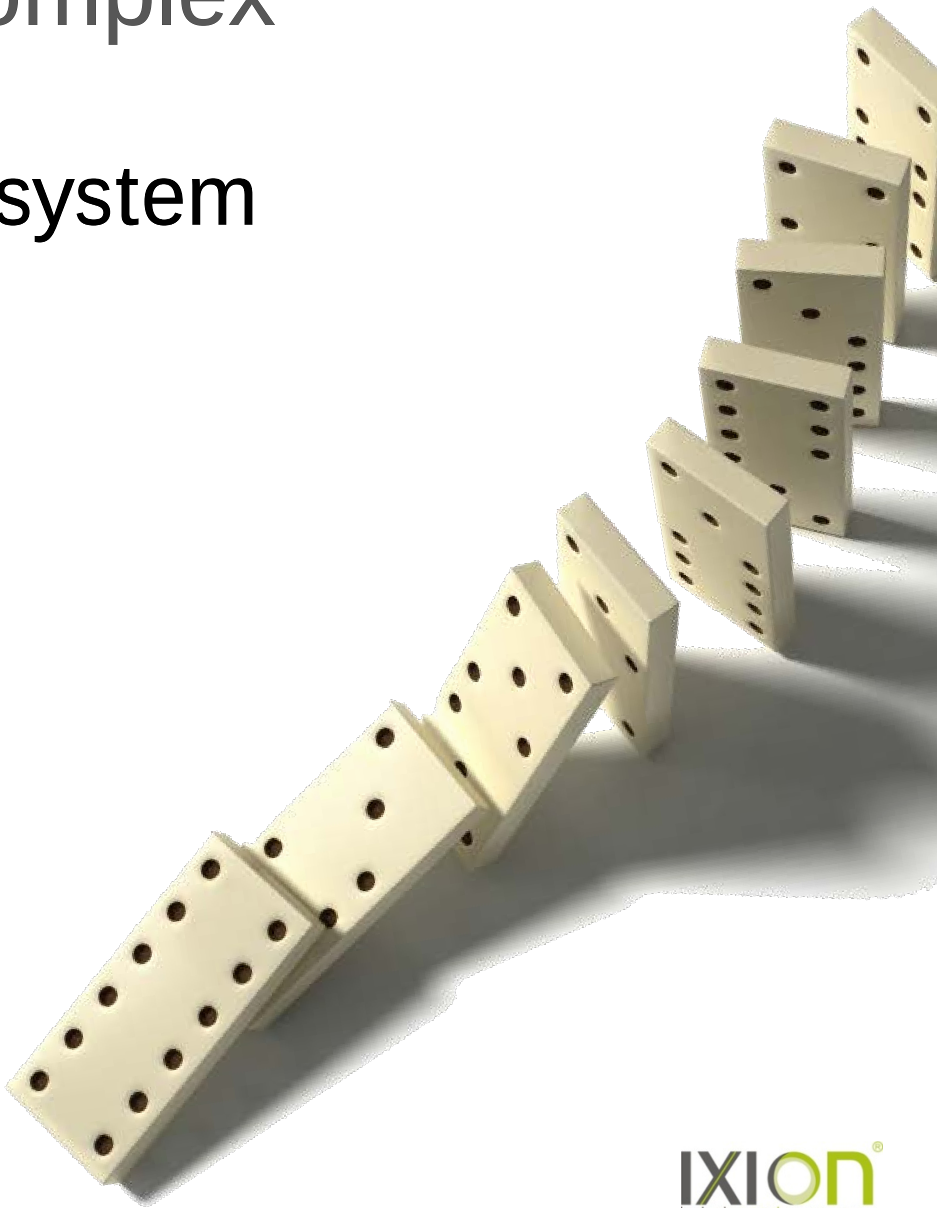
- TEAM:
  - IXION Industry and Aerospace, Madrid – Spain (N. Jimenez)
  - Delft University of Technology, Delft – The Netherlands (M. van Paassen)
  - University of Illinois – Chicago campus / State University of New York – Bufffalo (M. Bolton)

- ESA project officers
  - M. Trujillo
  - Y. Yushtein – R. Peldszus

IXION industry and aerospace

System Failure is Complex

Interactions between system components results in breakdowns

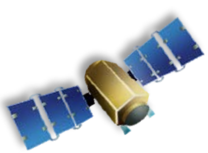


Human-automation Interaction (HAI):

A major contributor to failures in safety critical systems

75.5 % of accidents in general aviation

~ 50 % of accidents in commercial aviation



Many high profile accidents in space operations

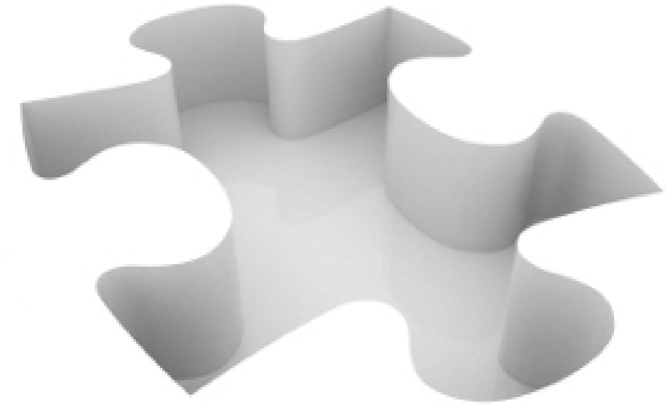
IXION industry and aerospace

A Systems Problem:

Need to consider the human operator as an integral part of the system

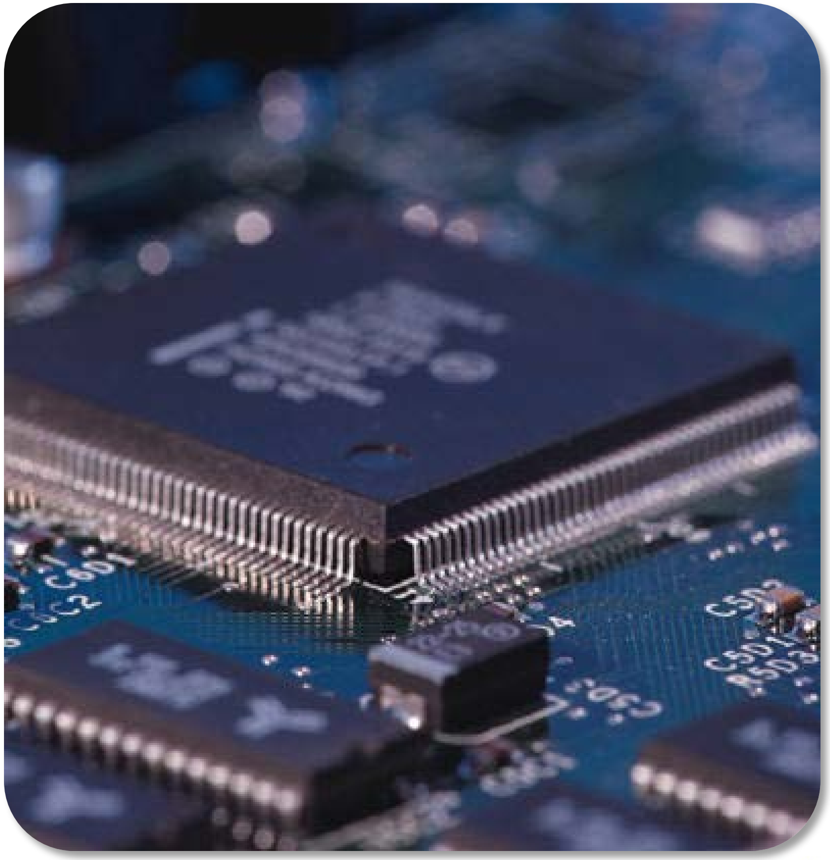
Evaluating System Safety with Human Behavior

- Experimentation and Testing: Human subject testing
- Modeling Expected Performance: Human performance modeling
- Simulation: Agent-based analyses
- Stochastic Analyses: Human reliability analysis
- Static Analyses: Searching interface models for preconditions to erroneous human behavior



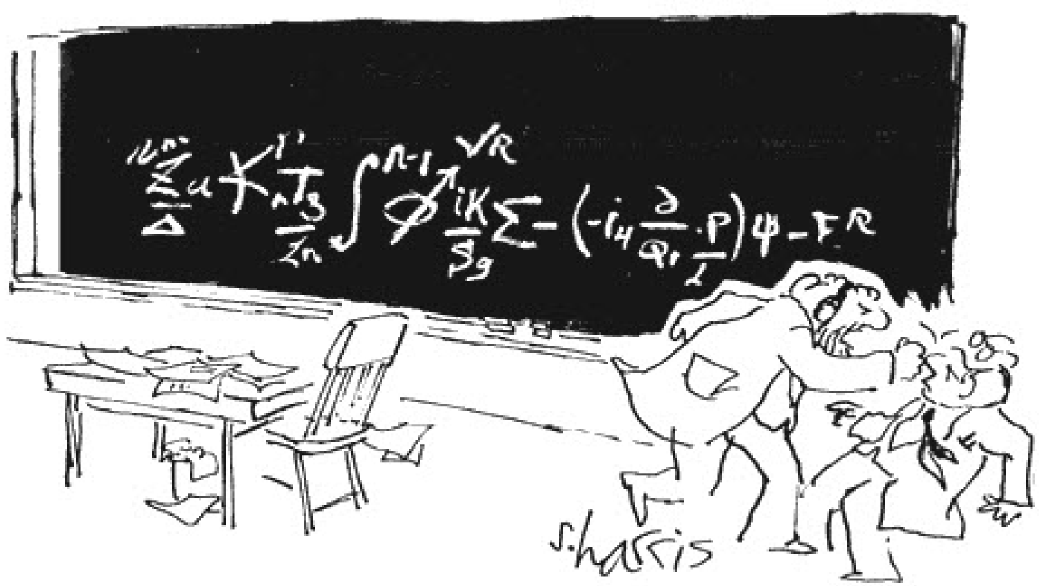
These techniques can miss human-system interactions that could lead to system failure

Computer hardware and software engineers have similar problems



Formal Methods:

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



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

Formal Methods:

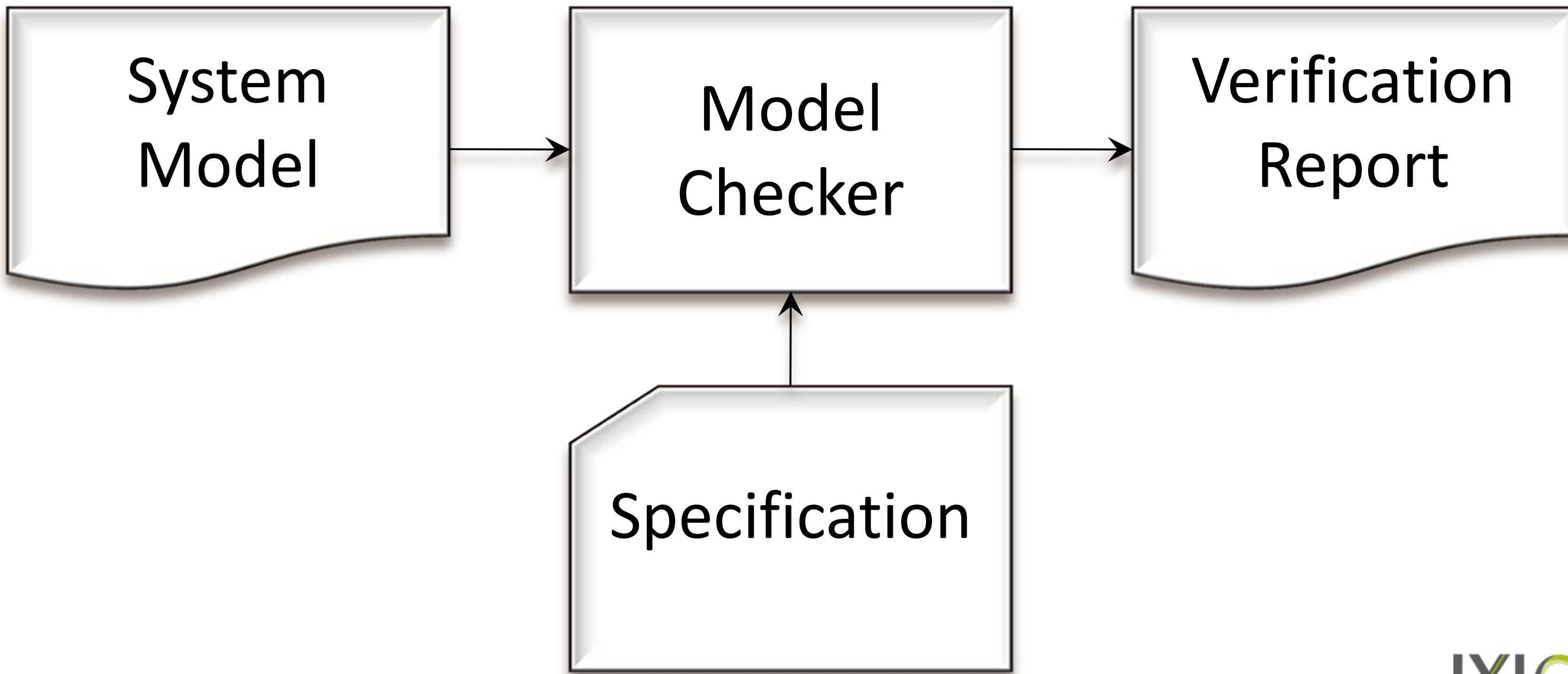
Tools and techniques for **proving** that a system will always perform as intended

- **Modeling** – Representing a system's behavior in a mathematical formalism
- **Specification** – Formally expressing a desirable property about the system
- **Verification** – Proving that the model adheres to the specification

IXION industry and aerospace

Model checking:

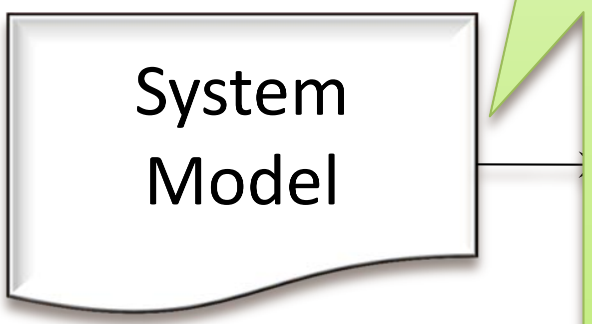
An automatic means of performing formal verification

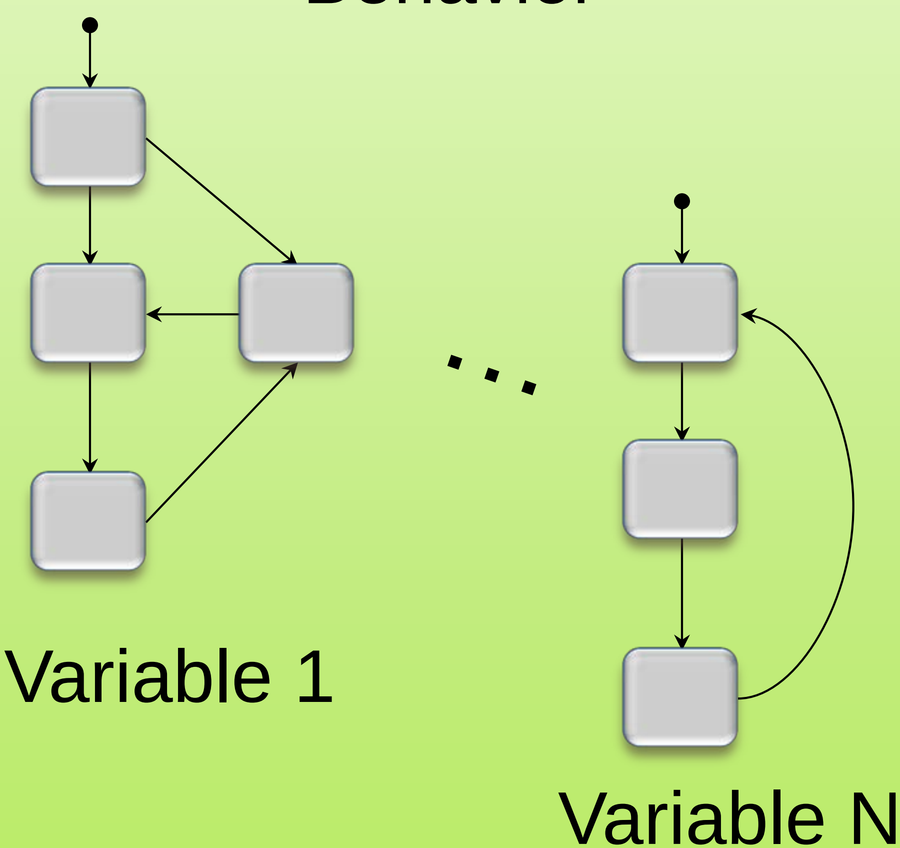


Model checki

An automatic
formal verific

A Finite State Machine Model Represents System Behavior





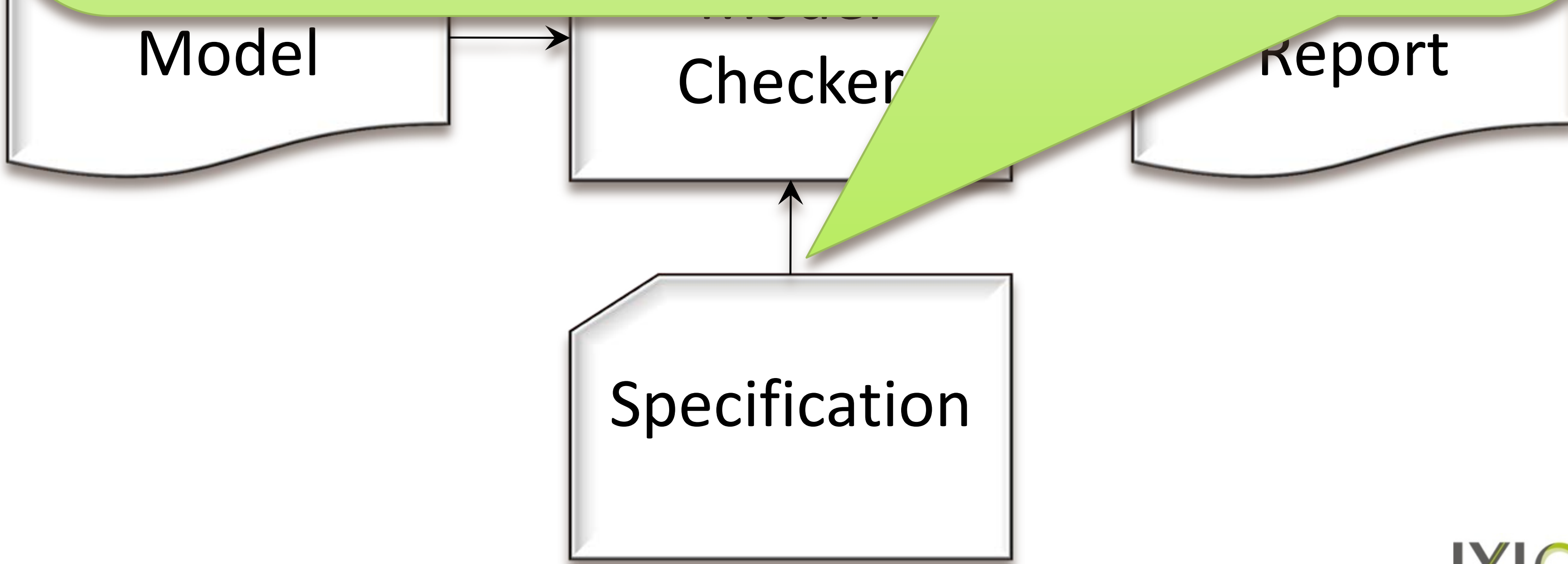
A Temporal Logic Specification Property Asserts Desirable Qualities About the System

For example: “The system should never reach unsafe state X”

$$G \neg (X)$$

Or, “The system should always eventually reach state Y”

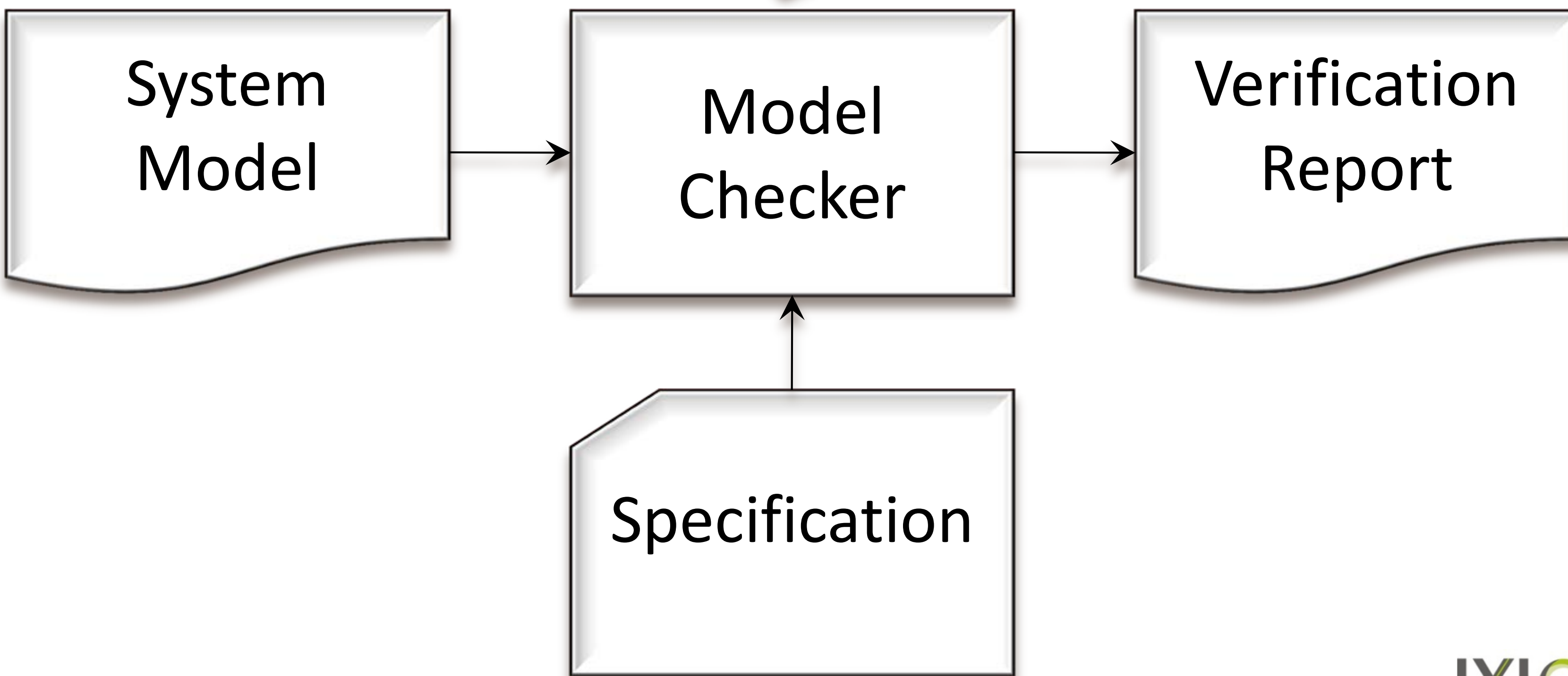
$$F\ (Y)$$



Model chec

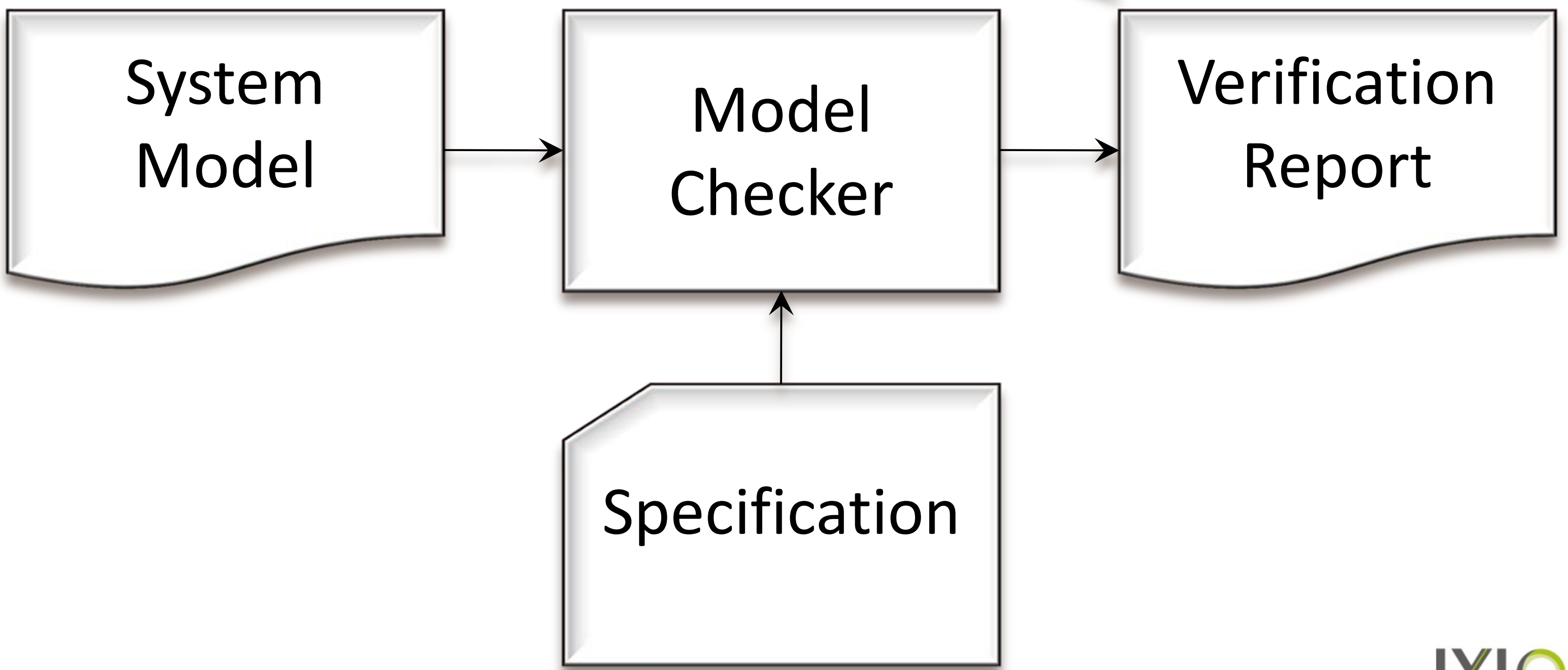
An automati
formal verifi

A model checker “searches” through the model’s statespace looking for violations

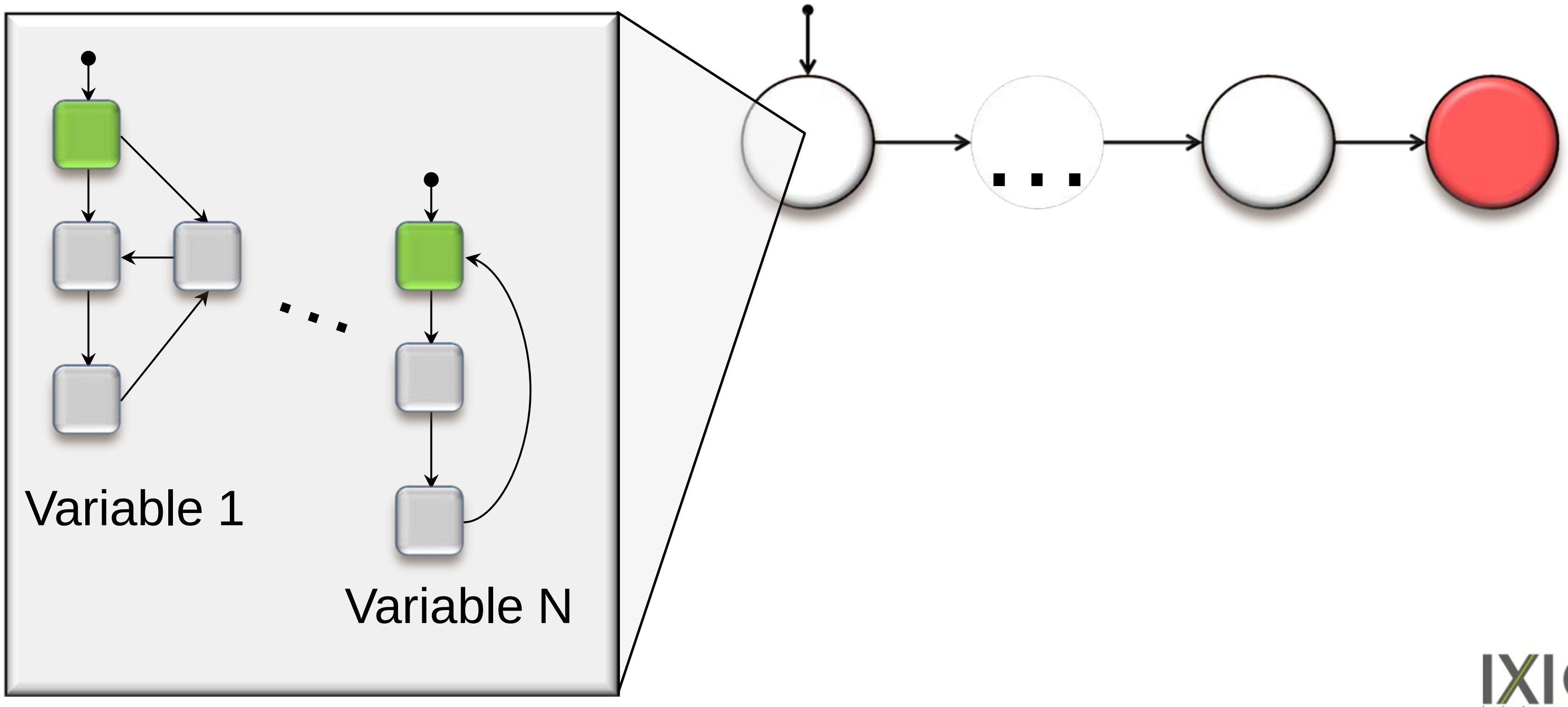


A confirmation or counterexample is returned

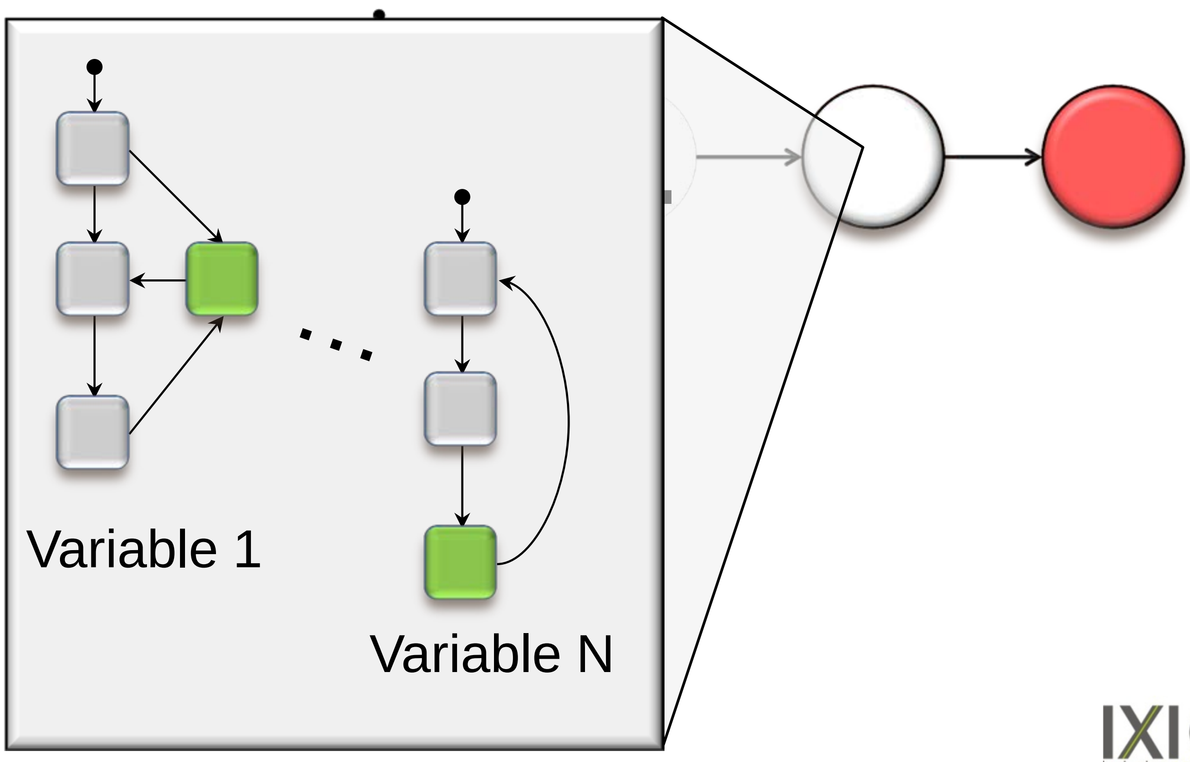
s of performing



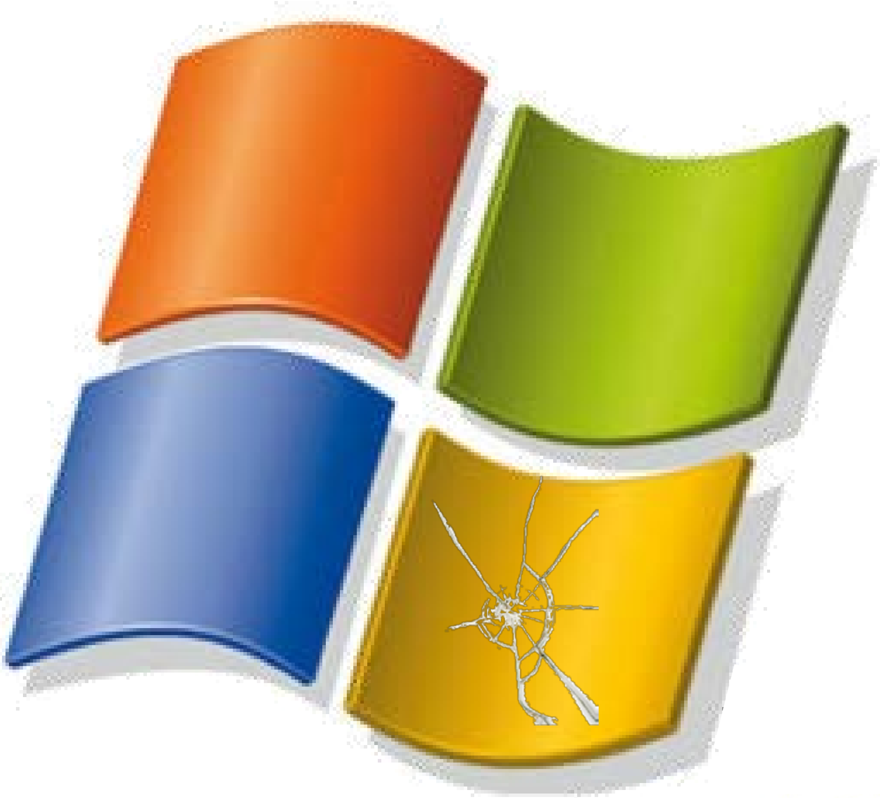
A sequence of states that led up to a violation



A sequence of states that led up to a violation



Model Checking Really Works!!



Formal methods for HAI

- Analysis of human-automation interfaces looking for usability problems and potential mode confusion
  - Just using interface models
  - Paring interface and automation models with mental models
- Analyses of systems with models of human behavior looking for safety and performance failures
  - Human behavior modeled using cognitive architectures
  - Human behavior represented using task models

Formal methods for HAI

- Analysis of human-automation interfaces looking for usability problems and potential mode confusion
  - Just using interface models
  - Paring interface and automatio... els with mental models
- Analyses of systems wit... behavior looking for safe... failures
  - Human behavior modele... architectures
  - Human behavior represented using task models

> - Primarily concerned with interfaces
> - Looks for human error potential, not system safety

Formal methods for HAI

- Analysis of human-aut
for usability problems
confusion
  - Just using interface m
  - Paring interface and a
mental models
- Analyses of systems with ... of human
behavior looking for saf... and performance
failures
  - Human behavior modeled using cognitive architectures
  - Human behavior represented using task models

- Can include human error organically
- Can be very complex
- Architectures not widely used

Formal methods for HAI

- Analysis of human-aut[…] for usability problems […] confusion
  - Just using interface m[…]
  - Paring interface and a[…] mental models
- Analyses of systems w[…] behavior looking for safet[…] performance failures
  - Human behavior mode[…]d using cognitive architectures
  - Human behavior represented using task models

- Less complex
- More widely used
- Does not provide cognitive explanation
- Errors must be explicitly included

Formal methods fo

- Analysis of human-au
  for usability problems
  confusion
  - Just using interface mo
  - Paring interface and automa models with
    mental models
- Analyses of systems with models of human behavior looking for safety and performance failures
  - Human behavior modeled using cognitive architectures
  - Human behavior represented using task models

Require analysts to explicitly assert properties to be checked

HAI Formal Verification Methodology

Objective:

Use model check to prove systems are safe with both normative and erroneous human-automation interaction

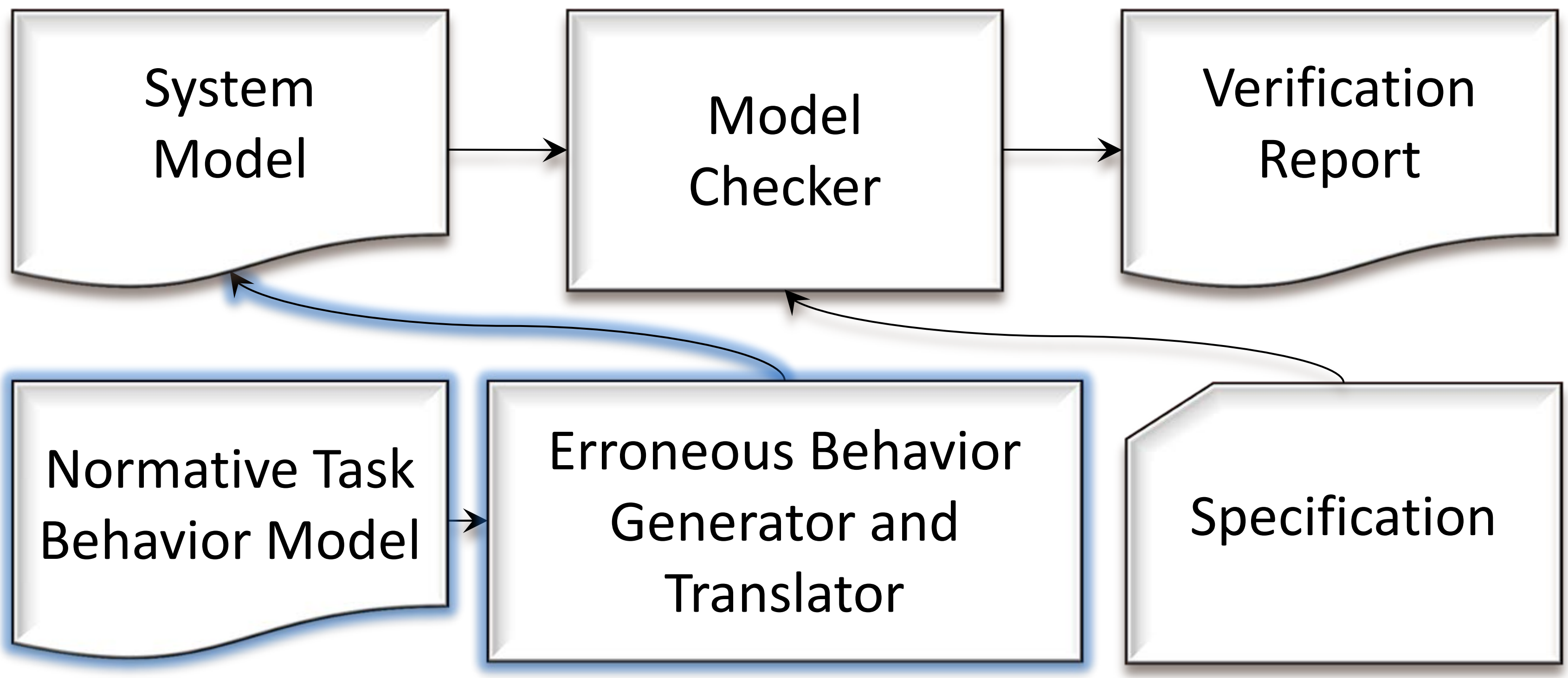
Objective:

Use model check to prove systems are safe with both **normative** and erroneous human-automation interaction

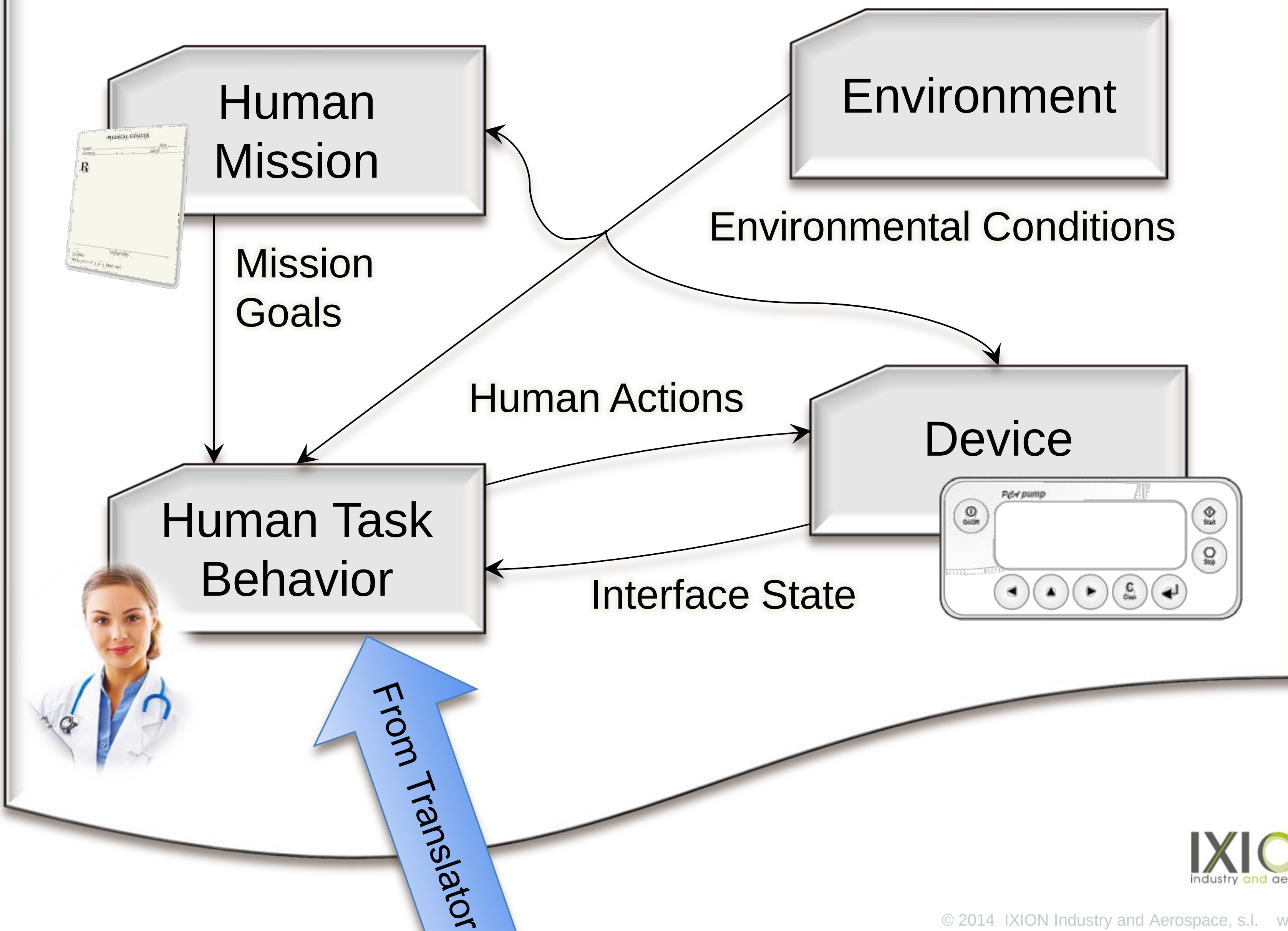
Objective:

Use model check to prove systems are safe with both normative and **erroneous** human-automation interaction

Model Checking with Human Task Behavior



System Model

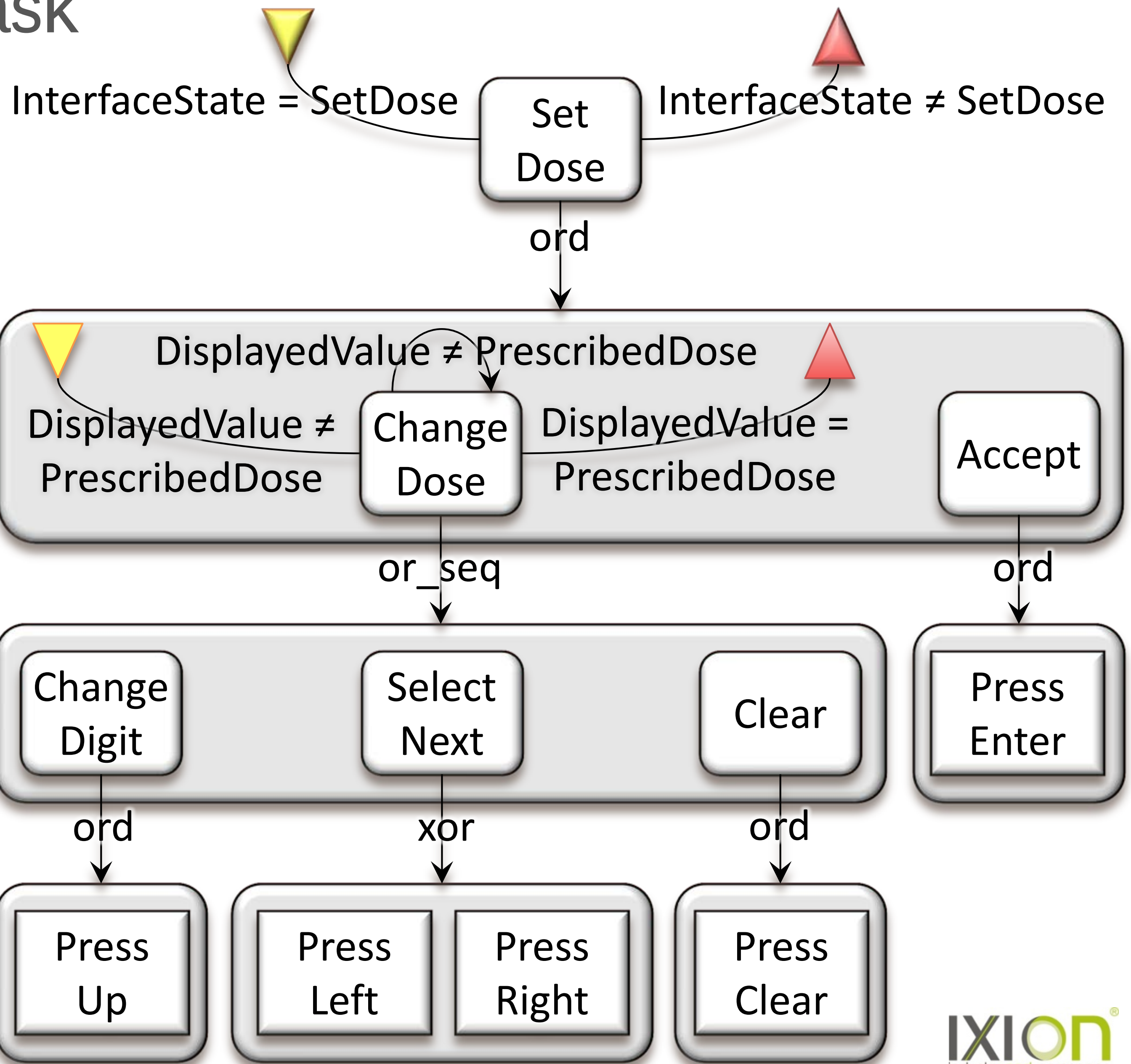


Enhanced Operator Function Model (EOFM)

A generic task analytic modeling formalism

- Formal semantics (and EOFM to SAL translator)
- Input output model
- Platform-independent
- XML notation
- Visual notation

Normative Task Behavior Modeling



But what if I make a mistake?

Two Methods for Erroneous Behavior Generation

- Bottom Up:
  Generating errors based on Hollnagels phenotypes of erroneous action
- Top Down:
  Generating errors based on Reason's slips

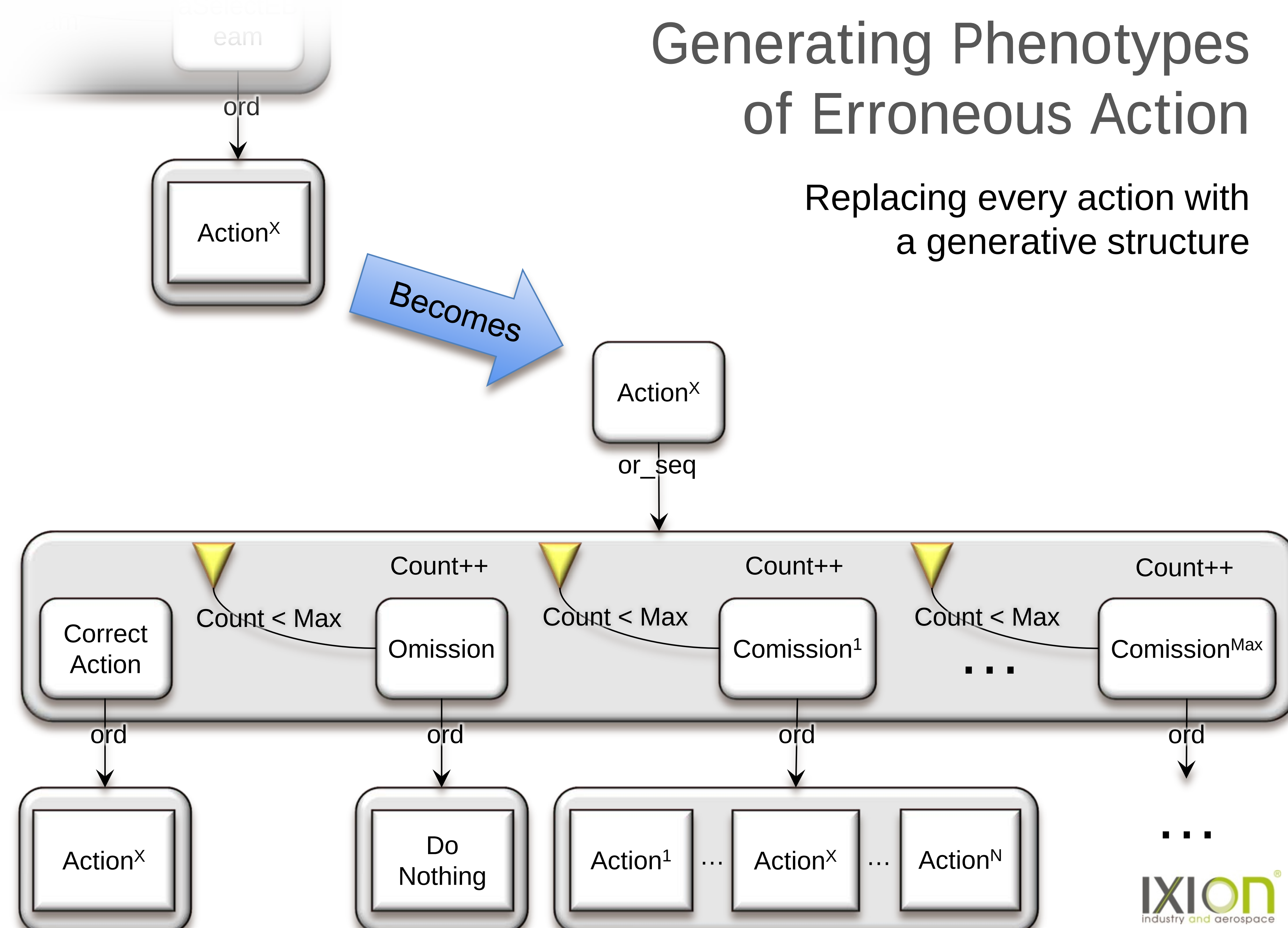
Erroneous Human Behavior

care of Eric Hollnagel

- Erroneous human behaviors can be classified based their **phenotypes**
  - Observable deviations from a normative plan of actions (a task)
- **All** erroneous behaviors (not related to timing) are composed of one or more “zero-order” phenotypes:
  - Omission
  - Jump (forward or backward)
  - Repetition
  - Intrusion

Generating Phenotypes of Erroneous Action

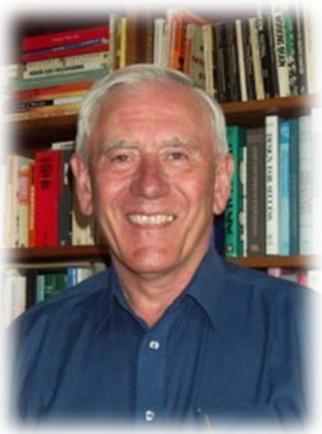
Replacing every action with a generative structure



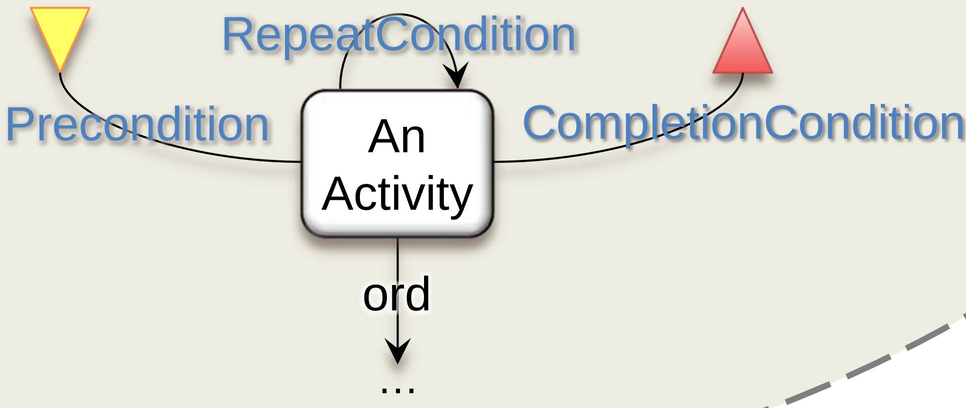
IXION industry and aerospace

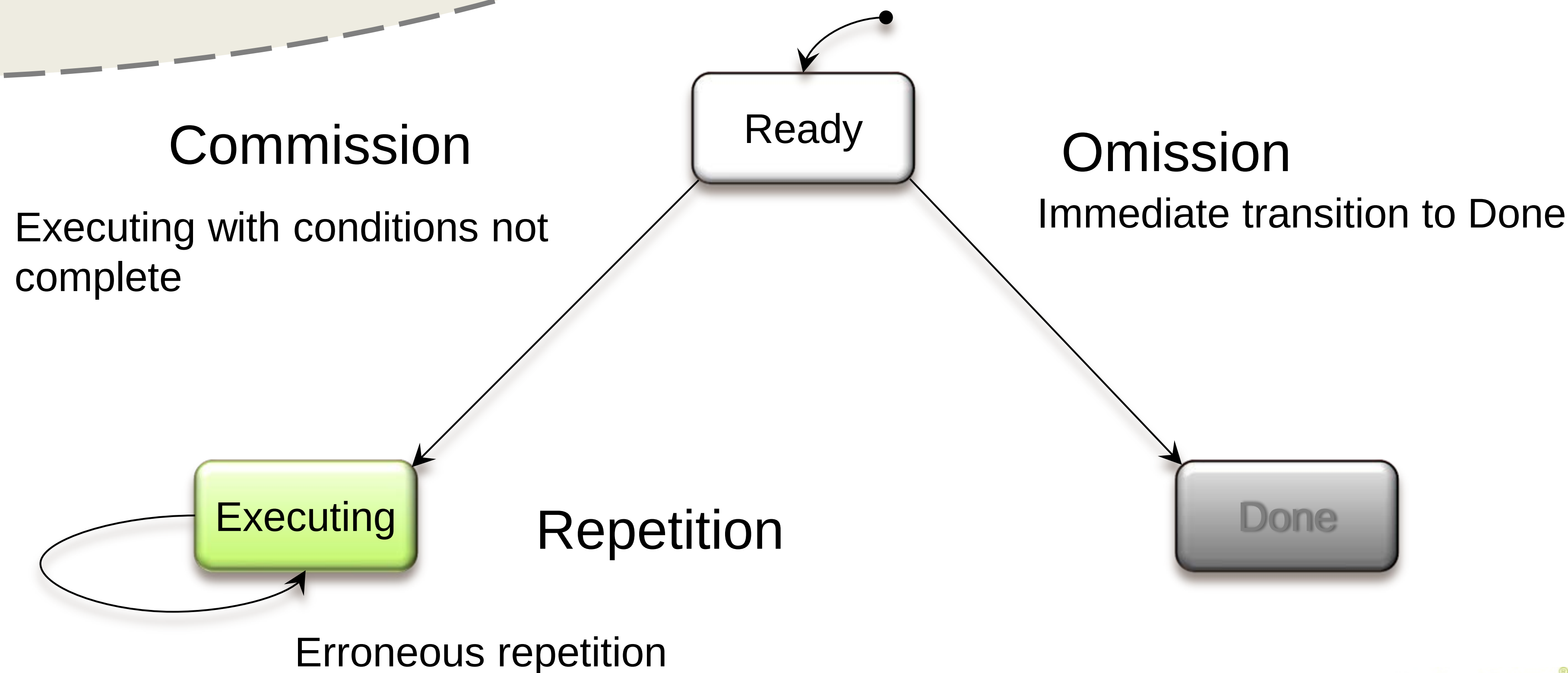
Reason's Slips: Failures of Attention

A person's inability to properly attend to the situation can cause them to perform a task erroneously



Generating Slips





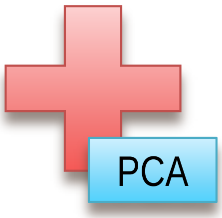
Constraining erroneous behavior:

Max = Maximum # of erroneous transitions

Count = Total # of erroneous transitions made

Erroneous transition can only occur if
Count < Max

Successful Application



PCA Pump

Verified that a correct prescription is always administered with normative and erroneous (slips) behavior

Automobile Cruise Control

Verified a red light would not be overrun with normative behavior

Instrument Landing Checklist Procedure

Verified that a before-landing checklist procedure would always prepare the aircraft for landing with normative behavior

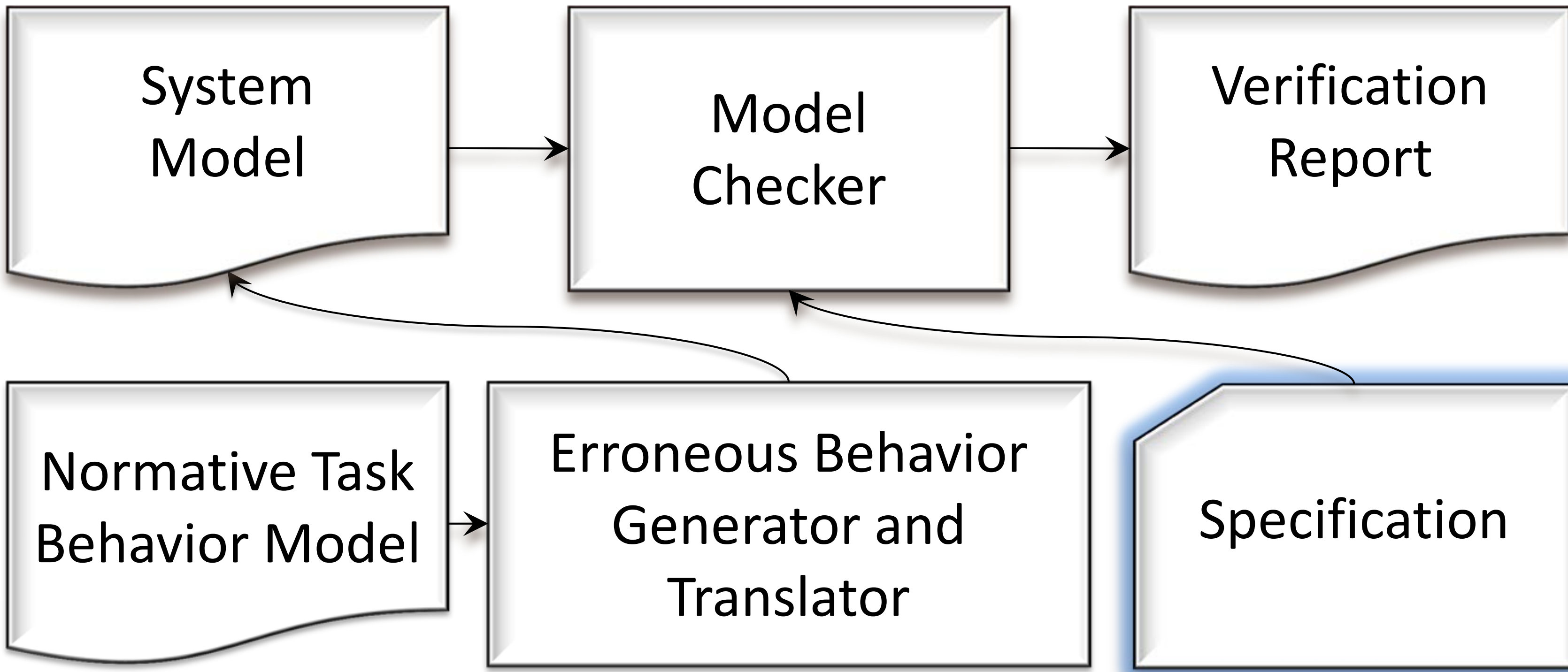
Radiation Therapy Machine

Verified that the machine would not irradiate patients with normative and erroneous (phenotypes) behavior

IXION industry and aerospace

Limitation:

The analysts must know what system safety properties they want to verify and formulate them as specification properties

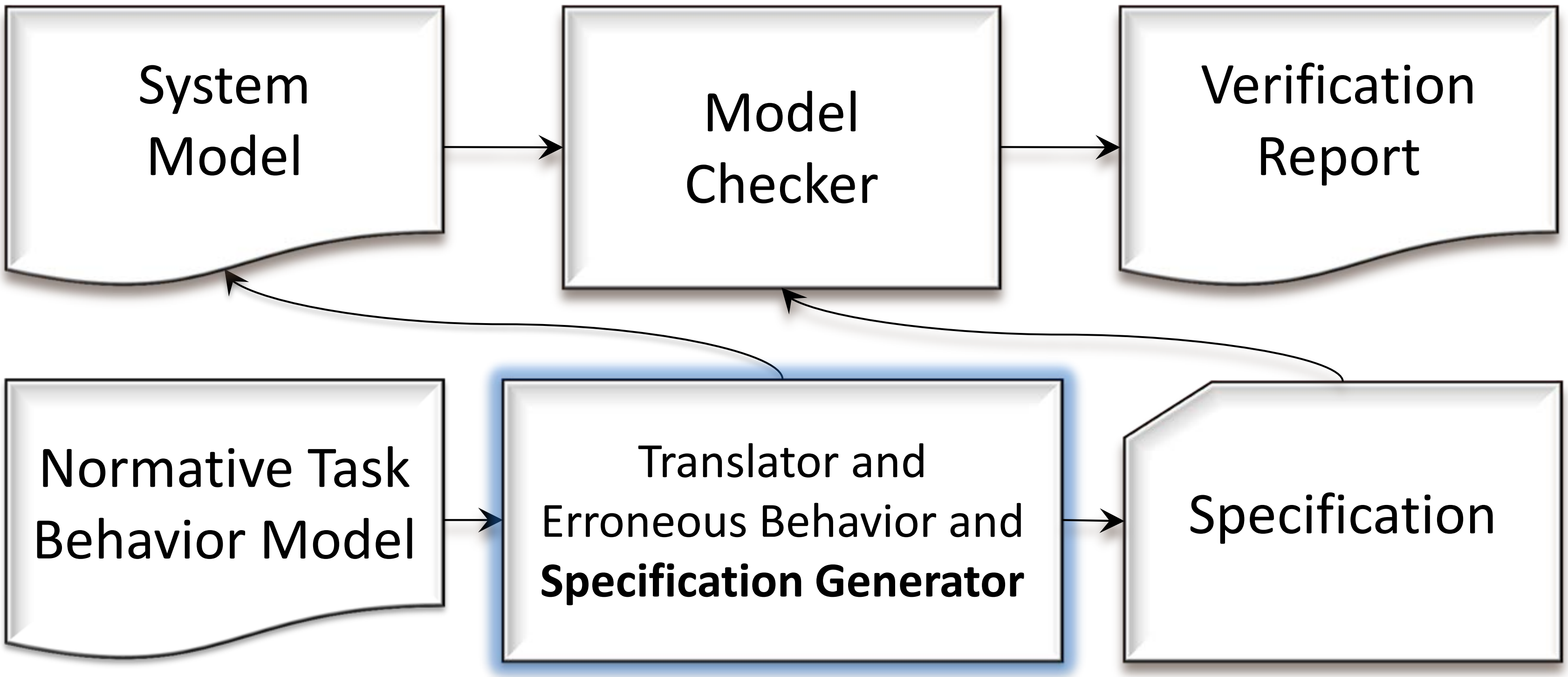


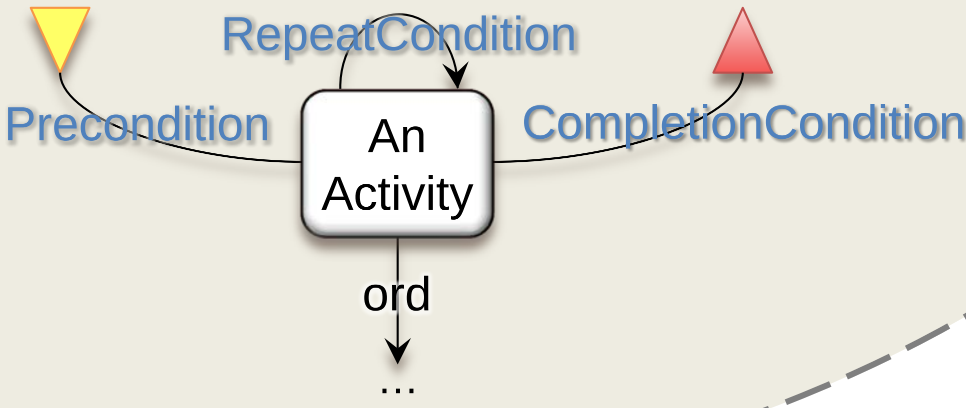
This is a problem because ...

- Specification notations can be difficult to learn, interpret, and use
- Analysts may not know what to check for
- Specifications are asserted in terms of failure outcomes and not their causes

Added Objective:

Specification properties are automatically generated from normative task behavior models





So what can we check for

- Human action properties
  - Startable, repeatable, finishable, skippable, completable, inevitable completability
- Interaction
  - Liveness

Interpreting Verification Results

- There are interrelationships between the properties
- Multiple reachability properties must be examined to get a full understanding of why a failure occurred
  - Examine the state coverage properties of each activity
  - Isolate the activity where the problem is originating
  - Use decision coverage properties to identify what strategic knowledge is associated with the failure
- The counterexample visualizer can be used to evaluate failures that produce counterexamples
- The report materials describe how to interpret the results more deeply

Implementation

- Modified the EOFM to SAL translator to automatically generate specification properties
- Note, generated specification properties **can** be used with other specifications (like safety properties)
- Generated specification properties **cannot** be used with erroneous behavior generation

Testing

- Artificial test cases were used to ensure that the generated properties would detect the desired conditions and not “false alarm”
- Generation was used to successfully evaluate and existing aerospace test case (before landing checklist procedure)
- The full method was used to evaluate two realistic test cases (discussed subsequently)

IXION industry and aerospace

Contributions to method

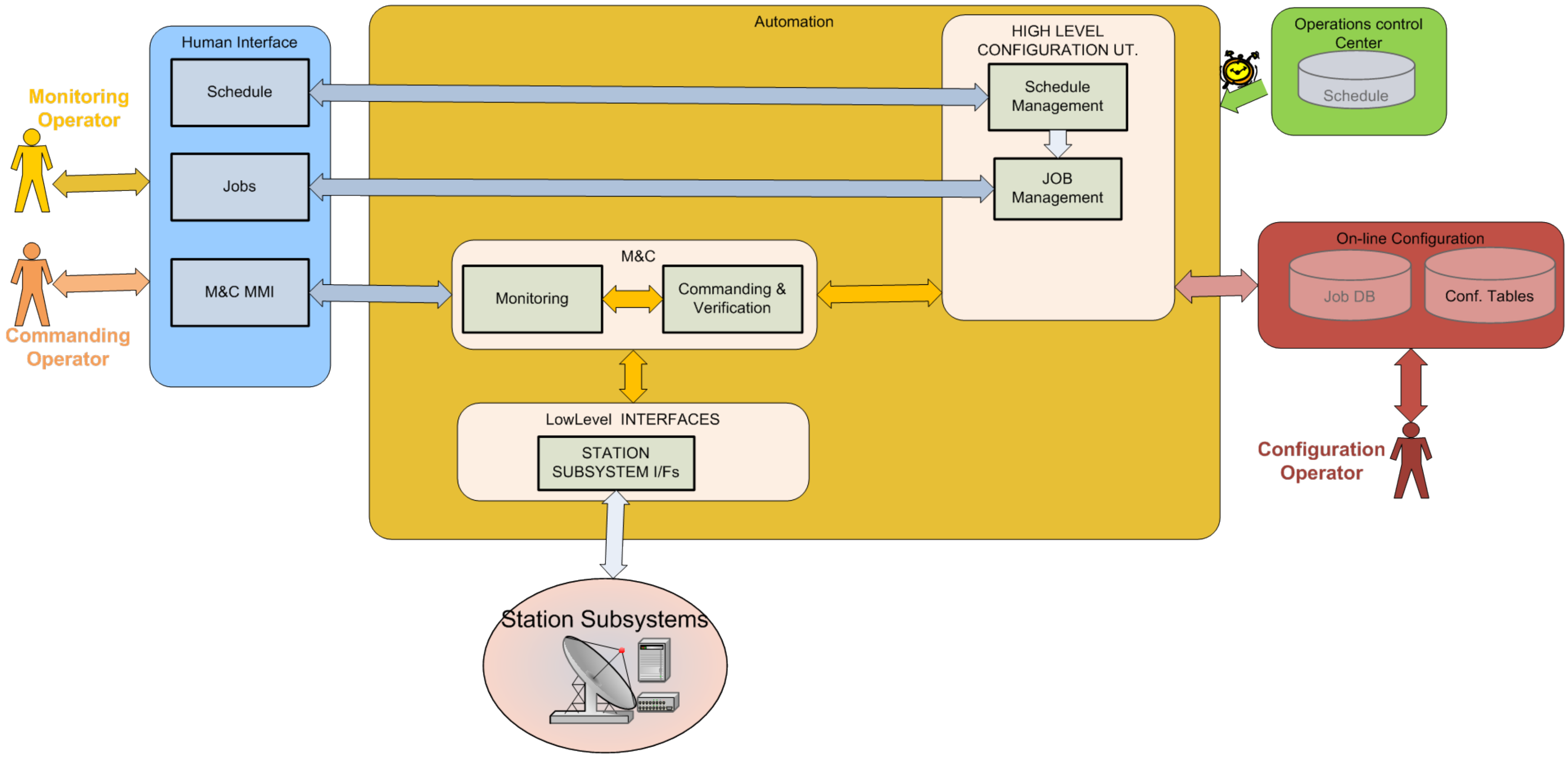
- An extension of the EOFM-supported infrastructure for formally verifying system safety with task analytic human behavior models
- A novel method for automatically generating specification properties from task analytic models
- The means to automatically check the system for human-system interaction problems using model checking

ESTRACK G/S Control System Test Case

CSMC: Monitoring & Control of Kiruna G/S

- Multi-antenna & multi-mission operations
- Fully automated: operation supervised from ESOC
- Permits Local and Remote (ESOC) Manual Operations
- Different Human roles

Architecture

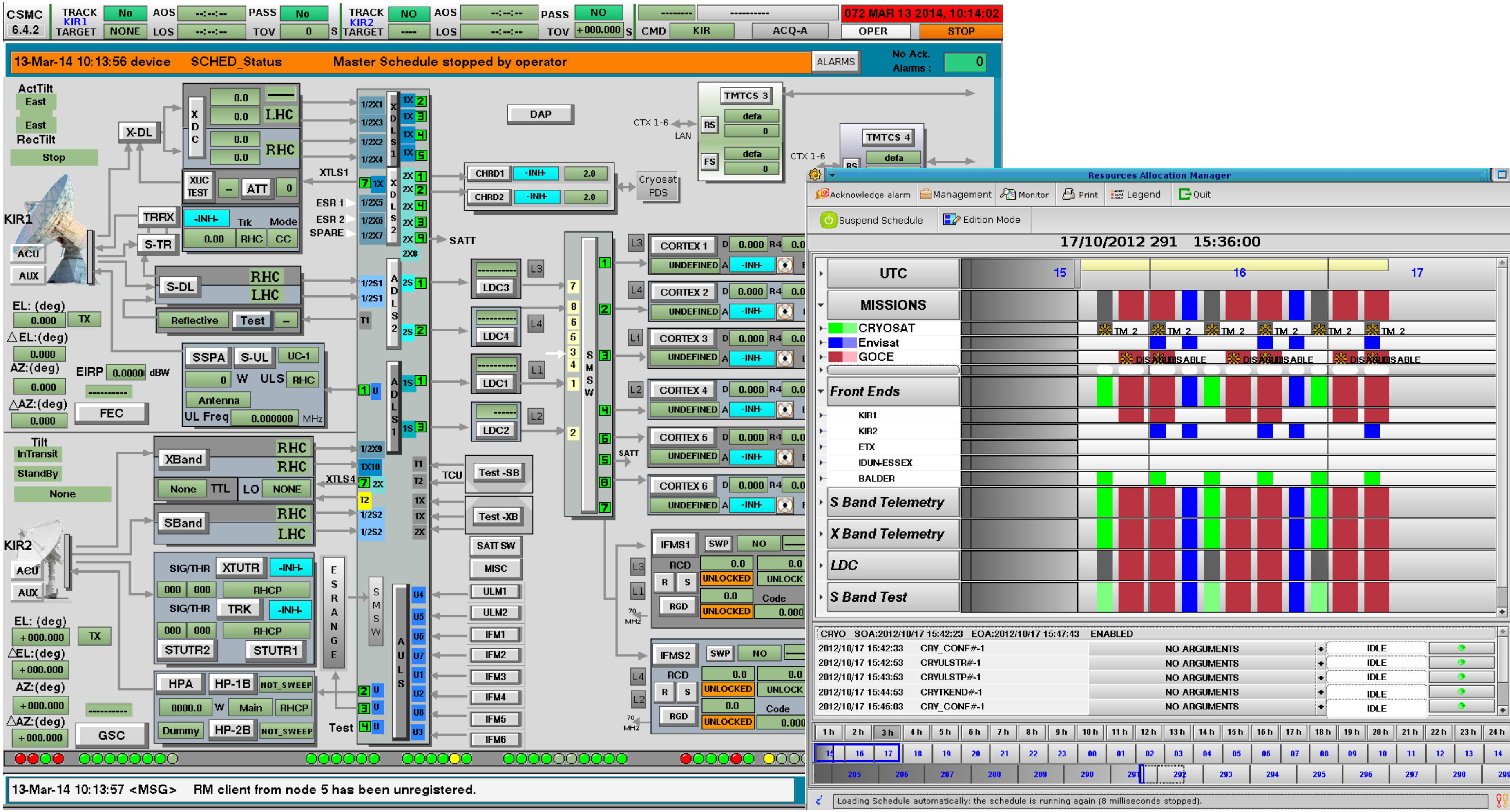


TU Delft – Delft University of Technology
University at Buffalo – The State University of New York
IXION industry and aerospace

Operation Concepts

- **Resources**: G/S elements than can be allocated to different missions to provide a service. Resources have attributes:
  - **Availability**: can it be used at all?
  - **Compatibility**: Can this mission/configuration use it?
  - **Allocation**: is being used now? How? (Yes/No, DL/UL, DL+UL…)
- **Activity/Pass**: Scheduled sequence of actions required to provide a service
- **Jobs**: predefined sequence of commands and their execution logic conditions
- **RAP:** Resource Allocation Plan that contains the timeline for passes
- **RAM**: Resource Allocation Manager. Allocates the resources and executes the schedule automatically.

User Interface



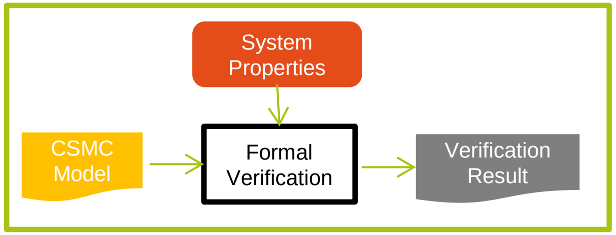
G/S View

Schedule View

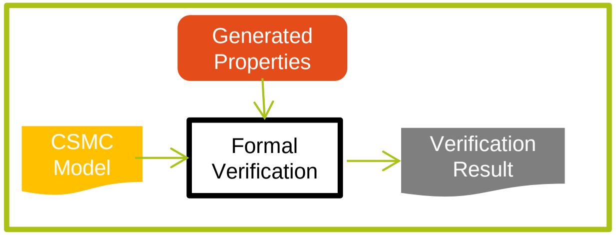
CSMC HAI Validation Approach

❑ HAI validation in nominal scenario

- Will the system achieve its goals? Is system safe?

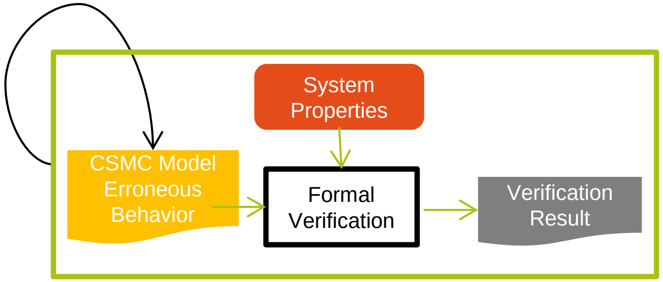


- Are there any unknown HAI erroneous conditions?



❑HAI validation in case of operator error

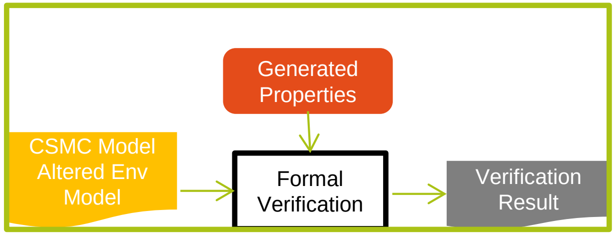
- Will the system be still safe in case of operator error?
- How many operator errors can the system support?



CSMC HAI Validation Approach

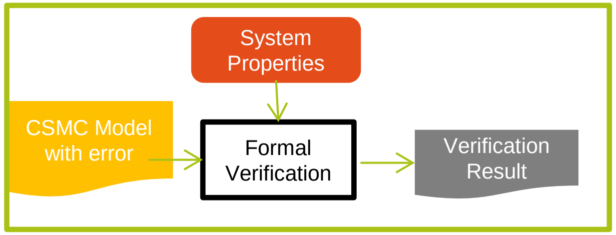
❑ HAI validation under stress conditions

- How will time constraints impact the HAI?
- Can we improve the procedures?



Methodology Validation Approach

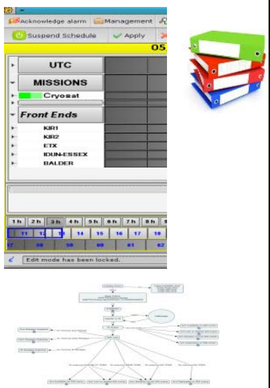
❑Methodology Validation with known defective version of the system (an old error)



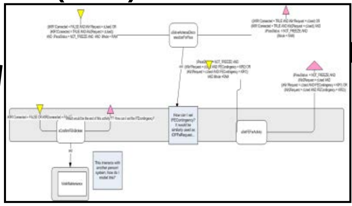
- Will the methodology find a known error?
- After implementing a solution, can we use the methodology to prove that the error is fixed?

Formal Verification Steps

CSMC Analysis

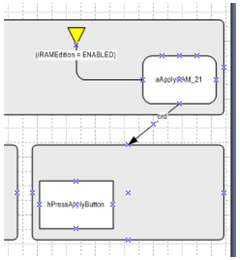


Human Task Model (Visio)



EOFM Model
<activity name="aApplyRAM">

<precondition>(iRAMEdition = ENABLED)
</precondition>
<decomposition operator="ord">
<action humanaction="hPressApplyButton"/>
</decomposition>
</activity>



EOFM toSAL

Human Tasks & Generated Properties

Operator Module Generated from EOFM
[]aApplyRAM_ExecutingDone:
aApplyRAM = actExecuting AND
hPressApplyButton_6 = actDone -->
aApplyRAM' = actDone;
Properties Generated from EOFM
hPressApplyButton_6_21_Executability:
THEOREM main |-
G(NOT(hPressApplyButton_6_21=actExecuting));

Validation

Specific Properties
fakeTiltAlarm:
THEOREM main |-
G(NOT((iAlarmTilt=ON)AND
(lAntennaRecTilt=NO_TILT)));

SAL

Refinement
EOFM, Automation properties

Manual Modeling in SAL

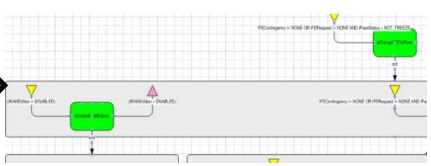
Interface, Automation and Environment modeling in SAL
lRAMEdition' = IF hPressEditionModeButton THEN ENABLED
ELSIF (hPressApplyButton) THEN DISABLED
ELSE lRAMEdition ENDIF;
lKir1Request' = IF (hPressApplyButton) THEN iKIR1Scheduled
ELSE lKir1Request ENDIF;
lKIR1Availability' = IF (hPressApplyButton) THEN iKIR1Availability
ELSE lKIR1Availability ENDIF;

Verification

Counterexample:
==============
Path
==============
Step 0:
aApplyRAM = actReady
aApplyRAM_21 = actReady

Visualizer

Results Interpretation
Counterexamples Visual Analysis



Verification Properties

❑ Specific Properties: properties that the system should fulfill

- **recTilt** the tilt should match the ACU recommendation
- **fixedUnavailability** in case of failure the pass will use the other antenna
- **fixSTDM** there should exist STDMs in the antenna before tracking starts
- **fakeTiltAlarm** checks that automation does not activate tilt alarm erroneously

The model checker should validate all of them (prove)

❑ Generated Properties: for detecting potential HAI problems

- 576 Properties created by the EOFMtoSAL translator for activities and actions in the EOFM model: act_startability, act_executability, act_finishability, act_completability, act_resetability, act_inevitablecompletability

The model checker should find a counterexample for all except for InevitableCompletability

Verification Results

❑HAI Validation in nominal conditions

Model checker ran during 4 days for 580 properties (Windows7, 8GB RAM)

- **No errors found for specific properties, all of them proved**
- No errors found on generated properties but the environment model had to be fine tuned to obtain valid results.

❑HAI Validation in case of operator error

Added 1 zero-order phenotype of erroneous action to task model:

- **No errors found on specific properties.** System is safe for 1 erroneous action

Added 1 attention slip to task model.

- **2 errors found on specific properties**: operator could execute erroneously the rectilt and fixstdm tasks. SAL returns the first violation found. Further iterations could be applied to find other errors.

Validated in 64-bit linux. SAL limitation in Windows: memory error building the BDD.

SAL took more than 24 hours to validate 4 properties.The same properties in the nominal model took less than 4 hours. The statespace generated is too large. No more tests were performed.

Verification Results

❑HAI Validation under stress conditions

Initial scenario modified to reduce the time that operator has for solving the antenna error. The model cheker failed to obtain counterexamples for

- Completability of solveUnavailableAntenna
- Startability of some of its subactivities.

The counterxamples visualization helped to understand the error conditions and find an alternative solution: as some of the subactivities are not constrained by time, the tasks could be re-ordered to achieve as much as possible.

The environment model helped to create artificial scenarios that would be difficult to test with the real system

❑Methodology Validation

Added 1 error in the automation module for tilt alarm. The model checker returned a counterexample for rectilt property showing the error condition (fake alarm raised)

The error is fixed in the nominal model. SAL returned a proved result during the nominal validaiton.

Test Case Conclusions

- EOFM is simple and easy to use to model human tasks
- LTL is challenging:
  - Only enumerated & boolean values in the test case to avoid state explosion
  - Specific properties in LTL are difficult to create and validate
  - 60% effort was dedicated to validate/refine the system model.
- Model checker results must be analyzed carefully to interpret the results
  - The counterexample visualization helped to follow the execution path
- EOFM Generated properties were extremely useful for model validation
- Once the model is created, small effort is needed to:
  - verify the behavior for specific scenarios
  - create stressing conditions that are difficult to test on the running system
  - verify corrections
  - analyse the impact of modifications

Test Case Conclusions

Drawbacks:

- Time is not supported
- Only one operator is supported in the EOFM version used for this project
- The complete model would need a big effort. It would help to generate the model from graphical designs
- Erroneous behavior was only partially verified due to the state explosion problem

CSMC HAI Results:

✔ No safety errors found in nominal conditions for modeled tasks

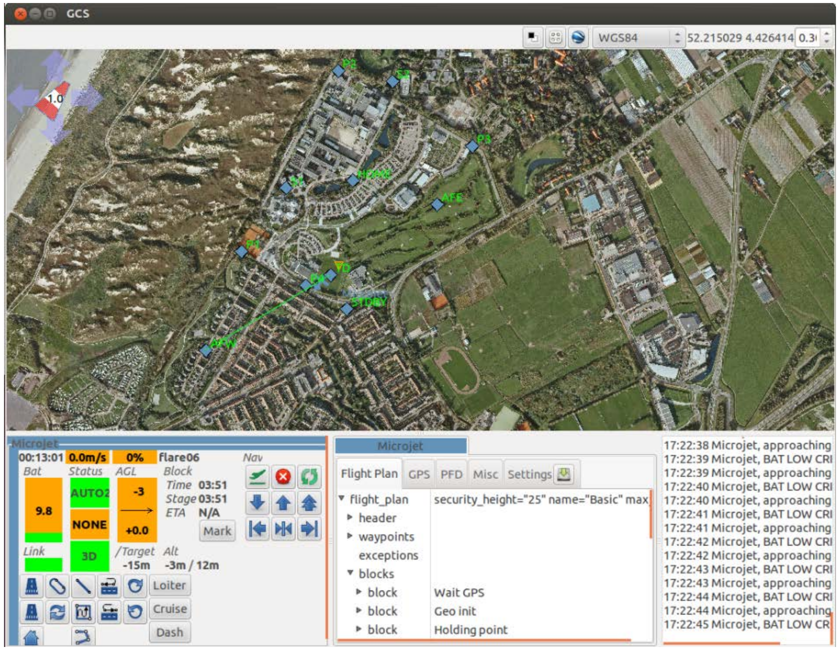
✘ Found improvements for the modeled tasks in case of stress conditions

✘ Additional confirmation dialogs should be added to avoid attention slips

✔ Error fix for false alarms was formally verified

UAV G/S Control System Test Case

UAV Application – Overview

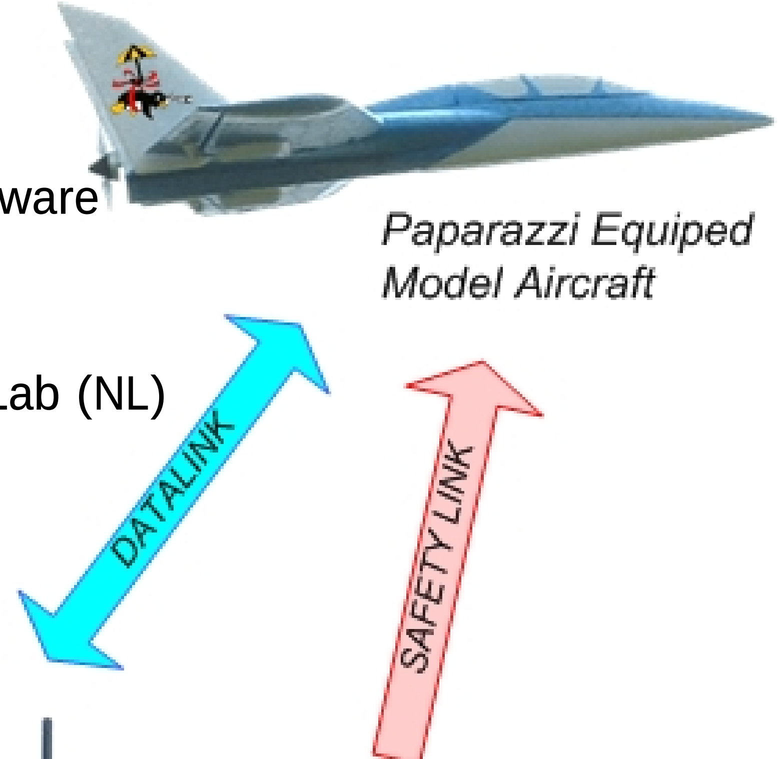


- Dynamic system – UAV does not stop
- Operator interface
- 2D/3D motion
- System monitoring needed
- Automation interaction – Flight Management System, implements path following

IXION® industry and aerospace

Implementation: Open source Paparazzi software

- Freely available, many users, developed primarily by ENAC (Fr) and TU Delft MAVLab (NL)
- Modifiable, simulation data is accessible through software bus
- Facilities for simulation

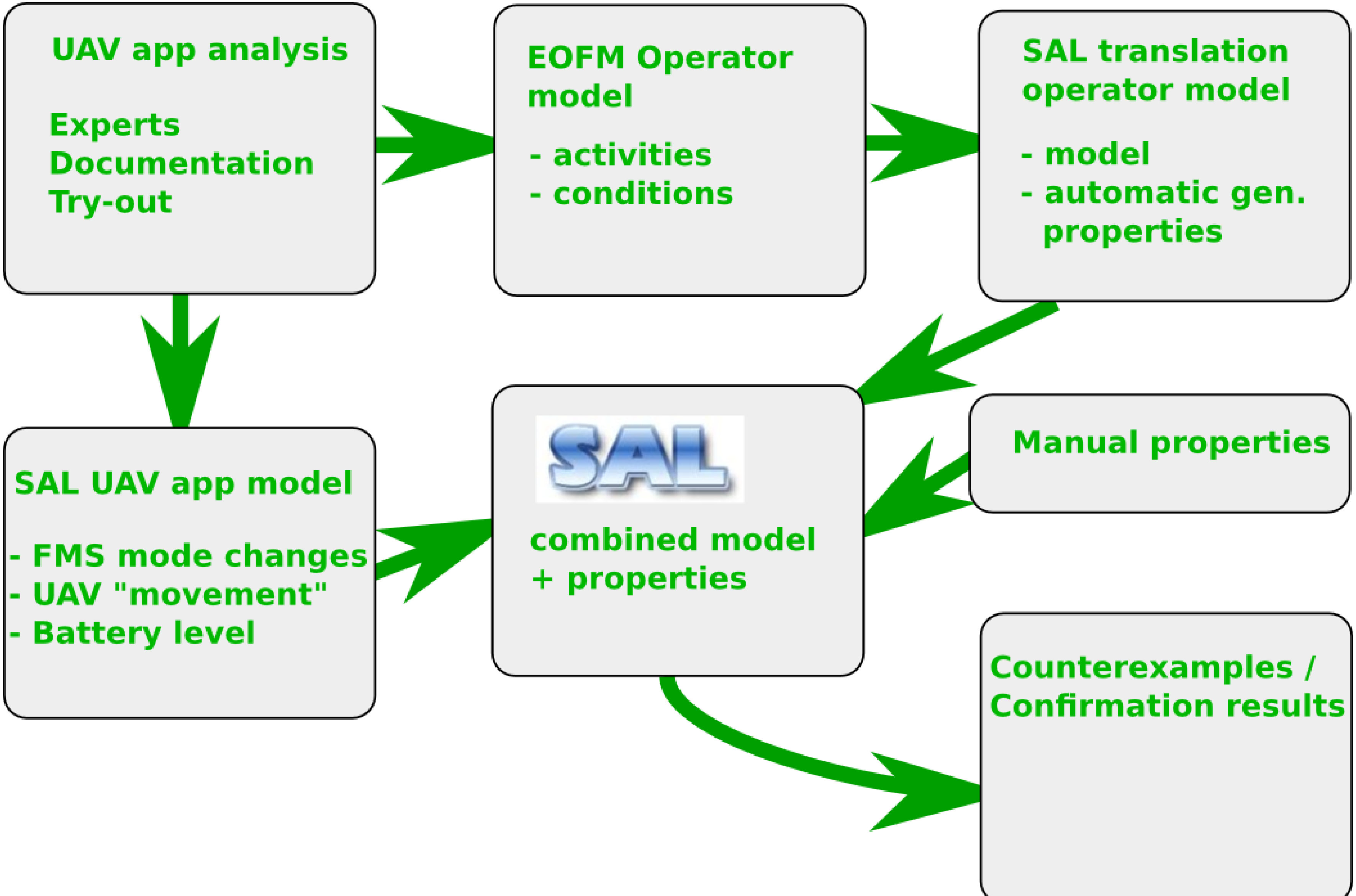


Interest and Focus for HAI methodology

Property of the application	Challenge for HAI verification
Dynamic behavior	Mix of immediate reaction of the interface and waiting for completion of the dynamic process
2D/3D (location + altitude) movement	Need to simplify 2D movement + “code” into discrete locations/state
Mixture of monitoring and action	Implement “parallel” activities
Openness and availability of the software, communication means	Opportunity to verify modeling approach by playing back HAI predictions in application

IXION industry and aerospace

Steps in the verification approach



Modelling problem

- Operator *should* distribute attention between two tasks; aFlightElements and aCheckAbortFlight
- Modelling the two tasks in EOFM makes their execution:
    - *Possible* – can check completion of both task
    - *Optional* – a flight can be made without aCheckAbortFlight
- To force the model to perform the aCheckAbortFlight activity, a virtual dead-man's switch has been implemented

Validation – Manual Theorems

- It should always land at the landing spot; so the FMS should end up in fpFlareNE or fpFlareSW:
  ```
  landelsewhere:
  THEOREM main |- G(NOT(iAltitude = altLanded
  AND iFlightPlanState /= fpFlareNE AND
  iFlightPlanState /= fpFlareSW));
  ```
  A counterexample indicates that a crash landing is possible; the theorem is proven for current parameters
- Varying the interval by which the operator checks, or the battery levels at which a return is initiated, affects the validity of the theorem.

IXION industry and aerospace

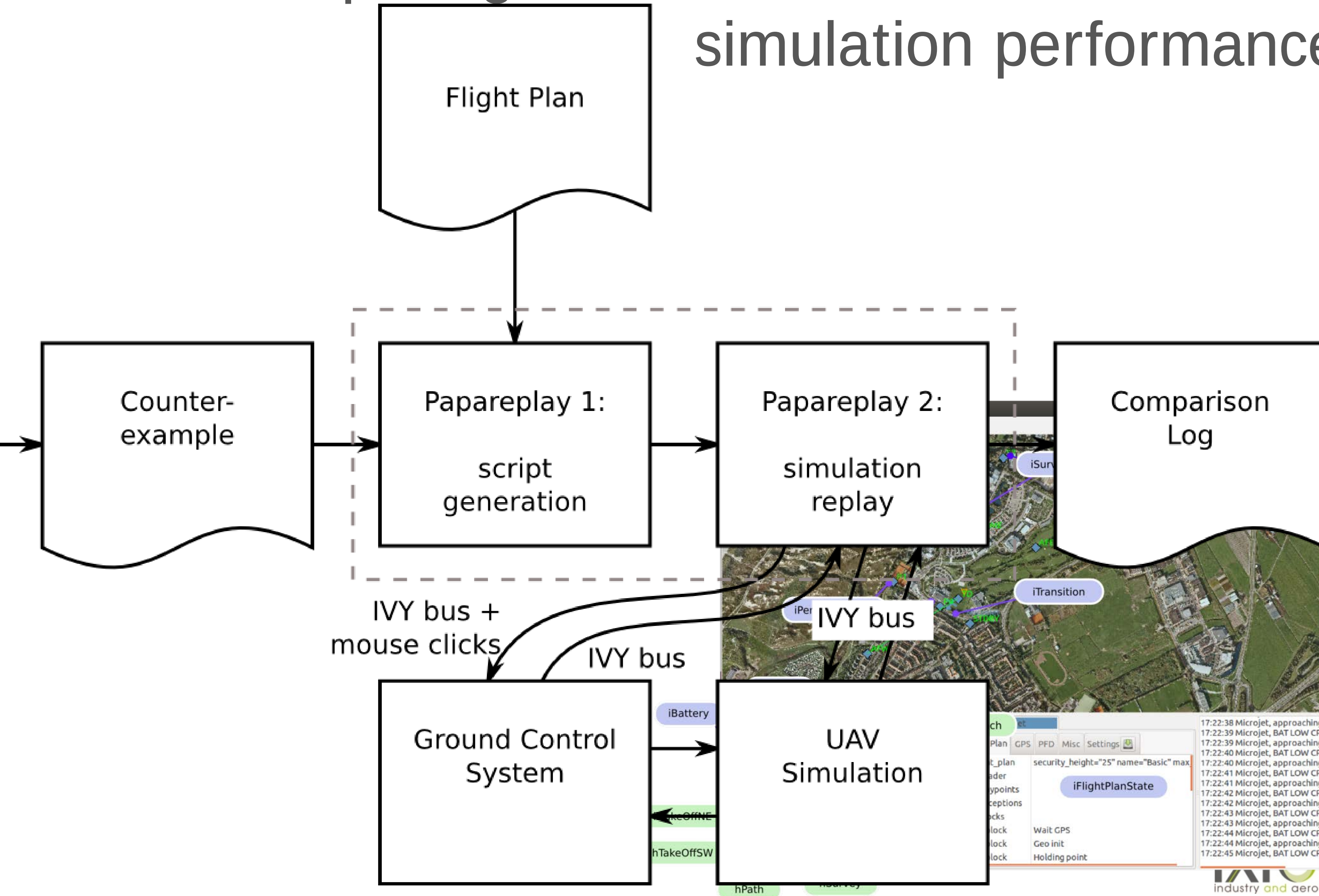
Validation – Manual Theorems

- Once given the result from `landelsewhere` , check how many surveys are possible:
  `survey4: THEOREM main |- G(NOT( iAreaDone > 3));`
  A counterexample is given for this theorem.
- Or how many times the perimeter can be checked:
  `perim2: THEOREM main |- G(NOT( iPerimeterDone > 1));`
  A counterexample is produced.

Verification results

- Automatically generated properties provide expected results
- Model scope causes not all activities to be repeatable (because re-charging the UAV is not modeled)
- Automatically generated properties are *very* useful in debugging phase

Comparing model verification traces with simulation performance



Results trace comparison

- “Round off” errors in the battery depletion level calculation
- All generated traces so far could be successfully played back
- Instructive to watch, relates counterexample trace to animation

UAV test case conclusions

- EOFM modeling matches well with modeling experience and task
- Forcing parallel work (monitoring/actions in this case) is tricky
- No new HAI errors discovered for the application
- Manipulation of the model gives expected results;
  - Reducing the initial battery level -> task element completion
  - Increasing the monitoring interval -> off-site landing
- Simplifications needed to model 2D movement with SAL
- Fair comparison between SAL-predicted traces and replay in Paparazzi
- Automatic property generation very useful
- Calculation times reasonable – on a fast computer!

IXION industry and aerospace

Project Conclusions

General Contributions

The formal verification methodology supported by EOFM serves its intended purposes:

1. HAI systems can be verified against manually created safety specifications
2. The novel property generation methods enable to detection of potentially unanticipated HAI issues
3. Erroneous behavior generation enables the impact of human error to be considered
4. The effort enabled a number of usability and stability improvements to be made to the supported tools
5. Novel mechanism for synergistically using formal methods with simulation

Dissemination of Results

Bolton, M. L., Jimenez, N., van Paassen, M. M., & Trujillo, M. (2013). Formally verifying human-automation interaction with specification properties generated from task analytic models. In *Proceedings of the Sixth IAASS Conference* (CD-ROM). Noordwijk: ESA Communications.

Bolton, M. L., Jimenez, N., van Paassen, M. M., & Trujillo, M. (ND). Automatically generating specification properties from task models for the formal verification of human-automation interaction. Submitted to *IEEE Transactions on Human-Machine Systems*. Accepted.

The project was profiled in the "Intelligent Systems 2013 Year In Review" article that appeared in the December 2013 issue of the AIAA's *Aerospace America Magazine*.

IXION industry and aerospace

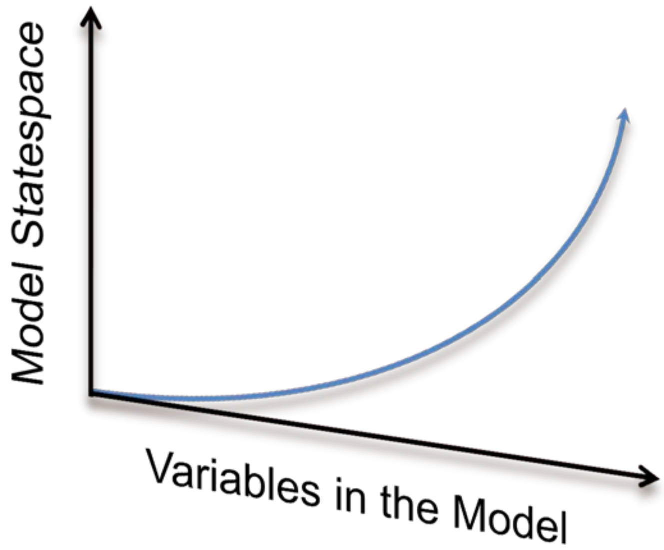
Limitations

Several limitations of tool usability could not be addressed:

- Task modeling with EOFM was straightforward, modeling other system components was not
- Incompatible features:
  - Phenotype generation
  - Slip generation
  - Specification generation
- Verification results could be slightly overwhelming

Problems with scalability:

- Formal models can quickly become too large to be analyzed
  (inherent problem with model checking)
- Erroneous behavior generation exacerbates this problem
- Parallel efforts have improved EOFM scalability but does not currently support all of the EOFM features



Limitations

Not a panacea:

- Does not address basic ergonomy of the interface (readability etc.)
- Limited application for dynamical systems
- Human operator modeling “procedural”

Future Work Identified

Property Generation

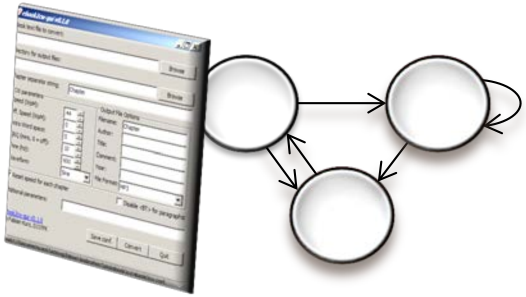
Explore other criteria related to the computation of task models and/or concepts from cognition

CTL / *LTL*

$\mathbf{G}\neg(Omission)$

Generate properties to reason about human errors

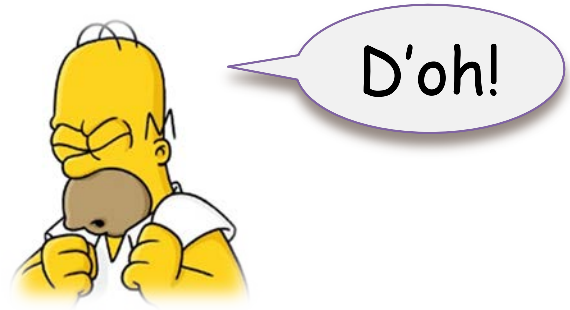
Account for interface and automation state



Include multiple human operator communication and coordination

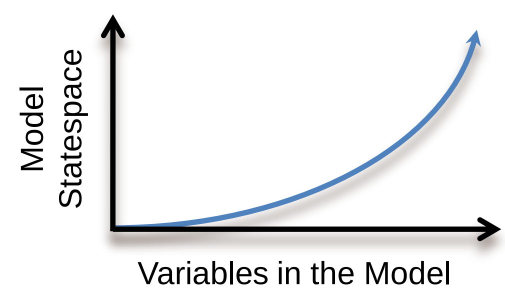
EOFM Methodology

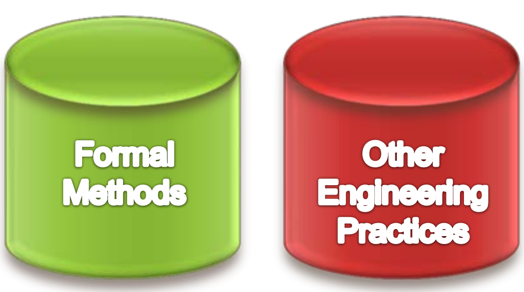
Extend erroneous behavior generation



Add cognitive and perceptual infrastructure

Improve EOFM and error generation scalability



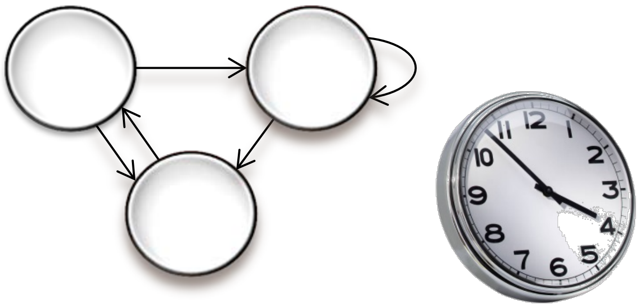


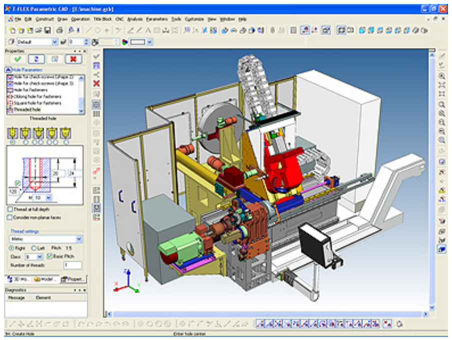
Improve synergy with other methods

General Formal Verification of HAI

Improve Usability and
Learnability of Formal Modeling

Explore timing analyses





Better integrate formal methods
into design

Thank You!