Formally Verifying Human-automation Interaction with Specification Properties Generated from Task Analytic Models

M. L. Bolton
University of Illinois at Chicago

N. Jimenez
IXION Industry and Aerospace

M. M. van Paassen
Delft University of Technology

M. Trujillo
European Space Agency

Supported by Verification Models for Advanced Human-Automation Interaction in Safety Critical Flight Operations from the European Space Agency
Human-System Interaction:
A significant 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
A Systems Problem:
Need to consider the human operator as an integral part of the system
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.
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
Automatically generate specification properties that will look for system conditions suggestive of problems with the human’s interaction with the rest of the system
Method:
Specification properties are automatically generated from normative task behavior models
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
Each activity’s and action’s execution state is represented as a finite state machine.

Formal Semantics
Linear Temporal Logic (LTL) Specification

Specification properties use model variables, Boolean logic operators, and **temporal** operators to assert properties about **all** paths through a model

<table>
<thead>
<tr>
<th>Name</th>
<th>Operator</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>$G \phi$</td>
<td>$\phi$ will always be true</td>
</tr>
<tr>
<td>NeXt</td>
<td>$X \phi$</td>
<td>$\phi$ will be true in all next states</td>
</tr>
<tr>
<td>Future</td>
<td>$F \phi$</td>
<td>$\phi$ will eventually be true</td>
</tr>
<tr>
<td>Until</td>
<td>$\phi U \psi$</td>
<td>$\phi$ will be true until $\psi$ is true</td>
</tr>
</tbody>
</table>
Linear Temporal Logic (LTL) Specification

Linear temporal logic does not allow you to positively assert existence (that a state exists) ...

... but we can assert existence negatively:

- If we want to prove that $\phi$ exists, we tell the model checker that we never want $\phi$ to be true: $G \neg (\phi)$

- If $\phi$ exists, the model checker will return a counterexample showing how $\phi$ was reached
Method:
Specification properties are automatically generated from normative task behavior models
... 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:**

\[ G \rightarrow (\text{Act} = \text{Executing}) \]
... 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 Completeness:**

\[ G \neg (Act = Done) \]
... 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:**

\[ G \left( (\text{Act} = \text{Executing}) \Rightarrow F (\text{Act} \neq \text{Executing}) \right) \]
... 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:**

\[
\text{G} \rightarrow \left( \text{F} \left( \text{G} \left( \forall \text{RootActivities} \left( \bigwedge_{\text{Act} \in \text{RootActivities}} \text{Act} \neq \text{Executing} \right) \right) \right) \right)
\]
Application

A pilot performing the before landing checklist of an aircraft
Application

A pilot performing the before landing checklist of an aircraft

Ignition ..................................Override
Landing Gear ...... Down, Three Green
Spoilers.....................................Armed
Flaps ..................Extended, 40 Degrees
Human Task Behavior

GSIndicator ≤ Inactive

aPrepare ForLanding

ord

GSIndicator ≤ Inactive

^ IgnitionLight = Off

IgnitionLight = On

aOverride Ignition

ord

GSIndicator ≤ Alive

^ ThreeGearLights = Off

ThreeGearLights = On

aDeploy LandingGear

ord

GSIndicator ≤ OneDot

^ FlapsGauge ≠ 25

FlapsGauge ≥ 25

aSet Flaps25

ord

GearDoorLight = Off

^ SpoilerIndicator = Off

SpoilerIndicator = On

aSet Spoilers

ord

GSIndicator ≤ Capture

^ FlapsGauge ≠ Flaps40

FlapsGauge = Flaps40

aSet Flaps40

ord

FlipIgnition Switch

PullGear Lever

Set Flaps25

Arm Spoiler

Set Flaps40
Environment

Alive  One Dot  Two Dots  Capture

0 s  5 s  10 s  15 s  18 s
Human Interface

- Ignition Switch
- Gear Lever
- Spoiler Lever
- Flaps Selector
- GS Indicator
- Ignition Light
- Three Gear Lights
- Spoiler Indicator
- Flaps Indicator
- Gear Door Light
Method

System Model → Model Checker → Verification Report
Human Task Behavior → Translator and Specification Generator → Specification
Method

System Model → Model Checker → Verification Report

Human Task Behavior → Translator and Specification Generator → Specification
Method

System Model → Model Checker → Verification Report

Human Task Behavior → Translator and Specification Generator → Specification
Verification Results

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

- Act Executability and Act Compleatability were satisfied for all activities and actions
- Task Liveness was satisfied
- Inevitable Compleatability was satisfied for all but three activities:
  - aPrepareForLanding
  - aSetFlaps40
  - aSetSpoilers
Verification Results

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

- Act Executability and Act Completability were satisfied for all activities and actions
- Task Liveness was satisfied
- Inevitable Completability was satisfied for all but three activities:
  - aPrepareForLanding
  - aSetFlaps40
  - aSetSpoilers

All three violations constitute serious safety problems
Verification Results

All three of these specifications must wait for the landing gear doors to fully open.
Contributions

– 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
– An example that demonstrates the method’s power
Limitations and Future Work

Investigating additional applications

Generating additional specifications

Providing analysts with guidance and decision support
Questions?
References