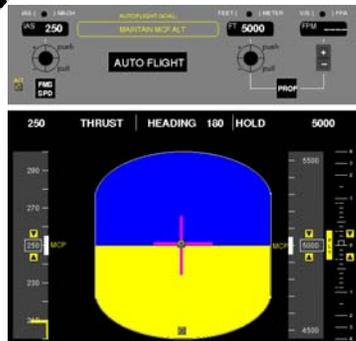
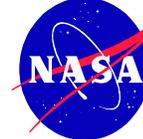
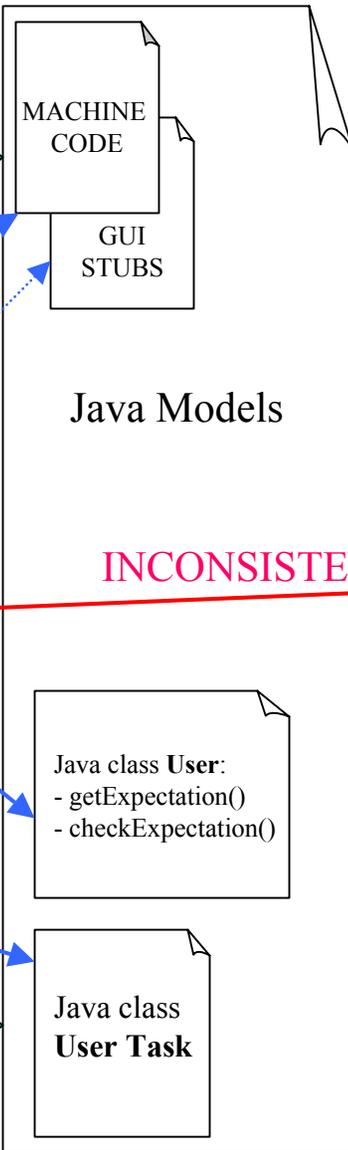


USING MODEL CHECKING TO DISCOVER AUTOMATION SURPRISES



STUB GENERATOR



JPF MODEL CHECKER

FAULTY EXECUTION

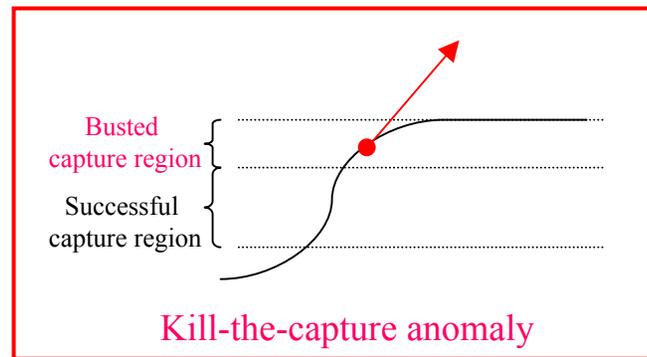
TRACE TRANSLATOR

start
incrMCPAlt
incrMCPAlt
pullAltKnob
fly
fly
incrMCPVS
fly

INCONSISTENCY

```
start >
ncrMCPAlt^{1,10} >
pullAltKnob >
(pilotExp > fly)^{1,10} >
incrMCPVS^{1,10} >
(pilotExp > fly)^5;
```

DRIVER GENERATOR

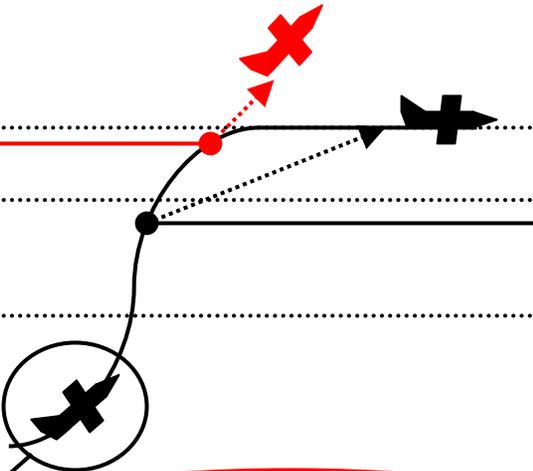


Kill-the-capture anomaly

Explanation of Accomplishment

- **POC:** Guillaume Brat and Willem Visser
- **Shown:** The aim of this work is to develop an automated approach to identify mode confusion problems (e.g., when a pilot thinks a plane is one mode when the plane is actually in another mode) in automation used in modern aircraft and spacecraft. Specifically, we aim to develop a framework that supports 1) automatic extraction of machine and interface models from code of automation, 2) extraction of user and task models from manuals and training materials, 3) encoding of user tasks in an intuitive notation, 4) automatic code generation from task specifications, and 5) analyzing the obtained models for mode confusion problems.
- **Accomplishment:** We demonstrated our approach on the example of a Java web-based autopilot tutorial used at NASA for pilot training. The main approach is to identify the four models of the system: the machine, the interface, the user, and the user task. The user task is described, using regular expressions, as a collection of sequences of actions performed on the display, and the corresponding Java code is generated automatically. The user task plays the role of a driver that synchronously executes the remaining models in the system. The JPF model checker is used to explore all possible executions of the task and to check the consistency of the states across the models. When an inconsistency is discovered, JPF records the (faulty) execution path. We also automatically analyze these faulty execution paths to produce scenarios that illustrate inconsistencies in terms of the actions that the user performs on the display.
- **Future Plans:** The focus of the current work is on the extraction of the machine, interface, user, and the task models from given GUI applications. As an extension to the ongoing work, we see ourselves working on 1) verification of several additional GUI applications used for simulation of cockpit and shuttle automation, 2) improving the framework for processing GUI components and extracting meaningful models from them, 3) developing a framework for interface design and code generation for interfaces, and 4) construction of more elaborate user models.

Kill-the-capture region:
if the pilot tries to capture an altitude in this region, the command is ignored by the flight control system, and the plane keeps going past its desired altitude.



Capture region:
if the pilot tries to capture an altitude in this region, the command is accepted by the flight control system, and the plane will level off at the desired altitude.

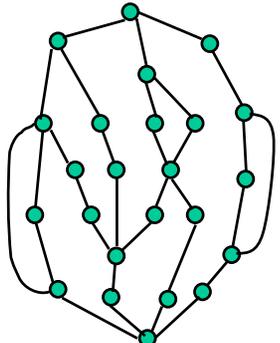
kill-the-capture anomaly

The error trace is translated into an anomalous scenario

The pilot's expectations and task can be modeled in a program, which can drive the code of the flight control system of a plane

JAVA PROGRAM

JPF Model Checker



```
void climb (int alt) {  
  ...  
}  
void move_studder() {  
  ...  
}
```

JPF explores all possible executions of this code, finds erroneous behaviors, and produces an error trace

Explanation of Accomplishment

- **POC:** Guillaume Brat, Oksana Tkatchuk and Willem Visser
- **Shown:** The aim of this work is to develop an automated approach to identify mode confusion problems in automation used in modern aircraft and spacecraft. These problems arise when a pilot thinks a plane is one mode when the plane is actually in another mode. For example, when a pilot tries to capture an altitude in the wrong region, the plane will keep climbing (or descending) rather than leveling off. In our work, we have developed a framework that supports 1) the automatic extraction of machine and interface models from flight control code, 2) the creation of user models in Java, 3) the encoding of user tasks in an intuitive notation that is automatically translated into Java code, and 4) the analysis of the obtained models for mode confusion problems.
- **Accomplishment:** Previous work in this area required to model the machine (flight control system), the user (pilot), its task, and the interface (cockpit display) in a formal language. In our work, all can be done using the actual code that is flown. Hence, we achieve a substantial gain in terms of fidelity of the analysis. We also simplify and automate the task of verifying these interactive systems.
- **Future Plans:** The focus of the current work is on the extraction of the machine, interface, user, and the task models from given GUI applications. As an extension to the ongoing work, we see ourselves working on 1) verifying several additional GUI applications used for simulation of cockpit and shuttle automation, 2) improving the framework for processing GUI components and extracting meaningful models from them, 3) developing a framework for interface design and code generation for interfaces, and 4) constructing more elaborate user models.