Enhancing Autonomy with Trust: Pilot license to the autonomy

Presented by S. Bhattacharyya

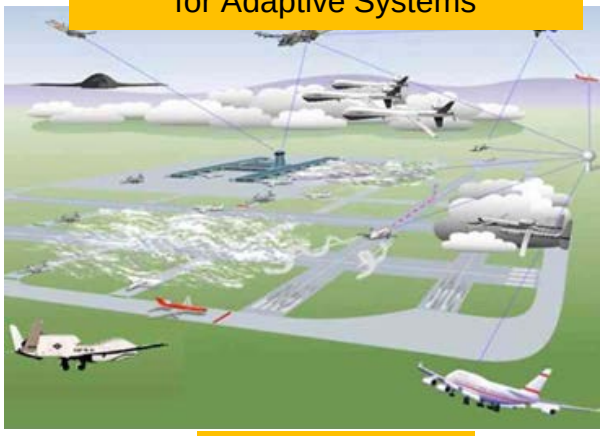
Rockwell Collins

UAVs are H

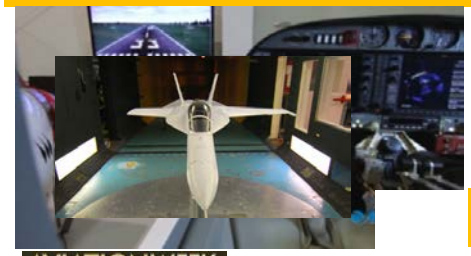
Rockwell Collins

Autonomy / Verification Context

NASA Certification Consideration for Adaptive Systems

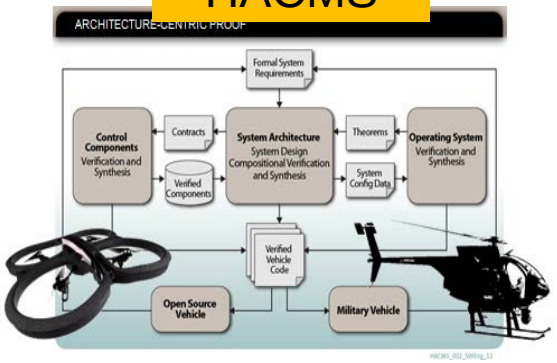


Damage-Tolerant Flight Controls
Emergency Mission Mgmt System
Control Toolkit



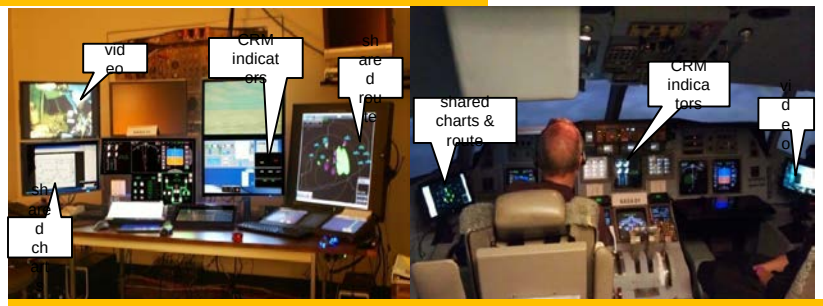
NASA TASAR – bring-aboard avionics, flight deck automation

HACMS

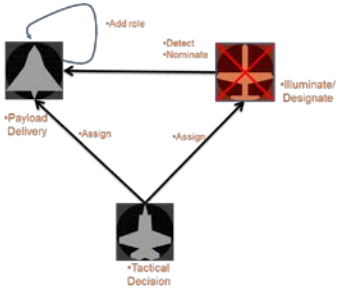


Distributed Flight Management

Advanced Flight Deck HMI

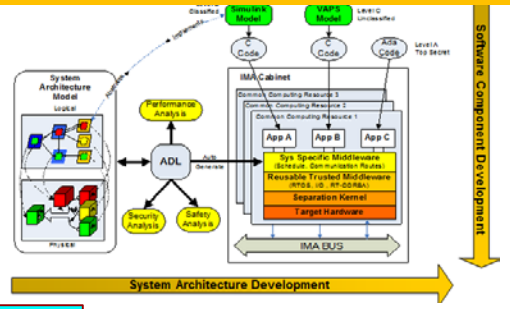


NASA Single Pilot Operations: Crew Resource Management



Distributed Skill Chain Example

SWPI: Complex System Architecture Analysis



Automated systems and formal methods align in technology and business cases

Motivation

- **Certifiable trust for autonomous systems**
  - **Robonaut, ALIAS…**
- "Certifiable trust" → biggest challenge.
- "When we certify avionics, we test every input, every path. When we certify pilots, we decide if they will probably do the right thing, but we do not test every response. It is more about behavior and probability".
- How can we assure correct pilot behavior?
  - **"New methods for verification and validation?"**
- Other aspects of trust not covered here:
  - Interpersonal trust is based on "information, integrity, intelligence, interaction, intent and intuition"
  - Security based trust

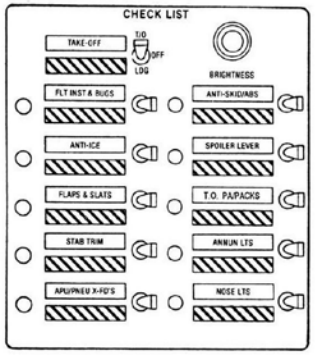
Overview

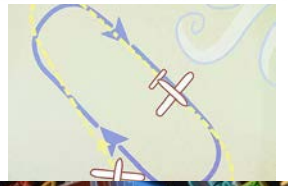
- **Desire: autonomy implements pilot behavior**
  - The notion is to create “pilot licensed” behaviors that work with human intent to support more autonomous mission operation
- **One of the approaches: Use of formal methods**
  - Model autonomous behaviors in formal methods framework
  - Translate from design environment to formal environment
  - Understand considerations and potential near-term limitations on autonomy
- **Limitation: developing trust for intelligent systems**
  - Non-determinism
  - Analytical reasoning about the systems

Technical Summary

- **Evaluated training manuals to identify requirements for expected pilot behavior**
  - Practical Training Standards
  - ONR
- **Evaluated and selected cognitive architectures**
  - Identified ACT-R (synthetic teammate), Soar for agent based behavior modeling
  - Evaluated learning mechanism
    - Implemented Reinforcement learning from FIT
- **Developed a verifiable, architectural and formal approach to modeling the pilot behavior**
  - Gain trust in autonomy with models in UPPAAL and ACL-2
- **Developed translation formalisms from cognitive architectures to formal approaches**
  - Maintaining architectural integrity
  - Algorithms to translate from ACT-R, Soar to UPPAAL, ACL-2

Pilot model: Requirements (short list)





1. **The system shall be capable of determining whether aircraft systems and equipment are functioning normally**
    1. in this case, via checklists
2. **The system shall be capable of recovering from flight plan deviations.**
    1. Implemented in formal tool UPPAAL for VOR navigation (similar approach as for preflight check list)
3. **Communications management**
4. **The system shall be capable of recovering from lost communications.**

Guarantee that the autonomy always executes the correct behavior as indicated in the FAA standards

- readily implement-able
- modular architectural approach

The Structure of cognitive architectures

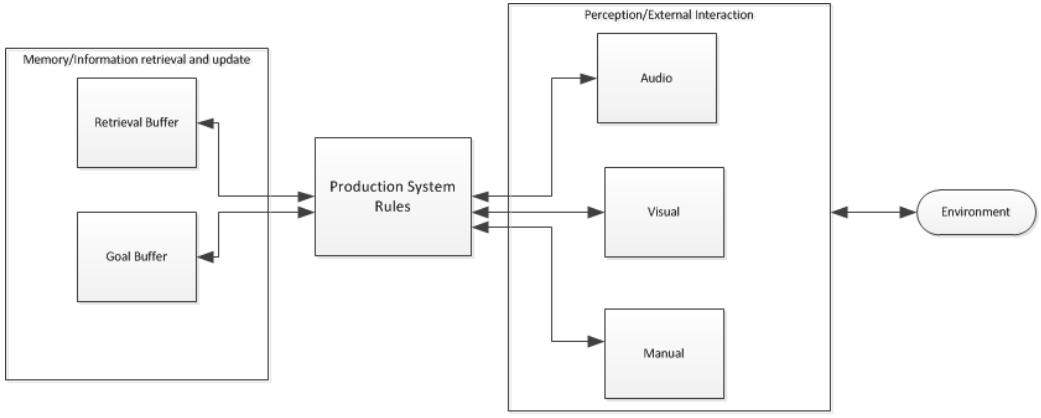
- **Evaluated some of the cognitive architectures to implement pilot behavior**
  - ACT-R, Soar
- **Understanding architectural integrity crucial in maintaining the formalisms**
- **Agent architecture**
  - Integration of several components
    - Perception, Memory, Production systems

Why Soar ?

ACT-R: Common Lisp based → challenging to verify

Soar: Tree structured working memory→ structured approach to representation

ACT-R



Soar

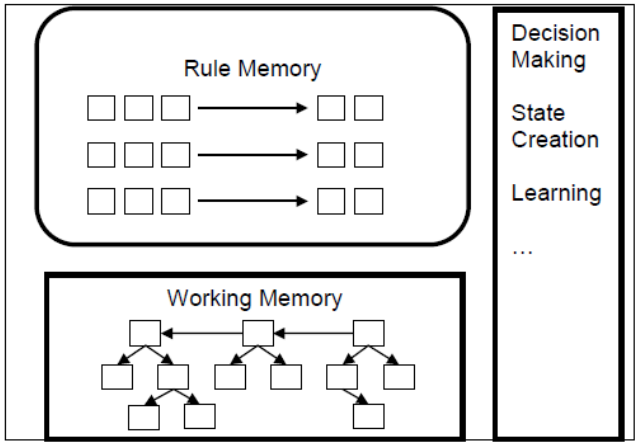
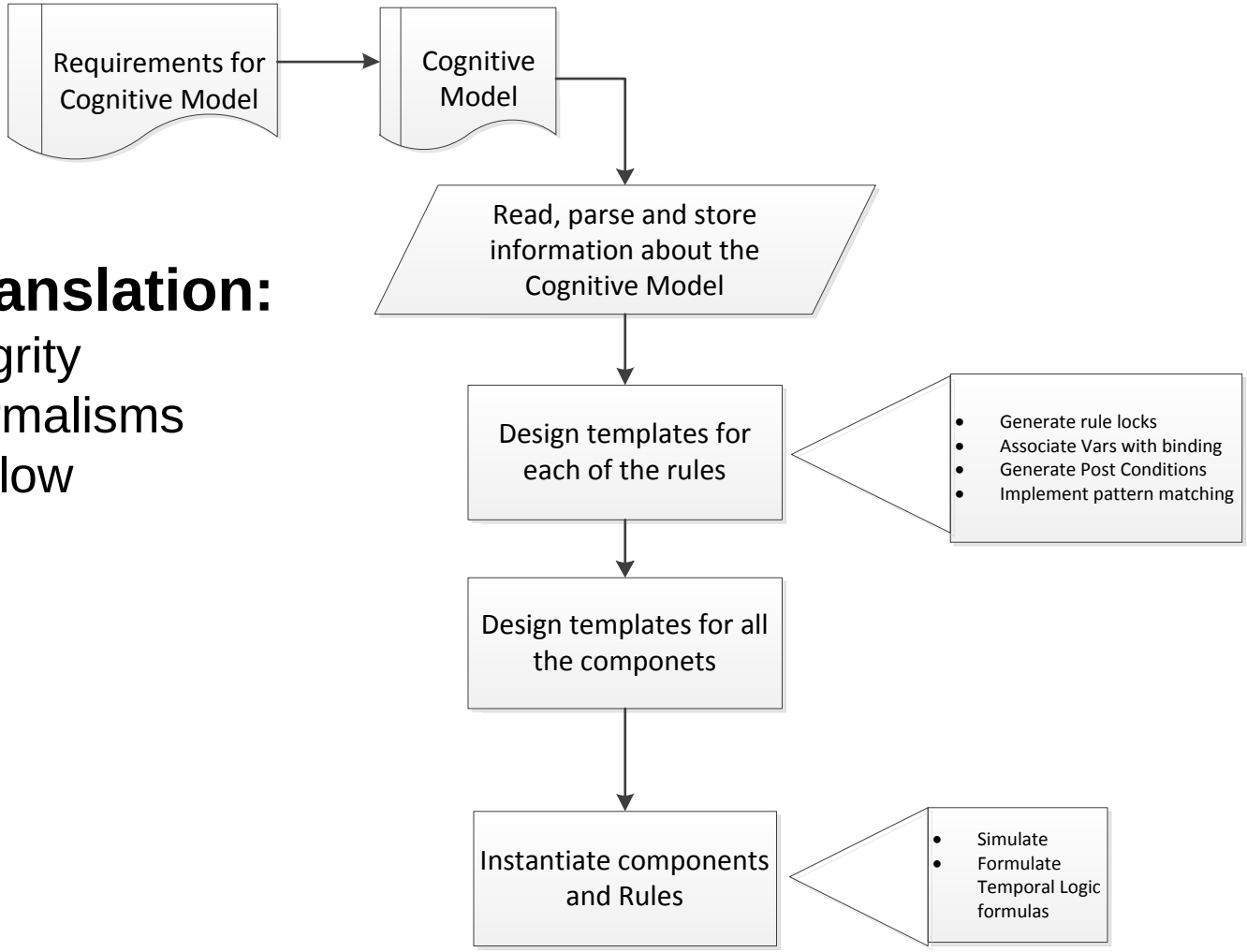


Image Source: Soar Tutorial Part 1

Flowchart for Translation from Cognitive Architectures to formal methods

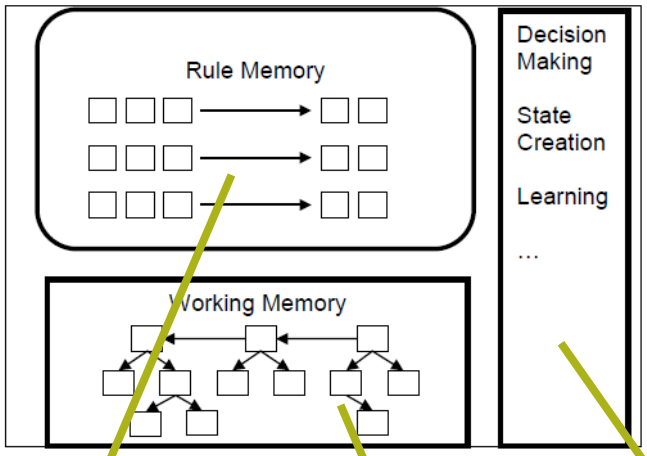


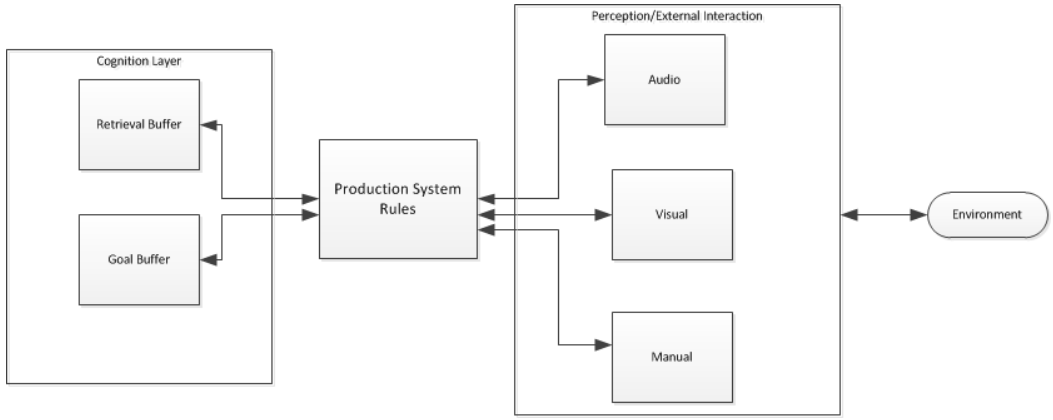
Challenges in translation:

- Architectural Integrity
- Rule execution formalisms
- Cognitive engine flow

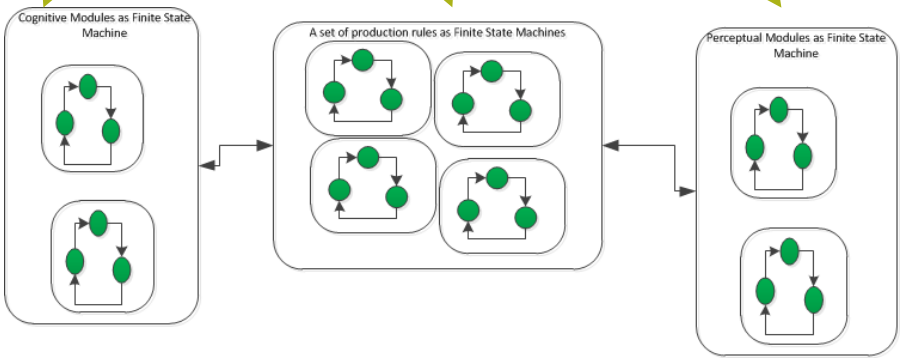
Cognitive architecture to Formal modeling environment

Cognitive Architectures





Formal Modeling Environment

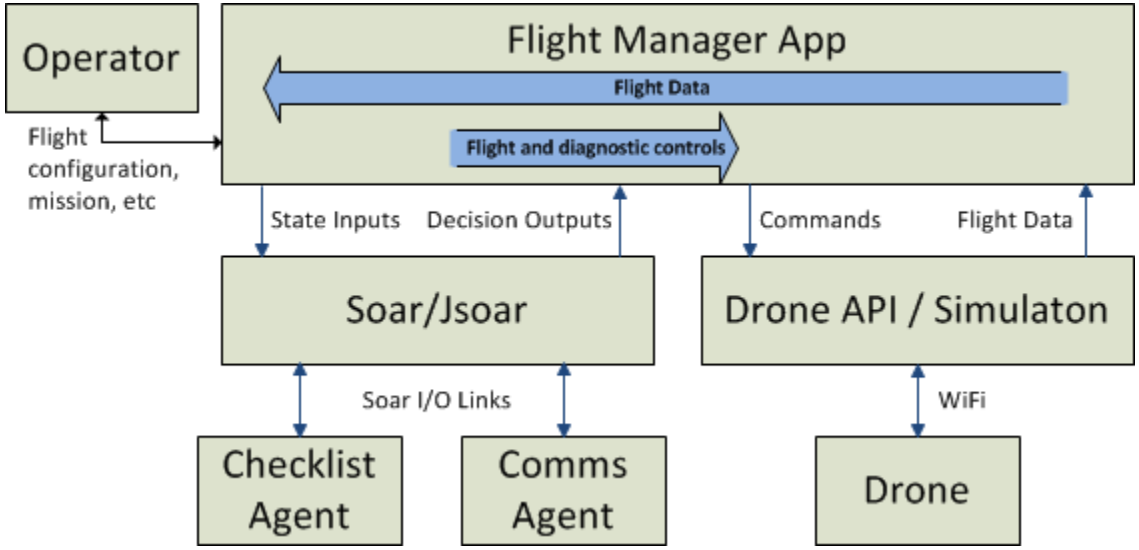



```
(defun counter*apply*increment (wmem)
  (if (and (wmem-match wmem "s1" "name" "counter")
           (wmem-match wmem "s1" "operator" "o")
           (wmem-match wmem "s1" "num" t))
      (let ((num (nth 2 (wmem-match wmem "s1" "num" t))))
        (if (and (wmem-match wmem "o" "name" "increment")
                 (wmem-match wmem "o" "first" num))
            (let* ((wmem (cons (list "s1" "num" (+ num 1)) wmem))
                   (wmem (remove (list "s1" "num" num) wmem)))
              wmem)
          wmem))
    wmem))
```

The process of translation and verification has been accomplished with multiple tools

Intelligent Learning System

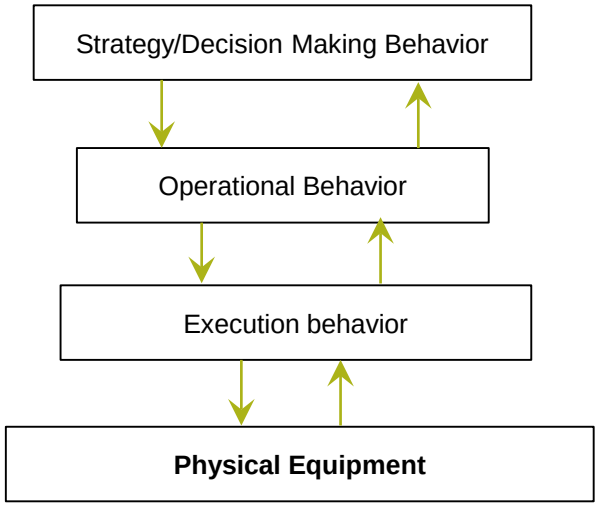
- Applied Reinforcement Learning for the Pilot as the agent
  - Agent evaluates multiple paths of accomplishing a task
- Learns the optimal path
  - Modifies existing Soar rules
- Analyzing potential conflicts or violations
  - Compositional analysis of rules
  - Generate monitors for runtime assurance
- Limitations of the approach discussed
  - Analysis of non-linear behaviors (Hybrid Systems reasoning)
    - Scalability of analytical reasoning tools for non-linear dynamics
  - Dealing with uncertainty during runtime for runtime assurance



Reinforced Learning as a Method for Rule Modification

Checklist Procedure Formal Modeling, V&V

- **Requirements**
  - http://greggordon.org/flying/CFIIPTSIIICCockpitCheck.htm
    - Communication equipment
    - Navigation equipment
    - Altimeter
- **Architecture**

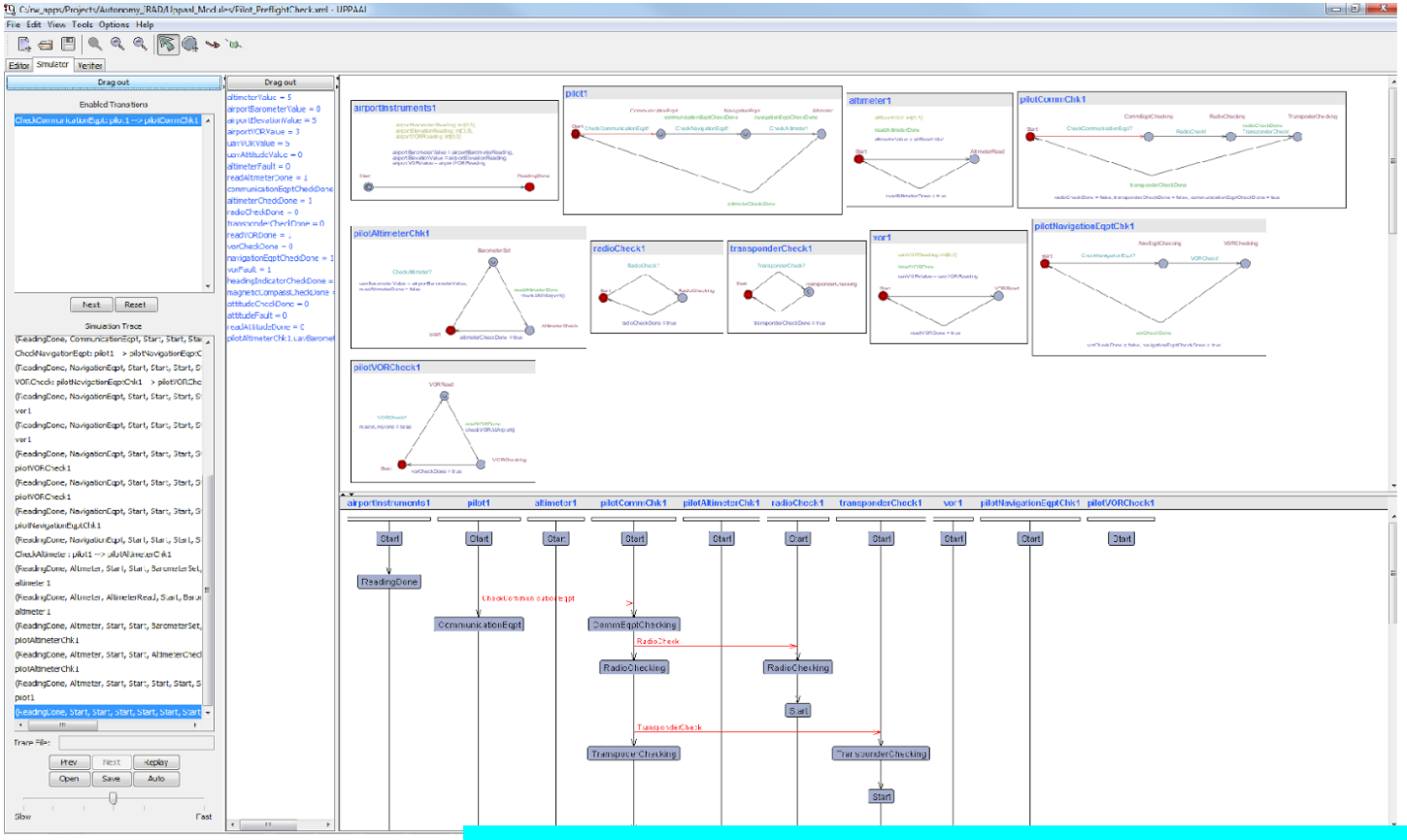


Implementation

Sample Formal Analysis Representation

Checklist Operation

- **Property (V&V)**
  - *"Always eventually the pilot checks the equipment (instance: radio) "*
    - A<> radioCheck1.RadioChecking
  - *"Pilot responds to faulty instrument (instance: altimeter) "*
    - ((airportElevationValue-altimeterValue) >=1 || (altimeterValue-airportElevationValue) >=1)&& pilotAltimeterChk1.AltimeterCheck --> altimeterFault
  - Reachability analysis to prove the properties



Sample V&V

Challenges in translation

- **Maintaining rule formalisms in the formal environment:**
  - how do we limit execution of one rule at a time?
  - implementing critical section or locking mechanism through the translator
- **Maintaining the cognitive architectural lifecycle:**
  - the execution of rules go through phases of operation, how to determine and implement the phases?
  - what kind of algorithm needs to be implemented during the translation process to maintain appropriate occurrence of phases?
- **Addressing concurrency:**
  - performing selection check to identify all the possible execution of rules
  - implementing methods to check for all the possible rules that can fire
  - executing one among several rules that can fire
- **Maintaining architectural integrity:**
  - how to maintain the architectural integrity of the cognitive engine?
  - implementing methods to deal with the flow, binding, evaluation of the condition during translation
- **Representation of knowledge:**
  - How best to represent knowledge or learned information?
  - Do we automate the knowledge representation through translation?
  - Specific knowledge information might need to be added manually

Future direction

- **Enhancing autonomy and maturing machine intelligence working with humans as team**
  - Integrating more behaviors on aircraft procedures and operations for autonomous missions
    - Including more safety constraints and functionalities as rules
  - evaluate learning to develop trust on the machine knowledge
    - Learning modifies a rule
    - Learning adds a new rule?
  - compositional analysis of the dynamic rule set
  - cognitive analysis, interactions, enabling human machine trust
  - Evaluating hybrid systems approaches

Rockwell Collins

Questions?