NRC
NRA on Autonomy

Dave Vos, Ph.D.

August 28, 2013

Systems Level Perspective on Autonomy

Relevant Vos Background

- MIT Aero/AStro Ph.D. 1992.
  - World's first successful Autonomous Unicycle Robot
- Founder, CEO, CTO Athena Technologies, Inc. Acquired by Rockwell Collins 2008. Control, Nav and Guidance Systems specialists
  - UAS Solutions
  - Certificated Light Sport Aviation Engine Controls for BRP Rotax engine
  - Manned Aviation solutions
- Example UAS: US Army Shadow Nav/Guidance/control system
  - Thousands of flights per month
  - Operational hours passed I Million in 2011
  - Automated Launch, Flight, Mission, Recovery (tailhook autoland)
- These are Personal Comments and Thoughts on Autonomy
  - Not affiliated with Athena or Rockwell Collins

Levels of Autonomy
All must Work as Advertised at Top Level

Baby Steps
MIT 1992

Robotic Unicycle tracks Heading and Speed Inputs

Autonomous Damage Tolerance
Rockwell Collins & DARPA 2009

Push Button to launch flight
Autonomous takeoff, execute flight plan, recover from damage, reroute to autonomous landing

A word on "Autonomous"

- FAA Currently is Allergic to the word "Autonomous". We need to help clarify
- Two main Perspectives (in English)
  - Deterministic systems: The resulting action due to a stimulus is always predictable
  - non-Deterministic systems: The resulting action due to a stimulus is not necessarily predictable

This Discussion Contemplates Deterministic Autonomous Systems

Some Recent Events – Manned and UAS

- Air France Airbus A330 en-route from Rio to Paris
  - Stall From 38 kft into the ocean after air data discrepancies
  - Autonomous system could have made this a non-event

- Airbus A320 in Hudson river
  - Precious time spent figuring out what had happened
    - Only remaining realistic option was land in the river - an heroic accomplishment under the circumstances
  - Autonomous system could very likely have landed back on the runway at LaGuardia

- UAS: Predator B crash - US Border
  - Switching crew consoles confused the vehicle configuration
  - Autonomous system cross-checks in GCS could have prevented the mishap

- Boeing 777 San Francisco
  - Slow & low approach
  - Autonomous system could have ensured a safe landing

Bad Luck or Human Error or Inadequate Systems Level Autonomy?

We Must Commit to the Advertised Level of Autonomy

- Design Philosophy is critical, Either we assume
  - Crew are Superhuman
    - Make no mistakes
    - Can resolve any complex situation arbitrarily quickly whilst performing other tasks
- Or
  - Crew are Human
    - Will make mistakes
    - Able to manage the system, but not <u>be</u> the system

- Don't expect Crew to Multi-task and Figure out Multi-Level Problems and Determine Emergency Solutions in Real Time
  - We must make the level of automated backup and recovery support the advertised level of Autonomy
  - We cannot require a Superhuman Crew to resolve lower level problems

Need Full Envelope Designs, Not Just the Allowable Normal Envelope

- For Example:
  - All-attitude body axis control to keep the trajectory on track regardless of attitude
    - "Up-elevator" can have dire consequences if inadvertently inverted at low altitude
  - Graceful degradation of systems when limits are exceeded

Darpa and Rockwell Collins 2010
Autonomous all-attitude flight

Certification Requirements Drive Architecture Decisions

- Current Certification philosophy allows Dependence on Superhumans as ultimate backup to resolve complex failure scenarios

- FMECA-based Systems Engineering must be employed and iterated upon to reach architecture design
  - Assume the Crew is a normal human, ie not very good at simultaneous real time multi-tasking and problem solving
  - Build appropriate levels of redundancy, including analytic redundancy to resolve problems automatically and keep flying
  - Enable Crew to Coordinate Emergency Actions vs Solving Emergency Problems

System Level Avionics, SW, Algorithm Design process

- Set Top Level Functional Requirements Including Requisite Levels of Autonomy for Flight Plans or Missions
- Define Notional architecture and algorithms
- Use FMECA to establish adequacy in terms of failures
- Iterate until converged
- Prove in Test (Simulations/Simulators, HWIL etc)
- Iterate until converged
- Flight test
- Iterate until converged

Software Development Challenges

- Safety Critical SW Development is Still a Relatively Immature Engineering Discipline
- By Definition, SW enables Designers to "fiddle Infinitely". Death to a business
- SLOC (Lines of Source Code) cost can range from $75 to many $100s per line
  - Can rapidly become Cost Prohibitive
  - Tough business decisions need to be made
    - Adjust fielded level of autonomy accordingly
    - Often leads to ultimately depending on a Superhuman crew
- Cost, Simplicity and Reduced SLOC count go hand-in-hand

Summary: Some Key Themes for Autonomy

- Commit to Autonomy
- Focus on the System Architecture and Necessary levels of Autonomy to meet System Requirements
  - Fundamentally driven by solid Systems Engineering
- Each level of Autonomy needs to be delivered without need for Crew to be Superhuman in cases of Failures or Mode selection

Summary: Some Needed Technologies and Tools for Enabling Broad use of Autonomy

- Mature Systems Engineering and Software Engineering Disciplines
- Structured System Architecture Design Techniques and Tools
- Safety Critical Systems Design through Systems Engineering and use of FMECA as driver
- Versatile Requirements Tracking Tools and Techniques
- Automated testing Tools and Techniques
- High Reliability Automatic code generation
- Tools for Tracking and Mapping of Test Plans and Results to Requirements
- Methods for Reducing System Complexity