Contingency Software in Autonomous Systems

NASA OSMA Software Assurance Symposium
July 18-20, 2006
Technical Briefing

Robyn Lutz, JPL/Caltech & ISU
Ann Patterson-Hine, NASA Ames

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and at NASA Ames Research Center, under a contract with the National Aeronautics and Space Administration. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA Software IV&V Facility. This activity is managed locally at JPL through the Assurance and Technology Program Office.
PROBLEM STATEMENT

Autonomous vehicles currently have a limited capacity to diagnose and mitigate failures. We need to be able to handle a broader range of contingencies (anomalous situations).

A contingency is an event or condition (as an emergency) that may but is not certain to occur [Merriam-Webster]
1. Speed up diagnosis and mitigation of anomalous situations.
2. Automatically handle contingencies, not just failures.
3. Enable projects to select a degree of autonomy consistent with their needs and to incrementally introduce more autonomy.
4. Augment on-board fault protection with verified contingency scripts
Contingency Software in Autonomous Systems

Overview of this year’s accomplishments (1)

• Completed Autonomous Rotorcraft Project (UAV) case study
  – Documented process & results (1 published & 2 submitted papers)
  – Performed hardware-in-loop testing of diagnostic tree
  – Project applied results, modifying camera controller to enable autonomous switching between color and video cameras
• Modeled MER Critical Pointing software to be reused on MSL
  – Called if commandability lost & before trajectory-correction maneuvers
  – Auto-generated diagnostic tree from TEAMS model of what is known when a “quit-failed” signal occurs
  – Supplemented available project documentation before reuse
Contingency Software in Autonomous Systems

Overview of this year’s accomplishments (2)

• New case study
  – ADAPT emulates a typical spacecraft power system with redundant power buses, a solar panel, and battery storage
  – The approach for developing contingencies resulted in critical function identification and preliminary identification of required contingency plans
• Described work to date at the Mini-SAS at JPL
• Data availability potentially high (needs packaging, sanitizing of models)
• Technology Readiness Level:
  – FY05: 3 ("Experimental demonstration of critical function &/or proof of concept")
  – FY06: 4 ("Validation in a lab environment") on grounded rotorcraft
What is a contingency?

• Contingencies are obstacles to the fulfillment of a system’s high-level requirements that can arise during real-time operations
  – Failures: camera fails due to hardware or software problem
  – Operational situations of concern: lens cap left on means that all images are black, so can’t land unassisted
  – Environmental situations of concern: strong crosswind interferes with imaging, thus with finding landing site
• Contingency-handling involves requirements for detecting, identifying and responding to contingencies.
• Contingency handling includes, but extends, traditional fault protection
• Something previously not done automatically is now done by the software
  – Previously done manually, or
  – Previously could not be done
• Example of incremental autonomy:
  – Collision avoidance (not hitting buildings)
    1. Remote control by pilot steering from ground
    2. Path calculated on ground, loaded into system, path-plan executed in flight
    3. Path calculated in flight based on real-time imaging
• Autonomy allows system to detect and respond to a broader class of anomalies in many more ways
• The rotorcraft software is safety-critical:
  – Requires collision-avoidance
  – Requires autonomous take-off & landing in populated areas
  – Used for critical missions: finding lost hikers, downed pilots; detecting highway accidents; imaging (early warning) forest fires
1. *Identify contingencies* that risk mission-critical functions in a power system testbed (using S-FTA, S-FMECA, Obstacle Analysis)
2. *Model contingencies* & autonomous recovery actions using TEAMS (Testability And Engineering Maintenance System, QSI)
3. *Analyze contingencies*: TEAMS produces diagnostic tree of checks needed to detect & isolate contingency, identifies missing checks and recovery actions
4. *Code contingencies*’ diagnosis & recovery behavior in the project’s planner scripting language (auto-translation from TEAM’s XML output)
5. *Verify contingency scripts* with hardware-in-loop simulation

Using the above steps:
- Verify contingency plans used by NASA projects
- Investigate issues in coverage of contingencies
- Test results on power system testbed
Contingency Software in Autonomous Systems

Contingency analysis

• Used Bi-Directional Safety Analysis to find contingencies
  – Forward analysis from potential failures to their effects (Software Failure Modes, Effects & Criticality Analysis)
  – Backward analysis from failures to contributing causes (Software Fault Tree Analysis)
• Guided thinking about possible ways to handle contingencies:
  – Use “Mitigation” column in SFMECA
  – Remove leaf nodes from SFTA graphs
  – Use obstacle resolution patterns [van Lamsweerde & Letier, 2000]
• TEAMS produces a diagnostic tree of checks needed to detect & isolate contingencies; identifies missing checks and recovery action
  – “Testability Engineering and Maintenance System”
  – Modeling & analysis toolset
  – Won NASA Space Act Award
  – Used successfully on 2nd generation RLV IVHM risk reduction program
Contingency Software in Autonomous Systems

Approach

1. TEAMS Model

Test:
- testLeftCameraNotTooDark

2. Diagnostic Tree auto-generated

Test:
- testRightCameraNotTooBright

Open Right Lens Cap

3. XML autotranslated to verify contingency handling on platform

Test:
- testRightCameraNotTooDark

Desaturate Right Camera

4

Test:
- testLeftCameraNotTooDark

No
Obstacle analysis approach

- KAOS framework for goal-oriented obstacle analysis [van Lamsweerde & Letier, 2000]
  - Goal is a set of desired behaviors
  - Obstacle is a set of undesirable behaviors that impede a goal
- Relevance to application:
  - Contingencies are
    - Obstacles to achieving goals, or
    - Indications that goals are unrealizable with available agents
- Advantages
  - Structured approach early-on (anticipatory planning)
  - Supports more formal analysis, as needed
Contingency Software in Autonomous Systems
Identifying contingencies & contingency-handling software actions

• Step 1. Identify the goals
• Step 2. Identify the agents
• Step 3. Identify the obstacles to the goals (these are the contingencies)
• Step 4. Identify alternative resolutions to the obstacles (the contingency-handling that can be done autonomously)
• Step 5. Select a resolution among the alternatives
• Requirements evolution
  – Use goal & obstacle analysis to refine requirements in a developing system [Anton & Potts]
• Maintenance
  – Focus on management of requirements changes [Bennett & Rajlich]
  – Evaluate in terms of traceability or change-impact [Cleland-Huang]
• Dynamic monitoring
  – Monitor operational systems for mismatch assumptions/environment & perform remedial evolutions [Fickas & Feather]
• Autonomous fault handling with AI planners [Brat et al., Chien et al.]
• Safety in autonomous systems [Fox & Das, ESA ESTEC]
• Vehicle health management [Patterson-Hine et al.]
Perception is a critical function in systems requiring obstacle avoidance, threat detection, science missions and “opportunistic” discovery.

SAS_06_Contingency_Lutz_Patterson-Hine_Tech_Briefing
Contingency Software in Autonomous Systems
Perception instrumentation onboard rotorcraft

- Gray scale wing tip (stereo vision)
- Left Wing
- Right Wing
- Scanning Laser Range Finder (SICK)
- Color Camera
- Firewire Hub
- Onboard Flight Computer
- RS232
- Firewire
• Cases in which the cameras are a critical system:
  – Cameras assigned responsibility during nominal ops
    • No line of sight -> Camera provides position info
  – Cameras are backup when other subsystems fail
    • Failed/degraded GPS -> Camera provides position info
    • Failed/degraded ARP -> Camera provides landing-site data
  – Images as mission objective (surveillance)
    • Failure of cameras can jeopardize success
• Thanks to Matt Whalley, Autonomous Rotorcraft Project Manager, & to Rob Harris, Chad Frost, Doron Tal, Stacy Nelson, Anupa Bajwa
What do we know when a “quit-failed” signal occurs?

Contingency Software in Autonomous Systems
Critical pointing for Mars spacecraft
Autonomous, contingency response for critical scenarios such as commandability loss, & before critical trajectory-correction maneuvers

Thanks to Tracy Neilson, MER/MSL
Contingency Software in Autonomous Systems
Emulates a typical spacecraft power system

Advanced Diagnostics And Prognostics Testbed (ADAPT)
Sensor Data and Control Command Flow Overview

Antagonist (ANT)
Avionics (AVI)
Observer Logger (LOG)
Data Acquisition (DAQ)
National Instruments Compact FieldPoint
User (USR)
Ethernet Switch

Thanks to Scott Poll, ADAPT manager
Contingency Software in Autonomous Systems

Preliminary identification of some required contingency plans

**Advanced Diagnostics and Prognostics Testbed (ADAPT)**

**Build 1 Software Architecture**

Notes:
- Future development in green
- Not yet implemented in blue
- Red circles are data sockets
- cFP is National Instruments Compact FieldPoint hardware

[Diagram of testbed architecture with network connections and software components]
Properties for each function, switch & test-point are entered into the TEAMS tools.

TEAMS builds a Dependency Matrix in which each row is a fault source (e.g., a camera that can fail) and each column is a test (e.g., whether we have a good Stereo image). Here, we select the normal or contingency scenario (camera OK or not) for the analysis.
Executing the Contingency scenario, we check that the behavior is correct: left COLOR camera is available (no red slash) & being used; confirm that tests can isolate failure to which camera.

Most useful: the automatic Diagnostic Tree:
--Shows best sequence of checks to detect & isolate
--Shows indistinguishable failures (“ambiguity groups”)

--XML output option auto- translated into rotorcraft’s planning language (APEX) to simulate contingencies on the vehicle
Contingency Software in Autonomous Systems

Potential applications

- Contingency management is essential to the robust operation of complex systems such as spacecraft and Unpiloted Aerial Vehicles (UAVs)
- Automatic contingency handling allows a faster response to unsafe scenarios with reduced human intervention on low-cost and extended missions
- Results, applied to the Autonomous Rotorcraft Project and Mars Science Lab, pave the way to more resilient autonomous systems
Investigate and model with TEAMS key contingencies involved in safe software reconfiguration of power distribution systems to support autonomous operations

Demonstrate and verify a subset of the contingency responses we have developed on available platforms

Support transfer to other NASA projects
• Improved contingency handling needed to safely relinquish control of unpiloted vehicles to autonomous controllers
• More autonomous contingency handling needed to support extended mission operations