Changing the way the world does software
The Cost of Autonomy

Colin O’Halloran
coh@drisq.com
Insurance

• Vehicles will need to have an acceptable safety case to get insurance
  – Cost prohibitive otherwise

• Over the horizon operations
  – Communication delays/interruptions
  – Operator cognition in doubt
  – If 100% operator cognition required there is a cost in training & monitoring & user acceptance.

• Assurance of behaviour required
Autonomy vs Pilots

• Avionics support pilot through automation
  – Pilot typically alerted to failures
  – Training to cope with failures
  – Pilot expected to have 100% environmental cognition
  – Even then Eurofighter's automated carefree handling required high assurance.

• Qantas QF32
  – Catastrophic engine failure
  – Multiple systems failures – plenty of warnings!

• Autonomous systems will have to be able to cope
  – Cannot just dump information onto the [remote] pilot
The Problem – RAND Study (Cars)

- Current fatality rate (USA) is 1.09 fatalities per 100 million miles
- Say we have a fleet of 100 autonomous vehicles driving 24 hours a day, 365 days a year, at an average speed of 25 miles per hour
- Need to demonstrate with 95% confidence their failure rate to within 20% of the death rate (1.09/100million miles)
- How many miles (years) would the autonomous vehicles have to be driven?
- Answer: 8.8 billion miles, which would take 400 years with such a fleet
- Conclusion: “...developers of this technology and third-party testers cannot drive [read ‘test’ or ‘simulate’] their way to safety.”
- Alternative methods to supplement real-world testing [are needed] in order to assess autonomous vehicle safety and shape appropriate policies and regulations
Assuring the ‘System’

• Have a set of requirements/behaviours
  – Must do  Routine to achieve using ‘conventional’ techniques
  – Must NEVER do Not achievable using ‘conventional’ techniques
  – Failure behaviour Very expensive using ‘conventional’ techniques

• This sets boundaries for the vehicle behaviour
Conventional Approach

• Some modelling may be done
  – Perhaps using tools such as Simulink/Stateflow
• Often code is developed first
  – Applying retrospective assurance is difficult [at best]
• Documentation can be sparse
• Programming language selection often fashion based
• Tested, and then some more, and then...
  – The cost of change is proportional to the size of the system
  – Not the size of the change
  – Leads to extended project time & budget overrun
• Support to safety case...tenuous?
Safety Critical Systems

• Require verification at all stages of development

• Evidence set by domain standard
  – ISO26262 automotive
  – DO-178C/ED-12C aerospace
  – …etc, but none for maritime systems

• Typically require evidence generated based upon requirements and verification of implementation
Standard Selection

- No software standard specific to safety of autonomous systems
- Aerospace DO-178B:
  - Over 20,000 certified jet aeroplanes in service worldwide since 1992
  - No hull-loss accidents in passenger service have been ascribed to software
  - DO-178C released in 2011 and authorised for use 2012
  - Verification cost: 50-80% of software production.
  - Prerequisite for Autonomous Air Vehicles – sets a baseline for autonomy?
Standard Selection

- Automotive ISO26262:
  - Only lower Levels (ASILS) typically or/and very simple systems
  - Tiny experience for autonomous driving relative to fleet of all cars
  - Of the reported crashes where an autonomous car has hit something it ‘shouldn’t have’:
    - Autopilot isn’t an autonomous system and the driver should always be ready to take control
    - Autopilot should only be used on highways where pedestrians and cyclists are not present as it can’t always detect them

Significant environment constraint

Give up – over to you! 😊
Need new approach to assurance

• We build systems beyond our ability to understand them
  – Space Shuttle
  – Ariane 5
• We do not understand the consequences of our high level designs (models) and low level designs (software).
• Assurance is a limiting factor for adoption of autonomous systems.
• If the assurance doesn’t matter then the market is significantly smaller.
• Conventional simulation and testing is expensive and insufficient to provide evidence of absence of unsafe/undesirable behaviour.
Automating & Hiding Formal Methods

• D-RisQ enabling users to extract benefit without expertise in formal methods
• Used for systems and software analysis against requirements (Modelworks®)
  – Automotive sector: Electro/hybrid vehicle system
  – Aerospace: Undercarriage system; Display
• Used for source code verification (CLawZ®)
  – Verified 350,000 Lines of Code
• Also used for binary verification (FEVER®)
  – In development
• Formal Methods Supplement to DO-178C [DO-333]
The USMOOTH Problem

- IUK Project led by ASV Ltd.
- How to design a decision making system that can be support Unmanned Systems (Maritime) Operations Over The Horizon for extended durations (weeks)?
- How to provide assurance that the software does what is required and nothing else?
  - How could we support a safety argument
- What standards could be used?
  - Not necessarily maritime standards
- At what cost and can the process be easily [cheaply] repeated?
Decision Making System

Decision Making

Target information
- range, range_rate
- bearing, bearing_rate

Human input (Override)

Own position
- speed, bearing

New heading
- New speed

Steering
- Propulsion
Systems, Software and Certification

- **System Requirements**
- **Software Requirements**
- **Software Design**
- **Source Code**
- **Executable Object Code**
- **Processor**

- **Table A-2 Objectives**
- **Table A-3 Objectives**
- **Table A-4 Objectives**
- **Table A-5 Objectives**
- **Table A-6 Objectives**
- **Table A-7 Objectives**

**DO-178C**

- Modelworks®
- CLawZ®
- FEVER®
- USMOOTH

**Tools and Standards**

- Modelworks®
- CLawZ®
- FEVER®
- USMOOTH

**DO-178C**

- Table A-2 Objectives
- Table A-3 Objectives
- Table A-4 Objectives
- Table A-5 Objectives
- Table A-6 Objectives
- Table A-7 Objectives
Summary

- Defining behaviour in English
  - Easy for all to understand
  - Translate to maths (hidden)

- Defining specification in model
  - Simulink/Stateflow
  - Translate to maths (hidden) and check against requirements

- Auto-translate model to source code
  - Independently automatically prove against model

- Compile to Executable Object Code
  - Independently automatically prove against source code
  - NB stretch target

- Use of DO-178C/DO-333 provides evidence to support safety case

- Automation makes adaptation easy, i.e. cost effective