Changing the way the world does software
Automated Certification from Soup to Nuts

Nick Tudor
njt@drisq.com
Introducing D-RisQ

• D-RisQ is a SME based in Malvern, UK.
• All personnel have a background in mathematics, engineering and computer science
  – Huge experience in analysis of complex systems and software across many sectors
  – Safety and security critical systems, automotive, aerospace, robotics...
  – Build automated formal analysis tools
  – Focus on cost/time reduction
  – Builds upon the use of commercially available and widely used development tools
D-RisQ FM Ecosystem

Verification

- Design
  - CSP
- Protocols
  - CSP
- Packet Analysis
  - CSP

Formal Methods

- Requirements
- English
  - CSP
- Design
  - CSP
  - Simulink/Stateflow
  - SysML
- Networks
  - CSP
  - Z
- Source Code
  - CSP
  - HOL
- Object Code
  - CSP
  - HOL
  - Z

- Viruses
  - CSP
  - HOL
- EOC
  - CSP
  - HOL
- Vulnerability
  - CSP
- Ada
  - Z
- C
  - Z
  - HOL
D-RisQ FM Ecosystem

Verification

Design

Protocols

Packet Analysis

CSP

Requirements

Modelworks®

Formal Methods

CSP

CLawZ®

FEVER®

Object Code

Source Code

Networks

Simulink/Stateflow

SysML

CSP

CSP

Z

CSP

CSP

Z

Z

CSP

CSP

Ada

C

CSP

HOL

CSP

HOL

CSP

Z

Z

HOL

Viruses

EOC

Vulnerability

CSP

HOL

CSP

HOL

CSP

Z

Z

HOL
The USMOOTH Problem

• Project led by ASV

• How to design a decision making system that can support **Unmanned Safe 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?
  – NB This is safety critical software
Acceptable Behaviour

Behaviour Boundary

Defined by:
User requirements
Regulations
Physics
Environment

• Acceptability demonstrated by:
  • Probing the boundary and
  • By design/development processes

• Aiming to ensure:
  • Nothing can penetrate or leak from the boundary
  • And everything inside is done correctly

Everything in here must be acceptable

The more complex, the more cost to verify conventionally
Software Engineering,
Formal Methods and Certification
Verification Objectives – DO-333

FM.A-3.2 Accuracy & Consistency
FM.A-3.3 HW Compatibility
FM.A-3.4 Verifiability
FM.A-3.5 Conformance
FM.A-3.7 Algorithm Accuracy

FM.A-4.8 Architecture Compatibility

FM.A-5.3 Verifiability
FM.A-5.4 Conformance
FM.A-5.6 Accuracy & Consistency
FM.A-5.8 PDI Complete & Correct
FM.A-5.9 PDI Verified

FM.A-5.10-13 Additional Objectives

FM.A-7.9 Property Preservation

FM.A-7.10 Additional Objectives

FM.A-3.8-11 Additional Objectives

Compliance: with requirements
Conformance: with standards
Verification Objectives – DO-333

Software, Certification and Costs

System Requirements
  (FM.A-2: 1, 2)

High-Level Requirements
  (FM.A-2: 3, 4, 5)

Software Architecture

Software Design
  (FM.A-2: 6)

Source Code
  (FM.A-2: 7)

Executable Object Code
Software, Certification and Costs

System Requirements

Software Requirements

Software Design

Source Code

Executable Object Code

Processor

DO-333

Table FM.A-2 Objectives

Table FM.A-3 Objectives

Table FM.A-4 Objectives

Table FM.A-5 Objectives

Table FM.A-6 Objectives

Table FM.A-7 Objectives

Development cost: 20-50%

Verification cost: 50-80%
IT STARTS WITH REQUIREMENTS
We Need Unambiguous Requirements

• Must be able to bound the required behaviour in a form that:
  – Enables it to be agreed that it is the ‘required behaviour’
  – Language understood by regulator and developer

• The language must enable formalisation
  – Enable checks for conflicting requirements

• Requirements language should allow ‘abstraction’
  – Minimal assumptions about architecture
  – Enables a wider possibility of solution
Software and Certification - USMOOTH

LRE System Requirements Document (SRD) v1.0

DO-333

Table FM.A-2 Objectives
Table FM.A-3 Objectives

LRE Software Requirements Specification (SRS) v1.0

Software Requirements

Software Design

Source Code

Executable Object Code

Processor
Software and Certification - USMOOTHE

- System Requirements
- LRE System Requirements Document (SRD) v1.0

Reviewed

Specifications and Design:
- Software Requirements
- DO-333
- Table FM.A-2 Objectives
- Table FM.A-3 Objectives

- Source Code
- Executable Object Code
- Processor

LRE Software Requirements Specification (SRS) v1.0
SOFTWARE DESIGN AND REQUIREMENTS VERIFICATION
Systems, Software and Certification

- System Requirements
- LRE Software Requirements Specification (SRS) v1.0
- DO-333
  - Table FM.A-2 Objectives
  - Table FM.A-3 Objectives
  - Table FM.A-4 Objectives
- Software Requirements
- Software Design
- LRE Simulink/ Stateflow Software Design v1.0
- Source Code
- Executable Object Code
- Processor
- Reviewed

- Reviewed
- Reviewed
Independent Experiments – mid Feb ’17

Now can being analysed by MW

More automation of Modelworks due shortly
SOFTWARE SOURCE CODE
VERIFICATION AGAINST DESIGN
Overview of the D-RisQ CLawZ® Process

- Simulink
- Autocoder
- Code
- User Interface
- Proof using Theorem Prover ProofPower
- Z
- Z Producer

Development

Verification

Independence
Software Source Code

Reviewed

LRE Simulink/ Stateflow
Software Design v1.0

DO-333

Table FM.A-2 Objectives
Table FM.A-3 Objectives
Table FM.A-4 Objectives
Table FM.A-5 Objectives

System Requirements

Software Requirements

Software Design

Source Code

CLawZ®

Autocoded using dSpace TargetLink to C

Modelworks®

CLawZ for C – not yet fully developed; Verification not done

Executable
Object Code
Processor
EXECUTABLE OBJECT CODE
VERIFICATION
Executable Object Code

System Requirements

Software Requirements

Software Design

Autocoded using dSpace TargetLink to C

DO-333

Reviewed

Modelworks®

CLawZ®

Source Code

Table FM.A-2 Objectives

Table FM.A-3 Objectives

Table FM.A-4 Objectives

Table FM.A-5 Objectives

Table FM.A-6 Objectives

Compiled to ARM

Executable Object Code Processor
D-RisQ FEVER™

Formal Executable object code VERification

Source Code → Compiler → Executable Object Code

User Interface

Formalised in HOL

Processor

Instruction Set Architecture

Formalised in HOL

Proof using Theorem Prover ProofPower (HOL)

Target is currently ARM

Work in progress
COVERAGE
Options for verification activities of Executable Object Code include:

1. Formal analysis of Executable Object Code can be used to satisfy the objectives if all of the following conditions are satisfied:
   - The requirements are formally expressed.
   - A formal model of the Executable Object Code is defined.
   - Formal evidence demonstrates that the formal model of the Executable Object Code satisfies the requirements.

2. Formal analysis of Source Code can be used to satisfy the objectives if all of the following conditions are satisfied:
   - The requirements are formally expressed.
   - A formal model of the Source Code is defined.
   - Formal evidence demonstrates that the formal model of the Source Code satisfies the requirements.
   - Complementary analysis shows property preservation between the Source Code and the EOC.
FMA-6 Our Approach

System Requirements

High-Level Requirements

Development Activity
Review/Analysis Activity
Test Activity
Formal Analysis

Compliance
Robustness

Modelworks®

Low-Level Requirements

CLawZ®

Software Architecture

Design

Compliance
Robustness

Formal Proof of Property Preservation

Source Code

FEVER®

Executable Object Code
Executable Object Code

System Requirements

Software Requirements

Software Design

Source Code

Executable Object Code

Processor

DO-333

Reviewed

Modelworks®

CLawZ®

FEVER®

Table FM.A-2 Objectives

Table FM.A-3 Objectives

Table FM.A-4 Objectives

Table FM.A-5 Objectives

Table FM.A-6 Objectives

Table FM.A-7 Objectives
COSTS
Locking in IP and Reducing Cost

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

DO-333

- Table FM.A-2 Objectives
- Table FM.A-3 Objectives
- Table FM.A-4 Objectives
- Table FM.A-5 Objectives
- Table FM.A-6 Objectives
- Table FM.A-7 Objectives

Automatic = Significant cost reductions

IP effort here
Summary

• Unmanned systems have to have a high integrity autonomous decision making system
• The business case for wider adoption is based on fast, cost effective assurance
• Automated formal proof from requirements to binary tools being built
• USMOOTH is showing that this is possible
  – Regulatory approval of safety case and approach using DO-333 as we propose, still required
• Demonstration to follow in 2017
Thank you

D-RisQ
SOFTWARE SYSTEMS

Changing the way the world does software