An Outline Workflow for Practical Formal Verification from Software Requirements to Object Code

Formal Methods in Industrial Control Systems (FMICS 2013)

Darren Sexton BSc. MSc. CEng (MIET)

Agenda

- **Introduction**
- Overview of work-flow
- Observer approach
- Conclusions

Context

Ricardo

- Global engineering consultancy
- Working in multiple domains
  - Automotive, off-highway, motorsport, rail, clean energy, defence…
- Engineering skills across many disciplines
  - Not just software
- Expertise is in engineering solutions
  - Not in formal methods
- Interested in how formal methods can:
  - Deliver high-quality
  - Support safety critical projects
  - Reduce effort

MBAT

- Model-Based Analysis & Test
  - Focussed on combination of analysis & test
  - Focussed on “near-term” research
- ~ 40 European organisations
  - Industrial end-users
  - Tool vendors
  - Research institute
- Currently ~ two years into three year programme

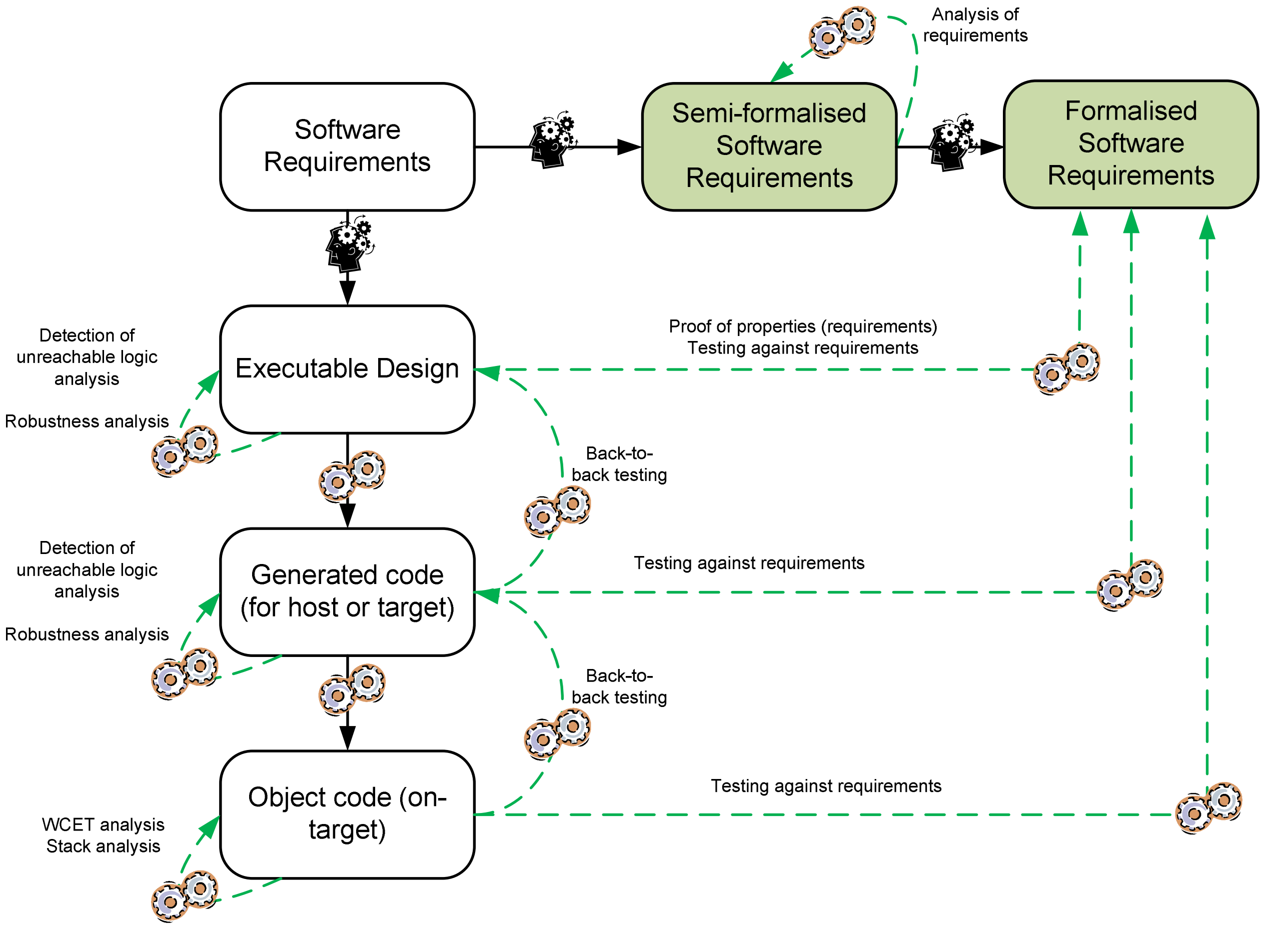
Agenda

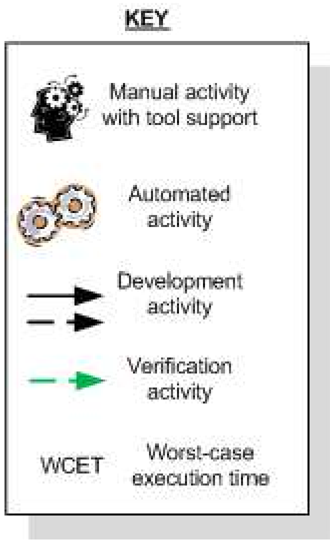
- Introduction
- **Overview of work-flow**
- Observer approach
- Conclusions

Overview of work-flow

Feedback loops not shown for clarity

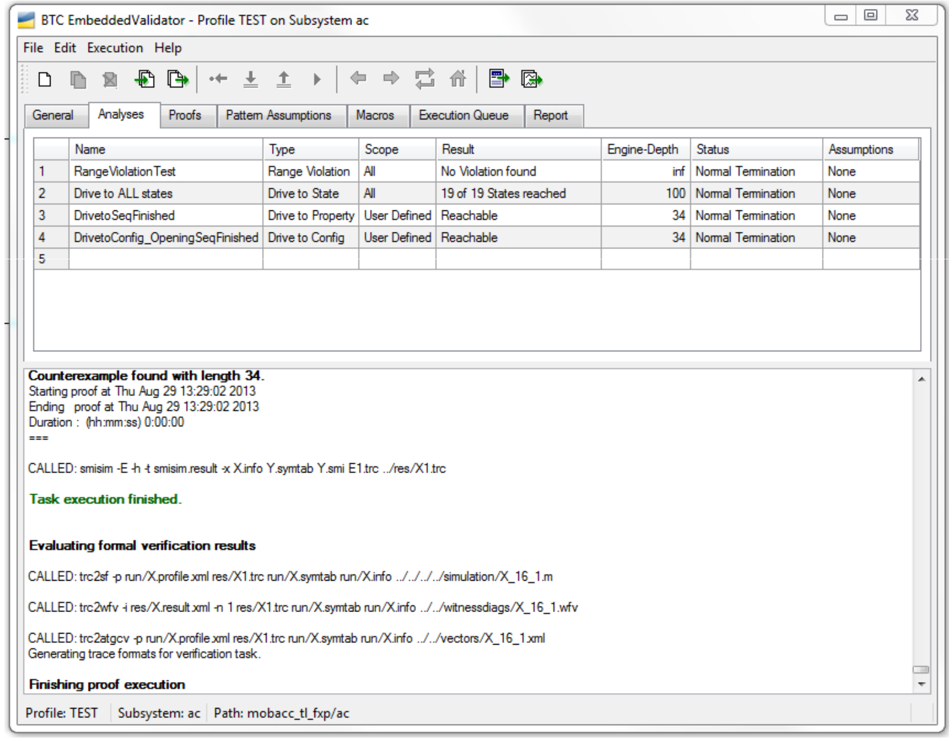
Normal V&V activities (e.g. peer review) not shown for clarity





Health / robustness checks on model

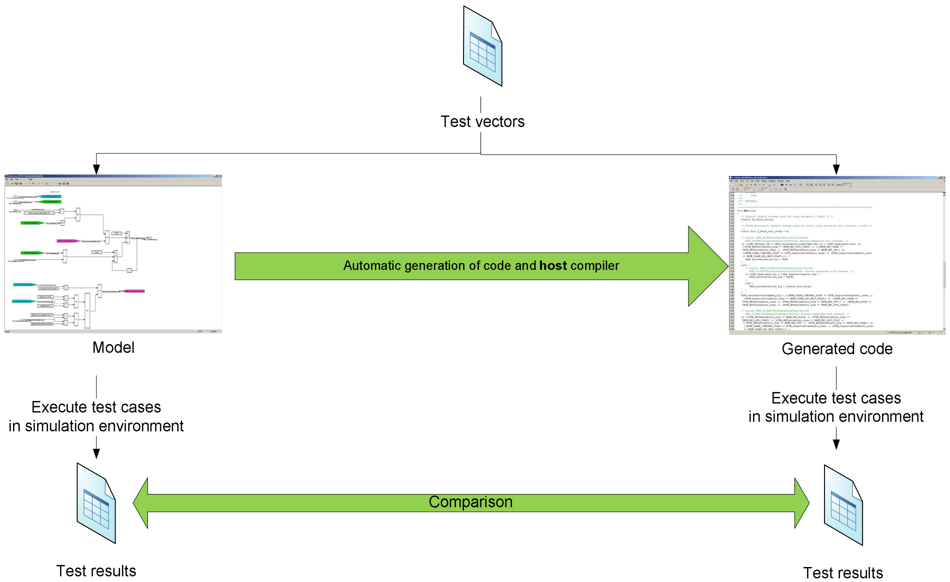
- **Objective**: Detect requirement-independent problems in model
  - E.g. Unreachable states, signal range checks, drive to specific outputs etc.
- **Approach**: Model checking techniques
- **Pre-requisites**:
  - Implementation model in TargetLink
- **Potential benefits**:
  - Eliminate basic errors *during model construction*
    - Thus reduce debugging time of later verification activities



Screenshot of defining basic health / robustness properties

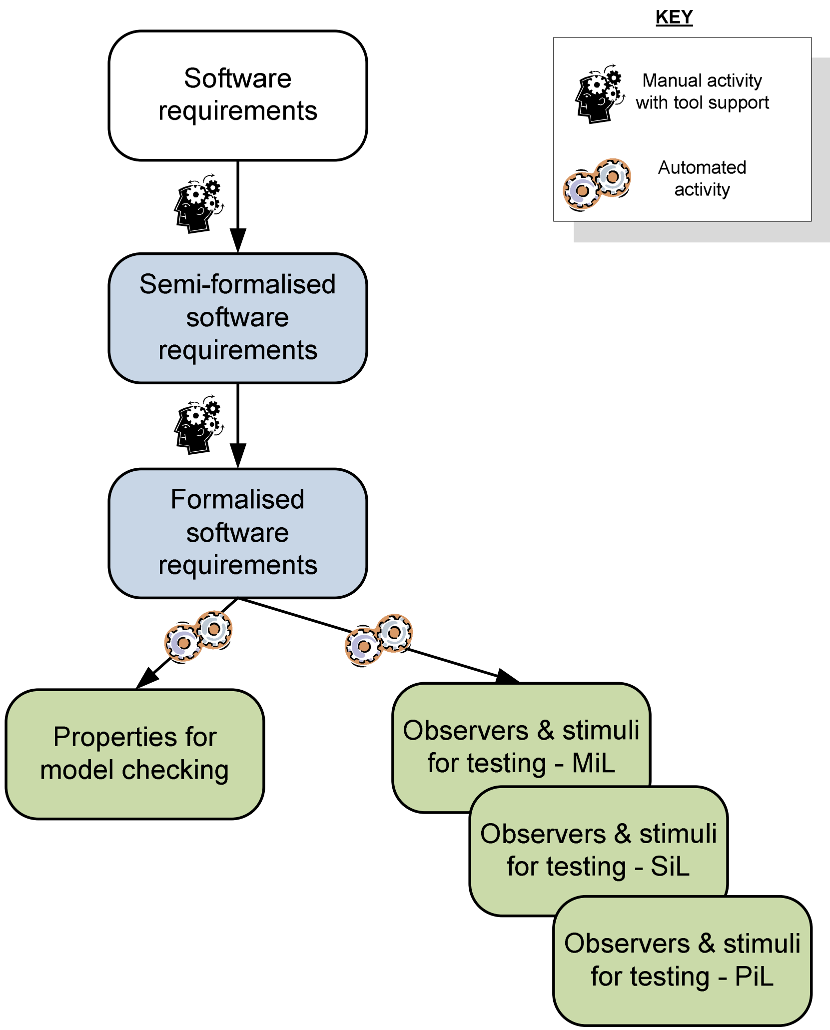
Automated back-to-back testing

- **Objective**: Gain confidence that generated code & object code matches models
- **Approach**:
  - Automated test stimuli generated to achieve high-structural coverage
  - Automated comparison of outputs in different environments (with tolerance)
  - Can be performed in advance of running requirements based tests
- **Pre-requisites**:
  - Implementation model in TargetLink
- **Potential benefits**:
  - Rapid indication of scaling errors, data-type issues, code generator / compiler errors *during model construction*



Requirements formalisation

- **Objective**: Translate natural language requirements to a notation:
  - With fully defined syntax and semantics
  - That can be used to support later verification activities (via ‘observers’)
- **Approach**: Tool support to map to patterns
- **Pre-requisites**:
  - (Semi-formal) Well structured natural language requirements
  - (Formal) Implementation models
- **Potential benefits**:
  - Improve requirements quality
  - Generation of ‘observers’ to support later analysis and testing activities



Use of formalised requirements as basis for analysis & testing

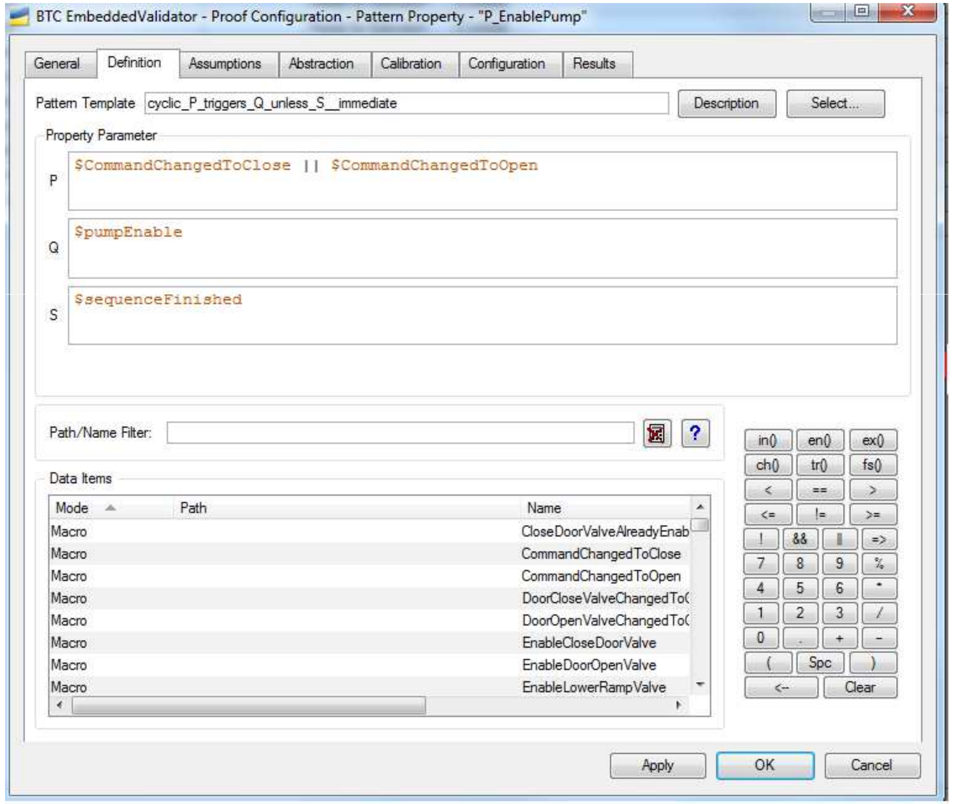
Proving formalised requirements

- **Objective**: *Prove* the implementation model complies with the formalised requirements

- **Approach**:
  - Import of patterns from formalised requirements phase
  - Model checking

- **Pre-requisites**:
  - Formalised requirements
  - Implementation model in TargetLink

- **Potential benefits**:
  - Rapid feedback to identify issues with implementation or formalisations
  - Witness trace for debugging where model violates requirements



Screenshot of defining a property to prove

Testing formalised requirements

- **Objective**: *Test* implementation model complies with the formalised requirements
- **Approach**:
  - Automatic generation of test vectors to test requirements (via 'observers')
    - *Requirements based* testing & analysis
    - Test vectors to drive signal ranges etc.
  - Running of tests in MiL, SiL, PiL environments
- **Pre-requisites**:
  - Formalised requirements
  - Implementation model in TargetLink
- **Potential benefits**:
  - Confidence in implementation (model, generated code, cross-compiler)
  - Reduce testing effort
  - Detailed measurement of requirements coverage, detect missing requirements

Observer Result Summary

Subsystem	External ID	Observer ID	Status
mobacc_tl_fxp/ac/Subsystem/ac	n.a.	CObserver1	fulfilled
	n.a.	CObserver10	fulfilled
	n.a.	CObserver11	fulfilled
	n.a.	CObserver12	fulfilled
	n.a.	CObserver2	fulfilled
	n.a.	CObserver3	fulfilled
	n.a.	CObserver4	fulfilled
	n.a.	CObserver5	fulfilled
	n.a.	CObserver6	fulfilled
	n.a.	CObserver7	fulfilled
	n.a.	CObserver8	fulfilled
	n.a.	CObserver9	fulfilled

Screenshot of requirements based test results

Agenda

- Introduction
- Overview of work-flow
- **Observer approach**
- Conclusions

Example: Natural Language to Semi-Formal Requirement

Identify key parts of the requirement

The [...] feature shall immediately disable the pump (until power-off & on) when the emergency stop button is depressed (e-stop input goes high)

Natural language

Example: Natural Language to Semi-Formal Requirement

Map key parts to pattern

The [...] feature shall immediately disable the pump (until power-off & on) when the emergency stop button is depressed (e-stop input goes high)

Natural language

Condition that triggers the action: “emergency stop button is depressed” – rising edge

The action: “disable the pump”

P_implies_finally_globally_Q_B

Semi formal

When the action must happen in relation to condition: “immediately”....

... But in reality we need to allow a small tolerance (justified by safety analysis)

Action latches: “until power-off & on”

Example: Semi-Formal to Formal Requirement

Map key parts to variables & expressions in the code

P_implies_finally_globally_Q_B

tr(acd_flg_eStop == TRUE)

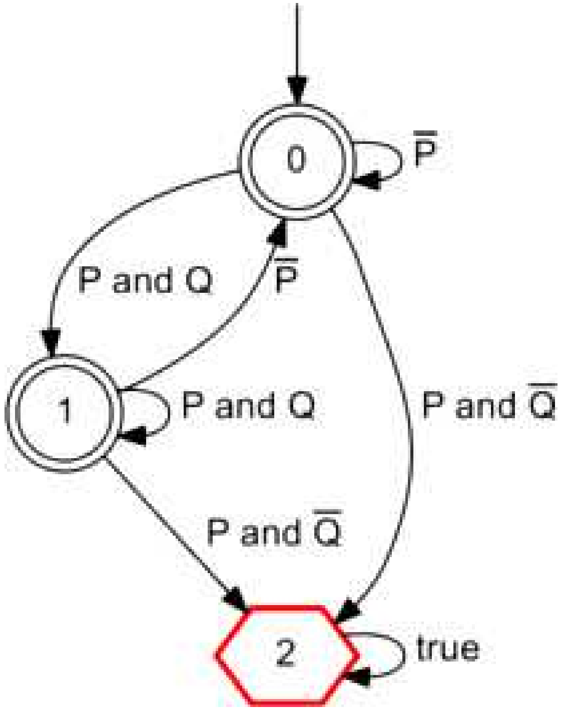
Built-in expression to detect rising edge

acd_flg_runPump == FALSE

Formal

Underlying formalism

- Formal notation uses patterns
  - Based on underlying notation of Büchi-Automaton charts
  - Capable of expressing LTL and more
- Engineers typically expected to select pattern based on names
  - Rather than having to examine underlying charts
- In practice:
  - Use of “boilerplates” to reduce gap between natural language requirements & patterns
  - Critical to provide systematic guidance for pattern selection
  - Necessary to refer to charts when debugging or deciding between several potential choices

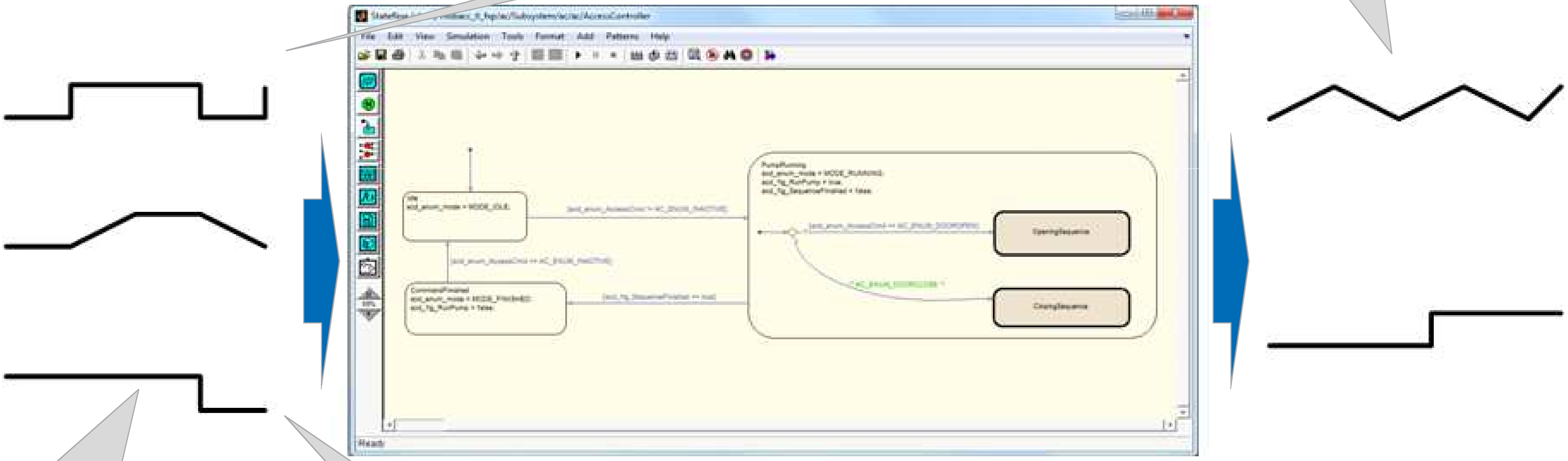


Example Büchi-Automaton chart for the pattern “cyclic_Q_while_P__immediate from BTC-EmbeddedSpecifier

Observer based testing & analysis

Test stimuli automatically generated from observers...

Observers monitor inputs and outputs to provide PASS / FAIL criteria



Feature under Test

... Can be extended based on implementation to achieve high-structural coverage, coverage of signal ranges etc...

... And limited by assumptions (e.g. rate of change)

Challenges & benefits of observer approach

Benefits

- ✓ Potential reduction in effort in verification
  - Rapid feedback from model checking
  - Reduction in human effort for test stimuli generation
- ✓ Verification is against formal requirements
  - "Formal verification"?
- ✓ Improved consistency of verification activities?
  - E.g. Reduce differences in testing style between test engineers

Challenges

- × Formalisation relies on appropriate style of natural requirements
  - So, must modify requirements writing process
- × Selecting correct patterns and...
- × ... ensuring consistent selection of patterns
  - So, must provide systematic guidance
- × Handling minor tolerance issues
  - So, must select tolerant patterns
  - Need some tool enhancements
- × Common cause failures between implementation and verification
  - So, must ensure other parts of process can detect these
- × Not appropriate for all types of functionality

Agenda

- Introduction
- Overview of work-flow
- Observer approach
- **Conclusions**

Conclusions

- Outline work-flow presented based on-going research programme
  - We have strong focus on what we can realistically deploy
  - Combining analysis & test to get confidence at different times

- Approach shows promise
  - But many challenges remain

- General view among team that formal approach increases initial effort
  - But provides higher quality
  - Potential for reduction in effort
    - Through later savings (less rework etc.)
    - Automation of testing?

- Formal approaches must focus on being “engineer friendly” to gain wide-spread adoption within automotive industry

Acknowledgements

The research leading to these results has received funding from the EU ARTEMIS Joint Undertaking under grant agreement n° 269335 and the UK Technology Strategy Board.

The author would like to thank Peter Gilhead and Rashiqua Quadir from Ricardo for their input.

