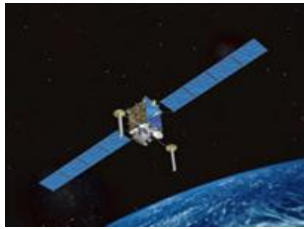


Unit Testing Made Easy for Safety Critical Software



Testing Compliance Challenges

Which regulation

- Mandated or highly recommended
- Variation by SIL

What and where

- From unit tests to system test
- Native and on-device

Techniques

- Functional requirements testing
- Test case design
- Structural code coverage

Certification

- Test evidence needed
- Demonstration of tool suitability



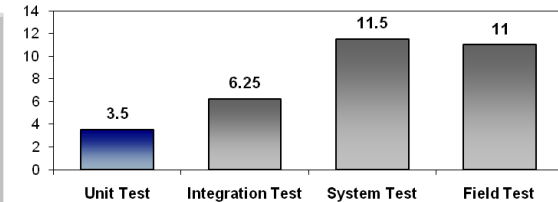
Balancing Compliance and Cost



Test platform



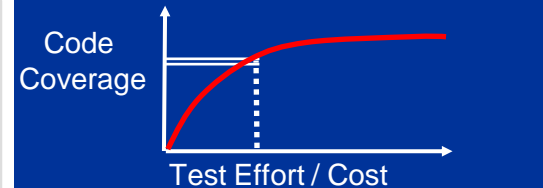
Levels of test



Test techniques



Coverage achieved by tests



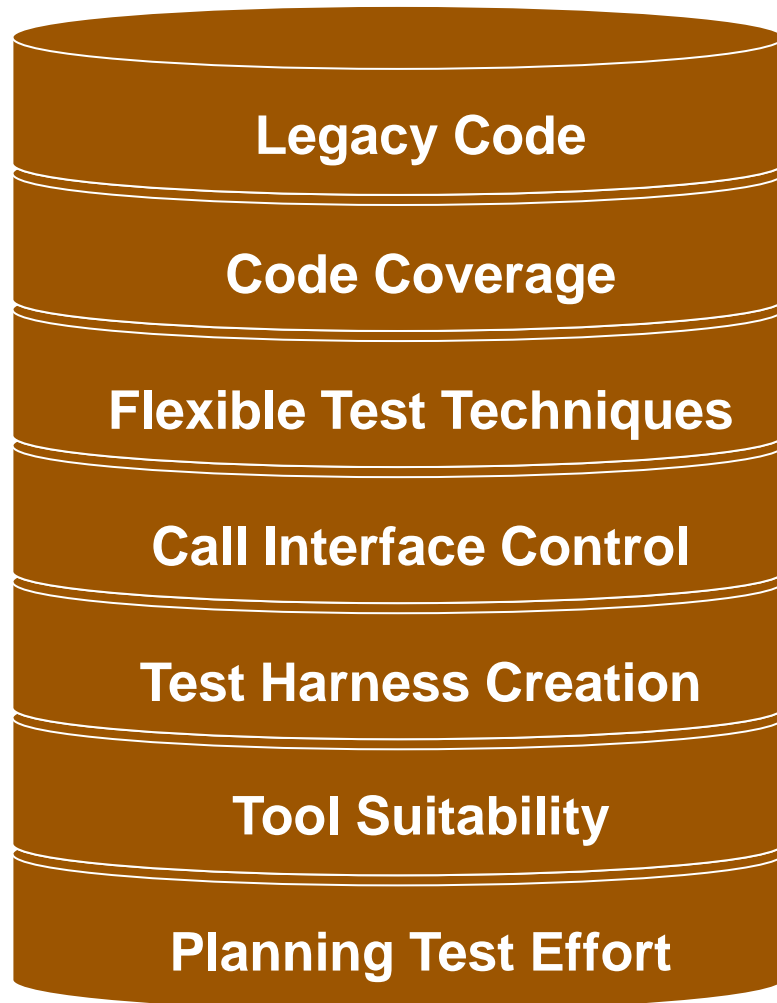
Test tool suitability / certification



Intelligent Test Automation



7 Automation Challenges for Test Tools



➤ Needing to create unit

➤ Confidence that my

➤ Having a full tool-kit

➤ Making it simple for

Knowing where to
focus my test effort
and how much work
it will be

standards use

Developer
Needs

Technical
Challenges

Intelligent
Solutions



BUILT ON
eclipse.

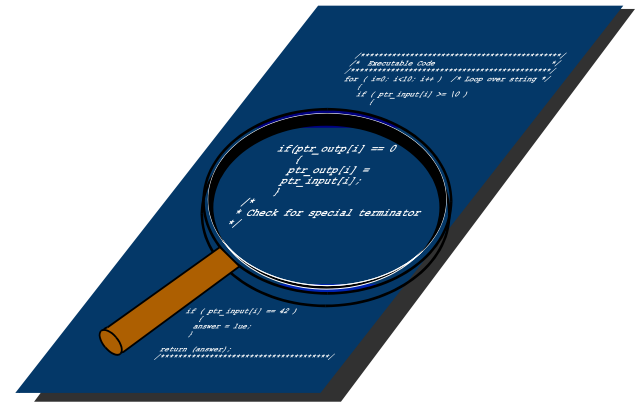


Cantata++

intelligent testing

1 Planning Test Effort

- Analyze code before testing
- Identify maintainability
- Aid test planning



➔ **Cantata++** measures...

● Procedural and OO Metric Sets

Myers

McCabe

Halstead

Hansen

McCabe Object Oriented

MOOSE

MOOD

QMOOD

Robert Martin

Bansiya's Class Entropy

● Reports in .csv format

2 Tool Suitability

- Reliability and maintainability
- Proven high integrity market use
- Integration with environment
- Support for certification against standards



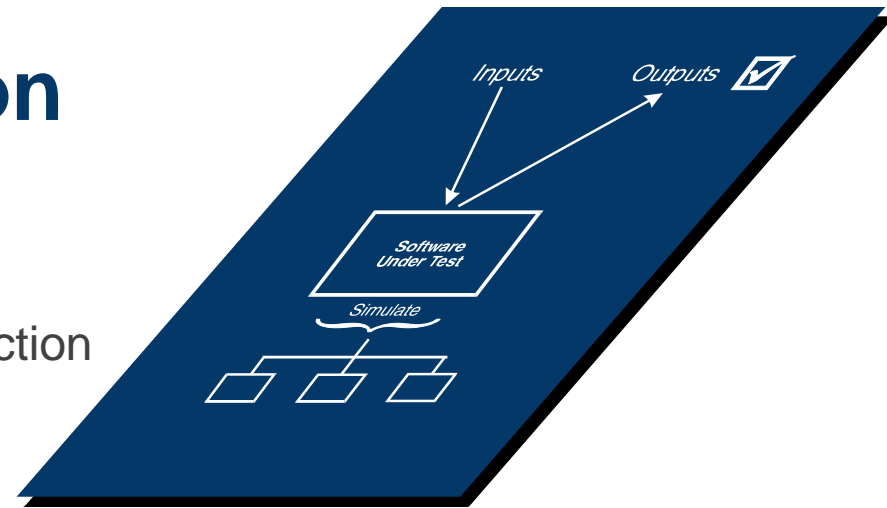
➔ **Cantata++** reassures with...

- Tool development and maintenance under certified QMS
 - DO-178B tool qualification package
 - Certification to other standards
- Used in all safety related markets
- Multi-target deployment technology built-in
- Standard specific guidance papers & safety manual
- Automated certification results
- Tool qualification/certification kit



3 Test Harness Creation

- Test driver harness for units
- Structured auto-repeatable test cases
- Automated checking and results production



➤ **Cantata++**

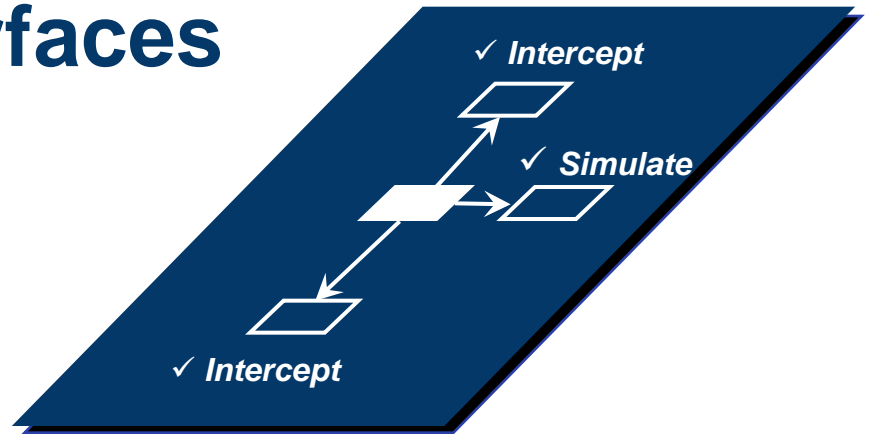
∞ automates through ...

- Parse of project code
- Automated structured test script
 - ☑ case per function/ method
 - ☑ parameters, call order, global data, returns
 - ☑ test case independence
- Full featured check library
- Positive and negative checking of global data
- Automated results production



4 Control of Call Interfaces

- Isolation of units from system
- Order of calls
- Different behavior on calls
- Checking/modifying parameters



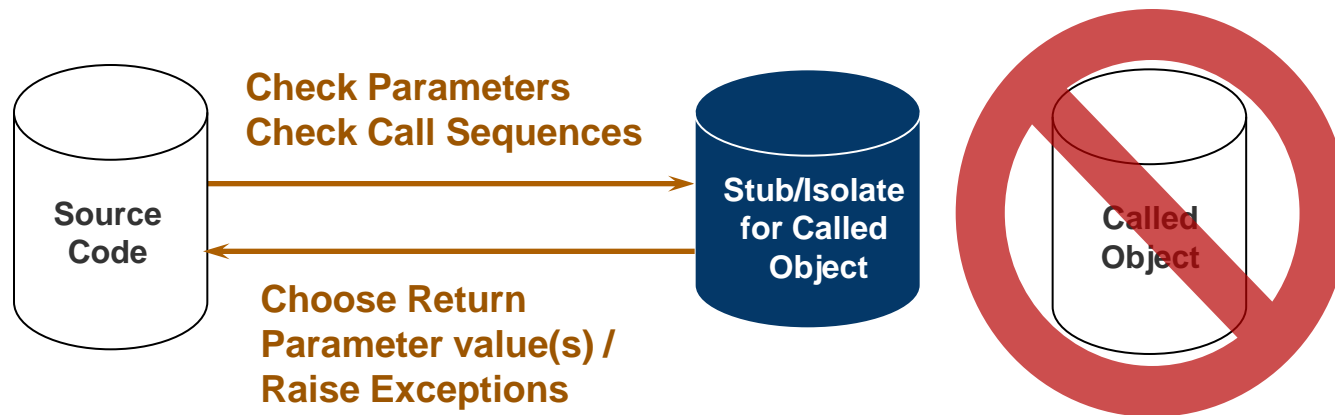
Cantata++

enables control by...

- Knowledge of calls made
- Automated generation of
 - 🔍 Stubs and Isolates to simulate calls
 - 🔍 Wrappers to intercept calls
- Programmable instances for each call
- Flexible call sequence control
- Automated checks on parameters

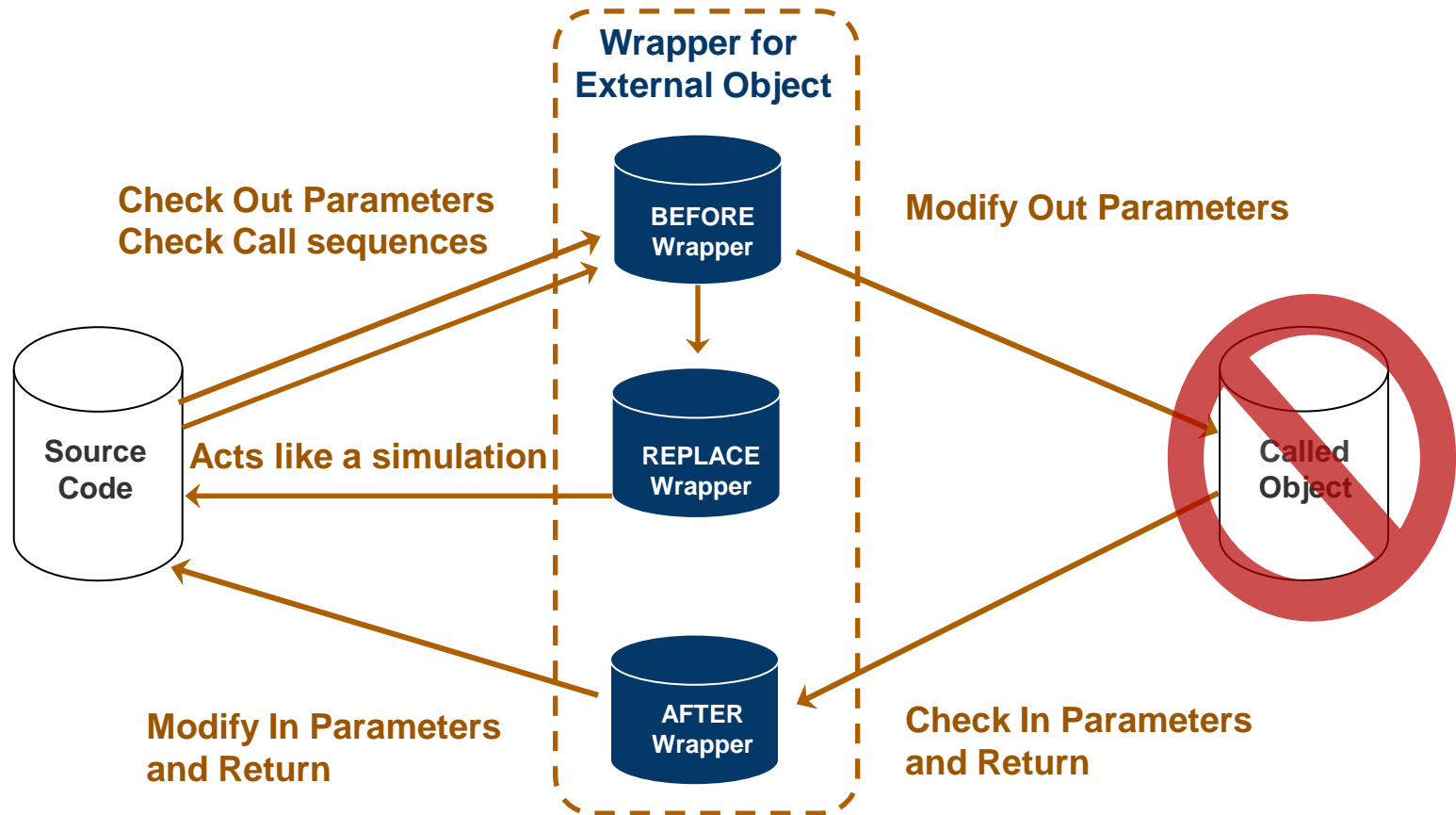
***Cantata++* Stubs and Isolates**

- A function/method inside test script with programmable instances
- Stub replaces called object at link time
- Isolate replaces specific calls to called objects at link time



Cantata++ Wrapping

- A function/method inside test script with programmable instances to select Before-After or Before-Replace Pairing
- Intercepts specific calls to called objects at link time

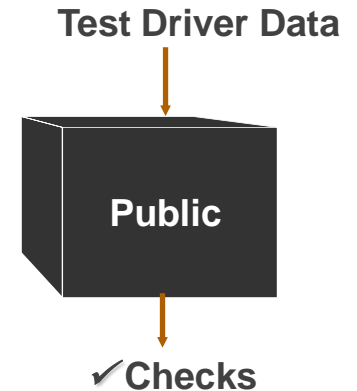


Call Interface Control Flexibility

Action	Stub	Isolate	Wrap
Called object simulated	Must	Must	✓
Called object included in link	-	✓	Must
Check inputs to called object	✓	✓	✓
Check outputs for called object	-	-	✓
Check call order	✓	✓	✓
Check or Set any data when object called	✓	✓	✓
Check or Set any data when object returns	-	-	✓
Set outputs from called object	Must	Must	✓
Modify inputs to called object	-	-	✓
Modify outputs from called object	-	-	✓
Control system calls	-	✓	✓
Control variadic function calls	✓	✓	-
Control class method calls	✓	-	✓

5 Flexibility: Black Box Testing

- Large data input sets
- Checking large output sets
- Robustness tests

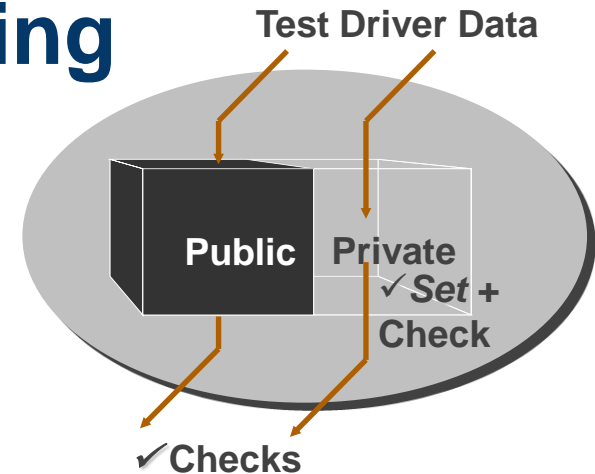


➔ ***Cantata++*** simplifies with...

- Table-driven test generation
 - ☑ Multiple values per parameter
 - ☑ Ranges of values
 - ☑ Functions calculating values
 - ☑ Combinatorial effect calculator
 - ☑ Checks on call sequences and returns
- Robustness rule sets for data types

5 Flexibility: White-Box Testing

- Call private methods/static functions
- Set/check private/static data
- Control of internal calls



➔ **Cantata++**

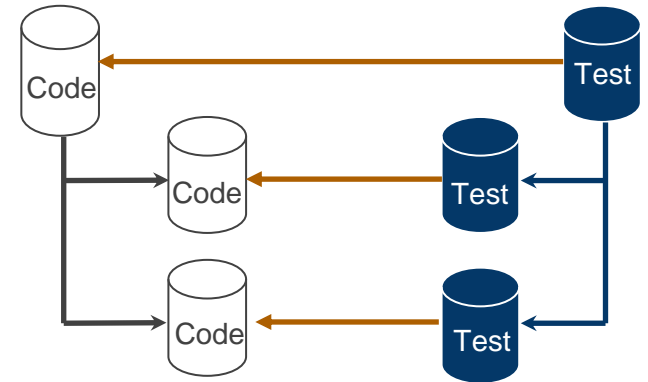
solves the problem through...

- Automated accessibility instrumentation
 - 🔍 Call private methods and static functions directly
 - 🔍 Set/check data which is private, in unnamed namespaces and declared static
- White-box Call Interface Control
 - 🔍 Calls internal to compilation unit
 - 🔍 Wrapping / Isolating OS library calls

5 Flexibility: Object Oriented Testing

- Test case re-use aligned with code
- Support for Templates
- Support for Inheritance
- Testing Abstract Base Classes

⑩ ●◆□



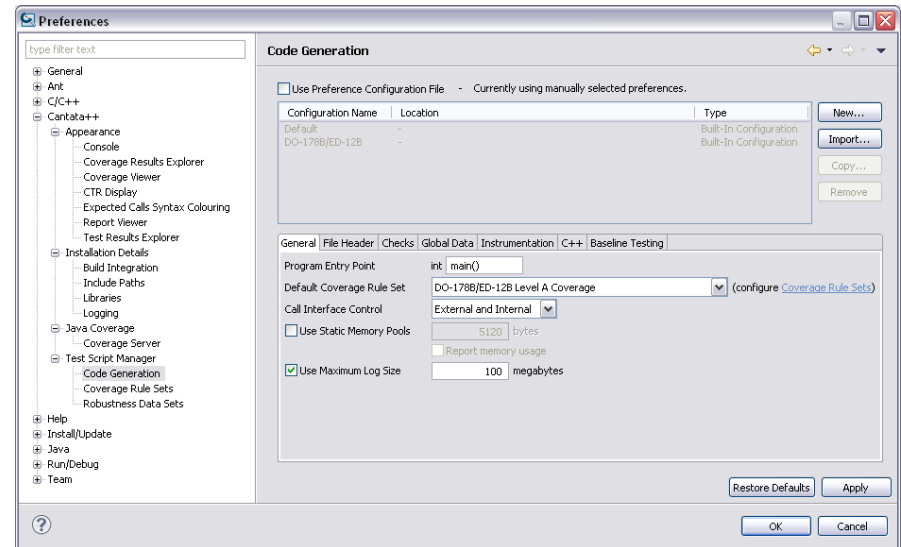
➔ ***Cantata++*** supports these with...

- Parallel hierarchy of code and test case re-use
 - ☑ Template instantiation
 - ☑ Inheritance and factory methods
- Abstract Base Class testing
- Object oriented code coverage
 - ☑ Derived inheritance context
 - ☑ User contexts (State machines or multiple threads)

5 Flexibility: Tool Configuration

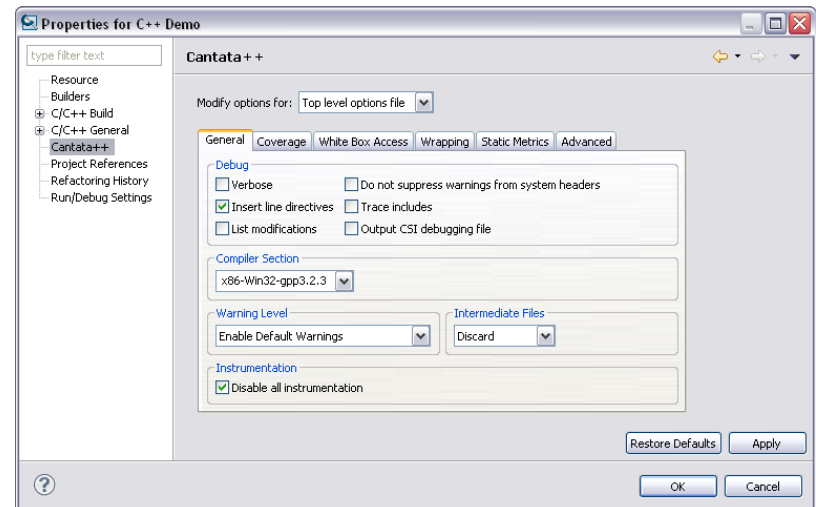
Workspace Tool Preferences

- Stored configurations
- Appearance of views / reports
- Installation details
- Java Coverage
- Test Script generation



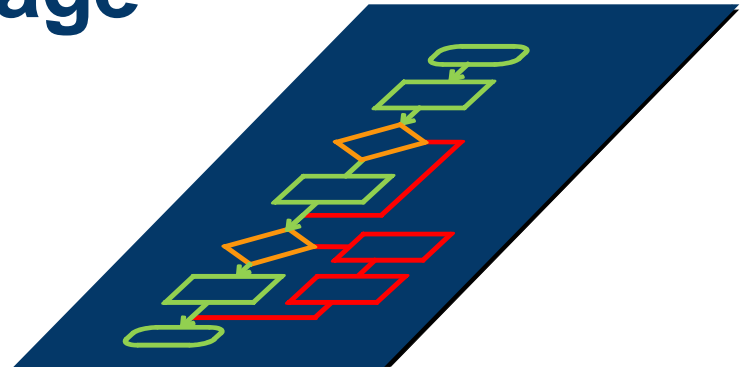
Project Specific Properties

- General & specific tool options
- Advanced options
- Whole project or sub-folders



6 Integrated Code Coverage

- Set coverage target in tests
- Diagnostics over stages / test runs
- Sensible coverage metrics
- Coverage redundancy optimization



➔ **Cantata++** advanced capabilities offer...

- Coverage Rule Sets
- Powerful drill-down views, filters and reports
- Coverage Metrics
 - ☒ Entry- Point Coverage
 - ☒ Statement Coverage
 - ☒ Decision Coverage
 - ☒ Call-return Coverage
 - ☒ Condition Coverage (including MC/DC)
- Automatic test case coverage optimization

7 Large Legacy Code Base

- Automatic generation of unit tests
- Reducing reliance on system regression tests
- Identifying dynamic testability issues



➔ ***Cantata++* Baseline Testing solves...**

- Automatic generation of passing C unit test baseline
- Automatic regression suite of makefiles
- Knowing testability limitations in code
 - ☒ Dynamically unreachable code
 - ☒ Crash scenarios
 - ☒ Data uninitialised or function static
 - ☒ Implicit function declarations
 - ☒ Compiler type truncation

Benefits of Intelligent Testing



● The 7 key Tool Automation Challenges

☞ Solved by intelligent solutions



● Eases tool adoption

☞ Eclipse IDE & target integration

☞ Tests in C/C++

● Increases efficiency

☞ Automation

☞ Flexible tool-kit

☞ Powerful diagnostics

● Reduces cost of compliance

☞ Standard compliant evidence

☞ Tool qualification / certification

Thank You



Any questions?



Demo: C C++ Baseline