

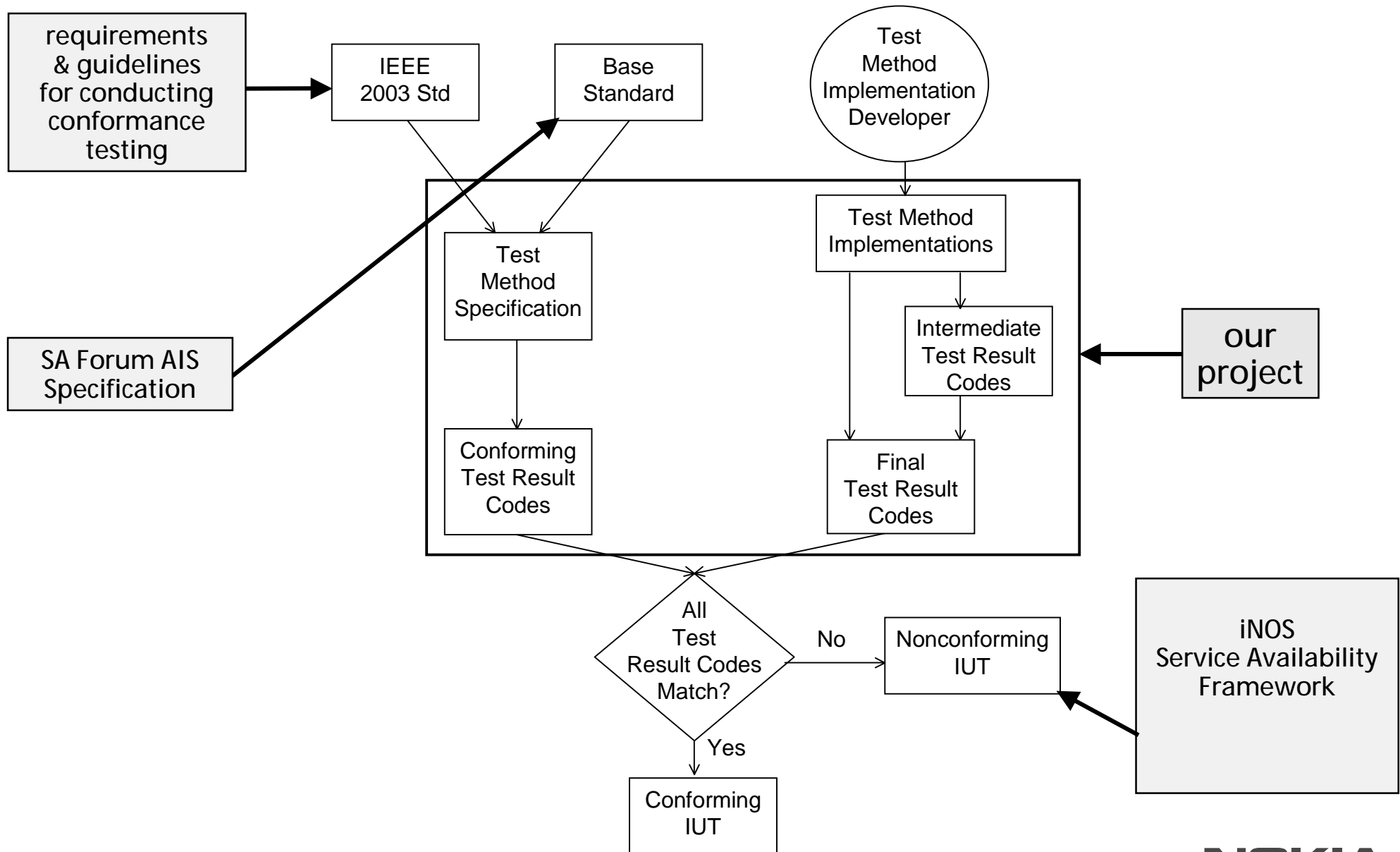
First Experience of Conformance Testing an Application Interface Specification Implementation

Francis Tam and Kari Ahvanainen
Nokia Research Center
Finland

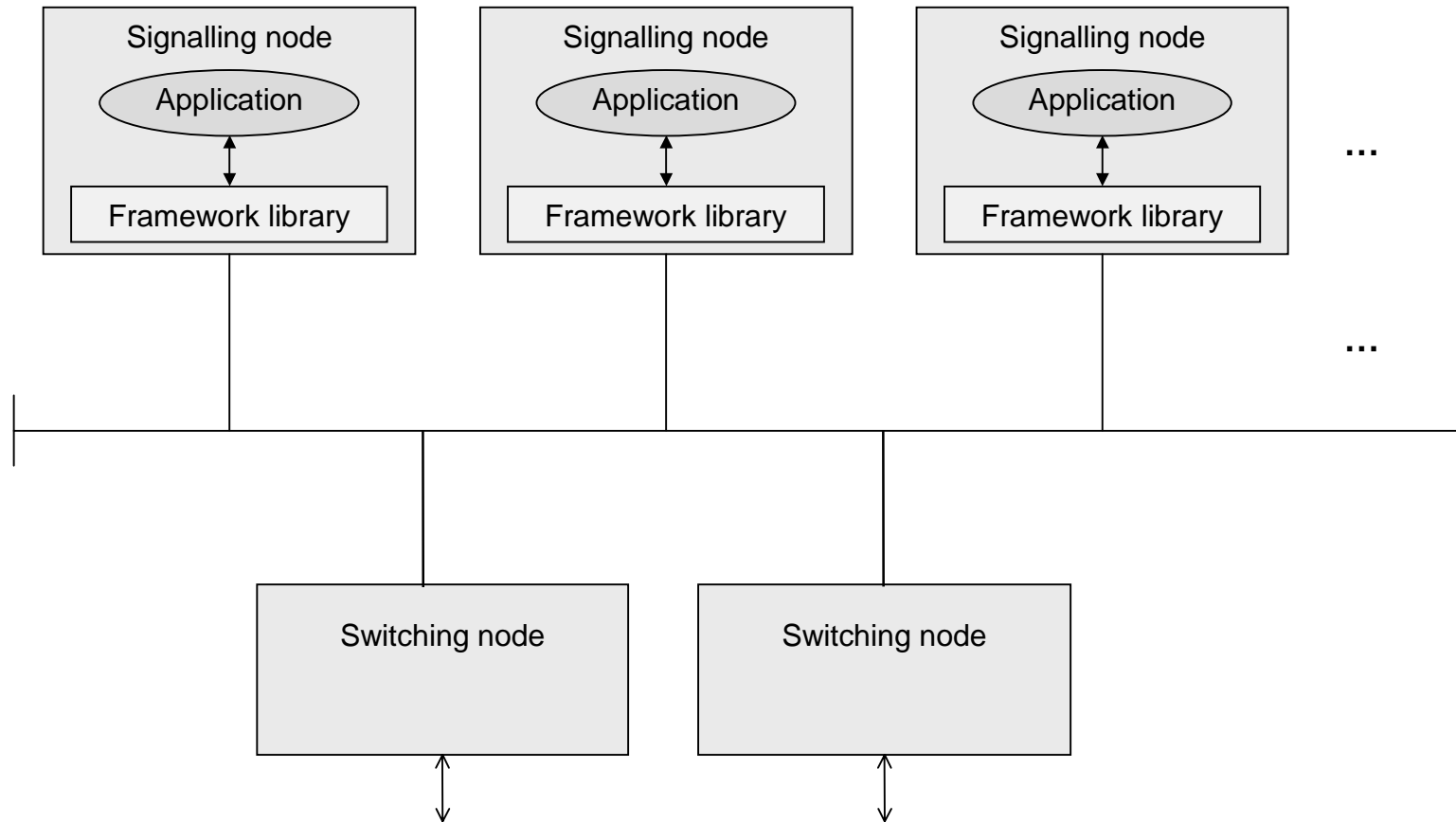
The Starting Point

- The objective was to investigate how we can perform conformance testing for the AIS specification
- IEEE Std 2003–1997
 - IEEE Standard for Information Technology – Requirements and Guidelines for Test Methods Specifications and Test Method Implementations for Measuring Conformance to POSIX[®] Standards
 - Rationale
 - Observation that the AIS specification is similar to a subset of POSIX – function call interfaces in C
 - Principle of following a standard if it is applicable
- iNOS Service Availability Framework
 - An implementation of a carrier-grade service platform based on a pre-release of the AIS specification (draft version 0.8)
 - A subset of the AIS Availability Management Framework (AMF) – 12 APIs implemented out of the 17 defined APIs (callbacks not counted)

Basic Model for Conformance Assessment



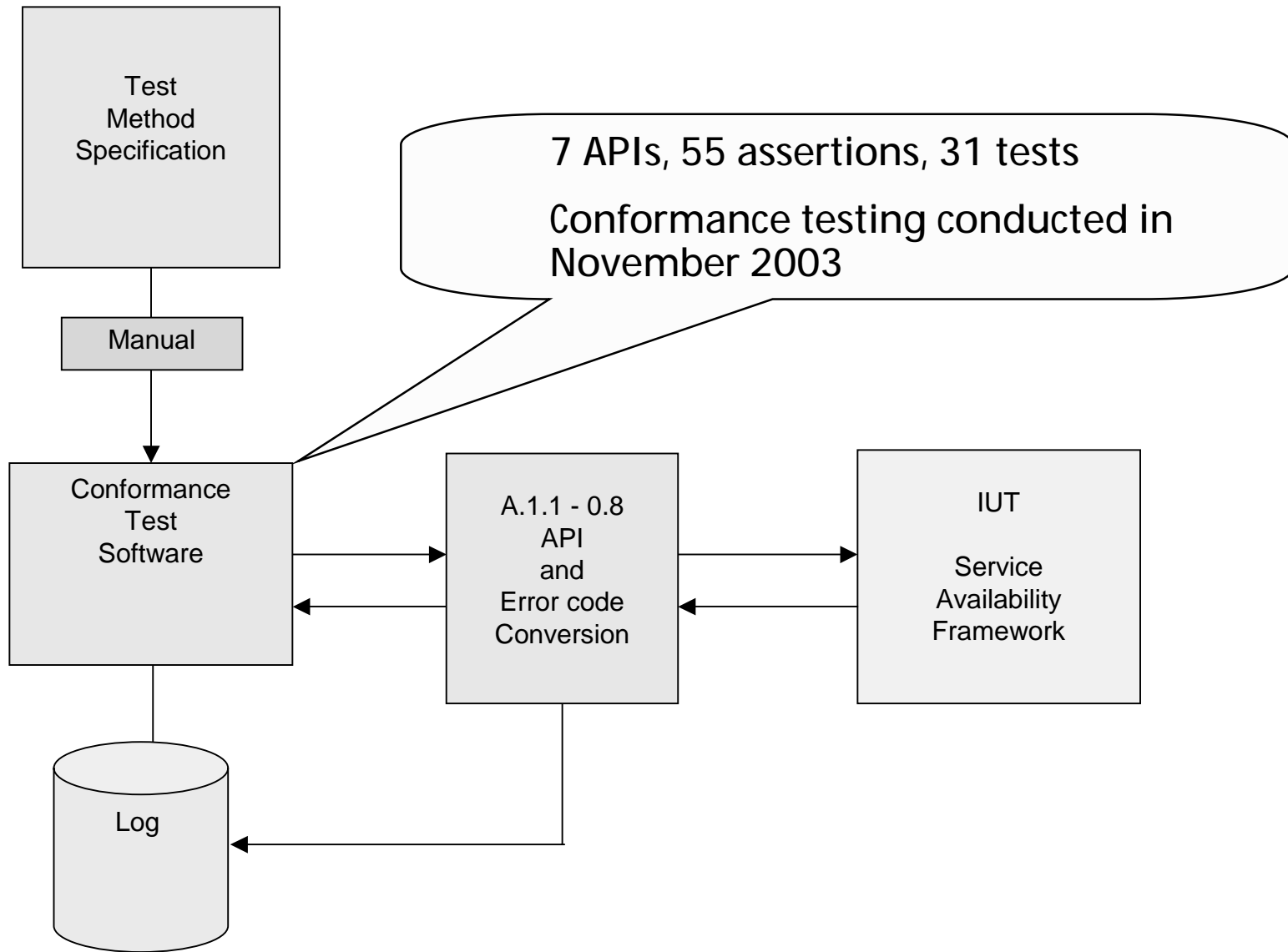
Carrier-Grade Service Platform



What the Book Says – In Plain English

- Test Method Specification
 - Expresses the required functionality and behaviour of the base standard as *assertions* and provide a complete set of test result codes
 - A combination of the defined structure, syntax and natural language is used in writing a number of assertions for each API (*element*)
- Test Method Implementation
 - For each API, each *assertion* is translated into software as test case(s) (*assertion tests*) and gets executed
 - Documentation containing:
 - information to install, configure, and execute the conformance test software; hardware and software requirements; etc.
 - instructions for performing the conformance test procedure
 - description to gather and interpret test results
- Guidelines for Testing Levels
 - Thorough testing
 - Exhaustive testing is infeasible
 - Test with different combination of input parameters (*option*) – permutation of parameters is not required
 - Return codes (*error conditions*) frame the testing method for each API

Reality on the Ground



Subset of the Generic Assertion Structure

<Assertion_identifier>

If <Option> **then**

If <Test_Support> **then**

(**Setup:** <Setup_Requirements>)*

Test: <Test_Text>

(**TR:** <Testing_Requirements>)*

(**Note:** <Notes>)*

Else <No_Test_Support>

Else <No_Option>

* — optional

Assertion Test – saAmfComponentRegister

Tests	amfHandle	compName	proxyCompName	Expected Return Code
OK-1	valid	valid	not used	SA_OK
E01-1	valid	NULL	not used	SA_ERR_INVALID_PARAM
E03-1	uninitialised handle	valid	not used	SA_ERR_INIT
E03-2	finalised handle	valid	not used	SA_ERR_INIT
E09-1	NULL	valid	not used	SA_ERR_BAD_HANDLE
E11-1	valid	previously registered	not used	SA_ERR_EXIST

Conclusions

- Requirements and guidelines of the IEEE std 2003–1997 seemed feasible
- Able to repeat all the tests and obtain consistent results
 - iNOS Service Availability Framework 1.0 in February 2004
- Internal reviews suggested that thorough level of testing has the right balance of confidence and manageability of tests
 - No falsely passed tests were identified
 - Provided useful insights into future implementations

- Prototyping the testing of callback functions is underway
- Exploit the notions of prologue and epilogue in high level scenarios involving multiple components and APIs

- Future work:
 - Implementation dependent error conditions such as `SA_ERR_LIBRARY` need to be considered in the context of the SA Forum Application Interface Specification compliance criteria
 - TTCN-3 instead of the informal, natural language to capture assertions in the Test Method Specification
 - potential of bringing in the whole tool chain for the testing environment
 - automatically generating test cases