



Welcome to the World of Standards



TTCN-3 Tutorial

ETSI Centre for Testing and Interoperability


- **Testing and Test Control Notation Version 3**
- **Internationally standardized testing language**
 - Product of the ETSI Technical Committee MTS (Methods for Testing and Specification)
- **A programming language that has been used for more than 15 years in standardization as well as industry**
 - Specifically designed for black box testing and certification
 - Constantly developed and maintained at ETSI by a team of leading testing experts from industry, institutes, and academia
- **A testing technology that applies to a variety of application domains and types of testing**
 - Knowledge of TTCN-3 is valuable both for employees as well as employers due to its wide applicability
 - Offers potential for reducing training and test maintenance costs significantly
 - Proven to work in very large and complex industrial tests, e.g., 3G network elements

The TTCN-3 Standards (available at <http://www.ttcn-3.org>)






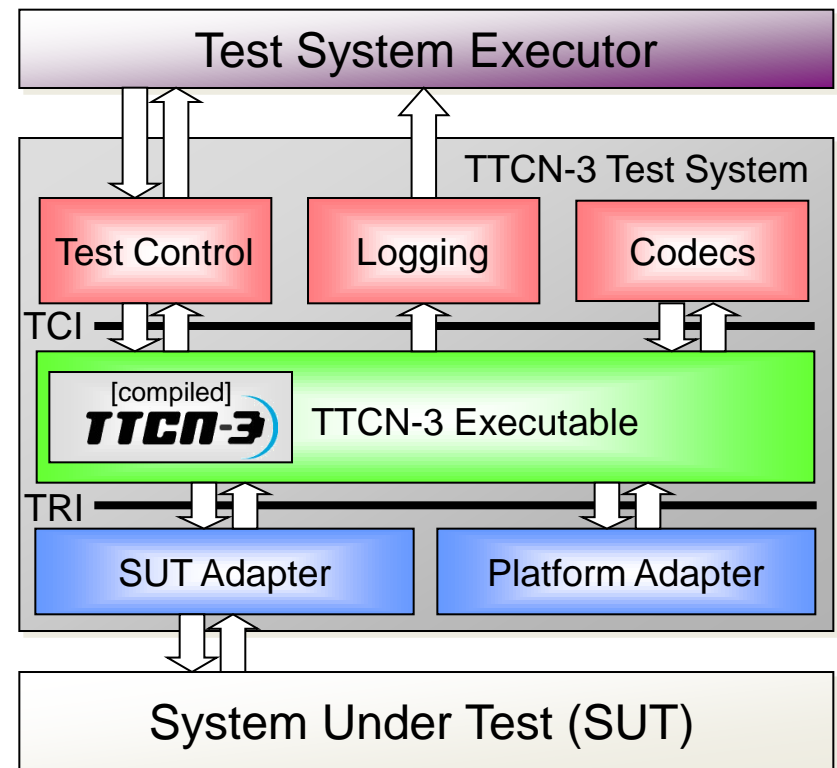
- ES 201 873-1 (Z.140)
 - TTCN-3 Core Language
- ES 201 873-2 (Z.141)
 - TTCN-3 Tabular Presentation Format (TFT)
- ES 201 873-3 (Z.142)
 - TTCN-3 Graphical Presentation Format (GFT)
- ES 201 873-4 (Z.143)
 - TTCN-3 Operational Semantics
- ES 201 873-5
 - TTCN-3 Runtime Interface (TRI)
- ES 201 873-6
 - TTCN-3 Control Interfaces (TCI)
- ES 201 873-7 and onwards (under development)
 - Using ASN.1, XML, IDL, C/C++ with TTCN-3

- From conventional programming or scripting languages?
 - Rich type system including native list types and support for subtyping
 - Embodies powerful build-in matching mechanism
 - Snapshot semantics, i.e., well defined handling of port and timeout queues during their access
 - Concept of verdicts and a verdict resolution mechanism
 - Support for specification of concurrent test behaviour
 - Support for timers
 - Allows test configuration at run-time
 - Tests focus only on implementation to be tested
- From a test tool or vendor proprietary testing language?
 - Not tied to a particular application or its interface(s)
 - Not tied to any specific test execution environment, compiler or operation system
 - TTCN-3 as such is not executable and requires a compiler/interpreter, adapter as well as codec implementations

-  TTCN-3 builds on top of TTCN-2 but extends it significantly
 - Core language has now look and feel of a regular programming language: much easier to learn
 - No longer uses protocol specific terminology like PCO, ASP, PDU, etc
 - Different presentation formats: tabular, graphical, ...
 - Completely dynamic test configurations
 - Support for synchronous communication
 - Support for testing distributed systems
 - Standardized test system interfaces (TRI & TCI)
 - Improved text string matching: regular expressions
 - Better harmonisation with ASN.1
 - Extension mechanism to integrate other type systems, e.g., XML, ASN.1, C, ...

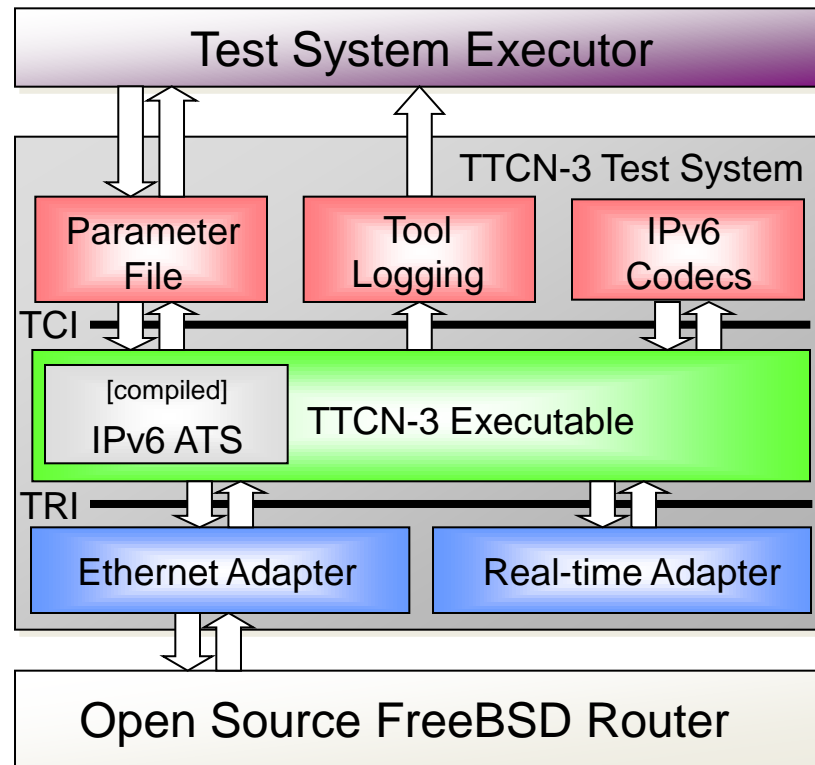
TTCN-3 test systems in a nutshell

- TTCN-3 specifies a test but a test system is needed for test execution
 - TRI and TCI standards define test system architecture
 - TTCN-3 tools are *required* to support internal interfaces
 - Allows reuse of test platforms with different tools but also for different SUTs
 - A test system requires
 - A TTCN-3 tool = TTCN-3 compiler and execution environment
()
 - A test platform for a specific device under test
( s +  s)
- Note: Tools come with default Test Control & Logging



TCI = TTCN-3 Control Interface
TRI = TTCN-3 Runtime Interface

An example adaptation: A IPv6 test system



- TTCN-3 is easy to learn
 - Look and feel of a regular programming language
- Unambiguous specification and execution of tests
 - Well defined syntax, static - and operational semantics
 - Enables completely automated test execution
- Off-the-shelf tools and test systems are readily available
 - Five different commercial TTCN-3 tools on the market
- Open source community now taking shape
 - Tools as well as test suites and useful modules
- Can be used to specify tests for standardization as well as proprietary product features
- Flexible testing technology
 - Virtually no limits to adapt a test system to your needs
 - Scalable – allows test systems to grow over time

At ETSI

- Used for development of any new conformance test suite, e.g., SIP (VoIP), IPv6 (Core, Mobility, Security), HiperMAN / WiMax, 3GPP IP Multimedia Subsystem, ...

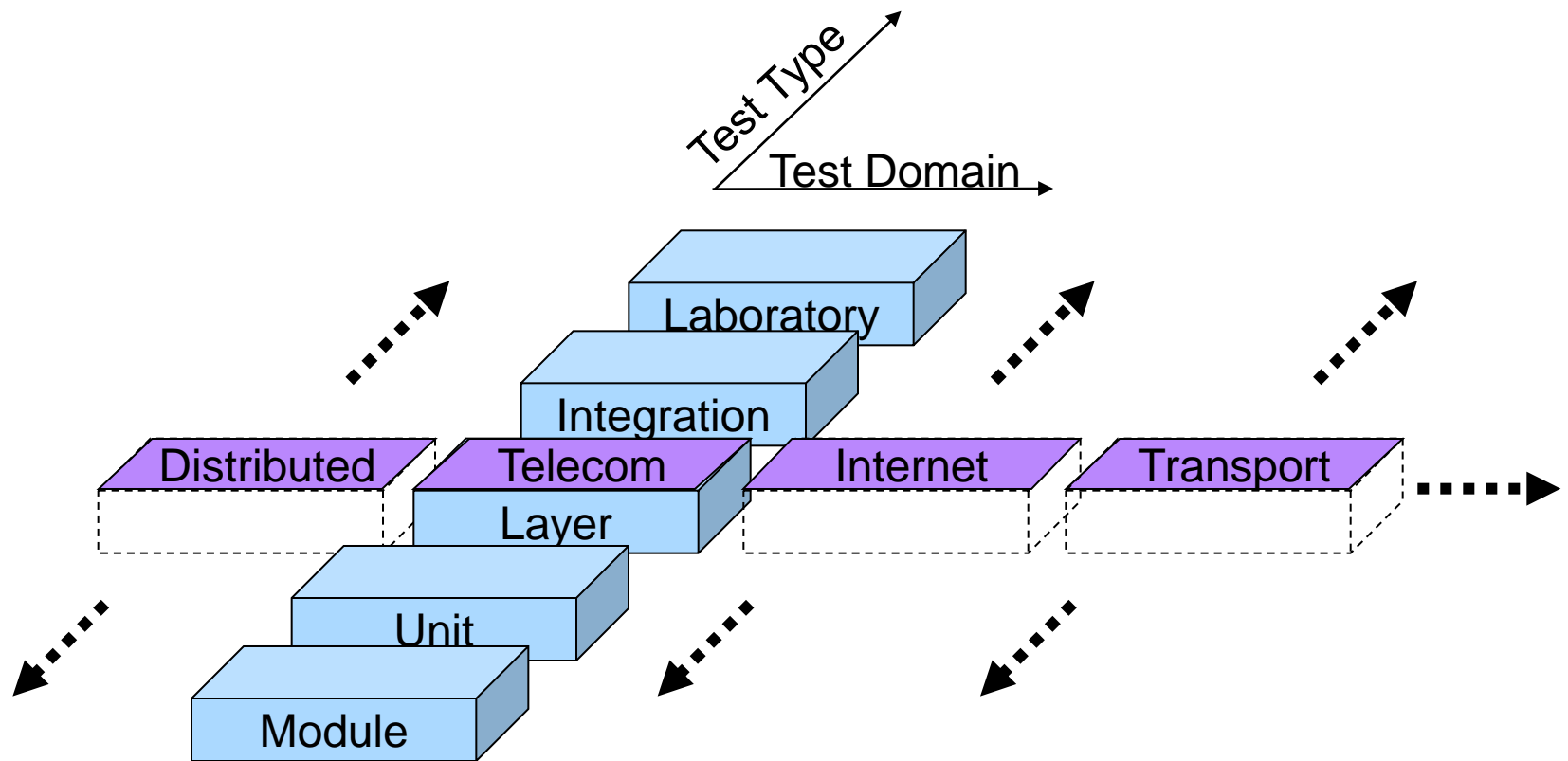
In industry

- Applied in a variety of application domains, e.g., telecom, automotive, financial, ... (see www.tt-medal.org)
- Ericsson reported 1,000 active licenses at TTCN-3 User Conference 2006
- Nokia experiences captured in IEEE Software 23(4) 2006
- Motorola reports doubling of testing productivity

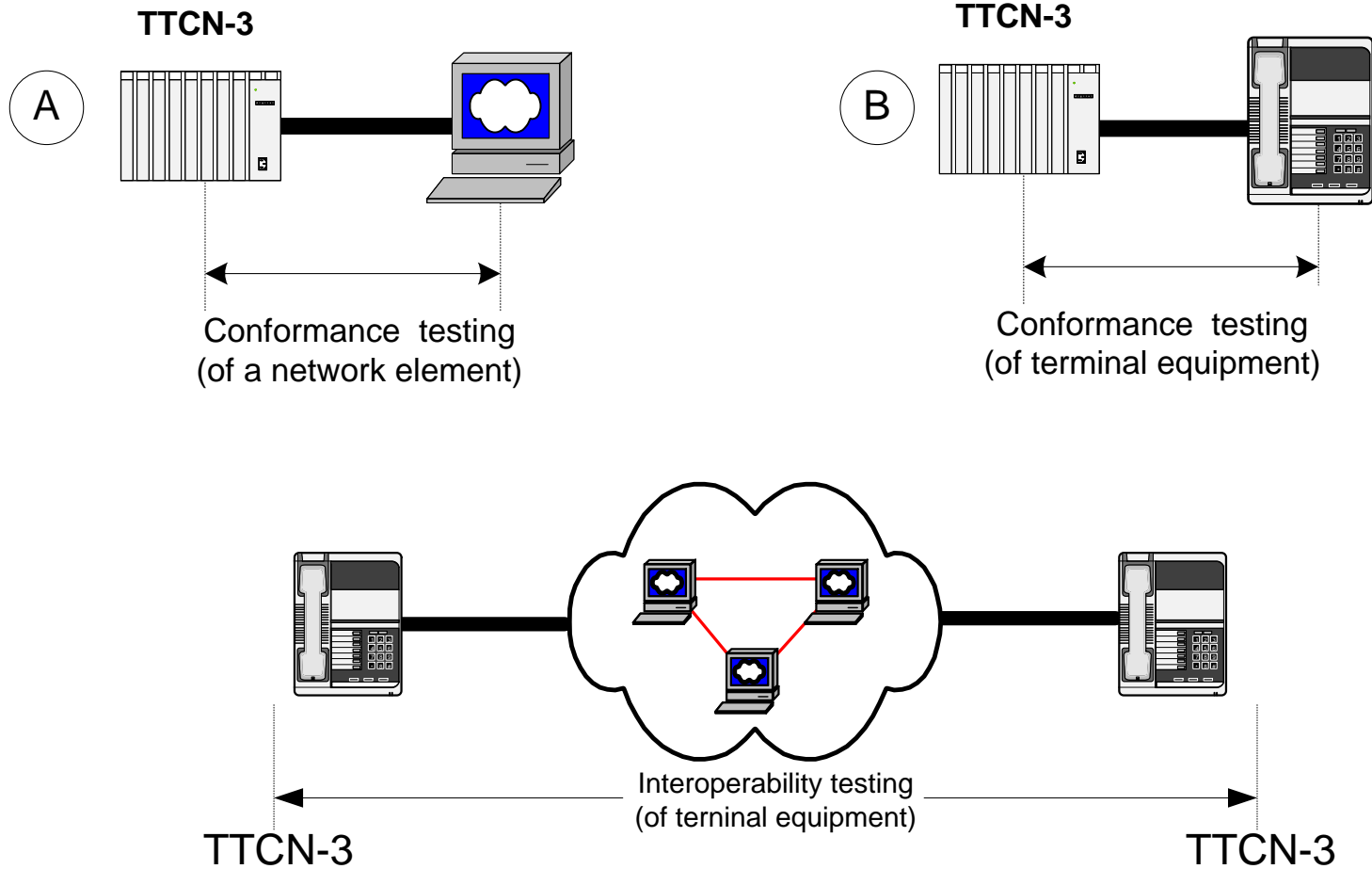
Also used beyond Europe

- Strong community in China

Expansion of TTCN-3 Use



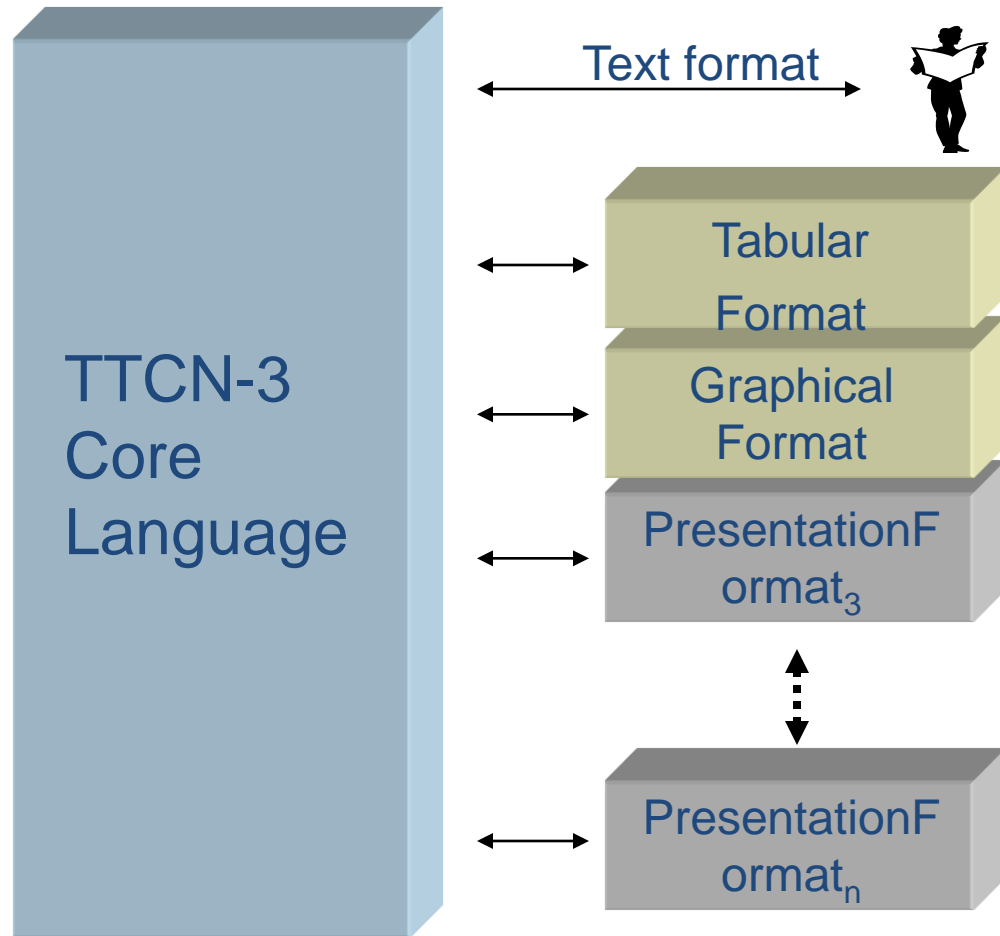
TTCN-3 can automate Conformance and Interoperability Testing



Main Capabilities of TTCN-3

- Dynamic concurrent testing configurations
- Various communication mechanisms (synch and asynch)
- Data and signature templates with powerful matching mechanisms (including regular expressions)
- Attributes for encoding, display or user-defined information
- Test suite parameterization
- Control of Test Case execution and selection mechanisms
- Control of complex test configurations
- Assignment and handling of test verdicts
- Harmonized with ASN.1 (XML and IDL coming)
- Different presentation formats
- Well-defined syntax, static - and operational semantics

The Core Language and Other Presentation Formats



● Core format is text based (most popular)

● TTCN-3 can be edited or viewed in other formats

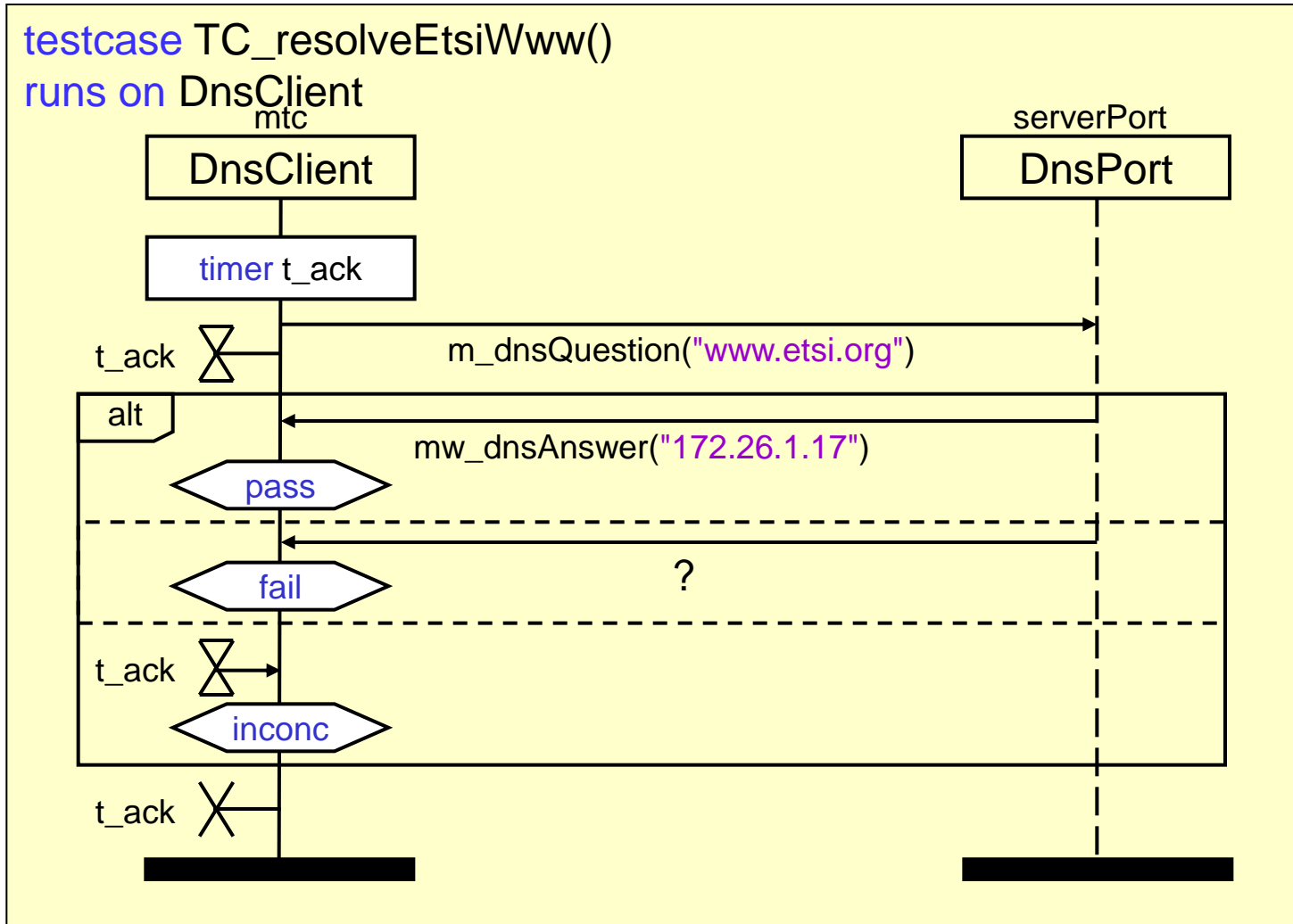
- Tabular format (for TTCN-2 people)
- Graphical format (good for visual overview)
- Other standardized formats in the future?
- Proprietary formats possible

Example Core (Text) Format



```
testcase TC_resolveEtsiWww() runs on DnsClient
{
    timer t_ack;
    serverPort.send(m_dnsQuestion("www.etsi.org"));
    t_ack.start(1.0);
    alt {
        [] serverPort.receive(mw_dnsAnswer("172.26.1.17")) {
            setverdict (pass);
        }
        [] serverPort.receive { // any other message
            setverdict(fail);
        }
        [] t_ack.timeout {
            setverdict(inconc);
        }
    }
    t_ack.stop;
}
```

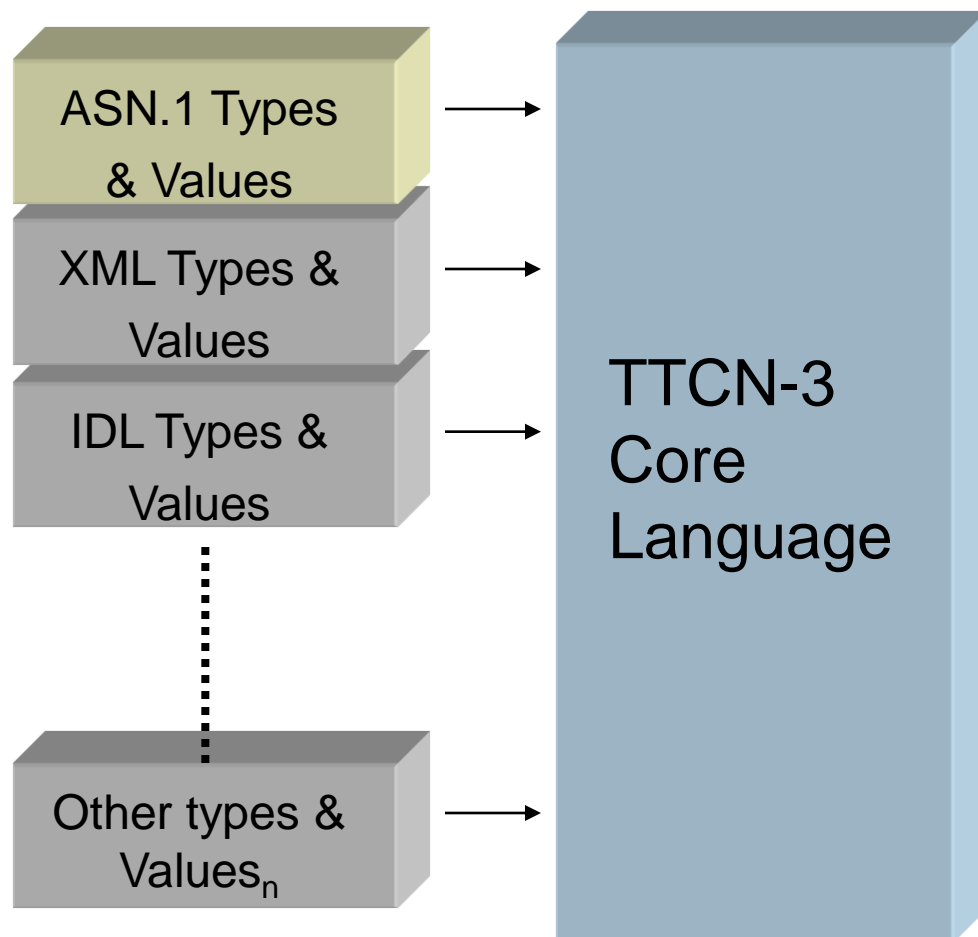
Example Graphical Format



Example Tabular Format

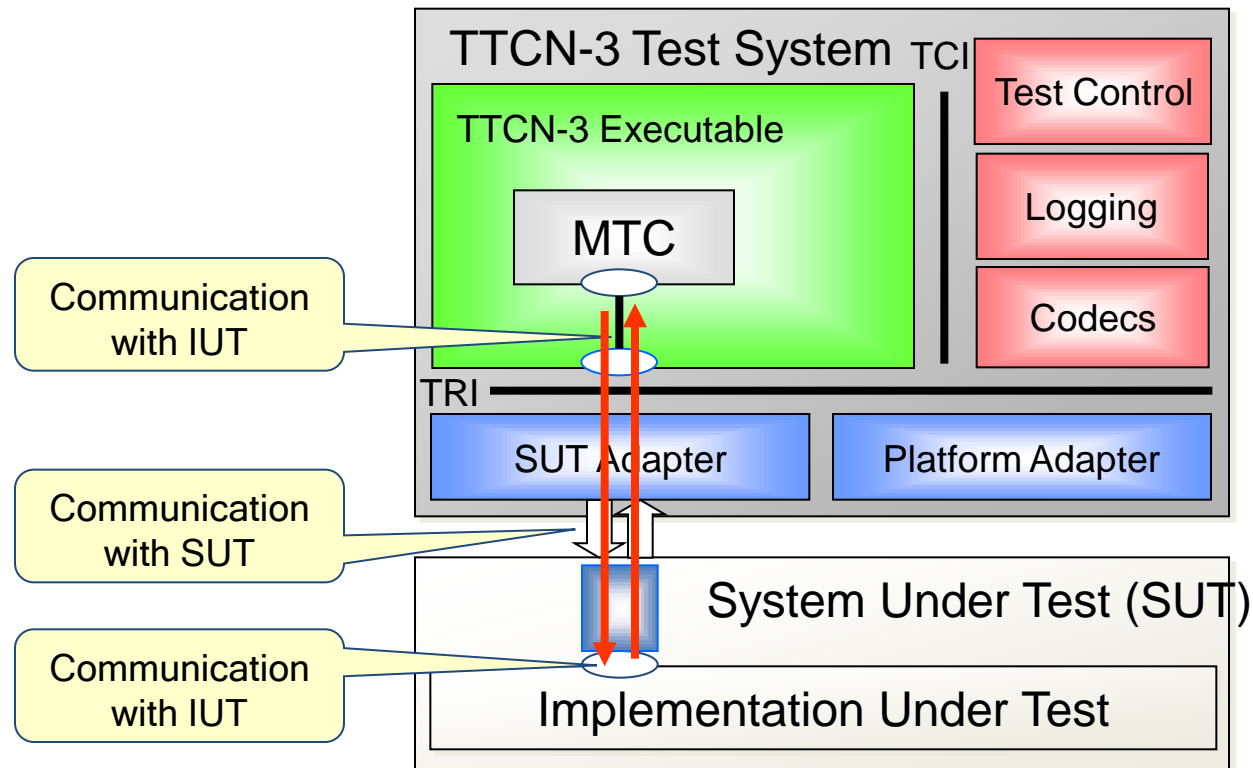


Testcase			
Name	TC_resolveEtsiWww()		
Group			
Purpose			
System Interface			
MTC Type	DnsClient		
Comments			
Local Def Name	Type	Initial value	Comments
t_ack	timer		
Behavior			
<pre> serverPort.send(m_dnsQuestion("www.etsi.org")); t_ack.start(1.0); alt { [] serverPort.receive(mw_dnsAnswer("172.26.1.17")) { setverdict (pass); } [] serverPort.receive // any other message { setverdict(fail); } [] t_ack.timeout { setverdict(inconc); } } t_ack.stop; </pre>			
Detailed Comments:			



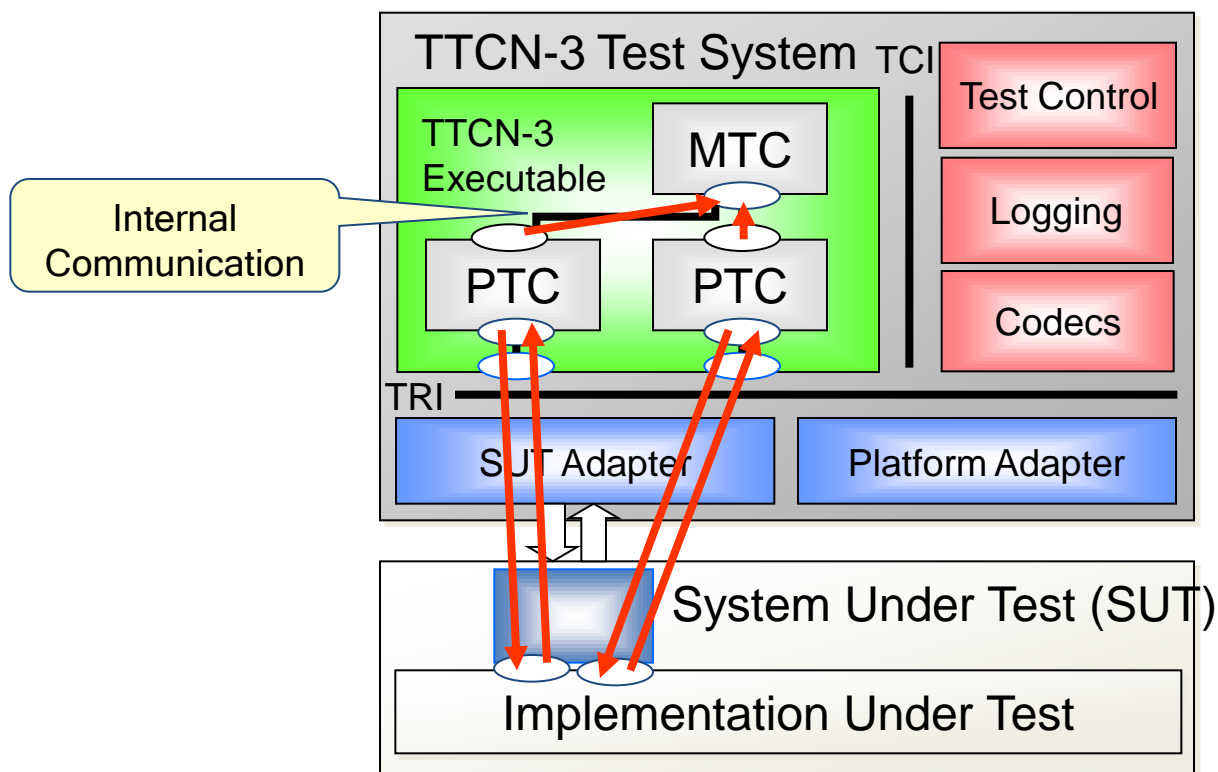
- TTCN can be integrated with types systems of other languages
- Fully harmonized with ASN.1 (1997)
- Harmonized with other languages
 - IDL, XML, C/C++

Minimal Test Configuration

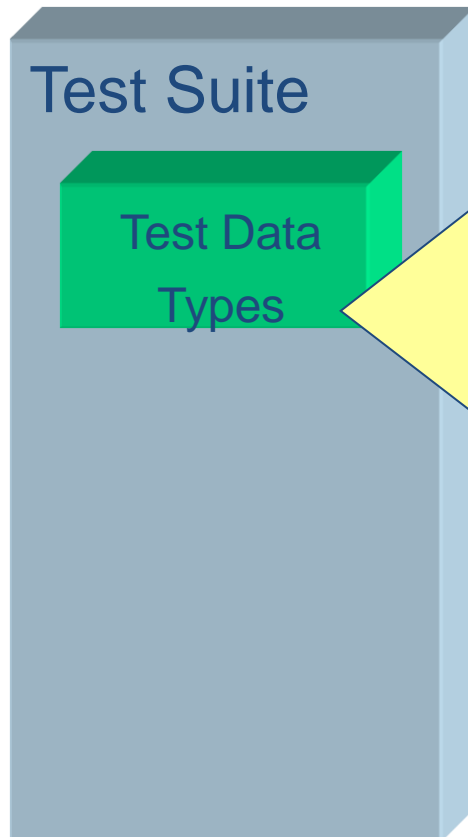


- All test behavior is executed on one (main) test component

Example Concurrent Test Configuration



- A test involves execution of many parallel test components
- Dynamic instantiation of components and communication links



Data types which specify

- Structure of messages or calls and their information elements (fields, parameters)
- Internal data structures (e.g., for computation)
- Possibly encoding or display information

Built-in basic types

`integer`, `boolean`, `float`,
`bitstring`, `hexstring`, `octetstring`,
`charstring`, `universal charstring`

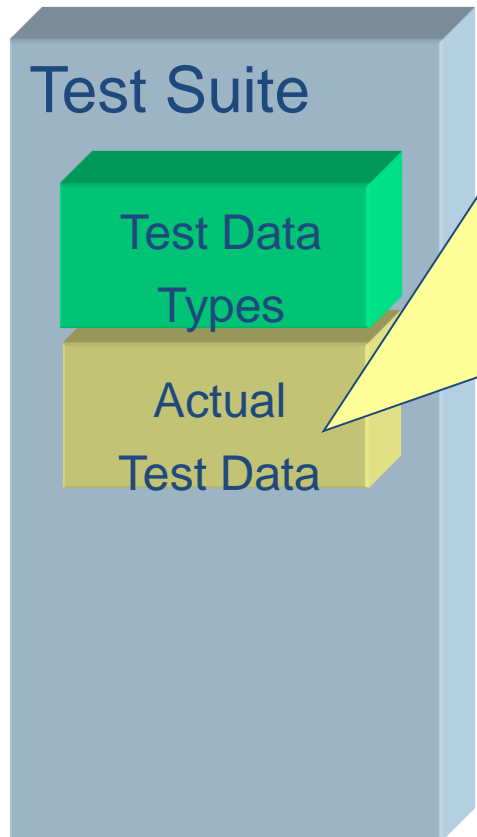
... and structured types

`record`, `record of`, `set`, `set of`
`union`, `enumerated`

... and special types such as

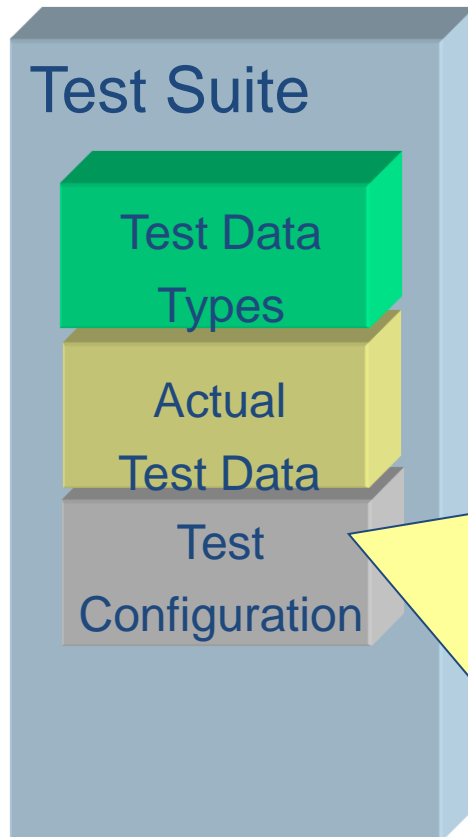
`component`, `port`, `verdicttype`, `default`, etc

Building blocks of a TTCN-3 Test Suite



Actual test data (values) used during testing

- Constants or Templates for specific message or call parameter values
- Matching expressions for allowing multiple message or call parameter values
 - value range, value list, wildcards, presence, length, size, permutation
 - regular expressions
- Using also template decomposition, parameterization and modification

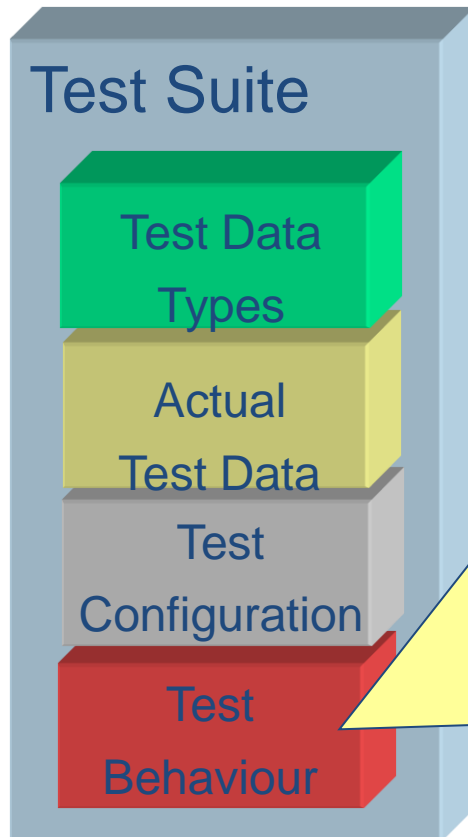


Static aspects

- Test component and port types

Dynamic aspects

- Dynamic instantiation and management of test components
- Mappings of test components to abstract test system interfaces
- Connections between test component interfaces
- Management of test components

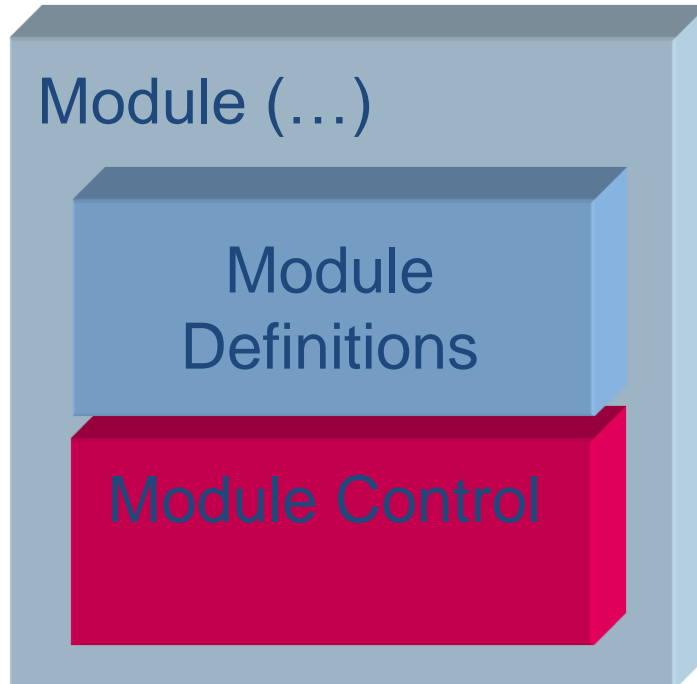


test cases

- specify sending/receiving messages, computation (e.g., checksums), and verdict assignment
- can be decomposed with functions and altsteps
- can (re)use default behaviour
- can use timers and timeouts

test execution control (optional)

- order, repetition, conditions, etc



```
module EtsiDnsTests
{
    // Test definition part

    control
    {
        // Test execution part
        // (optional)
    }
}
```

```
module EtsiDnsTests
{
    // Message structure
    // Actual test data
    // Test configuration
    // Test Case definitions
}
```

```
module EtsiDnsTests
{
  group MessageStructure
  {
    // Defintions of message types
  }
  group TestData
  {
    // Templates for messages instances
  }
  group TestSystemConfiguration
  {
    // Port and component types and mappings
  }
  group TestCases
  {
    // Test case definitions
  }
}
```

```
type record DnsMsg // simplified message structure!  
    {  
        DnsMsgKind kind,  
        charstring question,  
        charstring answer optional  
    }  
type enumerated DnsMsgKind {e_query, e_response}  
template DnsMsg m_dnsQuestion( charstring p_question )  
{  
    kind := e_query,  
    question := p_question,  
    answer := omit // no answer  
}  
template DnsMsg mw_dnsAnswer( charstring p_answer )  
{  
    kind := e_answer,  
    question := ?, // any question ok  
    answer := p_answer  
}
```

```
type port DnsPort message
{
  inout DnsMsg
}
```

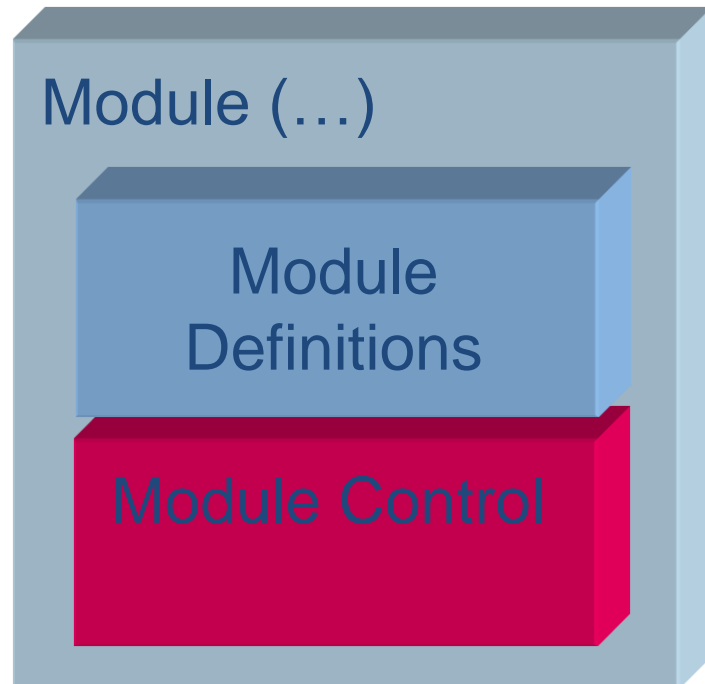
```
// Note: port types may also allow multiple different
//       message types or restrict the direction
```

```
type component DnsClient
{
  port DnsPort serverPort
}
```

```
// Note: component types can also define multiple port
//       instances of the same or different port type and
//       declare timers, constants or variables
```



```
testcase TC_resolveEtsiWww() runs on DnsClient
{
    timer t_ack;
    serverPort.send(m_dnsQuestion("www.etsi.org"));
    t_ack.start(1.0);
    alt {
        [] serverPort.receive(mw_dnsAnswer("172.26.1.17")) {
            setverdict(pass);
        }
        [] serverPort.receive { // any other message
            setverdict(fail);
        }
        [] t_ack.timeout {
            setverdict(inconc);
        }
    }
    t_ack.stop;
}
```

```
module EtsiDnsTests
{
    // Test definition part
    modulepar boolean mp_example;

    testcase TC_resolveEtsiWww()
    runs on DnsClient
    {
        // .. as in previous slide
    }

    // Test execution part
    control {
        if (mp_example) {
            execute(TC_resolveEtsiWww());
        }
    }
}
```

Where can I learn more?



- Visit ETSI's official TTCN-3 web site (www.ttcn-3.org)
 - Public TTCN-3 test suites, useful TTCN-3 modules
 - Links to commercial as well as open source tools
- Read well written TTCN-3 standard suites
- Join the ETSI mailing list (list.etsi.org/TTCN3.html)
- Take a course
- Read publications
 - Proceedings of Conference for Testing of Communicating Systems (TESTCOM)
 - Presentations of yearly TTCN-3 User Conferences in Europe or Asia (see www.ttcn-3.org)
 - Get a text book
<http://www.wiley.com/legacy/wileychi/ttcn-3/>
- Register for the next TTCN-3 user conference!



Contact Details:



www.ttcn-3.org

ttcn-3@etsi.org

Thank you!