TCI Meeting January 23-24

Participants:

Maximillian Frey, O2
Claude Desroche, STS
Johan Nordin, Telelogic
Thomas Deiss, Nokia
Jacob Wieland, TUB
Theofanis Vassiliou-Gioles, Testing Technologies
Stephan Pietsch, Testing Technologies
Ina Schieferdecker, FOKUS
George Din, FOKUS
Dimitrios Apostolidis, FOKUS

Agenda

Status

- All parts of the document are ready
- Document structure
  - Introduction
  - General structure
  - Abstract Types + InterfacesUse scenarios
  - Java and C Mapping
- TCI {set of abstract types and interfaces}
  - Abstract class **Type**
  - **Value** class
  - Integer, Float etc => inherit the Value
  - Interfaces divided: **required** and **provided**
    - Feature which comes from the component technology of components from OMG
    - Required = GREEN
    - Provided = RED
  - Defined from perspective of TCI Adaptor

Scenarios

- **10.3) For TCI-TM => to show the interaction between TCI and TM and the role of "required" and "provided".**
  - The user is in the front of the provided interface
  - TCI is defined such not only the control part is executed but also only one test case. The user can decide to start a TestCase.
  - The identification of TestCases is based on **knowledge** about the module.
  - Q: what if we load 20 modules? How do we identify one TestCase?
    - TCI designed that we load only one top-level module!!!
  - Context for modules => hence we can execute any test case which belong to a context
  - Q: TestSuite = 10 test cases distributed over 3 modules. One module does not contain TestCases but imports them. Which "control module" do we set ?
    - One control module at time
  - Q: setModule() !? What does it?
    - Initializes the TCI with top module
    - Useful to set the context. To execute a module we need to load the module code.
  - But if we have a fully qualified reference to the TestCase!? What need to have setModule()
    - We agreed that we have a top-level module. This is given to the TCI in order to find the parameters, test cases etc.
  - Can we not take the information about the module from the startTestCase() since the information about the module is contained within the TestCaseId?
  - Q. Is it possible to set the module A and to start a TestCase from module B?
    - YES. If the module B is visible from the module A.
  - Why do we need setTestModule() when we can take all the information we need from setTestCase() ? Both solutions are possible. One can either take all the information from setTestCase() (also the module which can be contained with the TestCaseId) or separate by using setTest Module() to set the module and setTestCase() to set the testcase. This is an implementation decision.

    - To decide
        - tciSetModule
        - tciInitModule
        - tciInitControl
        - tciSetControl**tciRootModule (was decided)**tciControlModules
    - Aggred!!! We have tciRootModule before tcistartTestCase
- **10.4) TM sends a Log message**
    - We add TestComponentID to tciLog()
- **10.5) How to get a module parameter**
    - Correct: getTypeName(ParameterTypeName)
    - Why not have tciGetModulePar(paramterId, parameterTypeName) ?
    - Why to ask separately for the Type?
        - To decode the parameter (ex. for cast)
    - To provide possibility to take the defaultValue ?
    - Back to the original version: one can ask often about the value but only once for the type. **TODO**: Modify the description in the text for the scenario**10.6) Stop a test case**
    - How can one control the execution? How can one stop a test case at a certain point?
- **10.7) Stop controll**
    - If we call tciStopControl() during a test case should we wait for the termination of the TestCase?
    - Termination of the control part results also in the termination of any TestCase.
    - Error outside the TestCase
- **10.8) Errorneous termination of Control**
    - Remove tciSetModule()
    - After receiving an error one should explicitly stop the control
- **10.9) Erroneous termination of a TestCase within the Control**
- **10.10) Erroneous termination of a TestCase**
    - If one runs a TestCase from the control part and the TestCase gets an error, should we stop the control as well?
    - NO The user decide what to stop: the TestCase or the control
- **10.11) Component Handling for Direct Test Case Execution within the Control**
    - User and three nodes
    - Completely agreed.
- **10.12) Component Handling for Direct Test Case Execution**
    - Modified the tciComponentTermincated(ptc, LocalPTCVerdict) -> it goes now to the Node with the description „Another Node with the MTC“
    - Within TCI, there is no need for a system component.
- **10.13) Local Control Component Creation**
    - Modify tciCreateComponentReq(control, null)
- **10.14) Remote Control Component Creation**
    - Modify tciCreateComponentReq(control, null)
- **10.15) Local MTC Creation**
- **10.16) Remote MTC Creation**
    - **CHANGE**: put the TTCN3 fragment after the diagram
- **10.17) Local Intercomponent Communication**
- **10.18) Internode’s Communication between Test Components**
- **10.19) Encoding**
- **10.20) Decoding**
    - TRI-SA and TCI-CD attached to one node.

- **IDL**

- **ANSI C Mapping**
    - Change all functions which are not yet prefixed by tri by putting tci
    - No pointer => but in the implementation can be used
    - TciType => declared as abstract type
    - Objid, TciObjidElem,UniversalCharstring: representations must be defined => mapped as *structs*.
    - UniversalChar mapped as long int[4]. Why not char[4] ?
    - Int defined **abstract** and when mapping one can decide if it is a big or short integer.
    - Proposal: fixed length for int
    - Put one flag for ints (big, small). A smart compiler, which check if it short or long and set the flag.

- void setBig(boolean);
- String: OMG recommendation
- Memory: copy, clone to share memory

- **Java Mapping**
  - Mapping of TciObjId => a collection of elements
  - Mapping of DataTypes => get/set pairs
  - Replace TciSignatureIdType with TriSignatureIdType
  - Remove the TciMessageType.

- **Comments**
  - **NetHawk**
    - TM should be able to set modules, give values for modules parameters
    - Parametrized test cases
    - **Open Issue: Should TM have knowledge about TestCases compiled into it or can it ask TE for that knowledge?**
    - There is no service TCI-TM
    - **> Information <**:
      - Tree of testcases and groups
      - TestCase parameters
      - Modules parameters & Default Values
      - What about proprietary attributes?
    - Decided: TM provides to TE
      - a list of TestCaseIds. (fully qualified).
      - getTestCaseParameters(testcaseid) => returns a list of mode and typeId
      - getModuleParameters() => returns ParamId, TypeId, DefaultValue
      - »(for all known module parameters; of the root module and imported modules)
      - **Open Issue:** how to handle external constants?
        - Outside the scope of TCI
  - **Nokia**
    - Where is the system? Do we have a system component in the TE? Is there one? Who creates it
      - One logical "system" port. Known be TRI.
      - At TCI level, system does not exist as a component – rather, the system ports can be distributed over many nodes.
    - Many to one mappings?
    - A system port can be distributed over many devices.
    - **TODO**: Remove getSystem() and all about system
    - saReset, executeTestCase, where are these executed ?
      - tciReset and tciExecuteTestCase (new function) have to be propagated by TCI
      - All distribution stuff must be managed by CH
    - Unsynchronized **done**
      - It should be synchronized.
      - Stop is atomic operation
      - New operation: tciTestComponentDoneReq, tciTestComponentDone
    - getEncodedVariant => what is enc. Variant ?
      - **Add**: getEncodedVariant, getEncodedValueVariant
    - Missing functions for: componentId, defaults, address ???
      - Do we need such functions for address ?
        - Address is user defined.
        - Decision:: No function will be added.

- Johan
  - TciStartComponent();
  - TriComponentId instead of TciComponentId
  - Definition of tciEnqueueMsgConnected =>missing signature
  - tciDisconnect(); Deconnect
  - Remove all about system from TCI doc
- O_2
  - C and Java mapping should be connected and enable interworking

- **Editing Session**Add scenarios per new function group:
  - tciRootModule, tciGetTestCases, tciGetTestCaseParameters, tciGetModuleParamters
  - tciMapReq, tciMap, tciUnmapReq, tciUnmap
  - tciExecuteTestCaseReq, tciExecuteTestCase
  - tciResetReq, tciReset
  - tciTestComponentDoneReq, tciTestComponentDone
- Add explanation about the system component
- Change the instance order in the use cases to increase readability
- Change red/green colour to increase readability
- Reorder the use cases: from simple to complex
- Check all the parameters in the use cases
- Delete getSystem and all system related things in TCI

- **Action Points**
  - Minutes 31 January
  - Operation Mapping etc. 14 Feb
  - Revised Document: by 21 Feb
  - Review 28 Feb
  - Issued/Upload on: after 1 March?