<testing_tech>

The execution of TTCN-3 tests

The TTCN-3 Runtime and Control Interfaces

Theofanis Vassiliou-Gioles

vassiliou@testingtech.de

phone +49 30 726 19 19 0

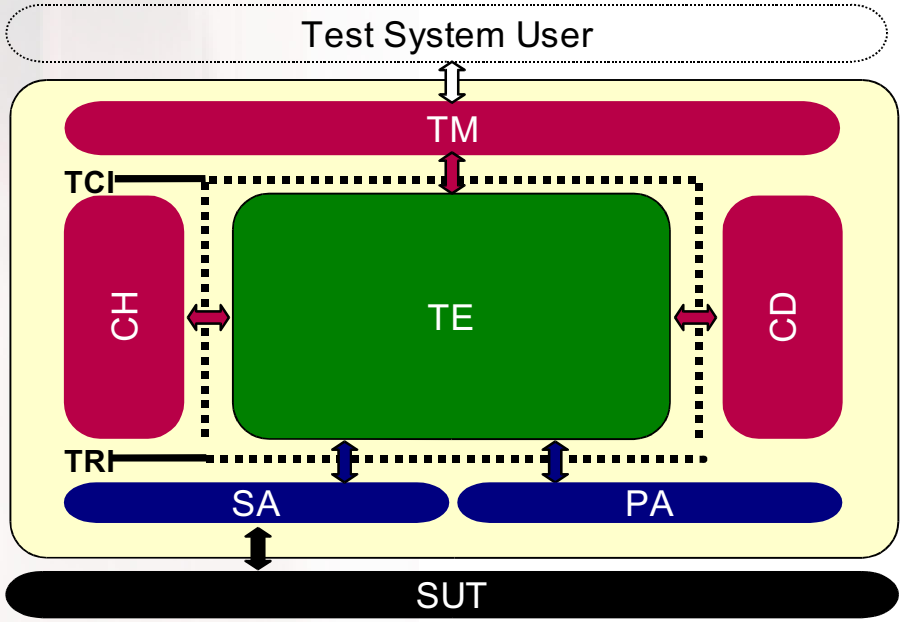
www.testingtech.de

<testing_tech>

Agenda

- The TTCN-3 Reference Architecture
- The TTCN-3 Control Interfaces (TCI)
  - Introduction
  - Coding examples with the Java binding
- The TTCN-3 Runtime Interface (TRI)
  - Introduction
  - Coding examples with the Java binding

A TTCN-3 Test System

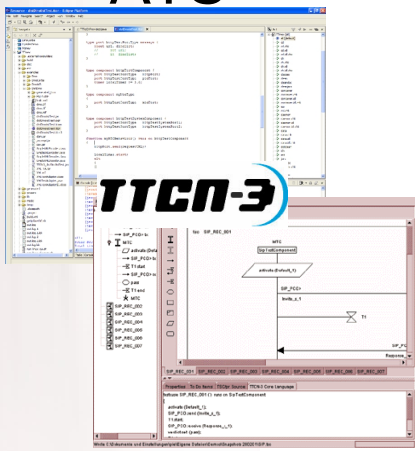


- TE – TTCN-3 Executable
- SA – System Adapter
- PA – Platform Adapter
- CD – Codec
- TM – Test Management
- CH – Component Handling
- SUT – System Under Test

- ETSI ES 201 873-1
  TTCN-3 Core Language (CL)
- ETSI ES 201 873-5
  TTCN-3 TTCN-3 Runtime Interface (TRI)
- ETSI ES 201 873-6
  TTCN-3 TTCN-3 Control Interfaces (TCI)

Implementation

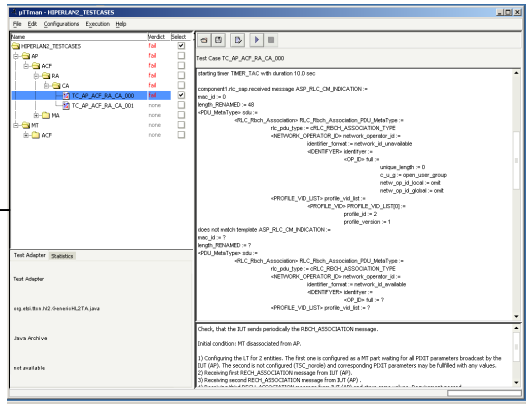
ATS



Test System

TE

+



SUT

communication

Steps To Implement TTCN-3

1) Translate TTCN-3 into executable code
2) Adapt runtime environment to test management and codecs
3) Implement communication aspects

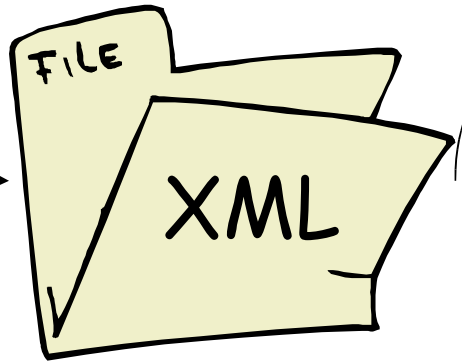
Steps To Implement TTCN-3

1) Translate TTCN-3 into executable code
2) Adapt runtime environment to test management and codecs
3) Implement communication aspects

<testing_tech>

The used example

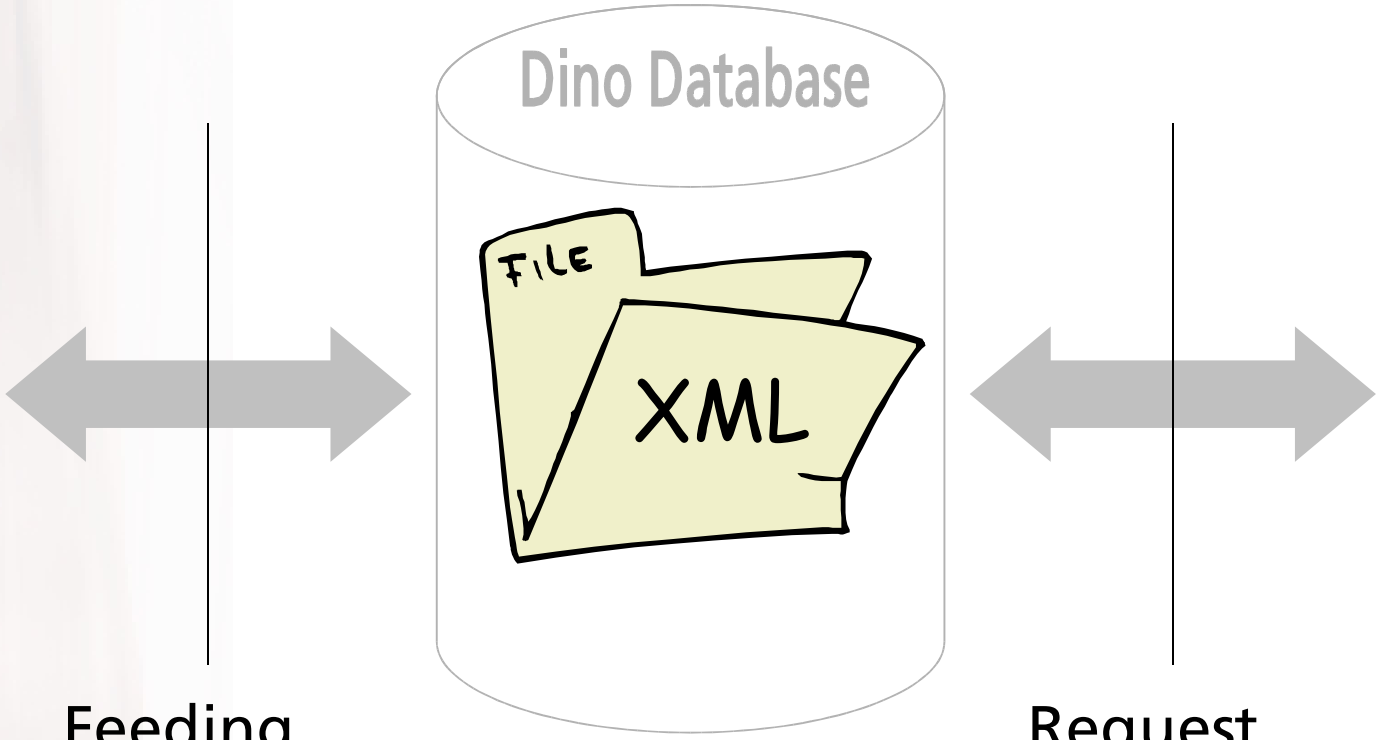
Example – The Dinosaur Database



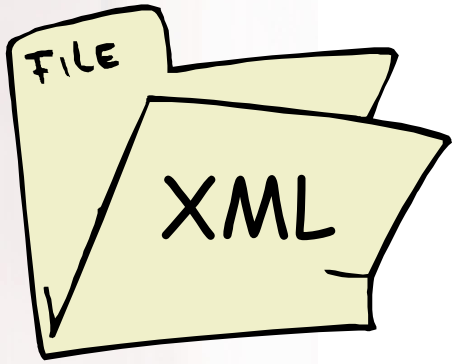
Paleontologist
feeding the database

Student
requesting data

Database Interfaces



The Request Interface



URL

http://www.testingtech.de/TTCN-3_Example/dinolist.xml

XML File

```
<?xml version="1.0" encoding="UTF-8" ?>
+ <!--  -->
<!DOCTYPE dinolist (View Source for full doctype...)>
- <dinolist>
  - <dinosaur>
      <name>Argentinosaurus</name>
      <len>35 m</len>
      <mass>80 tonnes</mass>
      <time>Albian to Cenomanian</time>
      <place>Argentina</place>
    </dinosaur>
  - <dinosaur>
      <name>Brachiosaurus</name>
      <len>22 m</len>
      <mass>30 tonnes</mass>
      <time>Kimmeridgian</time>
```

Module

```
module dinolistTest {
    // type declarations
    // configuration declarations
    // module parameter and
        // external function declarations
    // template declarations
    // function declarations
    // test case declarations
    // control part
}
```

Module

```
module dinolistTest {
    // type declarations
    // configuration declarations
    // module parameters and
        // external function declarations
    // template declarations
    // function declarations
    // test case declarations
    // control part
}
```

Coding

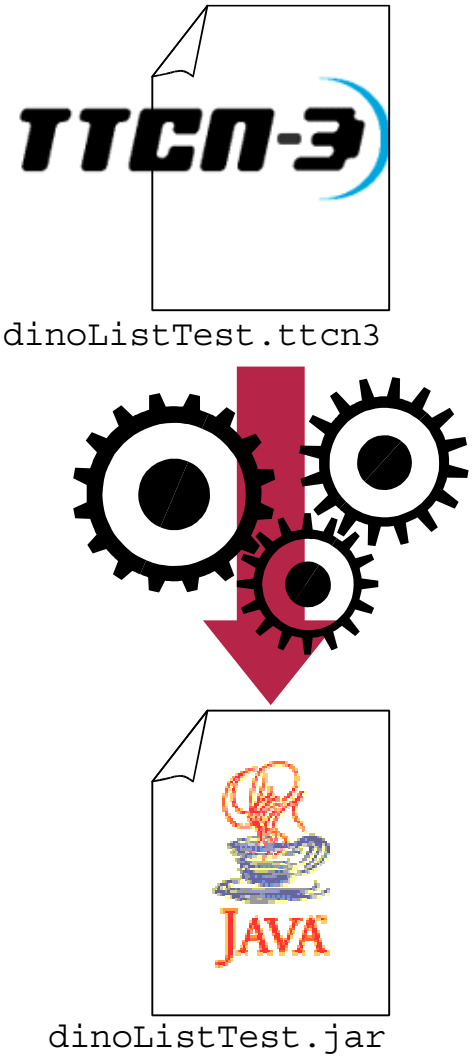
Communication

Execution Configuration

Translate TTCN-3 Into ' Executable Code

```
F:\Dino>TTthree dinoListTest
```

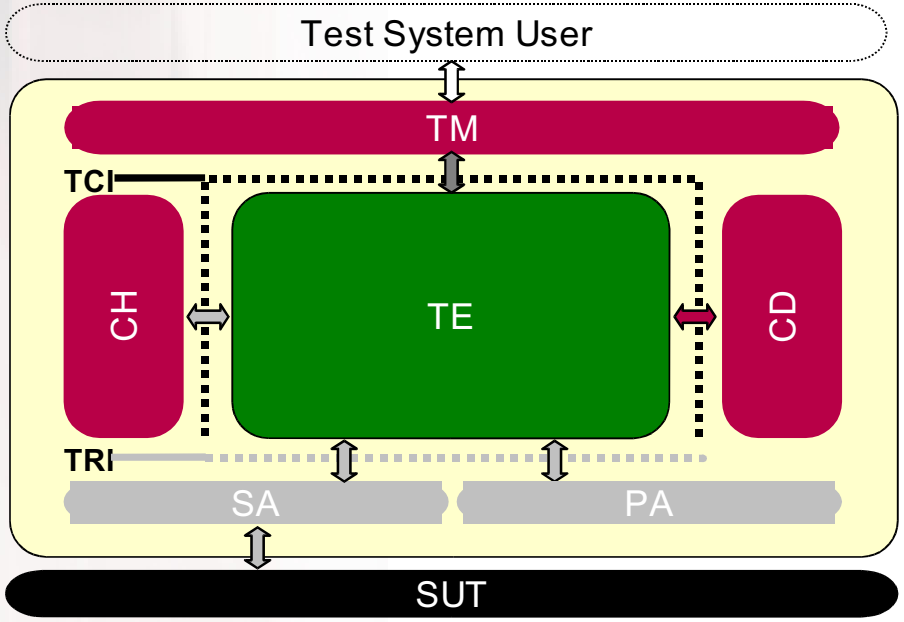
1. Reads module definitions written in the TTCN-3 core notation
2. Compiles them into Java sources
3. Java sources will be compiled into byte code class files and combined into a single JAR archive file



Steps To Implement TTCN-3

1) Translate TTCN-3 into executable code
2) Adapt runtime environment to test management and codecs
3) Implement communication aspects

TCI – Distribution,Management and Codec Adaptation



- *Facts on the TTCN-3 Control Interfaces (TCI)*

  Standardized (part 6)

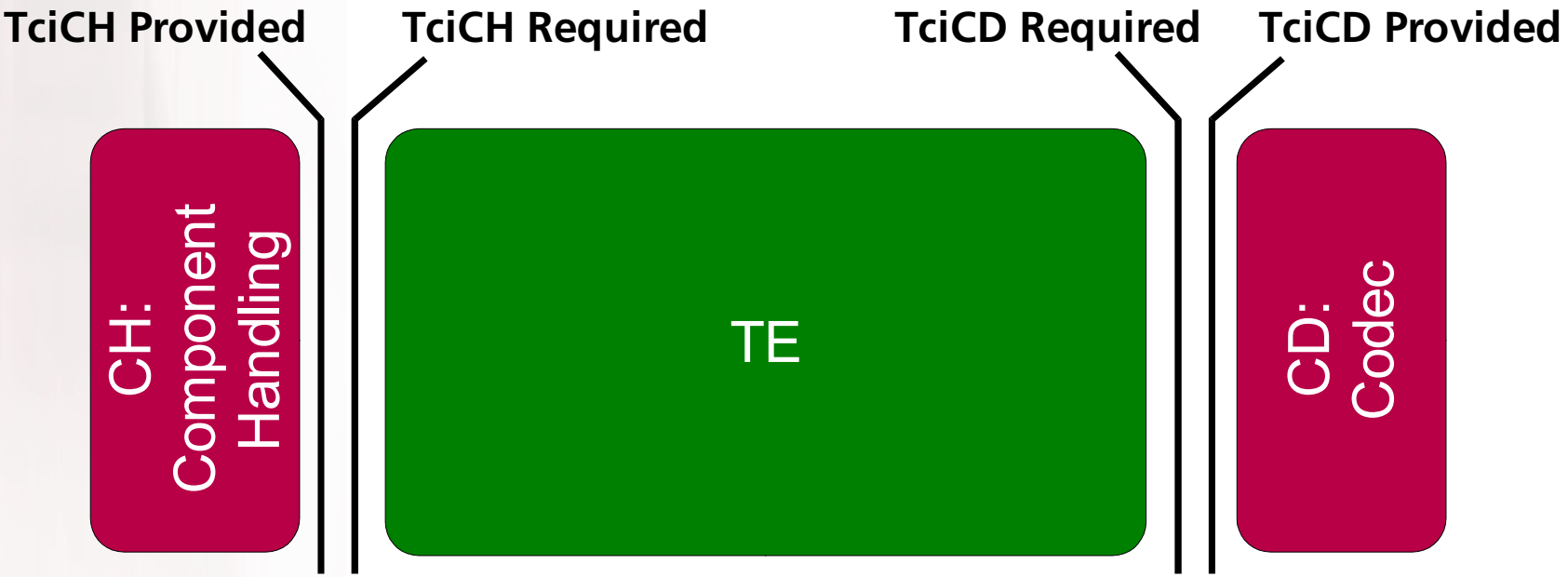
  Language independent specification

  Multi-vendor support

- TTthree – Management and Component Handling
- Basic encoder support
- Decoder TTCN-3 type dependend

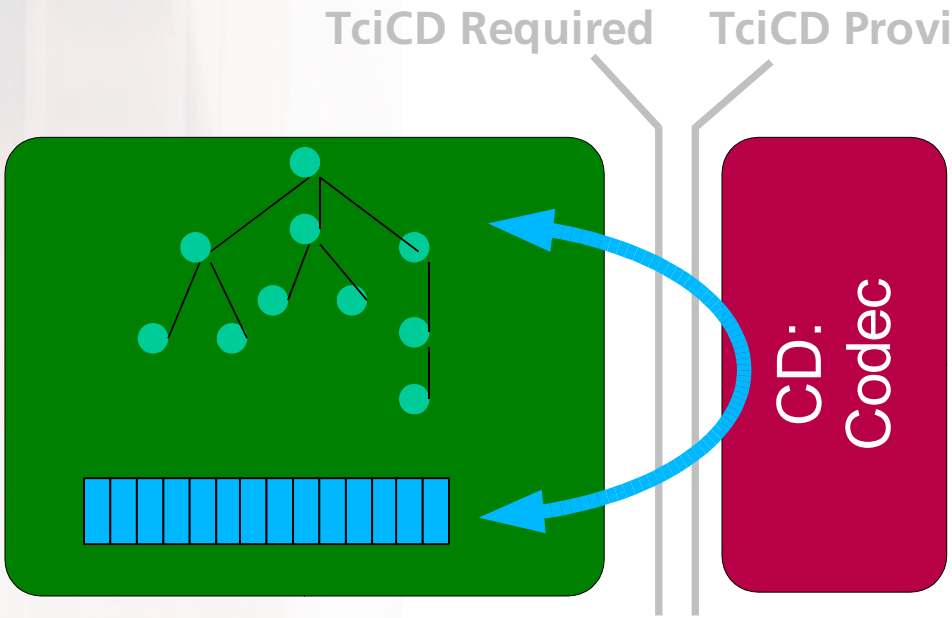
  Adaptation needed

Common Structure of Sub-Interfaces



- Have to be provided by the user
- Can rely on requested functionality of the TE
- Applies to all TCI interfaces

The Codec and Value Interface



- *Facts on the TTCN-3 Type and Value Interface*

  TE maintaines abstract type and data presentaion

  Codec translates between abstract and concrete presentation

- Management of different codecs
- Representation of TTCN-3 type system
- Representation of TTCN-3 value system

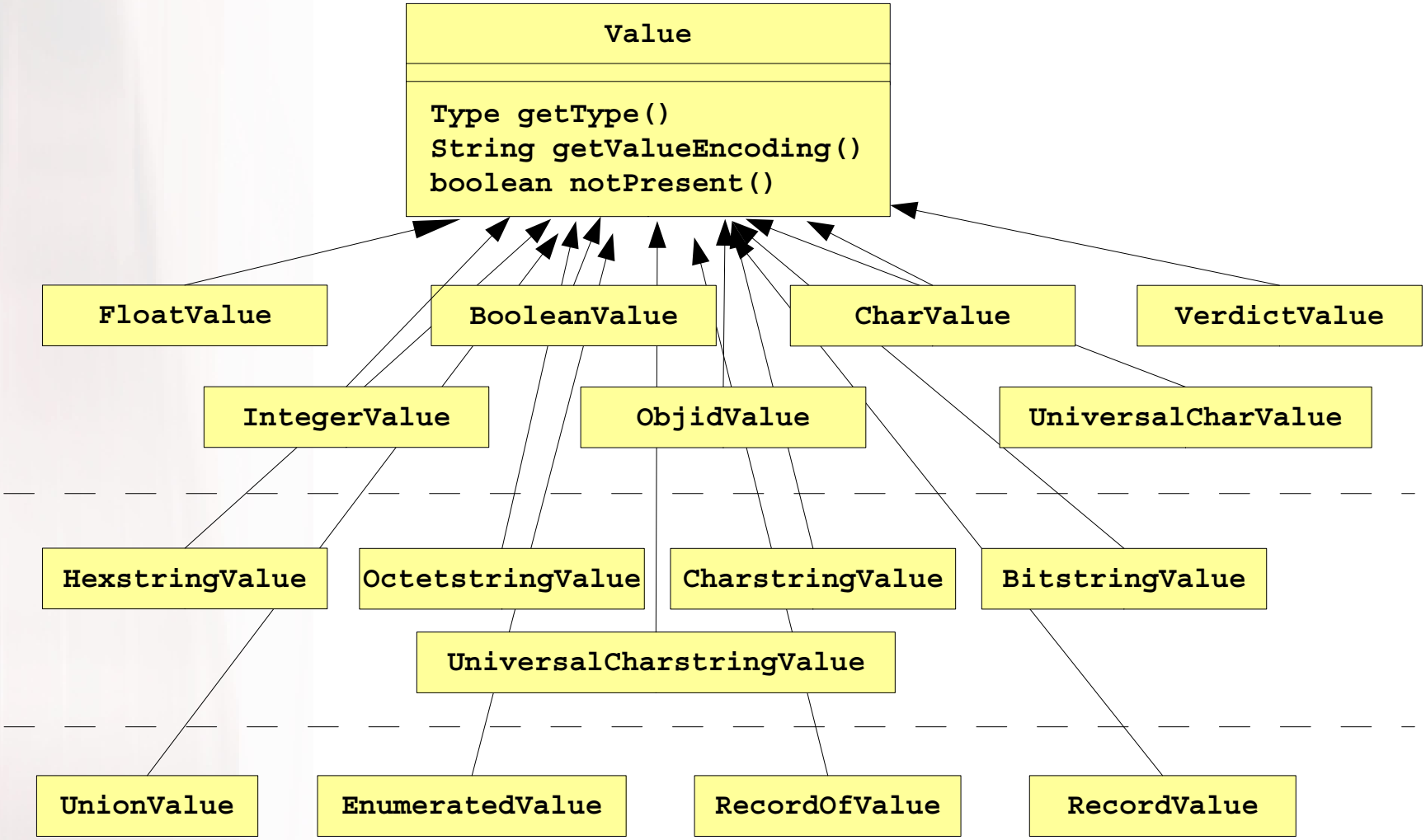
The Abstract Data Type (ADT) Model

- Modeling of TTCN-3 types and values
  - High-level description of data for communication
  - Passed to the encoder / decoder
  - Set of operations define the functionality of the type
  - Concrete representation of the types is done in the language mappings
- Model consists of two parts
  - Abstract data type `Type`
  - Different abstract data types for TTCN-3 values

ADT: `Type`

- Only types defined in TTCN-3 modules can be accessed
- No creation of user defined types at the TCI
- Complete set of operations
  - `TciModuleIdType getDefiningModule()`
  - `String getName()`
  - `TciTypeClassType getTypeClass()`
  - `String getTypeEncoding()`
  - `Value newInstance()`

ADT: Value


Value
Type getType()
String getValueEncoding()
boolean notPresent()
FloatValue
BooleanValue
CharValue
VerdictValue
IntegerValue
ObjidValue
UniversalCharValue
HexstringValue
OctetstringValue
CharstringValue
BitstringValue
UniversalCharstringValue
UnionValue
EnumeratedValue
RecordOfValue
RecordValue

The Codec Interface

- Encoding
  - Internal TTCN-3 data representation to bitstring
  - Access via the Value interface
  - Independend from the SUT
- Decoding
  - Bitstring to TTCN-3 data representation
  - Based upon a decoding hypothesis
  - TE may query multiple times for the decoding of the same bitstring
- Complete set of provided operations
  - `Value decode(in TriMessageType message, in Type hyp)`
  - `TriMessageType encode(in Value value)`

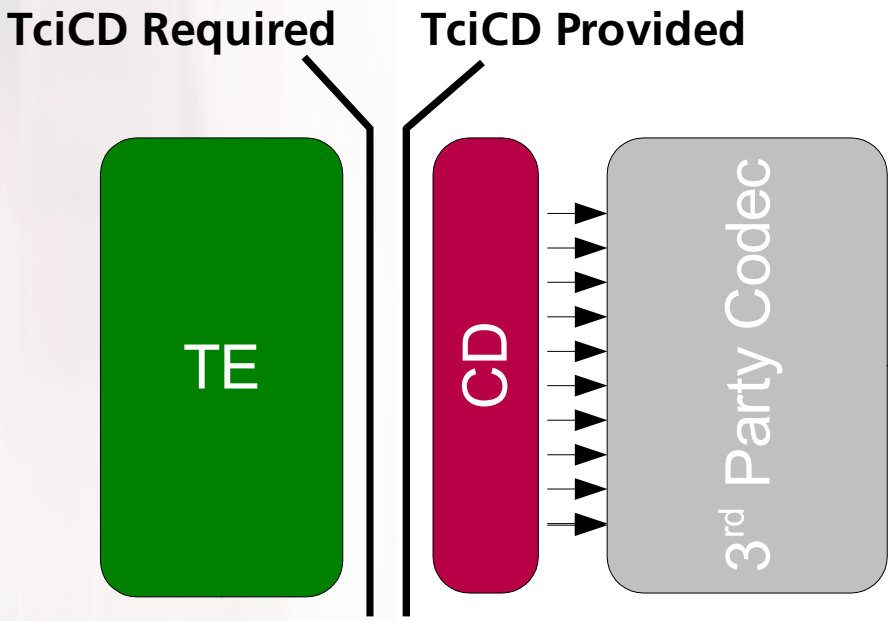
The `decodingHypothesis`

- The interpretation of an arbitary bitstring is context sensitiv
- Example: What is `'44494E4F'O` ?
  - An octetstring: `'44494E4F'O`
  - An integer: `1145654863`
  - A charstring: `"DINO"`
- Decode() can be read as follows:
  - Try to decode the provided bitstring, with the appropriated decoding rules into a value of given type
  - If you suceed, return the value
  - If you fails, return `NULL`

The Codec Interface (cont'd)

- Complete set of required operations
  - `Type getTypeForName(...)`
  - `Type getInteger()`
  - `Type getFloat()`
  - `Type getBoolean()`
  - `Type getChar()`
  - `Type getUniversalChar()`
  - `Type getObjid()`
  - `Type getCharstring ()`
  - `Type getUniversalCharstring ()`
  - `Type getHexstring ()`
  - `Type getBitstring()`
  - `Type getOctetstring ()`
  - `Type getVerdict()`
  - `void tciErrorReq(...)`

How to integrate external codecs



Depending on the codec ->
Generic solutions possible

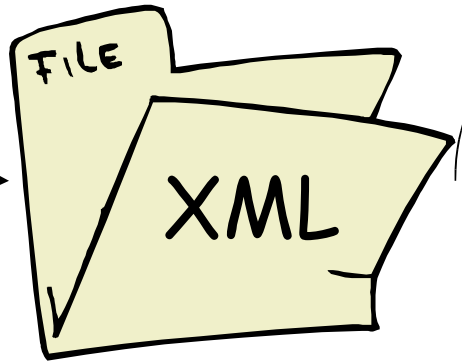
- Codecs provide
  - Operations to construct values
  - Operations to query values
  - Operations to encode values
  - Operations to decode bit-string
- CD provides
  - Operations to construct values
  - Operations to query values
- CD implementation maps TCI value structures into codec value structures

<testing_tech>

Some coding examples

Taken from the Dino example

Example – The Dinosaur Database



Paleontologist
feeding the database

Student
requesting data

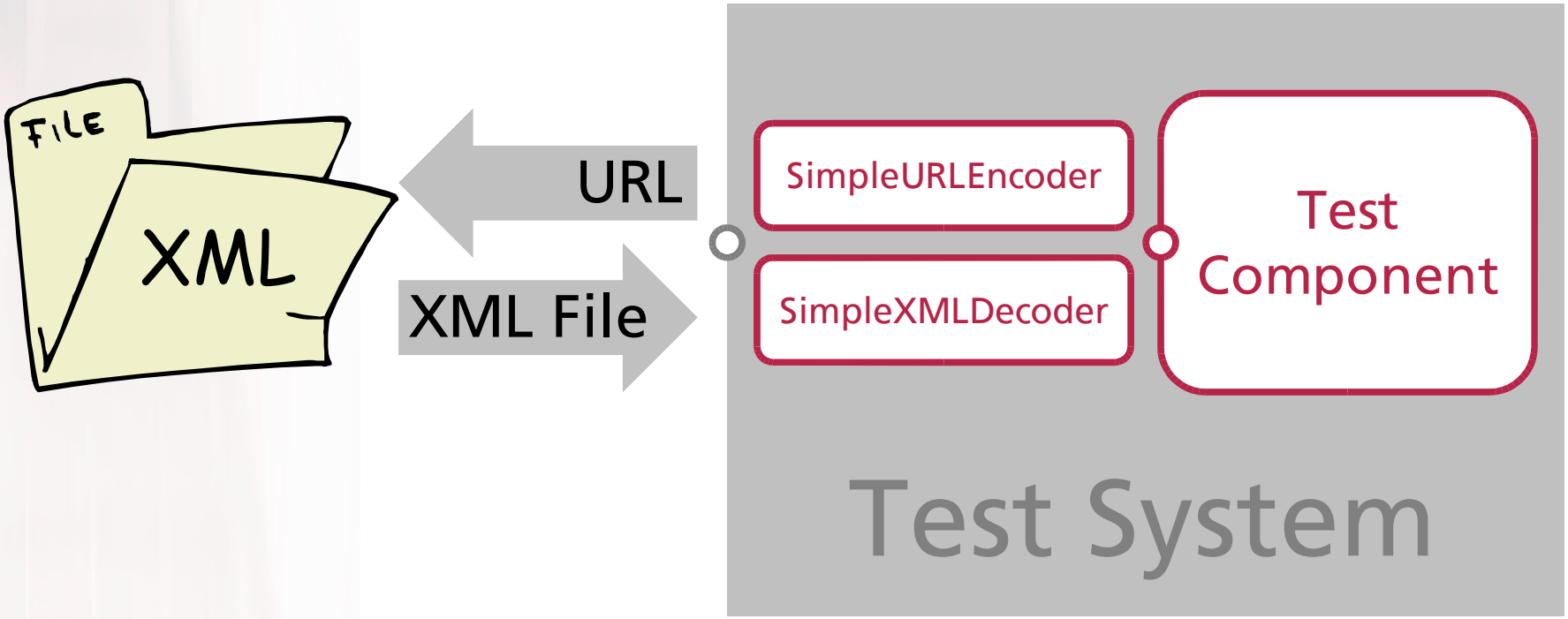
The Codec

- In order to send real data over a TTCN-3 port, encoding functionality has to be provided to the TTCN-3 runtime behavior
- Incoming data has to be decoded into a TTCN-3 value, to allow matching against TTCN-3 template definitions

Two entities have to be implemented

- Encoder – TTCN-3 data to real world data
- Decoder – real world data to TTCN-3 data

Codec practically



`SimpleURLEncoder` – TTCN-3 data to an URL

`SimpleXMLDecoder` – XML file to TTCN-3 data

TTCN-3 data as defined in the TTCN-3 module

Structured Type Defintion

```
type record url {
  charstring      protocol,
  charstring      host,
  charstring      file
}

type set of dinosaur dinolist;

type record dinosaur {
          charstring   name,
  integer         len,
  integer         mass optional,
  charstring      time,
  charstring      place
}
```

Relevant TTCN-3 Definitions

```
type set of dinosaur dinolist;
type record dinosaur {
    charstring      name,
    charstring      len,
    charstring      mass,
    charstring      time,
    charstring      place
 }
type record url {
   charstring       protocol,
   charstring       host,
   charstring       file
 }
template url requestURL := {
   protocol         := "http://",
   host             := "www.testingtech.de",
   file             := "/TTCN-3_Example/dinolist.xml"
 }
type port httpTestPortType message {
   out    url;
   in     dinolist;
}
```

Declaration of Database

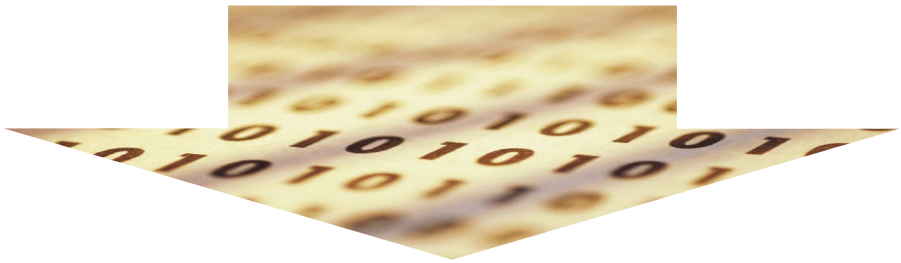
Declaration of URL

Declaration of Interface

SimpleURLEncoder

- Must implement the interface `org.etsi.ttcn.tci.TciCDProvided`
- Encodes a TTCN-3 record into an URL string

```
template url requestURL := {
    protocol        := "http://",
    host            := "www.testingtech.de",
    file            := "/TTCN-3_Example/dinolist.xml"
}
```



`http://www.testingtech.de/TTCN-3_Example/dinolist.xml`

public class **SimpleURLEncoder** implements TciCDProvided

```
public TriMessage encode(Value value) {
    Type type = value.getType();
    if (type.getName().equals("url") && (value instanceof Data)) {
        return new NativeMessage((Data)value);
    } else {
        tciCDReq.tciErrorReq("SimpleURLEncoder: Unsupported value");
    }
}
```

- Encodes only `url` types as defined in the TTCN-3 module
- Uses the “native” encoding, i.e. simply concatenates the data interpreted as a string

SimpleXMLDecoder

- Must implement the interface `org.etsi.ttcn.tci.TciCDProvided`
- Decodes a XML file into TTCN-3 set of and record

```
<?xml version="1.0" encoding="UTF-8" ?>
+ <!--  -->
<!DOCTYPE dinolist (View Source for full doctype...)>
- <dinolist>
  - <dinosaur>
      <name>Argentinosaurus</name>
      <len>35 m</len>
      <mass>80 tonnes</mass>
      <time>Albian to Cenomanian</time>
      <place>Argentina</place>
    </dinosaur>
  - <dinosaur>
      <name>Brachiosaurus</name>
      <len>22 m</len>
      <mass>30 tonnes</mass>
      <time>Kimmeridgian</time>
```

```
type set of dinosaur dinolist;

type record dinosaur {
    charstring    name,
    charstring    len,
    charstring    mass,
    charstring    time,
    charstring    place
}
```

public class **SimpleXMLDecoder** implements Decoder

```
public Value decode(TriMessage message, Type decodingHypothesis)
{
   try {
      ByteArrayInputStream bais =
        new ByteArrayInputStream(message.getEncodedMessage());
      return decodeNodes(type.newInstance(), parse(bais));
   } catch (TciException ex) { tciCDReq.tciErrorReq(ex.getMessage()); )
}

private Value decodeNodes(Value value2feed, Node node)
   throws TciException { /* ... */ }

private Element parse(InputStream input)
   throws TciException { /* ... */ }
```

- The received byte stream will be parsed by a XML parser
- The resulting DOM tree will be traversed and decoded recursively

SimpleXMLDecoder – XML parsing


```
private Element parse(InputStream input) throws TciException {
    try {
          DocumentBuilderFactory docBuilderFactory =
                  DocumentBuilderFactory.newInstance();
          docBuilderFactory.setValidating(true);
          DocumentBuilder docBuilder =
          docBuilderFactory.newDocumentBuilder();
          Document document = docBuilder.parse(input);
          input.close();
          return document.getDocumentElement();
    } catch (Exception e) { throw new TciException(e.getMessage()); }
}
```

- Uses Java API for XML Processing (JAXP) v1.2 by Sun Microsystems
- Parses the received XML file
- Returns the root element of the DOM tree

SimpleXMLDecoder – DOM Traversal


```
private Value decodeNodes(Value hypothesis, Node node)
    throws TciException {
    Type type = hypothesis.getType();
    switch (type.getTypeClass()) {
      case Type.RECORD:
      case Type.SET:
          // ...
      case Type.RECORD_OF:
      case Type.SET_OF:
          // ...
      case Type.CHARSTRING:
          // ...
      case Type.ANY:
          // ...
    }
}
```

- Decoding uses a decoding hypothesis given by the receive template, i.e. the value to be expected

SimpleXMLDecoder – DOM Traversal for TTCN-3 Record/Set


```
if (node != null && node.getNodeType() == Node.ELEMENT_NODE) {
    RecordValue record = (RecordValue) type.newInstance();
    String[] ttcnFieldNames = record.getFieldNames();
    NodeList nodeChildren = node.getChildNodes();
    for (int j=0; j<ttcnFieldNames.length; j++) {
          Value fieldValue = record.getField(ttcnFieldNames[j]);
          for (int i=0; i<nodeChildren.getLength(); i++) {
               Node child = nodeChildren.item(i);
               if ((child.getNodeType() == Node.ELEMENT_NODE) &&
                    (child.getNodeName().equals(ttcnFieldNames[j]))) {
                        record.setField(ttcnFieldNames[j],
                        decodeNodes(fieldValue, child));
               }
          }
    }
    return record;
}
```

- Records/sets and their fields have to be XML elements
- Element nodes will be decoded, if the node name is equal to the respective record/set field name

SimpleXMLDecoder – DOM Traversal for TTCN-3 Record/Set Of


```
if (node != null && node.getNodeType() == Node.ELEMENT_NODE) {
    RecordOfValue recordOf = (RecordOfValue) type.newInstance();
    Value elementValue = recordOf.getElementType().newInstance();
    recordOf.setLength(0);
    NodeList nodeChildren = node.getChildNodes();
    for (int i=0; i<nodeChildren.getLength(); i++) {
          Node child = nodeChildren.item(i);
          if (child.getNodeType() == Node.ELEMENT_NODE) {
                    recordOf.appendField(decodeNodes(elementValue ,child));
          }
    }
    return recordOf;
}
```

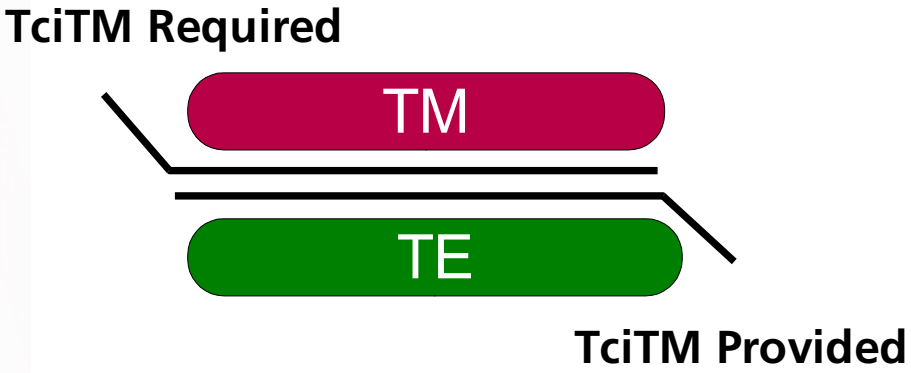
- Records/sets of and their fields have to be XML elements
- Element nodes will be decoded and appended to the record/set of

SimpleXMLDecoder – DOM Traversal for TTCN-3 Charstring


```
NodeList nodeChildren = node.getChildNodes();
for (int i=0; i<nodeChildren.getLength(); i++) {
    Node child = nodeChildren.item(i);
    if (child != null && child.getNodeType() == Node.TEXT_NODE) {
          CharstringValue charstring =
                   (CharstringValue) type.newInstance();
          charstring.setCharstring(child.getNodeValue());
          return charstring;
    }
}
```

- The data is located in the child XML node
- Charstrings have to be XML text nodes
- The character string value is the XML text node value

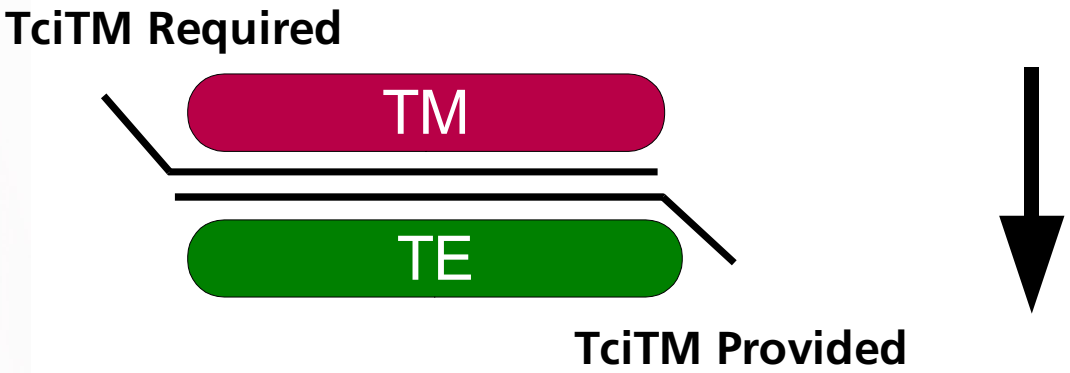
The Test Management Interface



- TM provides
  - User Interface, incl. Error reporting
  - Keeps track of test case execution
  - Module Parameter Resolving
  - Logging
- TE Provides
  - Entry points to the TE
  - Start/Stop test case
  - Start/Stop control part

- 6 operations provided
- 9 operations required

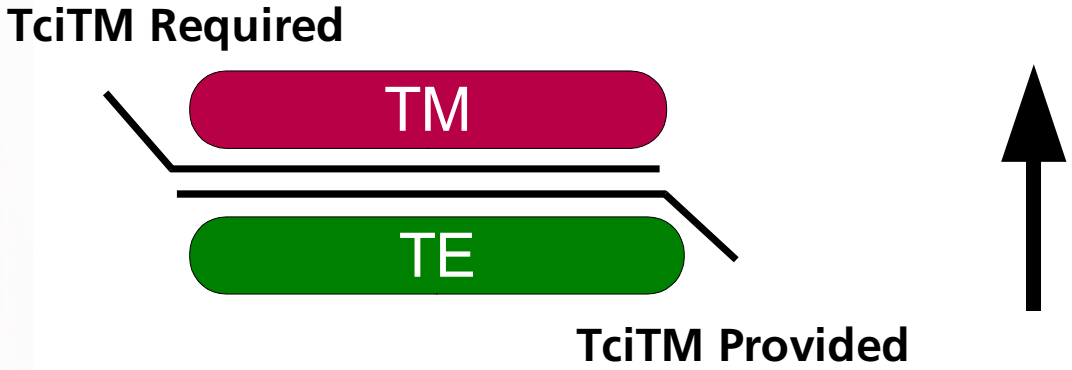
The TciTMRequired Interface



- TE offers the entry point for test case execution and some rudimentary database functionality
- Complete set of operations

```
void tciRootModule(moduleId) ;
TciModuleParameterList    tciGetModuleParameters(moduleId);
TciTestCaseIdList         tciGetTestCases();
TciParameterTypeList      tciGetTestCaseParameters(testCaseId);
TriPortIdList             tciGetTestCaseTSI(testCaseId);
void                tciStartTestCase(testCaseId, parameterList ) ;
void                tciStopTestCase() ;
TriComponentId      tciStartControl() ;
void                tciStopControl() ;
```

The TciTMProvided Interface



- TM implements user interface and resolves parameter resolution at runtime
- Complete set of operations
  - `void tciControlTerminated() ;`
  - `void tciError(errorMsg);`
  - `Value tciGetModulePar(parameterId);`
  - `void tciLog(testComponentId, message);`
  - `void tciTestCaseStarted(testCaseId, parameterList, timerVal);`
  - `void tciTestCaseTerminated(verdict, parameterList);`

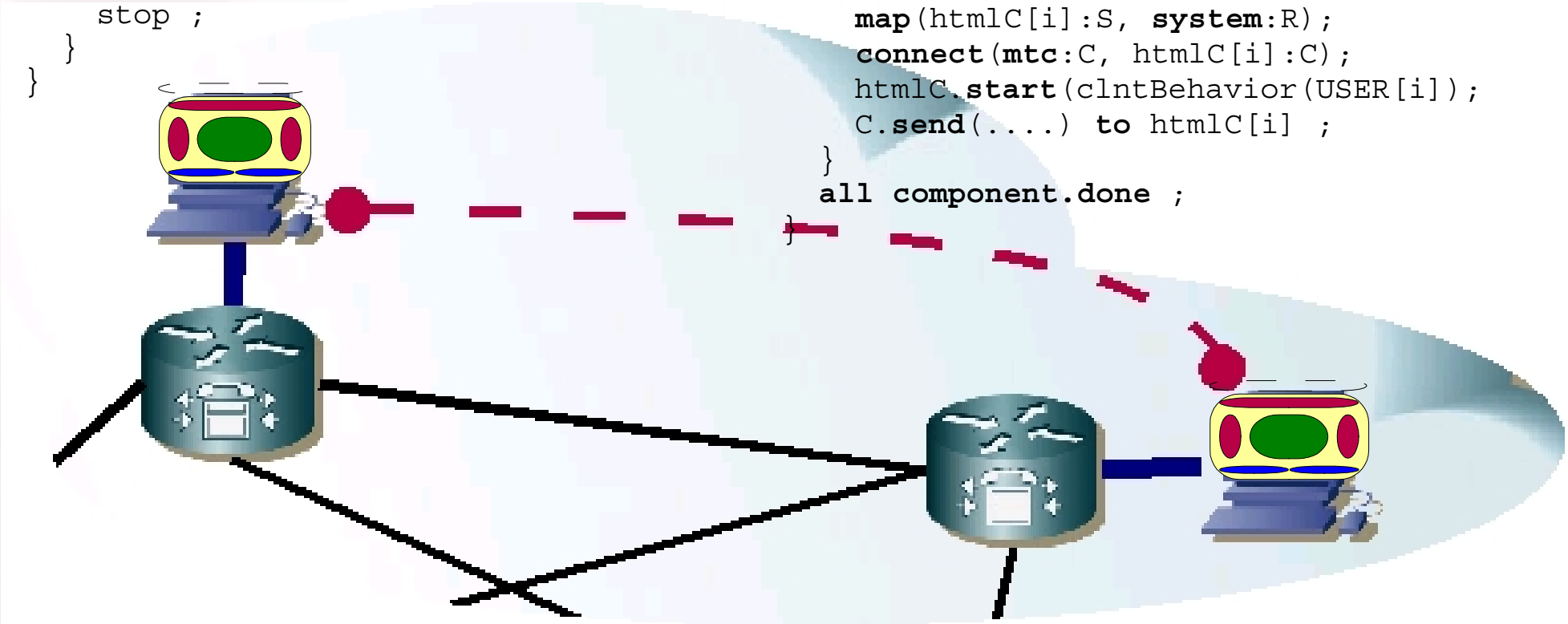
The Component Handling Interface

- Management of TTCN-3 components
  - No implementation of TTCN-3 functionality
  - Distribution of TTCN-3 configuration operations
  - Distribution of TTCN-3 intercomponent communication
- Concepts
  - Distributed TE, i.e. multiple TEs
  - A single Component Handling entity
  - Presence of a distinct TE*, i.e. the TE where a test case or the control part has been started
  - Distinct TE* responsible for final verdict calculation
- The most complex interface
  - 17 required operations
  - 17 provided operations

Example: HTML Scalability Testing

```
function clntBehavior(...) {
      runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

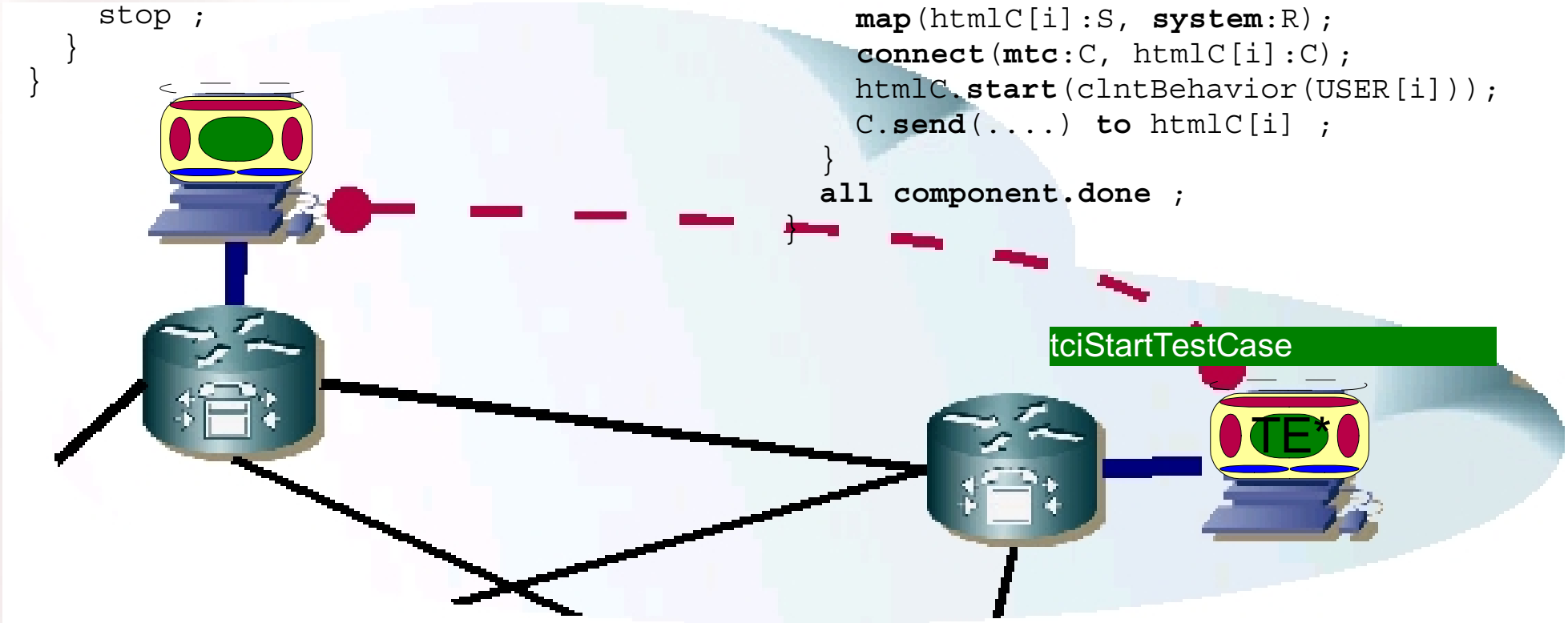
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
       runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

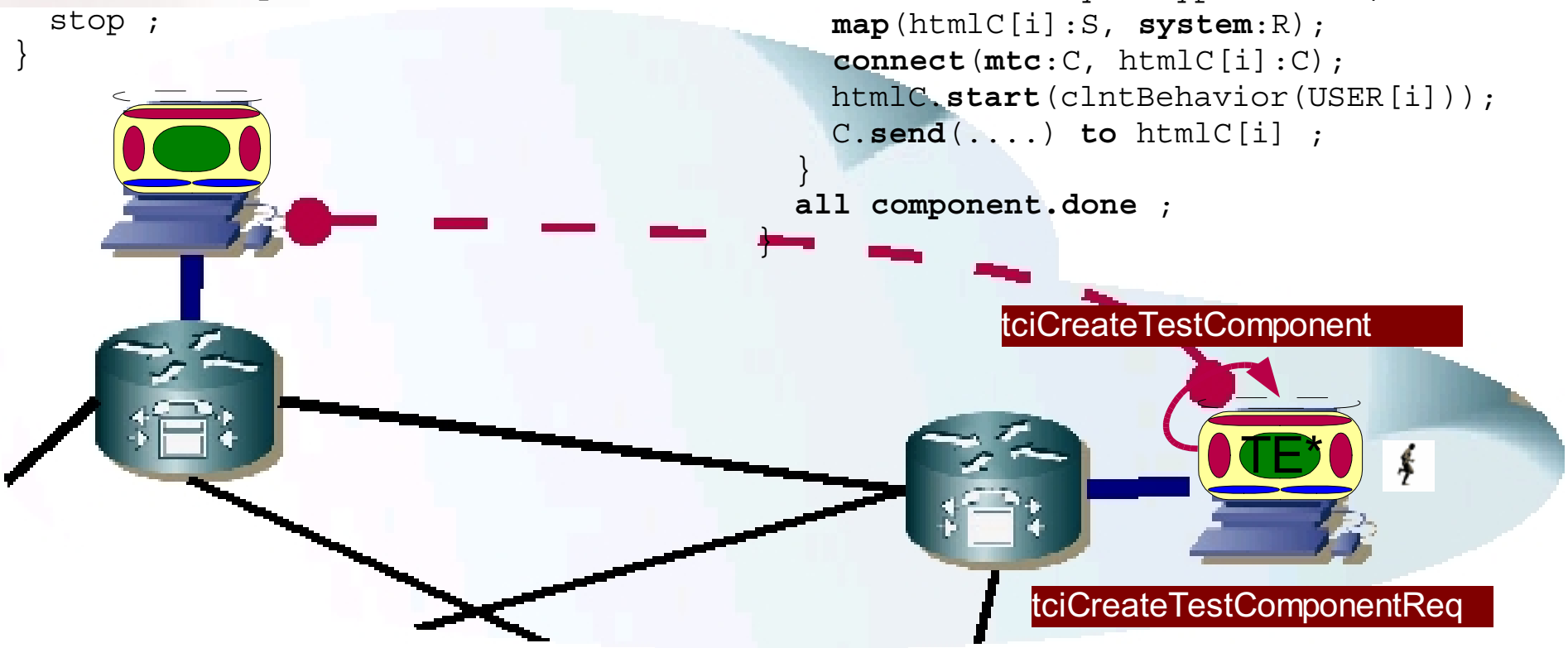
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
       runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

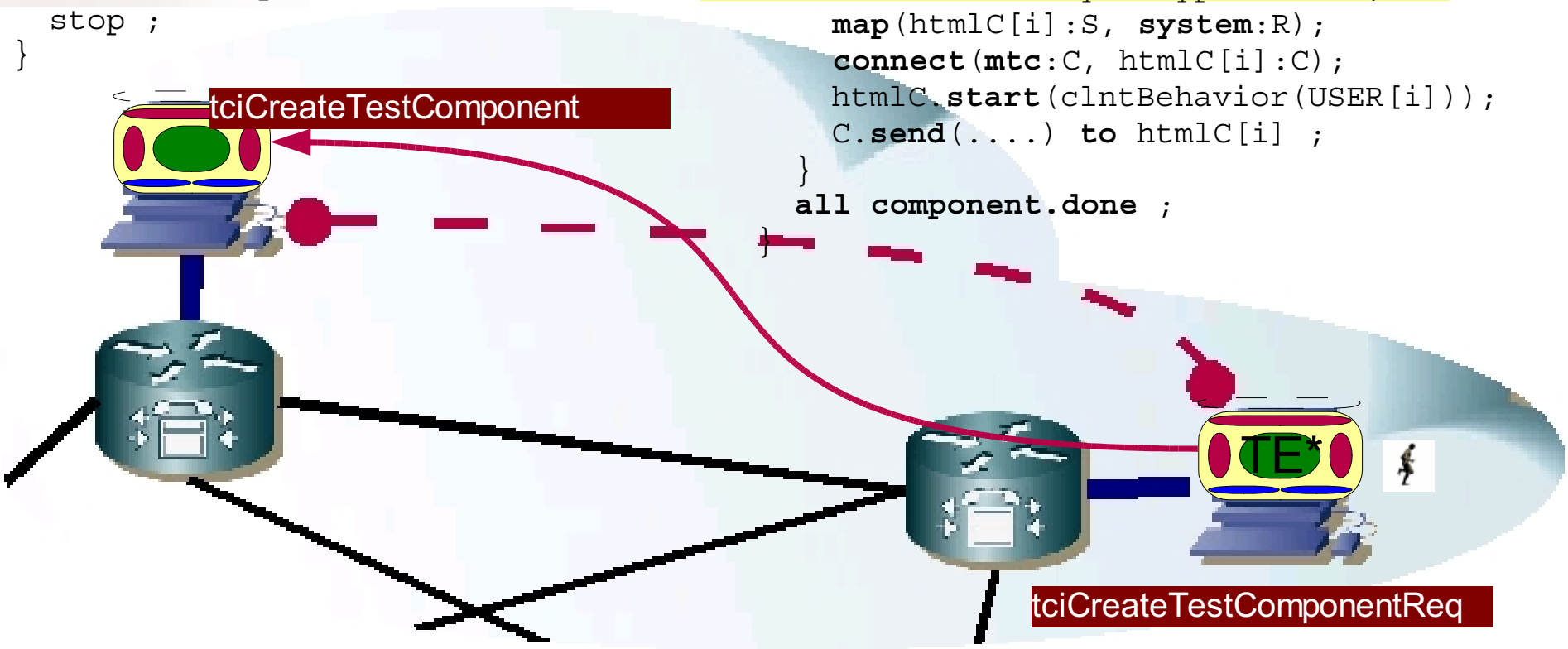
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
      runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

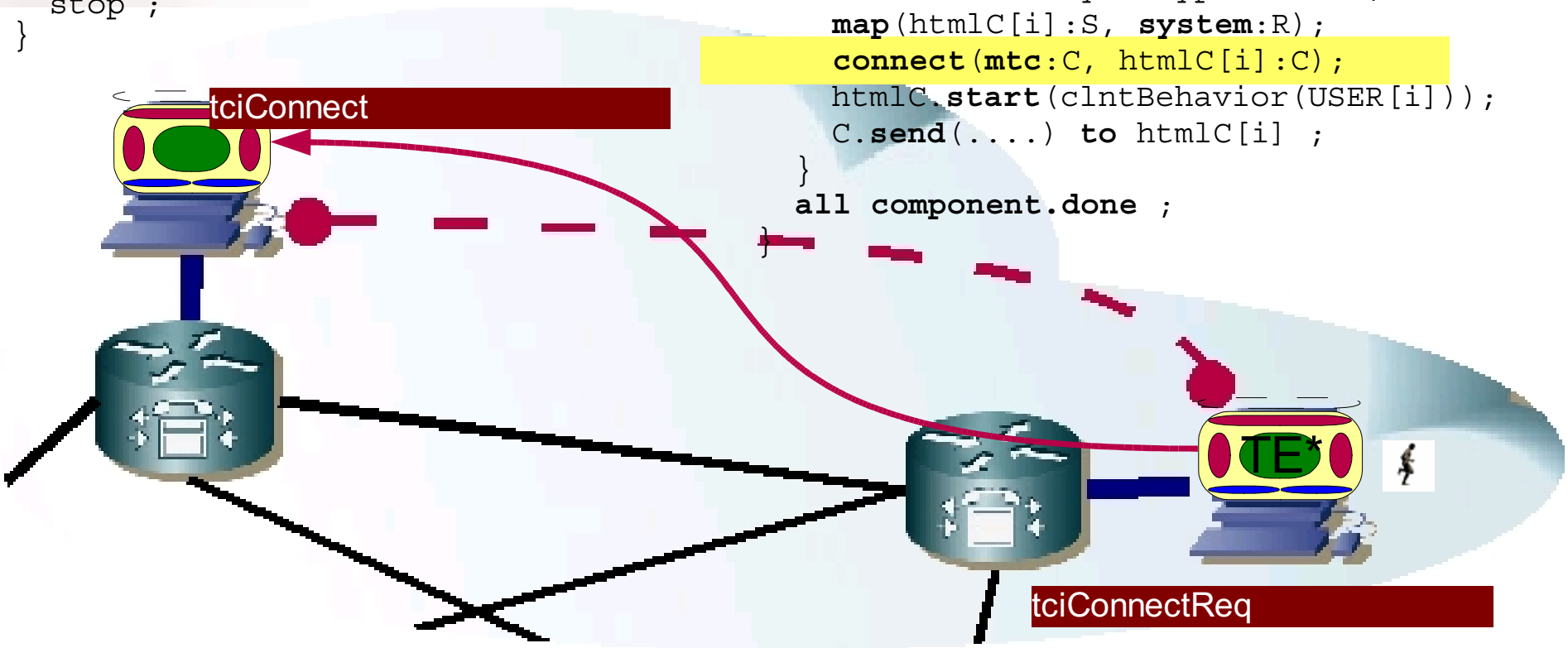
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
      runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

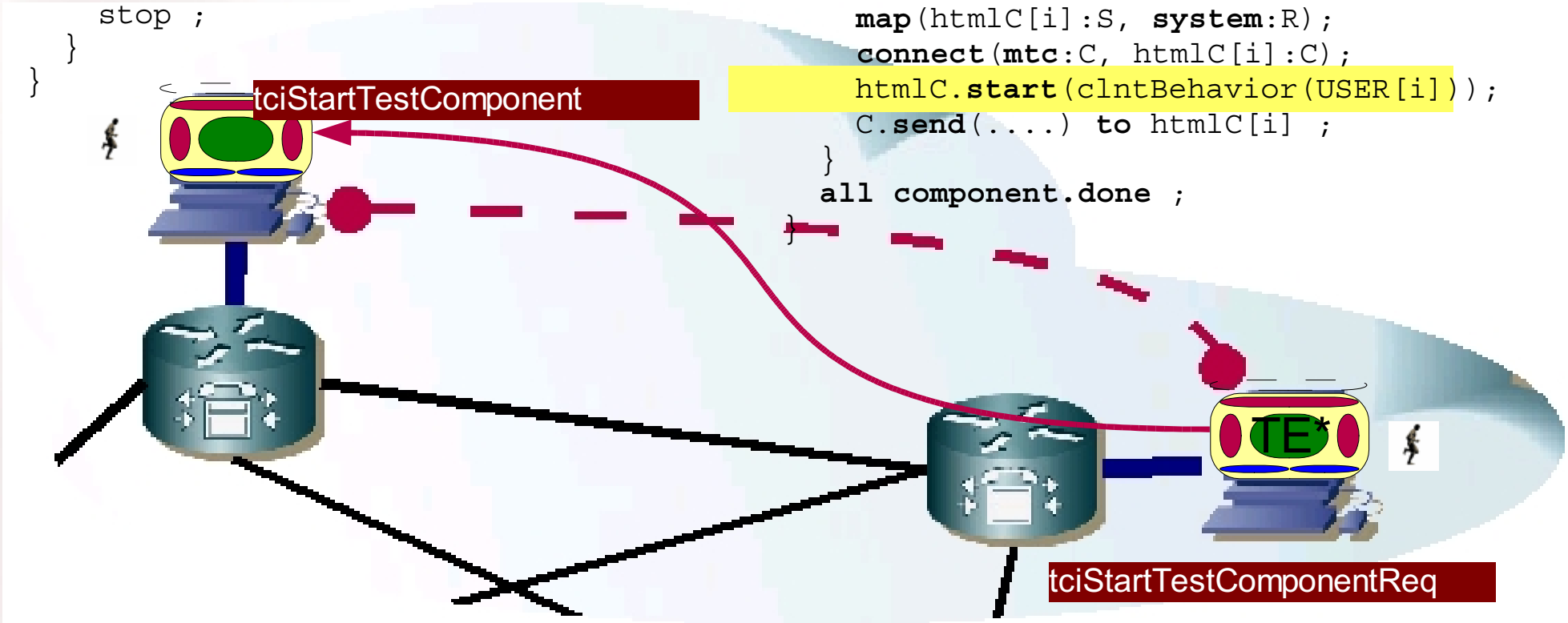
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
      runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

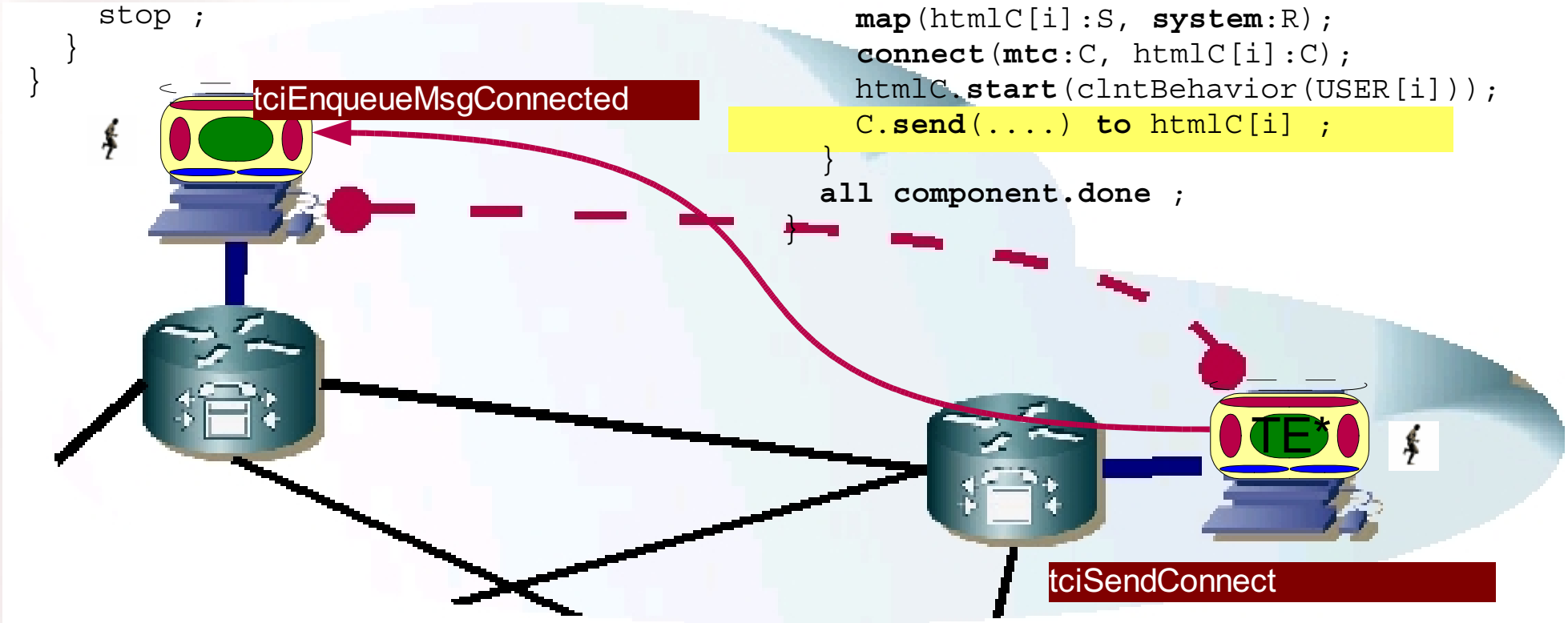
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
      runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

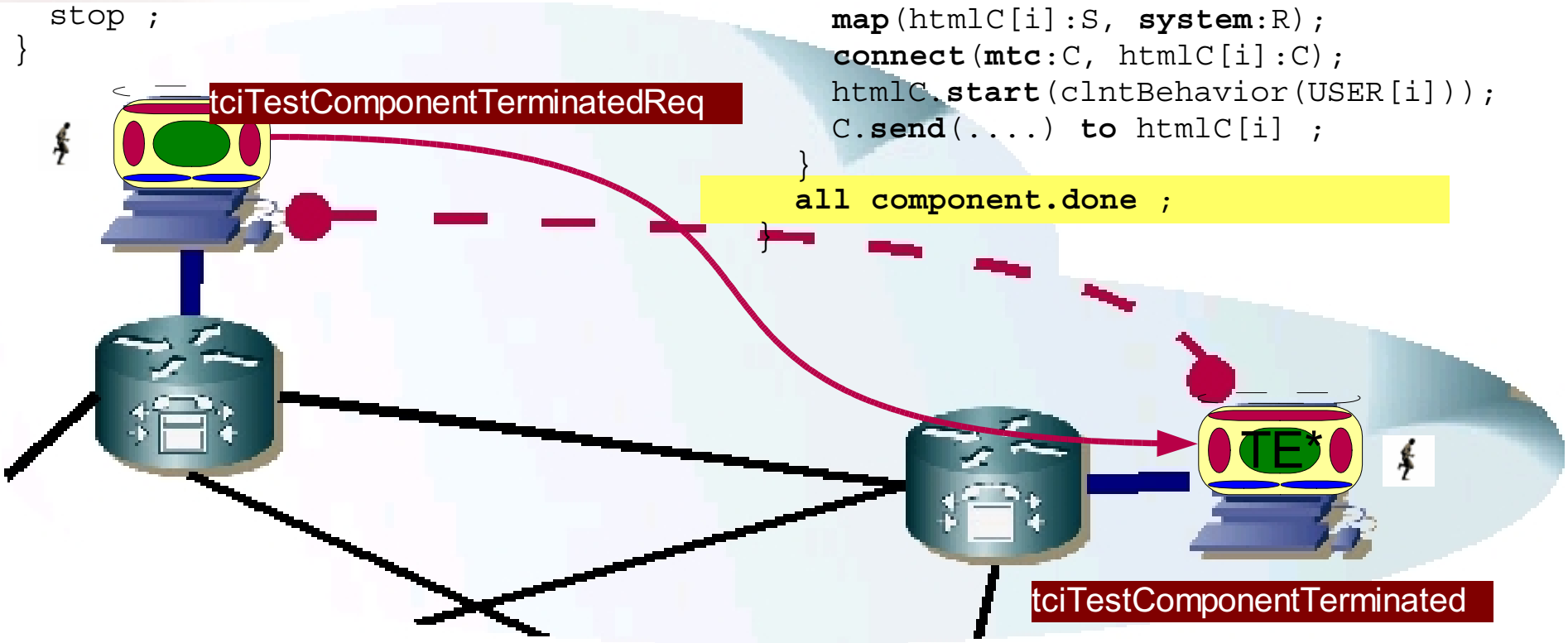
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
      runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

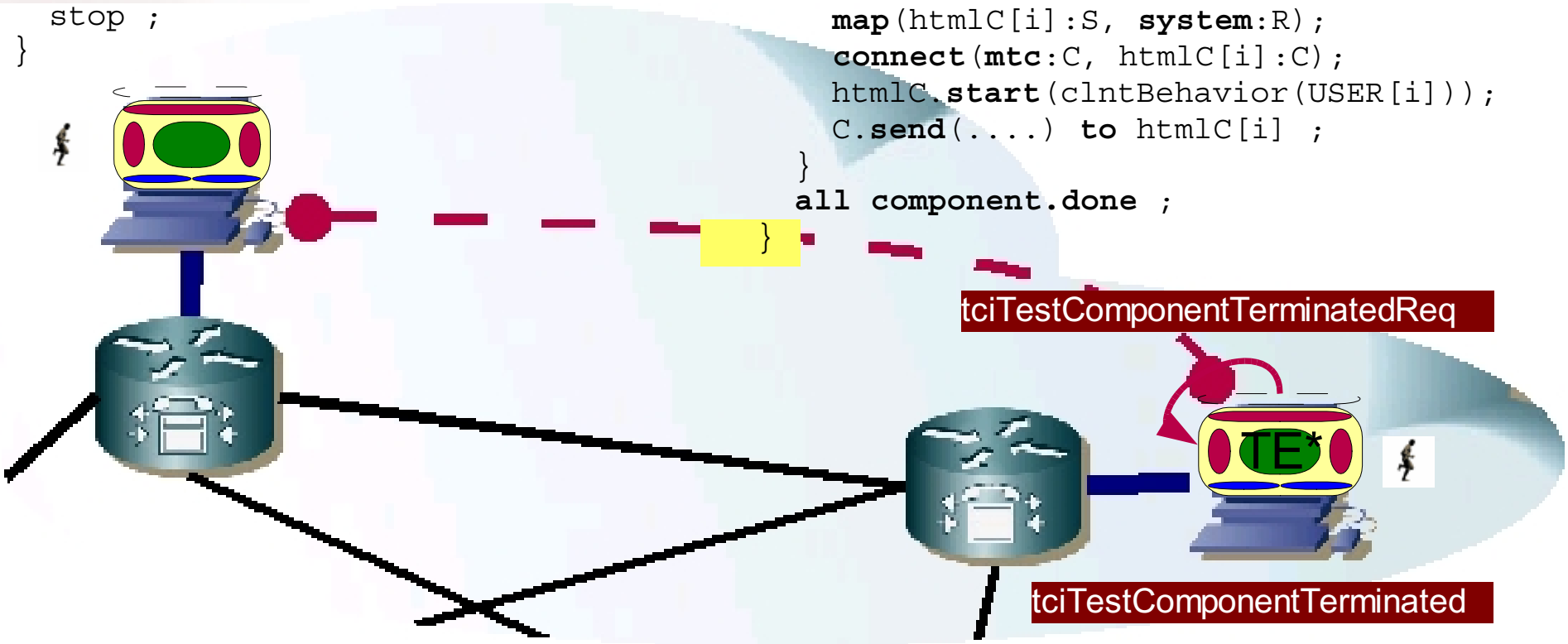
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
      runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

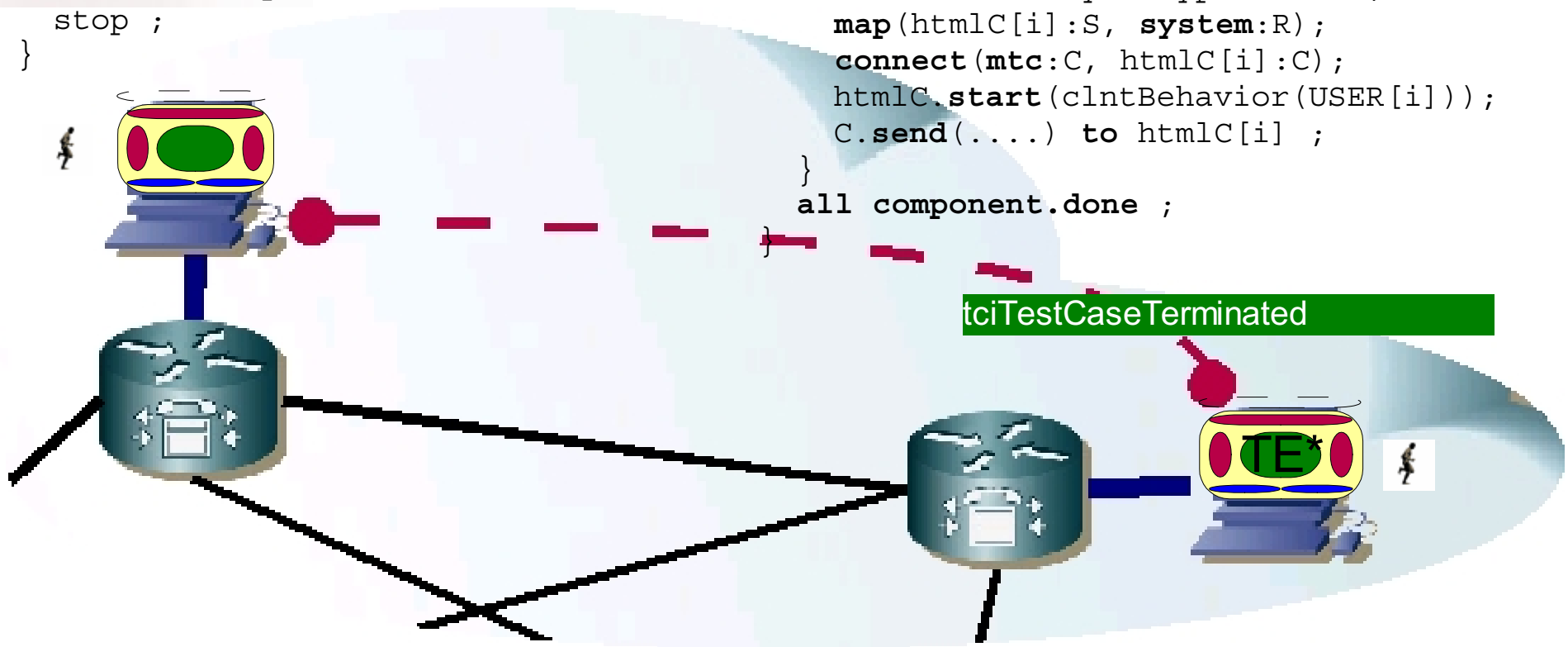
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Example: HTML Scalability Testing

```
function clntBehavior(...) {
      runs on MyPtcType
  {
   // Do what you have to do !
    setverdict(pass) ;
    stop ;
  }
}
```

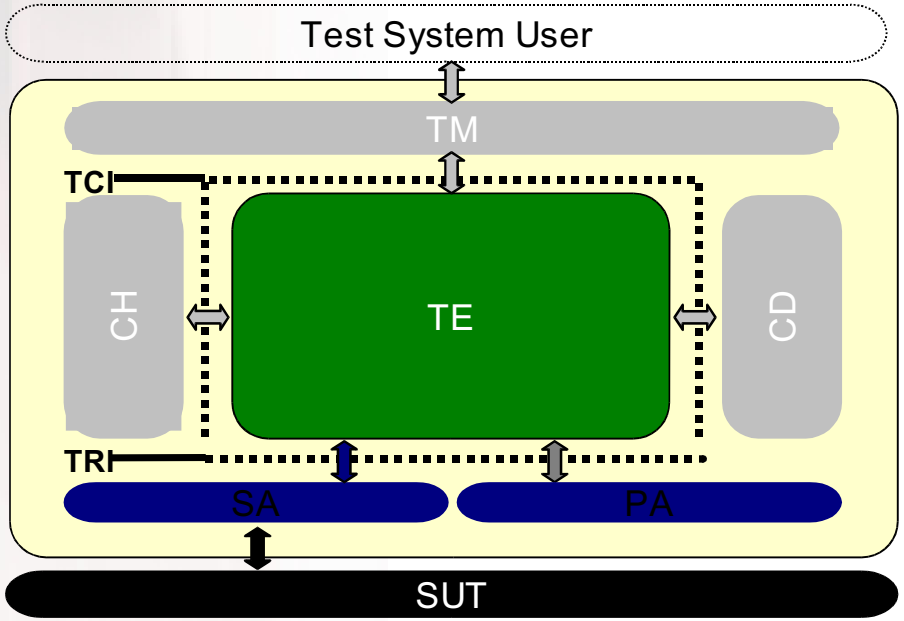
```
testcase scalabilityTest()
    runs on MyMtcType system MyTSI
{
  var integer i;

  for(i:=0;i<MAXNUMBER;i:=i+1) {
    htmlC[i] := MyPtcType.create;
    map(htmlC[i]:S, system:R);
    connect(mtc:C, htmlC[i]:C);
    htmlC.start(clntBehavior(USER[i]));
    C.send(....) to htmlC[i] ;
  }
  all component.done ;
}
```



Steps To Implement TTCN-3

1) Translate TTCN-3 into executable code
2) Adapt runtime environment to test management and codecs
3) Implement communication aspects

TRI – Communication Adaptation



- *Facts on the TTCN-3 Runtime Interfaces (TRI)*

  Standardized (part 5)

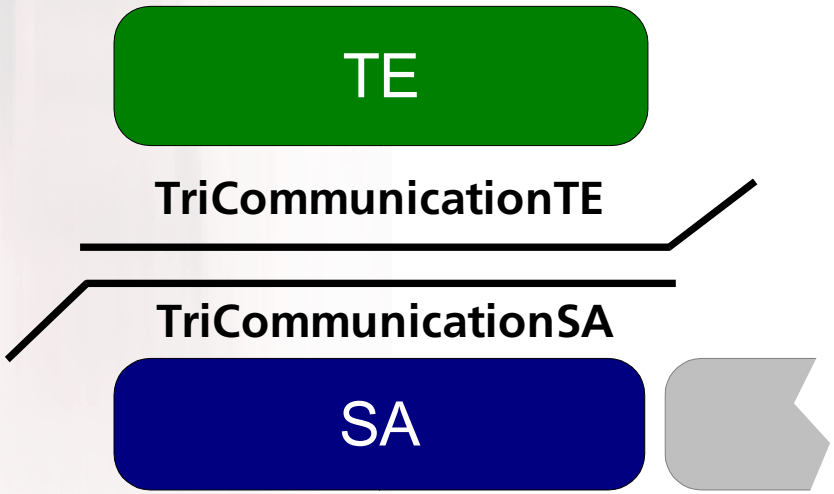
  Language independent specification

  Multi-vendor support

- TTthree – Default time implementation
- Configuration dependend

  For each configuration done once

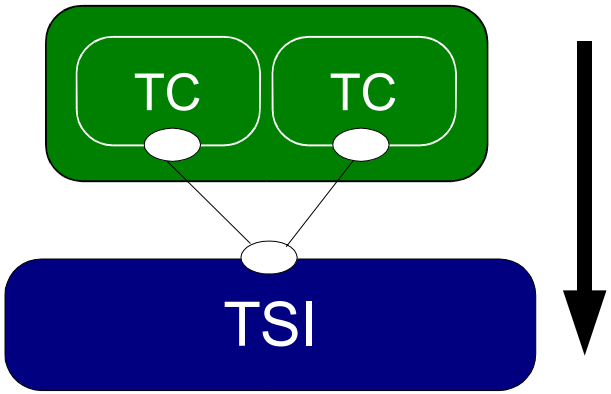
The TRI Communication Interface



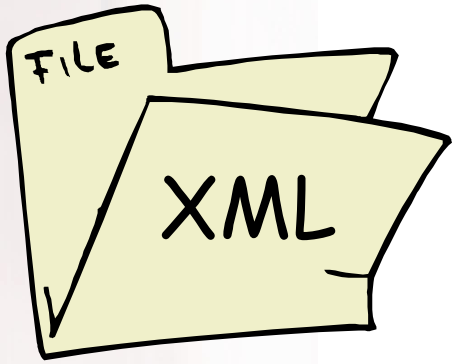
- Interface structure
  - Due to historical reasons different naming
  - Applies to all TRI inter-faces
  - SA reports status back
  - TE indicates error

TriCommunicationSA Interface

- Defines setting up configuration and sending of message to and/or calling of operations in the SUT
- Complete set of operations
  - `TriStatusType triSAReset();`
  - `TriStatusType triExecuteTestCase(...);`
  - `TriStatusType triMap(...);`
  - `TriStatusType triUnmap(...);`
  - `TriStatusType triSend(...);`
  - `TriStatusType triCall(...);`
  - `TriStatusType triReply(... );`
  - `TriStatusType triRaise(...);`
  - `TriStatusType triSUTactionInformal(...);`
  - `TriStatusType triSUTactionTemplate(...);`



Implementing the SA – The Request Interface



URL

http://www.testingtech.de/TTCN-3_Example/dinolist.xml

XML File

```
<?xml version="1.0" encoding="UTF-8" ?>
+ <!--  -->
<!DOCTYPE dinolist (View Source for full doctype...)>
- <dinolist>
  - <dinosaur>
      <name>Argentinosaurus</name>
      <len>35 m</len>
      <mass>80 tonnes</mass>
      <time>Albian to Cenomanian</time>
      <place>Argentina</place>
    </dinosaur>
  - <dinosaur>
      <name>Brachiosaurus</name>
      <len>22 m</len>
      <mass>30 tonnes</mass>
      <time>Kimmeridgian</time>
```

Configuration Declarations

```
/** component type definitions */
 type component httpTestComponent {
     port httpTestPortType    httpPort;
     timer                    localTimer := 3.0;
 }

 type component httpTestSystemComponent {
     port httpTestPortType    httpTestSystemPort;
 }
```

Test Adapter Practically – `XMLTestAdapter`

- Extends basic test adapter class
  `com.testingtech.ttcn.tri.TestAdapter`
- Overrides the following methods
  - `public TriStatus **triMap**(TriPortId compPortId, TriPortId tsiPortId)`
  - `public TriStatus **triUnmap**(TriPortId compPortId, TriPortId tsiPortId)`
  - `public TriStatus **triSend**(TriComponentId componentId, TriPortId tsiPortId, TriAddress address, TriMessage sendMessage)`
  - `public Encoder **getEncoder**(String encodingName)`
  - `public Decoder **getDecoder**(String encodingName)`
  - `public void **cancel**()`

```
public TriStatus triMap(TriPortId comp-
PortId, TriPortId tsiPortId)
```

- Maps a test component port to a test system interface port
- Typically starts the receiver loop

- Parameter `compPortId` is a port reference to the test component port
- Parameter `tsiPortId` is a port reference to the test system interface port

public TriStatus **triMap**(TriPortId compPortId, TriPortId tsiPortId)

```
public TriStatus triMap(TriPortId compPortId, TriPortId tsiPortId)
   {
       TriStatus mapStatus = CsaDef.triMap(compPortId, tsiPortId);
       if (mapStatus.getStatus() != TriStatus.TRI_OK)
       {
           return mapStatus;
       }
       return new TriStatusImpl();
   }
```

- Called whenever a TTCN-3 map statement executed
- Call **triMap** in the default SUT adapter
- **triMap** receives portIds of ports being mapped
- If succeeds returns **TRI_OK**
- Usage of predefined class TriStatusImpl()

```
public TriStatus triUnmap(TriPortId comp-
PortId, TriPortId tsiPortId)
```

- Unmaps a test component port from a test system interface port
- Typically stops the receiver loop

- Parameter `compPortId` is a port reference to the test component port
- Parameter `tsiPortId` is a port reference to the test system interface port

public TriStatus **triUnmap**(TriPortId compPortId, TriPortId tsiPortId)

```
public TriStatus triUnmap(TriPortId compPortId, TriPortId tsiPortId)
{
    TriStatus unmapStatus = CsaDef.triUnmap(compPortId, tsiPortId);
    if (unmapStatus.getStatus() != TriStatus.TRI_OK)
    {
        return unmapStatus;
    }
    return new TriStatusImpl();
}
```

- Called whenever a TTCN-3 unmap statements is executed
- Call `triUnmap` in the default SUT adapter

```
public TriStatus triSend(TriComponentId
componentId, TriPortId tsiPortId, TriAddress ad-
dress, TriMessage sendMessage)
```

- Is called when it executes a TTCN-3 send operation on a test component port
- Performs the real sending on the respective test system interface port
- Encoding of **sendMessage** has to be done prior to this operation call

- Parameter **componentId** is the sending test component
- Parameter **tsiPortId** is the test system interface port via which the message is sent
- Parameter **SUTaddress** optional destination address within the SUT
- Parameter **sendMessage** the encoded message to be send

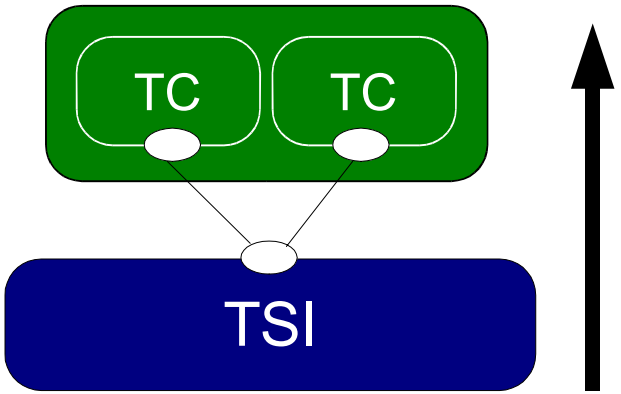
public TriStatus **triSend**(TriComponentId componentId, TriPortId tsiPortId, TriAddress address, TriMessage sendMessage) (2)

```
public TriStatus triSend(final TriComponentId componentId,
   final TriPortId tsiPortId, TriAddress address, TriMessage sendMessage) {
      try
       {
          byte[] mesg = sendMessage.getEncodedMessage();
          URL url = new URL(new String(mesg));
          final HttpURLConnection httpConn = (HttpURLConnection) url.openConnection();
          httpConn.setRequestMethod("GET"); httpConn.setUseCaches(false);
          //create the receiver thread
          Thread receiverThread = new Thread() {
             public void run() {
                  try { /* get the data */ } catch (Exception ex) {  /* do smthg */  }
                  httpConn.disconnect(); }}.start();
       } catch (Exception ex) {  /* do smthg */  }
       return new TriStatusImpl();
}
```

- Create a new URL from the already encoded message to be sent (Note: `TriMessage` is only a byte array container)

TriCommunicationTE Interface

- Defines receiving of messages and/or calling of operations in the TE
- Complete set of operations
  - `void triEnqueueMsg(...);`
  - `void triEnqueueCall(...);`
  - `void triEnqueueReply(...);`
  - `void triEnqueueException(...);`

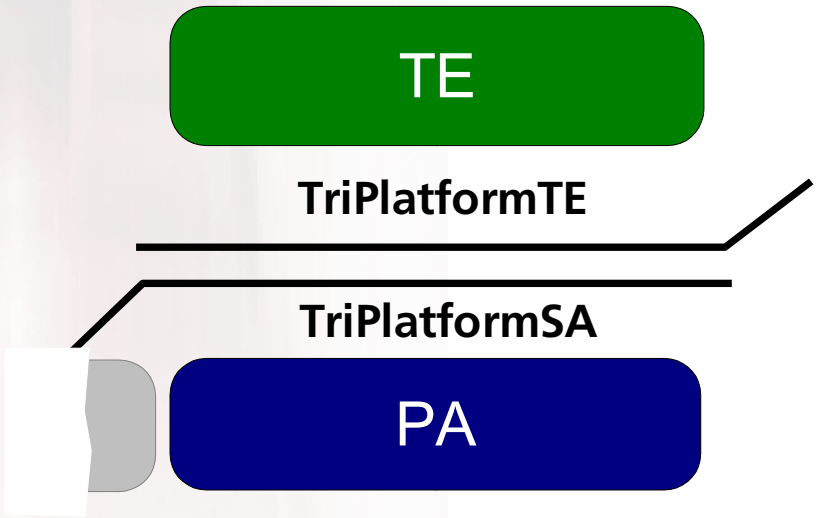


public void **triEnqueueMsg**(TriPortId tsiPortId, TriAddress address, TriComponentId componentId, TriMessage sendMessage)

```
...
     Thread receiverThread = new Thread() {
       public void run() {
          try {
            StringBuffer sb = new StringBuffer();
            /* read out the data and store in sb */
            TriMessage rcvMessage = new TriMessageImpl(sb.toString().getBytes());
            Cte.triEnqueueMsg(tsiPortId,new TriAddressImpl(new byte[] {  }), componen-
              tId, rcvMessage);
            } catch (Exception ex) {  /* do something */  }
                  httpConn.disconnect();
           } catch (Exception ex) {  /* do somthing */  }
    ...
```

- Enqueue the received data into the TE
- In this particular case this was just after the sending

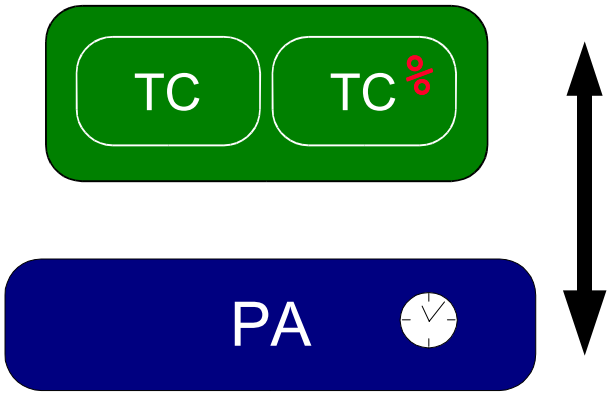
The TRI Platform Interface



- Interface structure
  - Implemenation of time and external functions
  - SA reports status back
  - TE indicates error

TriPlatform Interface

- Defines control of time and calling of external functions
- Complete set of operations (PA)
  - `TriStatusType triPAReset();`
  - `TriStatusType triStartTimer(...);`
  - `TriStatusType triStopTimer(...);`
  - `TriStatusType triReadTimer(...);`
  - `TriStatusType triTimerRunning(...);`
  - `TriStatusType triExternalFunction(...);`
- Complete set of operations (TE)
  - `void triTimeout(...);`



public TriStatus **triExternalFunction**(TriFunctionId functionId, TriParameterList parameterList,TriParameter returnValue)

```
TTCN-3:
     external function celsius2fahrenheit(float celsius) return float;

Java (MyExternalFunctionClass):
     public static FloatValue celsius2fahrenheit(FloatValue celsius)  {
          float tc = celsius.floatValue();
          float tf = ((float) ((9.0/5.0)*tc))+32;
          FloatValue fahrenheit = (FloatValue) RB.getTciCDRequired().
            getFloat().newInstance();
          fahrenheit.setFloat(tf);
          return fahrenheit;
     }
Java (TA):
     TestAdapter.addExternalFunctionImpl(MyExternalFunctionClass.class) ;
```

- Java operation signature follows TCI pendants of TTCN-3 external function declaration
- All operations have to be static, class implements **ExternalFunctionsDefinition**
- Registration of class implementing the external functions is necessary

<testing_tech>

Thank You !