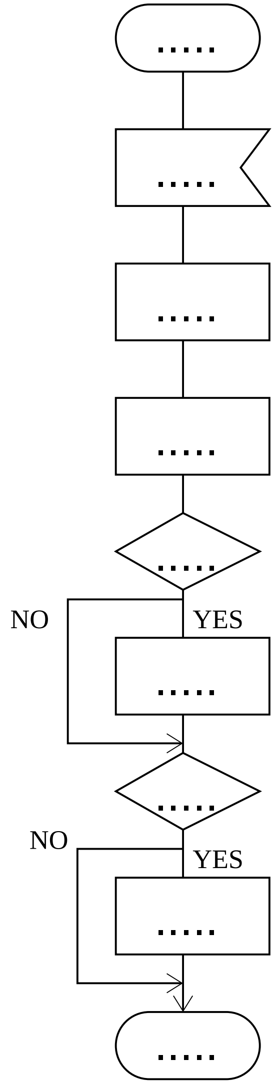
An Automated, Flexible Testing Environment for UMTS

Jan Bredereke, Holger Schlingloff

University of Bremen, TZi
Germany

Project Frame

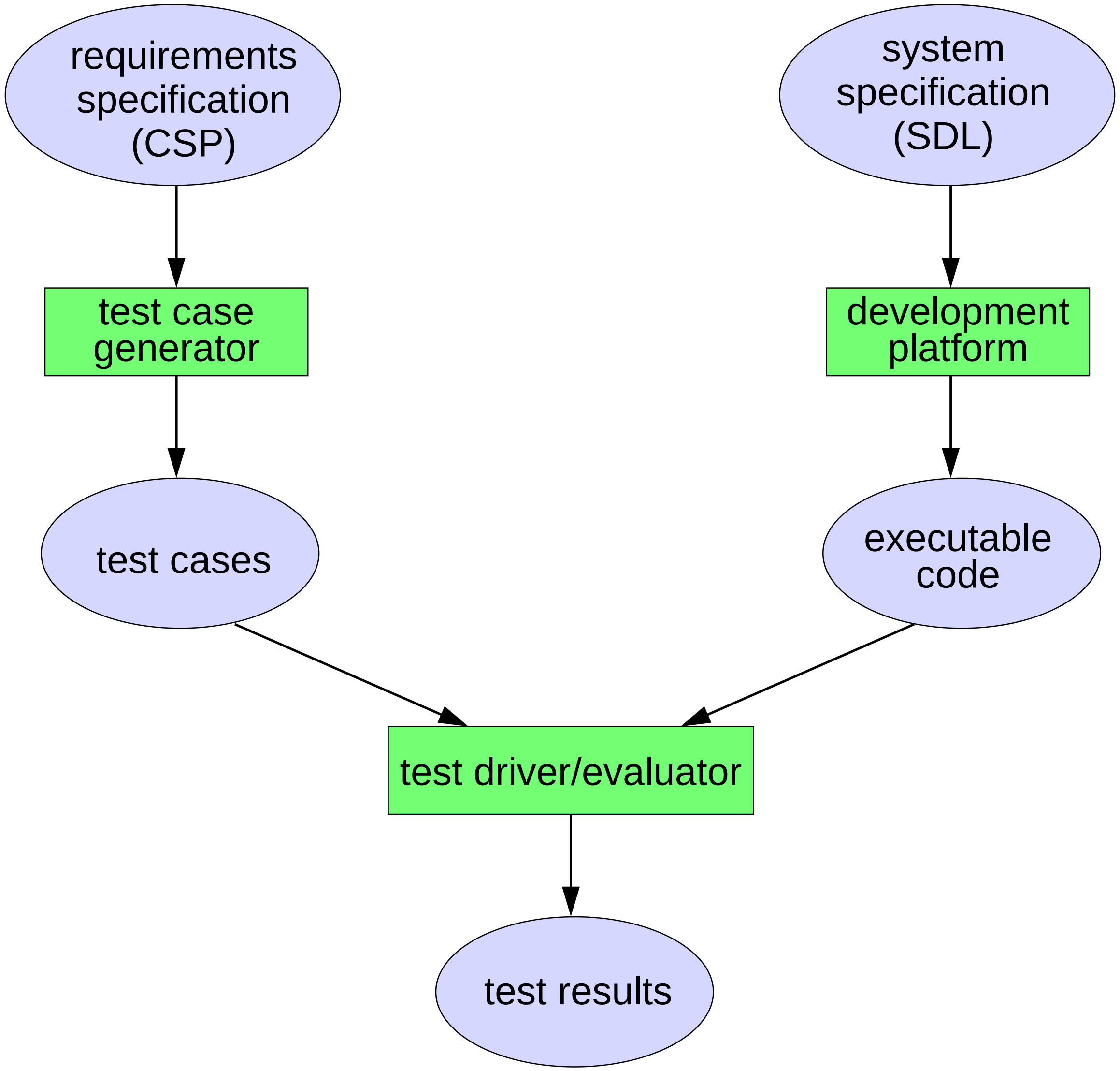
- **UMTS** protocol stack
- **Siemens AG**, Salzgitter:
  development in **SDL**
- **TZi**:
  black-box testing



Overview

- underlying **testing tool** RT-Tester
- **testing environment** for SDL
  - **automated interfacing** to SDL
  - **flexible maintenance** of test specification
- **application**: what did we find?

Specification Based Testing Approach



Testing Tool RT-Tester

- test specification in **timed CSP**
- **execution of** test **specifications**:
  1. automatic transformation into **graph of state transitions**
  2. on-the-fly, automatic, real-time
     generation / **execution** / **monitoring** / **evaluation**
     of test sequences
- tests of **arbitrary length/duration**
- both **functional** and **hard real-time** properties
- various visualization possibilities

Problem & Approach

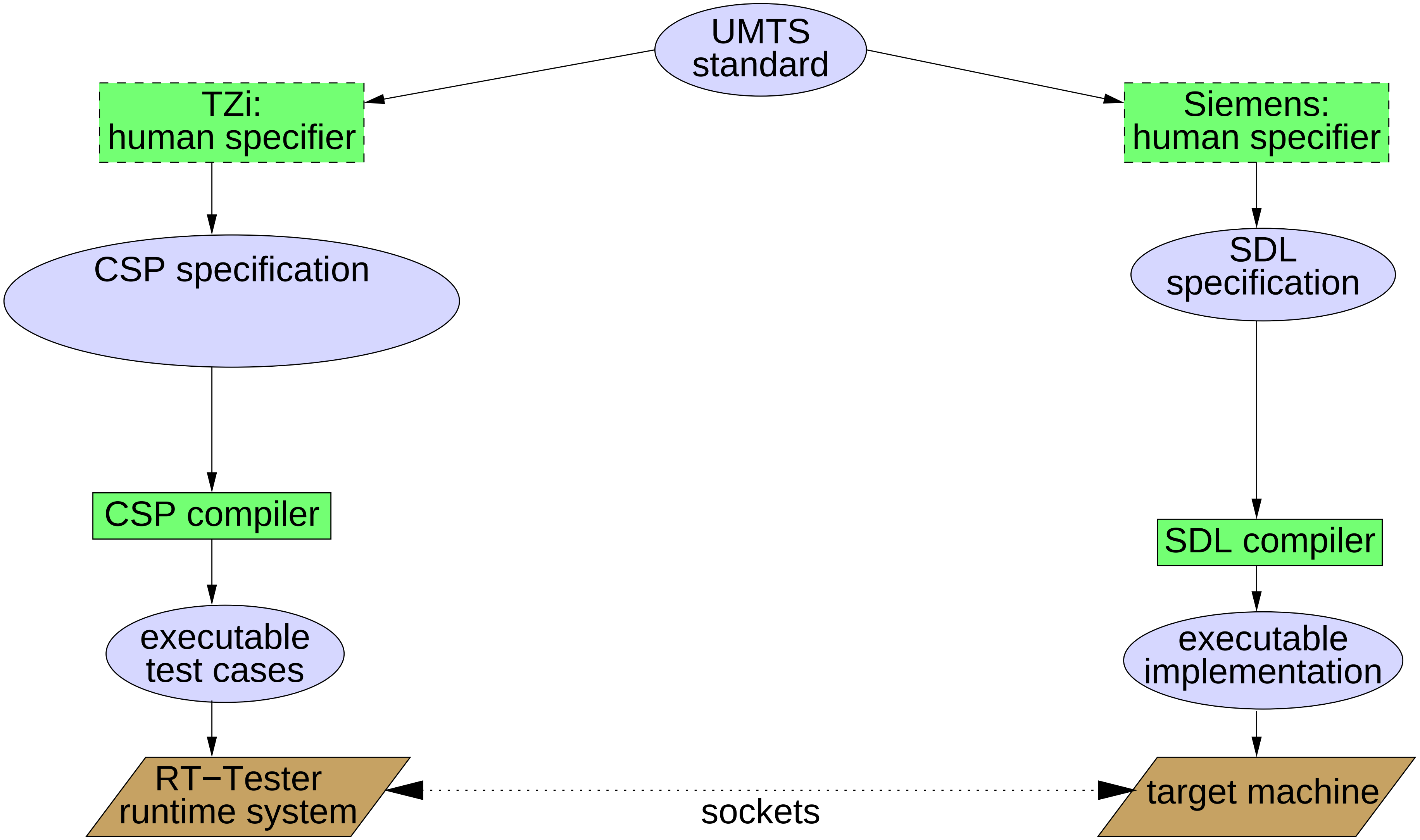
- **complex interface**, still **evolving**

$\Longrightarrow$

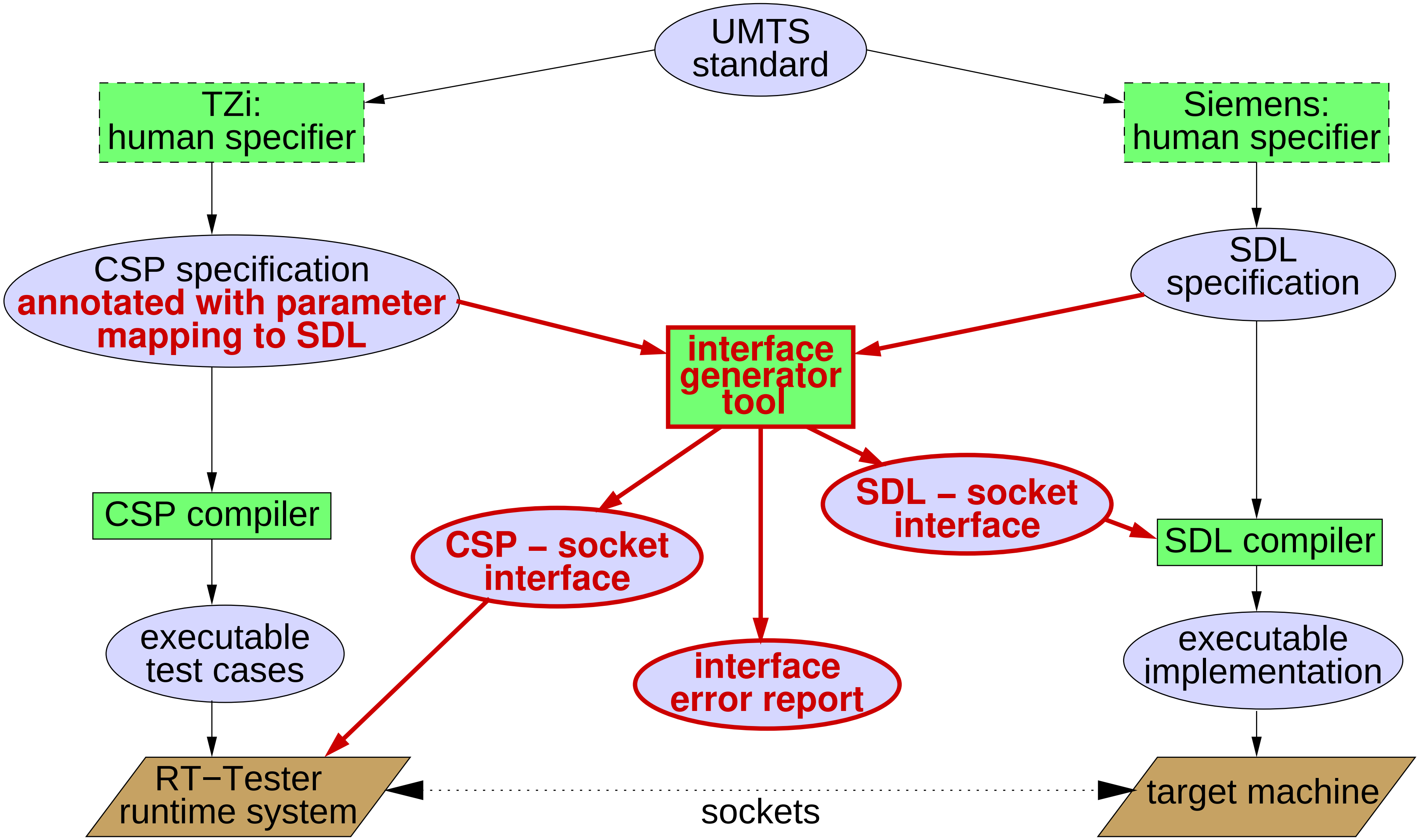
- **automation** of testing environment:
  - interface **in terms of SDL**, no low-level descriptions
  - automated **consistency check** of interface
  - interface **code generation**

- **flexible** test specification:
  - rules for **modularizing** test specifications

Manual Interfacing to SDL



Automated Interfacing to SDL



Example: Matching CSP and SDL Parameters

CSP

```
nametype Rb_identity = {0 .. maxRb_count}
datatype Rlc_data = dummy_value
channel rlc_tr_data_req :

    Rb_identity .

    Rlc_data
```

SDL

```
syntype RB_Identity = integer
    constants 0:MaxRb_Count
endsyntype;
synonym RLC_MAX_SDU_SIZE integer
     = 512;
newtype RLC_SDU_A
    carray(RLC_MAX_SDU_SIZE, octet)
endnewtype;
newtype RLC_SDU struct
    RB_Id          RB_Identity;
    RLC_SDU_Data   RLC_SDU_A;
    length         integer;
endnewtype;
signal RLC_TR_DATA_Req(
                        RLC_SDU);
```

Example: Matching CSP and SDL Parameters

CSP

```
nametype Rb_identity = {0 .. maxRb_count}
datatype Rlc_data = dummy_value
channel rlc_tr_data_req :
pragma SDL_MATCH \
        PARAM 1!RB_Id
    Rb_identity .
pragma SDL_MATCH \
        TRANSLATE dummy_value 0x99 * 16
pragma SDL_MATCH \
        PARAM 1!RLC_SDU_Data SUBSET_USED
    Rlc_data
pragma SDL_MATCH \
        SKIP 1!length DEFAULT_VALUE 16
```

SDL

```
syntype RB_Identity = integer
     constants 0:MaxRb_Count
endsyntype;
synonym RLC_MAX_SDU_SIZE integer
      = 512;
newtype RLC_SDU_A
     carray(RLC_MAX_SDU_SIZE, octet)
endnewtype;
newtype RLC_SDU struct
     RB_Id          RB_Identity;
     RLC_SDU_Data   RLC_SDU_A;
     length         integer;
endnewtype;
signal RLC_TR_DATA_Req(
                         RLC_SDU);
```

Overview

- underlying testing tool RT-Tester
- testing environment for SDL
  - automated interfacing to SDL
  - **flexible maintenance of test specification**
- application: what did we find?

Flexible Maintenance of Test Specification

- **late changes** to requirements
- **variants** of test suites:
  1. **adjust** test **coverage**
     - selected signal parameters
     - stimuli: random $\rightarrow$ increased probabilities $\rightarrow$ deterministic
  2. **component / integration** tests
     - different protocol layers
     - parallel instances of same layer
  3. **active / passive** tests

  $\Rightarrow$ a **family** of test suites

Rules for Modularizing Requirements

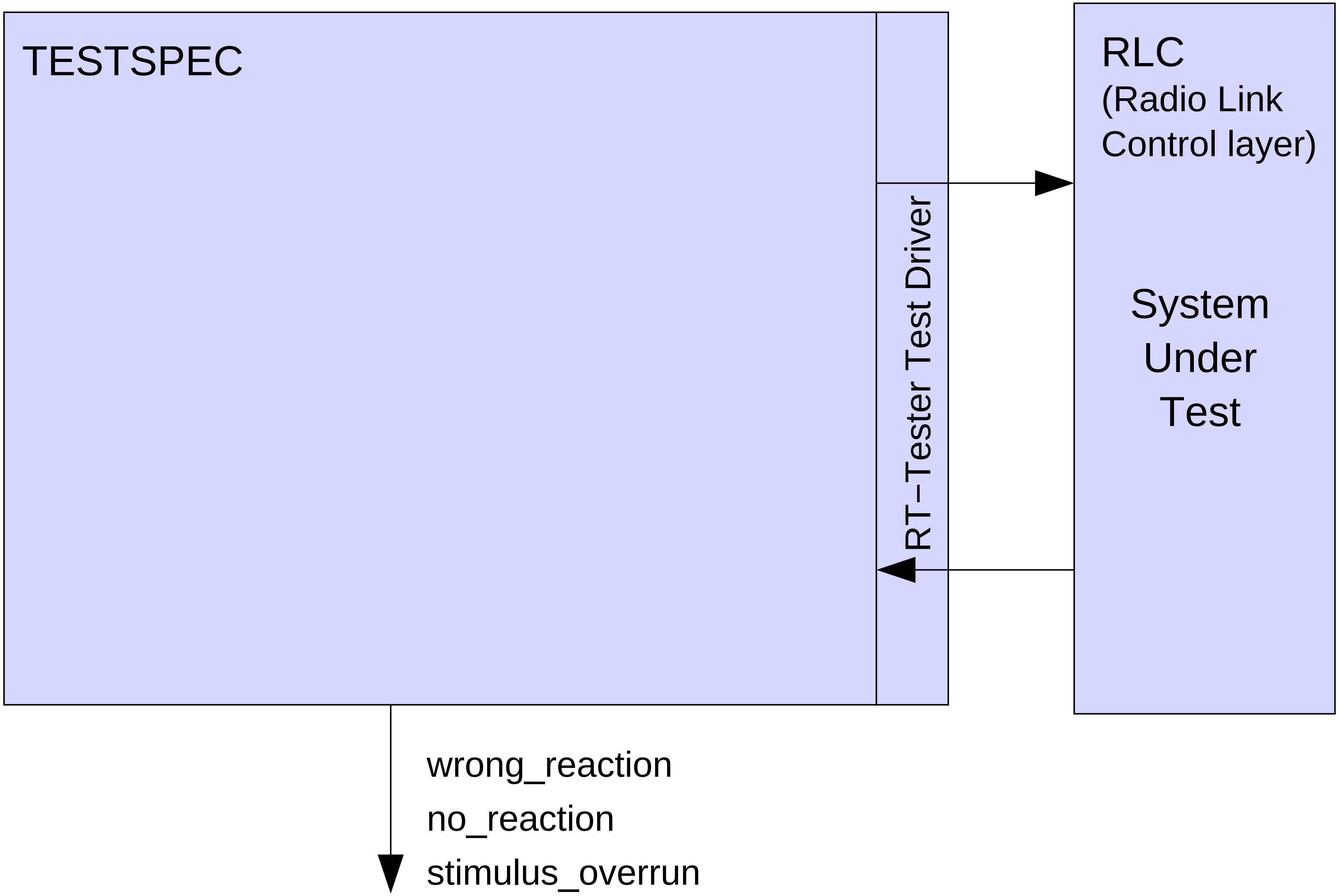
- **separate**: **signature / behaviour** of module

- **identify** req.s that will **change together**, put into **one module**

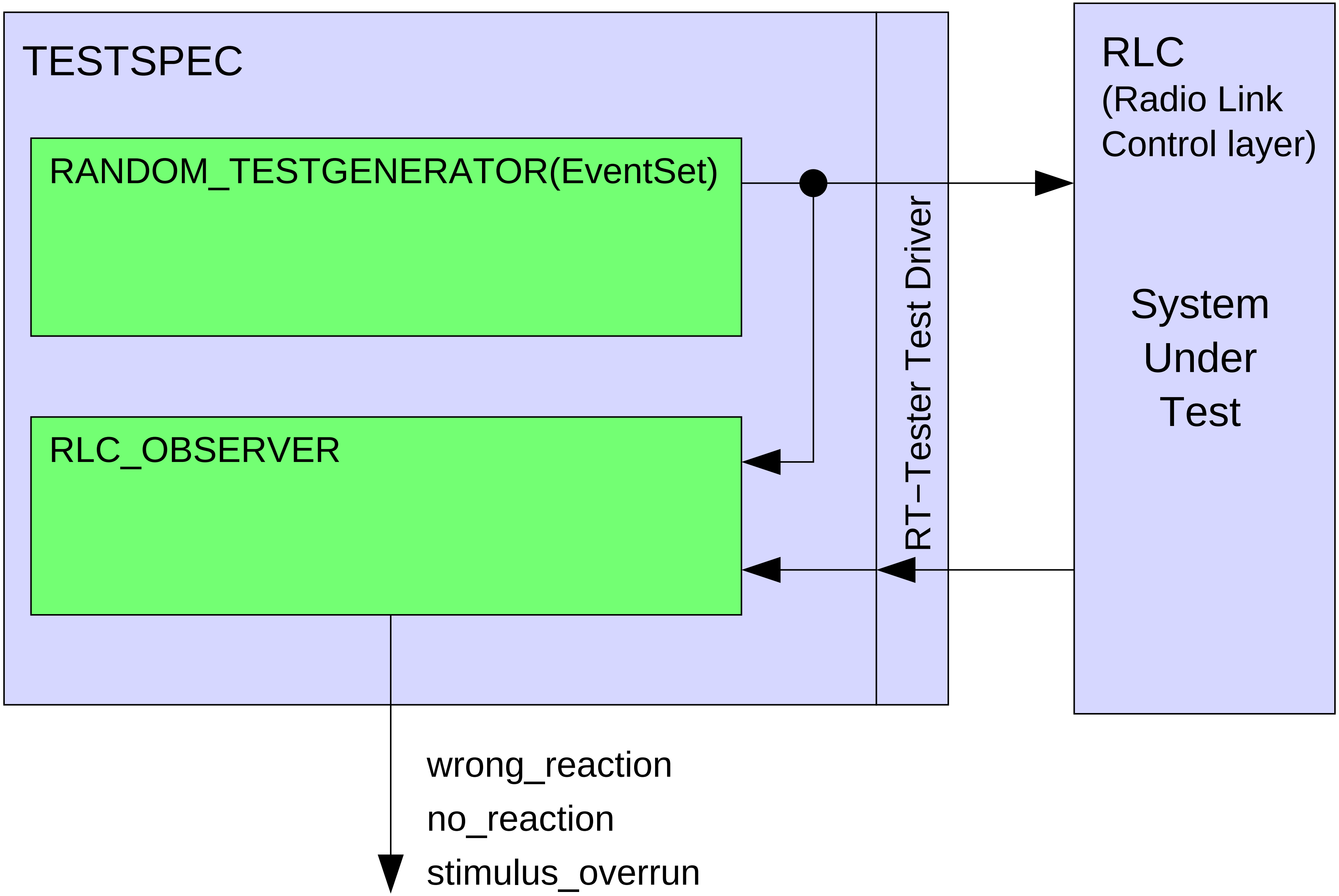
  specifically, **separate:**

  - **tester specific issues / application**
  - **timer handling / application**
  - **protocol layers**
  - **stimulus generation / test observation**

Separate: Test Stimulus Generation / Test Observation



Separate: Test Stimulus Generation / Test Observation

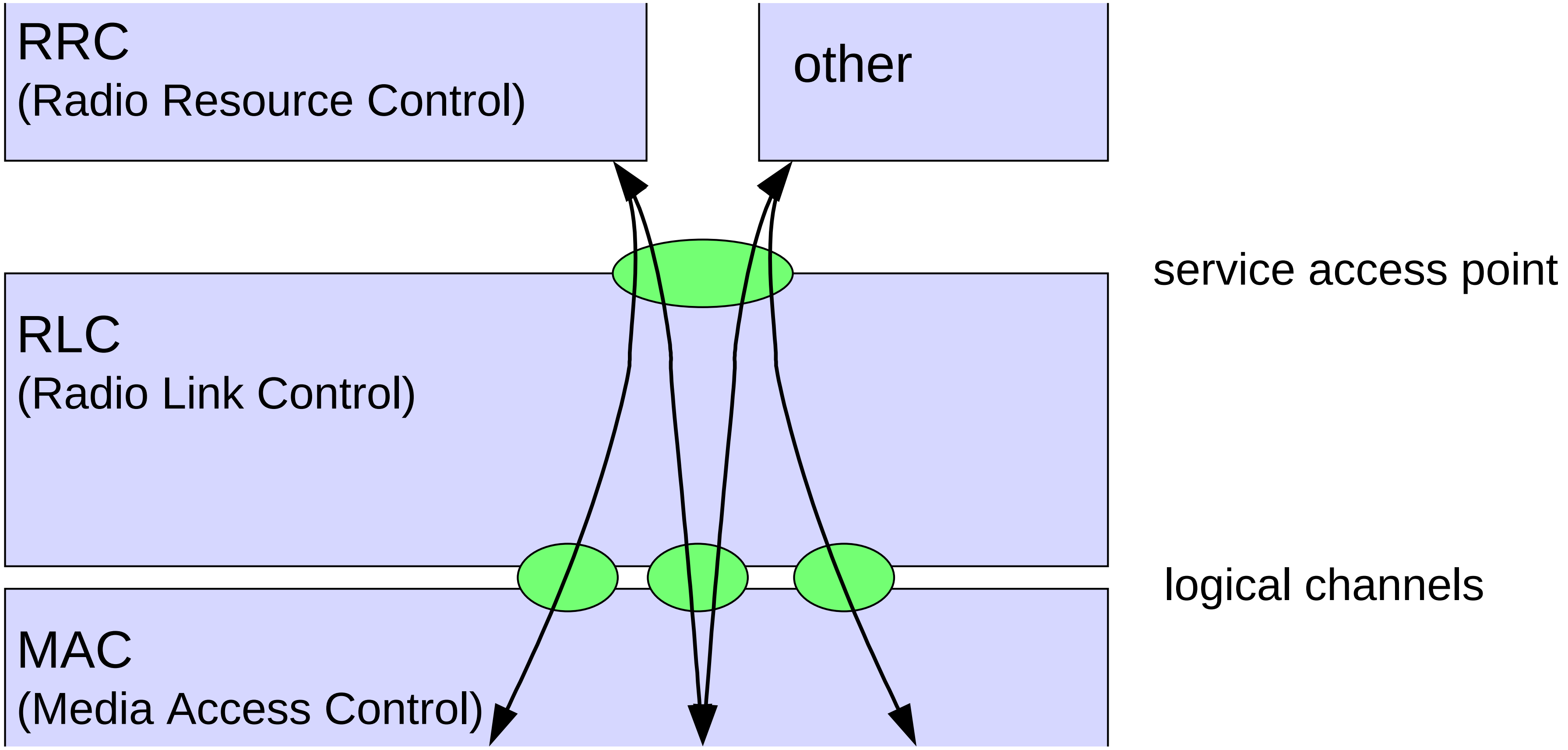


Overview

- underlying testing tool RT-Tester
- testing environment for SDL
  - automated interfacing to SDL
  - flexible maintenance of test specification
- **application: what did we find?**
  - ambiguities in the standards
  - unexpected behaviour of SUT

Found: Ambiguities in the Standards

Interface of RLC Layer

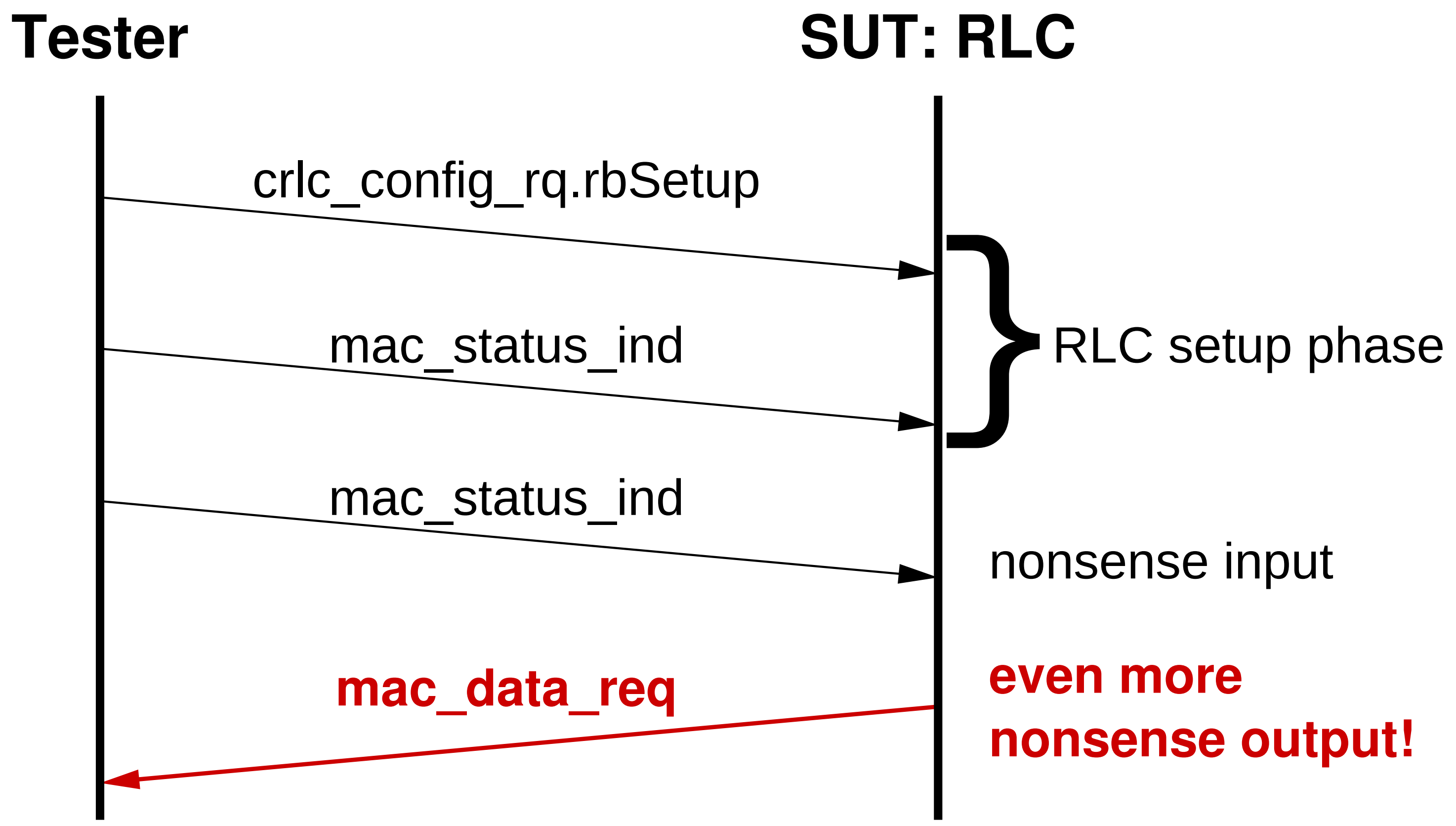


Found: Ambiguities in the Standards (2)

Properties of Service Access Points and Logical Channels

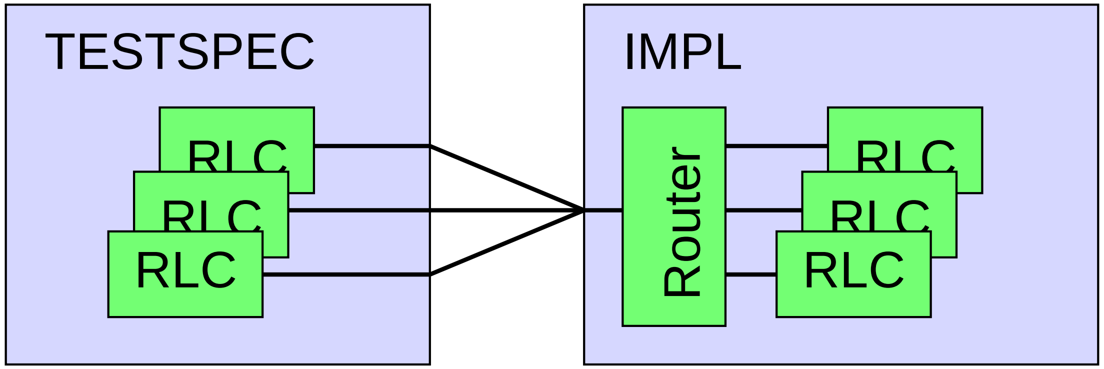
- **forward instantaneously / buffer?**
- **queueing discipline?**
- **single/multiple queues?**
- **min./max. delay** between delivery and availability?
- handling of **unspecified reception?**

Found: Unexpected Behaviour of SUT



Found: Unexpected Behaviour of SUT (2)

multiple RLC protocol machines in parallel



Summary

- **automatic generation** of **interface**
- **design rules** for **flexible, modular** family of test suites

profitable when:

- complex **requirements from 3rd party**
- **interface is** a **draft** most of the time
- **after completion** of requirements,
  short **time-to-market** is crucial