10th UCAAT User Conference on Advanced Automated Testing

ETSI The Standards People

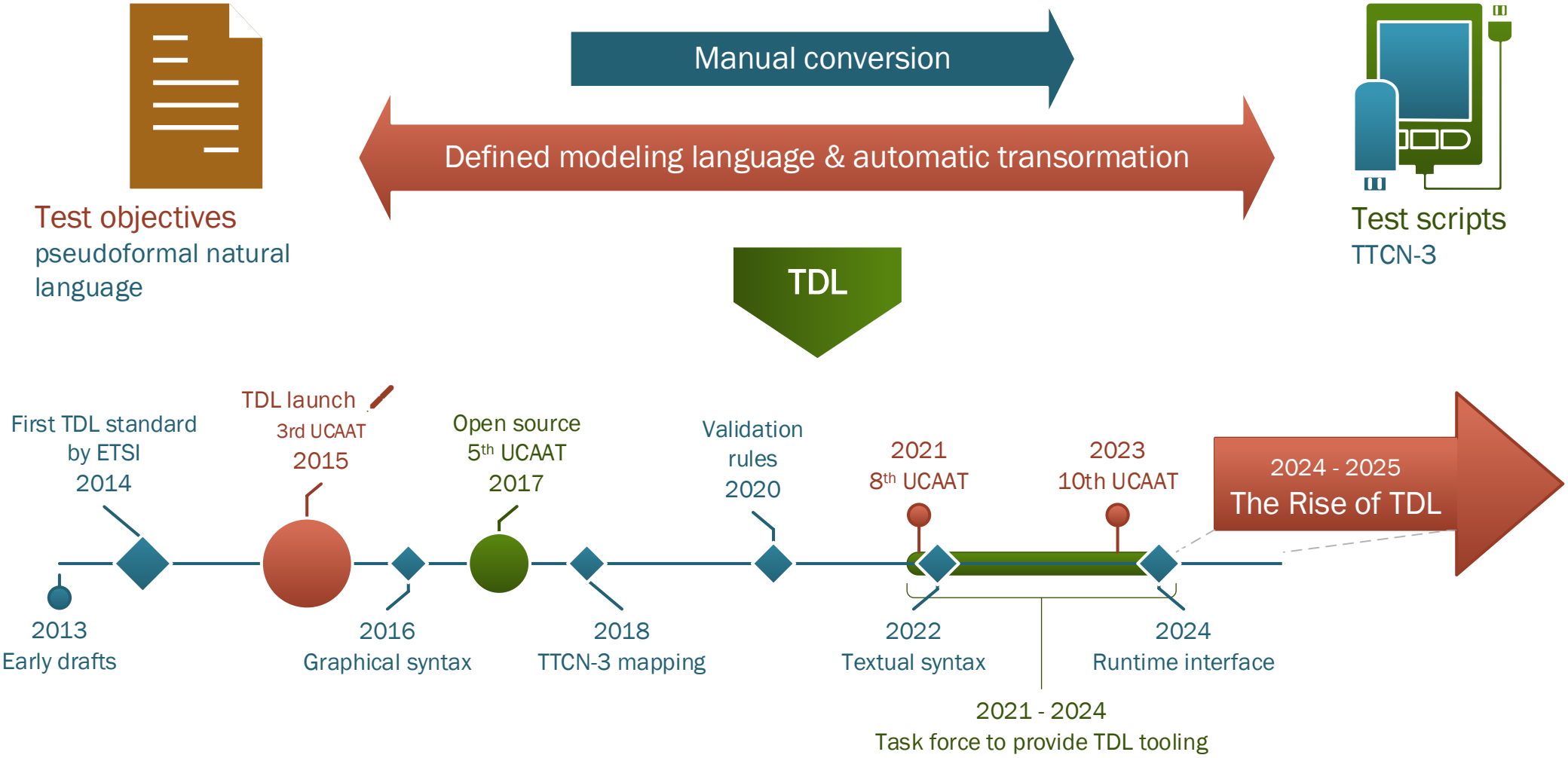
TDL: The Force Awakens

Martti Käärik / Elvior / ETSI TTF T034

30+ elvior

14/11/2023

Standardized test specifications in ETSI

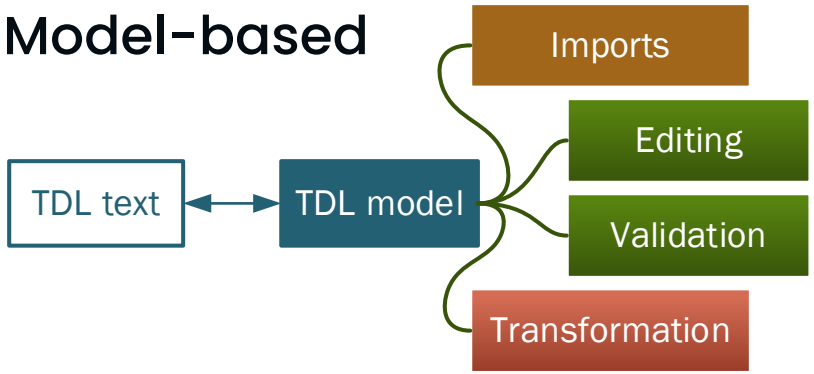


TDL standard

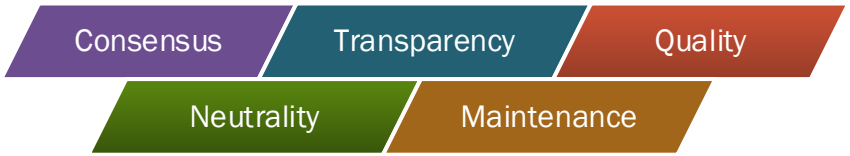
Textual syntax

```
Package DNSTester {
    Import all from TDL

    Enumerated MessageKind { MessageKind e_Question, MessageKind e_Answer }
    Structure DNSMessage (
        Identification identification,
        MessageKind messageKind,
        Question question,
        optional Answer answer
    )

    DNSMessage a_DNSQuestion (
        messageKind = e_Question
    )

    Test Description ExampleResolveNokia_tdl uses DNSClientServer {
        start client::replyTimer for 20 {Second}
        client sends a_DNSQuestion (
            identification = 12345,
            question = "www.research.nokia.com"
        ) to server
        client receives a_DNSAnswer (
            identification = 12345,
            answer="172.21.56.98"
        ) from server
    }
```

Model-based

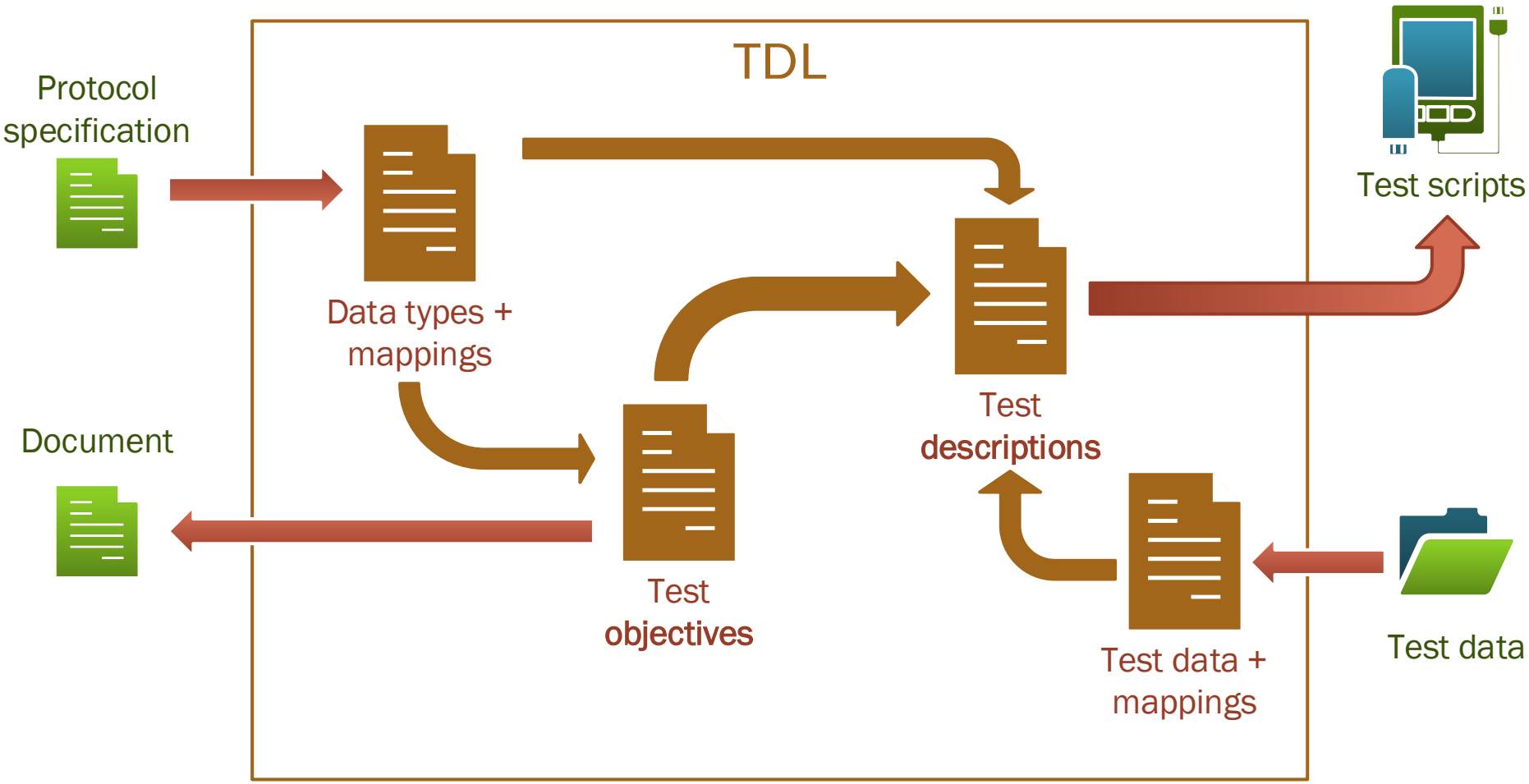


Standardized



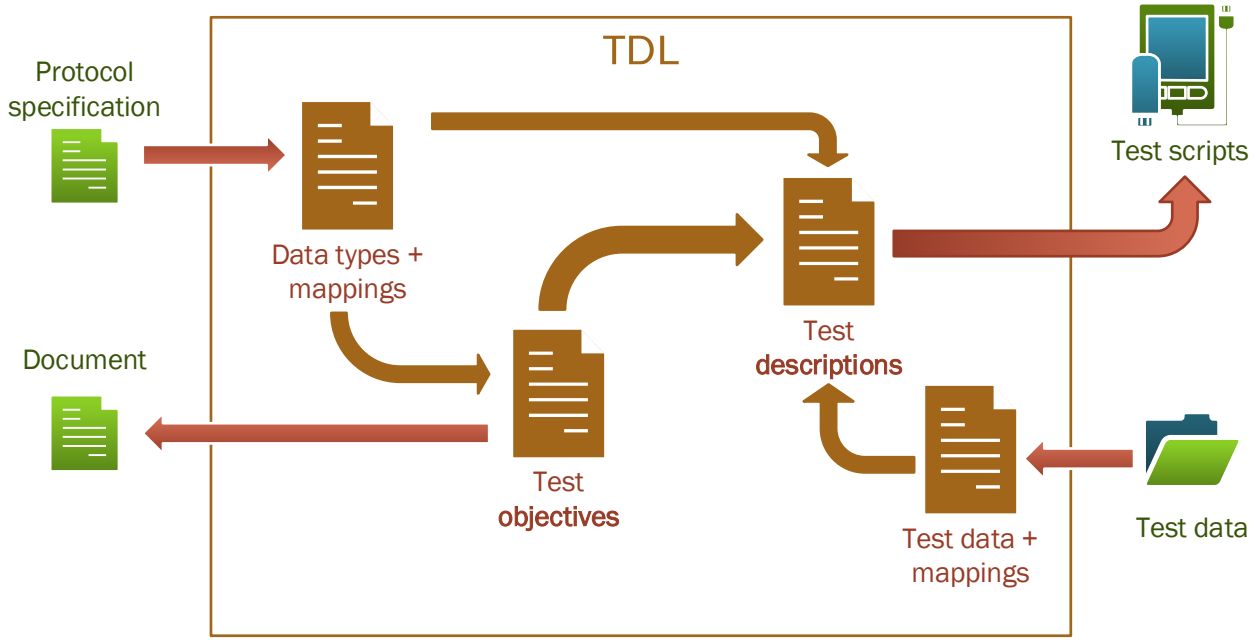
ETSI test development workflow

- Import OpenAPI
- Specify test objectives (TO)
- Export TOs for publication
- Refine TOs to test descriptions with test data
- Export executable tests for publication



TDL in test development process

- Test objectives and test description share the same meta-model
- External specifications and data can be imported without losing references
- TDL libraries can provide standardized mappings for various protocols
- Exporting to documents and other formats straightforward with models



Import OpenAPI to TDL


```yaml
openapi: 3.0.1
info:
  title: CompanyAPI
paths:
  "/auth/login":
    post:
      operationId: login
      requestBody:
        content:
          application/json:
            schema:
              "$ref": "#/components/schemas/Credentials"
      responses:
        '200':
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/Token"
        '403':
          description: Invalid credentials or user disabled
components:
  schemas:
    Credentials:
      type: object
      properties:
        email:
          type: string
        password:
          type: string
    Token:
      type: object
```

OpenAPI

```
Package CompanyAPI {
    Import all from HTTP.MessageBased Import all from Tdl

    Structure Credentials extends Body (
        String email,
        String password
    )
    Request login (
        uri = "/auth/login",
        method = POST
    )
    Structure Token extends Body (
        String token,
        String expires
    )
    Package JavaMapping {
        Import all from CompanyAPI
        Import all from Java Import all from Tdl

        @JavaPackage @MappingName : "Java"
        Use "com.company.product.model" as JavaTypesPackage

        @JavaClass
        Map Credentials to "Credentials"
                    in JavaTypesPackage as Credentials_Mapping {
            @JavaGetter : "getEmail"
            @JavaSetter : "setEmail"
            email -> "email",
            @JavaGetter : "getPassword"
            @JavaSetter : "setPassword"
            password -> "password"
        }
```

Data types + mappings

TDL

Test objectives and test cases

```
Objective Check_invalid_credentials_R01_1 {
    Description:
        "Server responds with status 403 to invalid login."
    References: "Requirement R01.1"
}

Credentials incorrect_login ()
Test Purpose Description Check_invalid_Credentials_TP {
    Objective: Check_invalid_credentials_R01_1
    Configuration: BasicClientServer
    Expected behaviour
    ensure that { when {
        client sends login(
            body = incorrect_login
        ) to server
    } then {
        server sends Response (
            status = 403
        ) to client
    }}
}
```

Objective

Test data + mappings

```
@JavaClass
@MappingName : "Java"
Use "<package name>.Authentication" as AuthenticationData
Map incorrect_login to "incorrect_login" in AuthenticationData
        as incorrect_login_mapping

Objective: Check_invalid_credentials_R01_1
Test Check_invalid_Credentials uses BasicClientServer {
    client sends login(
        body = incorrect_login()
    ) to server
        server sends Response (
            status = 403,
            body = omit
        ) to client
}
```

Test case

Exported document

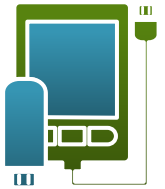
TP Id	Check_invalid_Credentials_TP
Test Objective	Server responds with status 403 to invalid login.
Reference	Requirement R01.1
Configuration	BasicClientServer
PICS Selection	N/A
Expected Behaviour	

```
ensure that {
         when
         {
              client sends login
                   body = incorrect_login
               to server
         }
         then
         {
              server sends Response
                   status = 403
               to client
         }
}
```

Document

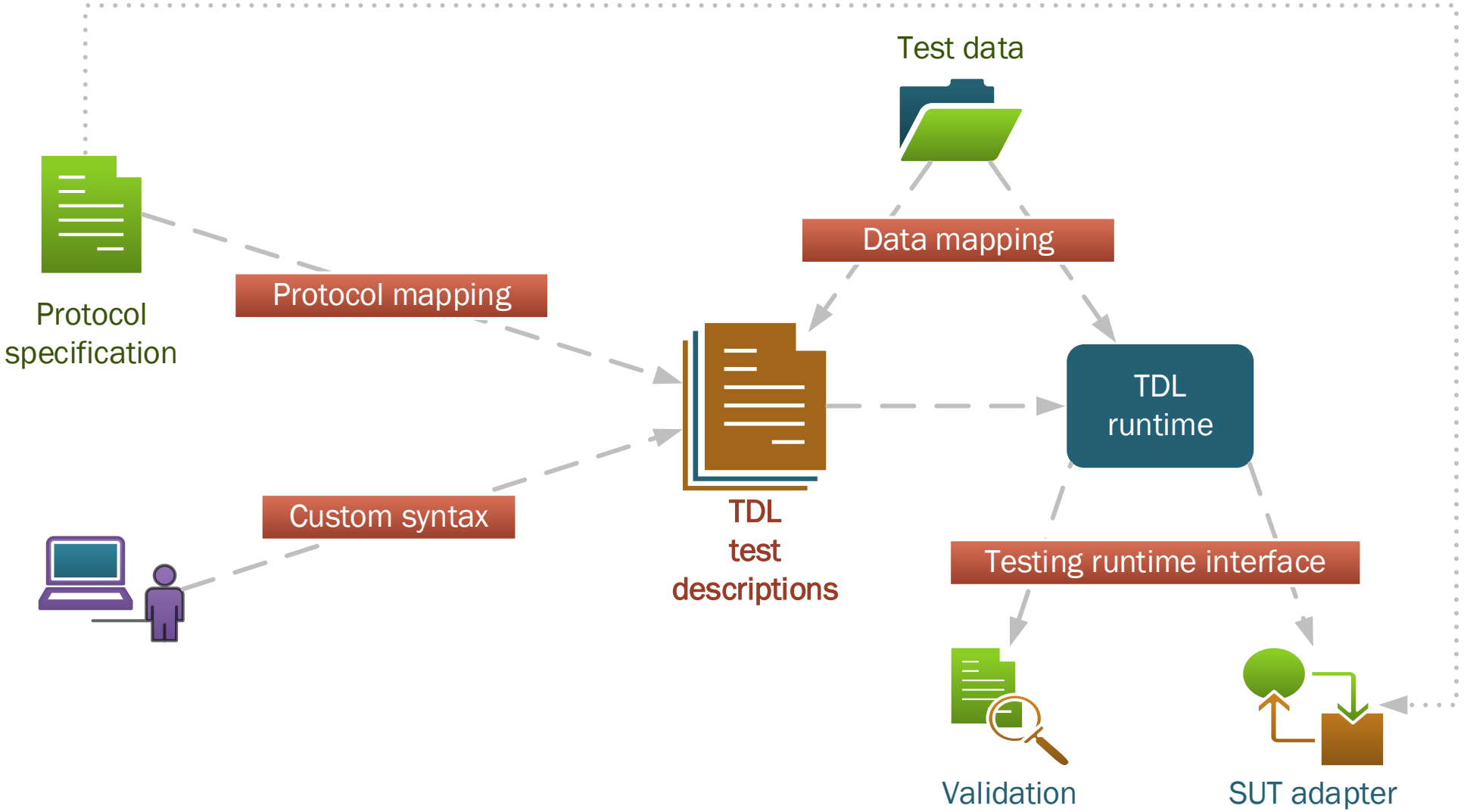
Exported test script


```
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class Check_invalid_Credentials
        extends org.etsi.mts.tdl.execution.java.examples.a.tdlgen2.exampleapi.configuration.Tester_Component {
    public void configure_Check_invalid_Credentials() {
    /** Server responds with status 403 to invalid login. */
    @Test
    public void test_Check_invalid_Credentials() {

        try {
            // CompoundBehaviour
            reporter.behaviourStarted("CompoundBehaviour", "CompoundBehaviour_4");
            try {
                HttpRequestData DataElementUse_4_datause_1;
                Credentials DataElementUse_5_datause_2;
                HttpResponseData DataElementUse_6_datause_3;
                Integer LiteralValueUse_2_datause_4;

                // Message
                reporter.behaviourStarted("Message", "Message_2");
                DataElementUse_5_datause_2 = org.etsi.mts.tdl.execution.java.examples.a.testdata.Authentication.incorrect_login;
                String LiteralValueUse_datause_5;
                LiteralValueUse_datause_5 = "/auth/login";
                HttpMethod DataElementUse_2_datause_6;
                DataElementUse_2_datause_6 = org.etsi.mts.tdl.execution.java.adapters.http.HttpMethod.POST;
                DataElementUse_4_datause_1 = new org.etsi.mts.tdl.execution.java.adapters.http.HttpRequestData();
                DataElementUse_4_datause_1.body = DataElementUse_5_datause_2;
                DataElementUse_4_datause_1.uri = LiteralValueUse_datause_5;
                DataElementUse_4_datause_1.method = DataElementUse_2_datause_6;
                systemAdapter.send(new PojoData(DataElementUse_4_datause_1),
                        getConnection("client", "http", "server", "http"));

                reporter.behaviourCompleted("Message_2");

                // Message
                reporter.behaviourStarted("Message", "Message_3");
                LiteralValueUse_2_datause_4 = 403;
                DataElementUse_6_datause_3 = new org.etsi.mts.tdl.execution.java.adapters.http.HttpResponseData();
                DataElementUse_6_datause_3.status = LiteralValueUse_2_datause_4;
                DataElementUse_6_datause_3.body = null;
                Future<ExecutionResult> receive_Message_3 = receive(new PojoData(DataElementUse_6_datause_3),
                        getConnection("client", "http", "server", "http")).execute();
                List<Future<ExecutionResult>> receive_Message_3_exceptionals = executeExceptionals();
```



Test scripts

Tailoring the test environment



Download TDL tools

TDL open-source project at Eclipse Marketplace

TDL IDE + meta-model implementation

- File management, validation, utilities

Textual editors

- Test descriptions, test objectives, types and mappings

Importers/exporters

- OpenAPI, Word

Code generators

- TTCN-3, Java

Execution engine

- Java + Junit + HTTP/JSON adapter

Developers – top.etsi.org

Any further questions?

Meet us at the ETSI booth!

tdl.etsi.org