21

Implementation of TTCN Operational Semantics in Estelle

[†]Luoming Hou, [‡]Jean-Philippe Favreau, and [‡]Debra Tang

[†]Department of Electronic Engineering and Computer Science
George Washington University, Washington, DC, USA

[‡]System and Network Architecture Division, Computer Systems Laboratory
National Institute of Standards and Technology, Gaithersburg, MD, USA

With an upgraded version of the Tree and Tabular Combined Notation (TTCN) that encompasses concurrency, it is now possible to foresee a new generation of Abstract Test Suites (ATSs) and Means Of Testing (MOT) for communication protocols. Indeed, the inclusion of concurrency/parallelism allows the complete description of ATSs in TTCN, including the upper tester behavior. This paper introduces a new prototype technology that automates the production of MOT from an ATS completely specified in TTCN. This technology is based on an implementation of the operational semantics of TTCN in Estelle.

1. INTRODUCTION

This paper[1] presents some of the results from a broad initiative on conformance testing. The Computer Systems Laboratory (CSL) of the National Institute of Standards and Technology (NIST) is engaged in an international initiative in conformance testing. CSL partners include the Telecommunications Laboratories (TL), Taiwan, The George Washington University (GWU), United-States, and the Institut National des Télécommunications (INT, France. The objective of this initiative is to design and implement a generic test tool generator that takes as input an abstract test suite completely specified in the Tree and Tabular Combined Notation (TTCN) and generates a test tool, then to apply this resulting test tool to the Transaction Processing protocol (OSI/TP) [1]. The project incorporates a number of fundamental blocks intended to secure the foundations for protocol testing, including the following components:

- Realization of the COS/POSI/SPAG (CPS) Technical Framework Specification [2,3];
- Realization of a TTCN to Estelle Test Engine Generator;
- Validation of the Estelle OSI/TP formal description; and
- Validation of a subset of the OSI/TP Abstract Test Suite (ATS).

[1]This work is a contribution of the National Institute of Standards and Technology and is not subject to copyright.

T. Mizuno et al. (eds.), *Protocol Test Systems*

This paper focuses on the TTCN to Estelle Test Engine Generator. This Test Engine Generator takes as input a TTCN ATS, performs syntax and semantics checking, and generates the kernel of a Means of Testing (MOT) for the relevant protocol including all Lower Tester and Upper Tester components. Our intent is to use the Test Engine Generator with a subset of OSI/TP ATS, and generate a test engine for an OSI/TP MOT. The resulting test engine will be driven by the implementation of the CPS and applied to OSI/TP products.

2. MOTIVATIONS

The previous version of the Tree and Tabular Combined Notation (TTCN) [4] did not include concurrent TTCN [5]. This limitation did prevent definition of Abstract Test Suites (ATS) for multi-party testing and the TTCN description of the behavior of upper tester(s). Concurrent TTCN resolved these problems but created a higher level of complexity in the implementation of the ATS. This has lead to our decision of implementing the operational semantics of TTCN in Estelle [6]. Indeed, Estelle supports most of the required attributes, in particular: Estelle is a member of Formal Description Technique (FDT) family which has a formal semantics and Estelle supports all the distributed and parallel/concurrent properties that TTCN Extension requires. Moreover, NIST has built a set of tools, the Portable Estelle Translator (PET) [7] and the Distributed ImplemeNtation GeneratOr (DINGO) [8], which supports the distributed implementation of Estelle formal descriptions.

Another motivation was related to the current problems in the area of ATS validation. Most of the times - if not all - ATSs are the results of manual generations. These types of generation result in ATSs that are error prone and difficult to maintain. A tool that automates the implementation of the ATS would accelerate the availability of Means of Testing (MOT) that could be run against reference implementation or simulated protocol formal description. Therefore, such a tool would allow a better validation of the ATS and result in a simpler ATS and MOT maintenance process.

3. TECHNICAL APPROACH

Rather than to develop a specific test engine for a specific ATS and for a specific protocol, the goal is to develop a generic test engine for any test suite described in IS TTCN (including the complete specification of the upper tester behavior). The inputs to the tool are the TTCN.MP file, the Protocol Information Conformance Statement (PICS) and Protocol Implementation eXtra Information (PIXIT). The tool outputs Estelle, C++ and ASN.1. When these pieces of code are compiled and linked, the resulting Executable Test Suite (ETS) is to be driven by a test engine driver (i.e., implementation of the CPS Technical Specification).

The tool development has been structured into the following stages:

- **TTCN parser**: The input of the parser is the TTCN.MP containing the ATS. The ATS is validated against TTCN syntax and semantics rules.
- **TTCN Operational Semantics**. The TTCN operational semantics is defined using a combined model of Estelle and ASN.1. In this approach, TTCN Abstract Service Primitive and Protocol Data Unit (ASP/PDU) are mapped into ASN.1; and TTCN Test Configurations are mapped into Estelle Constructs. A tree interpreter written in Estelle captures the dynamic semantics of TTCN behavior tree.

- **Code generation**: This is a stage that implements the mapping (transformation) between TTCN and Estelle/ASN.1 and generates C/C++ code. The C code for ASN.1 ASP/PDU is produced by PEPY [9] and C++ code for TTCN test configurations and trees is produced by a translator and PET/DINGO.

In parallel with this work, another team has been implementing the basic functionality of the test engine driver. The test engine driver is based on the Technical Framework Specification [2,3]. The Technical Framework Specification results from an initiative of the Corporation for Open Systems (COS), the Standard Promotion and Application Group (SPAG) and the Promoting Conference for OSI (POSI), Japan, and seeks to establish common functions, process architectures, and criteria which may be used throughout the world for the development, procurement and use of conformance test systems. The objective of CPS is to define an architecture for conformance test tools which is consistent with the OSI testing methodology, to rationalize operating environments, and to define interchangeable and reusable components of test systems. This initiative recognizes that test systems implement some of the functionality required by the OSI framework for conformance testing. A Process Model is introduced which defines a model for activities that take place during conformance testing. To each activity corresponds functions that could be made available in test systems in order to support the test operator in carrying out these activities. Functions include PICS and PIXIT administration, MOT configuration, static conformance review, test campaign management, test case selection, Protocol Conformance Test Report (PCTR) management, and System Conformance Test Report (SCTR) management.

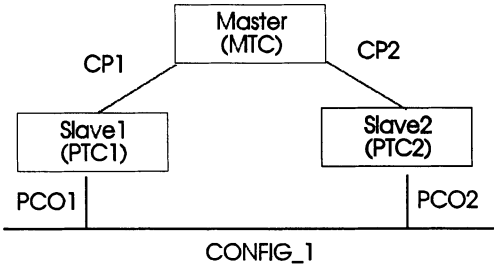

Figure 1. Tree-Like Test Configuration in TTCN

4. TTCN OPERATIONAL SEMANTICS IN ESTELLE

4.1. Parallel/Concurrent Operational Semantics

In the IS TTCN with Extensions [4,5], the test configuration notation is used to describe the parallel/concurrent testing behaviors. In this notation, an ATS contains one or more test configurations. Each test configuration is a tree-like structure consisting of one Main Test Component (MTC) and zero or more Parallel Test Components (PTCs). Cooperation Points (CPs) are employed to connect MTC and PTCs for exchanging the Cooperation Messages (CMs); Points of Control and Observation (PCOs) are employed to connect the MTC/PTCs with the test environment. The body of the MTC/PTC is the normal TTCN behavior tree where an MTC acts as a test case and each PTC acts as a test step. Every test

case adheres to one particular test configuration. Different test steps may run in parallel. ASPs/PDUs may be distributed over different PCOs.

Example 1. Tree-Like Test Configuration. In Figure 1, a test configuration (CONFIG_1) is defined as three test components: Master, Slave1 and Slave2. CONFIG_1 connects the Master (MTC) with Slave1 (PTC1) and Slave2 (PTC2) by two cooperation points CP1 and CP2. Two points of control and observation PCO1 and PCO2 are used in turn to connect Slave1 and Slave2 to the testing environment. Master will initiate Slave1 and Slave2 using the CREATE statements. Then, Slave1 and Slave2 will run in parallel and report the results to Master through CP1 and CP2.

4.2. Implementing the Parallel/Concurrent Semantics in Estelle

The tree-like test configuration in parallel/concurrent TTCN can be fully implemented by Estelle using the following strategy[2]:

- We define a generic Estelle specification $MOT_{Estelle}$ which has two Process modules -- $TE_{Estelle}$ (Test Engine) and $PCO_{Estelle}$. The $TE_{Estelle}$ module has its own sub-module $TC_{Estelle}$ (Test Component). $TC_{Estelle}$ has two different module bodies -- $MTC_{Estelle}$ and $PTC_{Estelle}$. This specification $MOT_{Estelle}$ is independent from the abstract test suite being processed.
- The MTC and PTCs in TTCN are mapped into $TC_{Estelle}$ instances. The CPs connecting MTC and PTCs in TTCN are mapped into channel instances connecting $TC_{Estelle}$ instances. The CMs exchanged through CPs between MTC and PTC or between PTCs are mapped into interactions going through channels in Estelle. In this approach, test management is realized via the actual implementation of Estelle semantics of the parallel $TC_{Estelle}$ instances.
- The PCOs in TTCN are mapped into the $PCO_{Estelle}$ module instances, each of which is connected to an instance of $TC_{Estelle}$ corresponding to MTC or PTC in TTCN. The $PCO_{Estelle}$ module body employs primitives to interface with PEPY and ISODE tools so that the ASN.1 ASP/PDU from the Test Engine (or the IUT) can be encoded (or decoded) and passed down (or delivered to) the IUT (or Test Engine). The advantage of this approach results from the properties of DINGO which allows the Estelle module instances to be distributed over different machines. Therefore, the multi-party and distributed TTCN operational semantics is implemented by distributing $PCO_{Estelle}$ module instances over different machines.
- We translate all the ASPs/PDUs/CMs types/constraints in TTCN into PEPY oriented ASN.1 types/values. Later, PEPY generates the encoding/decoding functions automatically and the encoded/decoded ASN.1 values are passed/obtained to/from ISODE tool. PEPY provides for the automated implementation of ASN.1 encoding/decoding rules.
- We write a TTCN tree interpreter which is the core part of the $TC_{Estelle}$ module body. Together with the C++ implementation of the Estelle primitives, this tree interpreter handles all the TTCN expressions, timer operations, assignments as well as constraint references.

---

[2]In this Section acronyms/names associated with Estelle constructs are tagged with a subscript "Estelle" (e.g., $SPEC_{Estelle}$).

Example 2. Mapping TTCN Test Configuration Into Estelle. In this example, the test configuration CONFIG_1 given in Example 1 is mapped into one instance of Estelle specification $MOT_{Estelle}$.

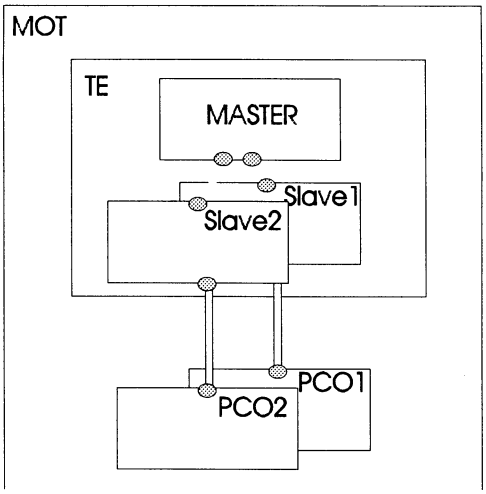

Figure 2. Mapping TTCN Test Configuration into Estelle

A full implementation of TTCN operational semantics requires mapping all the TTCN sources into Estelle and ASN.1. The core parts of the mapping mechanisms are summarized in Table 1.

TTCN Source	**ASN.1/Estelle Objects**
ASP Types	ASN.1 Type definitions
PDU Types	ASN.1 Type definitions
ASP Constraints	ASN.1 Value Definitions
PDU Constraints	ASN.1 Value Definitions
Test Configurations	Estelle Module Instances
Test Components	Estelle Module Instances
Cooperation Points	Estelle Channel Instances
Cooperation Messages	Estelle Interactions
Behavior Tree Logic	Number Tables
TTCN Statement	Numbers input to Tree Interpreter

Table 1. Mapping Mechanisms

5. AUTOMATED GENERATION OF ETS FROM ATS

Based on the TTCN operational semantics defined in Estelle and ASN.1, we are developing a tool set that automatically derives the ETS from the ATS. This tool set consists of three major sub-tools: (1) a TTCN parser, (2) an ASN.1 translator and (3) an object code generator. The general flow is presented in Figure 3. The text below describes briefly the different steps in terms of input used and resulting output.

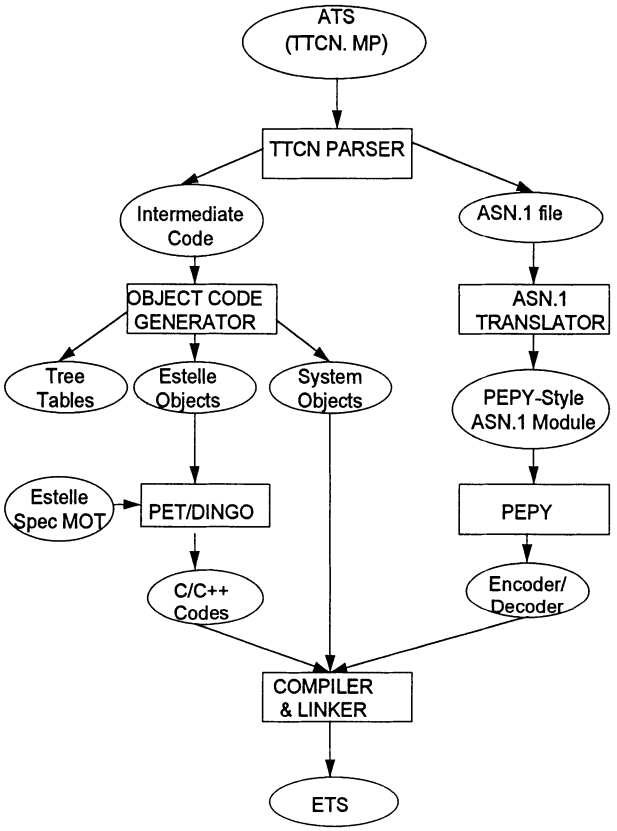

Figure 3. ATS to ETS - Illustrated Process

TTCN Parser:

Input: The input for the TTCN Parser is the ATS (TTCN.MP). This ATS makes use of concurrent TTCN to describe the behavior of the Upper Tester (i.e., no pseudo code or comments describing the Upper Tester).

Output: The TTCN Parser produces two types of intermediate codes: Estelle oriented intermediate code and ASN.1 oriented intermediate code. The Estelle oriented intermediate code contains all source code information of the TTCN.MP that passed syntax/semantics

checking (except for ASN.1 definitions) and can be easily translated into different object codes. The ASN.1 oriented intermediate code contains all the source ASN.1 type definitions, ASN.1 ASP/PDU/CM type definitions and their constraints definitions found in the TTCN specification.

ASN.1 Translator:
Input: The ASN.1 Translator uses the standard ASN.1 module definition combined with the restrictions/extensions from the TTCN standard;
Output: The tool produces an ASN.1 module definition in a format compatible with PEPY [9] (i.e., the PEPY-style ASN.1 module definition).

Object Code Generator:
Input: The Object Code Generator takes as input the intermediate code which defines all the global/local TTCN tabular objects classified and indexed and all the TTCN behavior trees numbered into tables.
Output: Three types of outputs result from the execution of the Object Code Generator.
1. The fully numbered tables for all TTCN trees, which will be dynamically loaded and run by the tree interpreter;
2. The Estelle module instances and variables;
3. The system dependent C/C++ code which supports the tree interpreter.

6. APPLICATION TO THE TRANSACTION PROCESSING ATS

Current plans include applying the tool to the Transaction Processing (OSI/TP) ATS. This section provides some background information about OSI/TP, the current status of the ATS, and the type of application planned.

The OSI Transaction Processing protocol (OSI/TP) [1] defines services and functions to support distributed tasks for which it is important to maintain the ACID properties: Atomicity, Consistency, Isolation, and Durability. The Atomicity and Consistency properties combined guarantee that transactions are atomic operations and that all sub-tasks involved in a transaction have the same outcome (success or failure). The Isolation property guarantees that separate transactions progress independently from each other. Finally, the Durability property guarantees that adequate logging facilities exist to maintain a trace of the outcome of transactions. The OSI/TP protocol is based on a two-phase commit protocol, with additional recovery mechanisms to handle communication failures and system crashes. Typical uses of the OSI/TP protocol include large distributed database systems.

ISO has started to work on the specification of a conformance test suite for the OSI/TP protocol. Currently, the activities in ISO focus on the specification of a test suite structure and test purposes. In Japan, INTAP has defined a very extensive list of test purposes and test bodies for the transaction processing protocol. The ultimate goal is to reach international agreements on a complete abstract test suite. Completion requires the validation of the INTAP ATS. This ATS is being reviewed via a joint effort between Europe, North-America and Japan.

The OSI/TP ATS is very complex and offers a good test for the evaluation of the Test Engine Generator. On the other hand, there are some technical problems to resolve. Though, the INTAP ATS is being updated to IS TTCN with Extension and the Test Management

Protocol Data Units (TMPDUs) are not specified using concurrent TTCN. Because of these situation, the plans are to select a subset of the ATS and bring it to the level required by the tool. When this step is completed, we plan to run the resulting OSI/TP test tool against an OSI/TP product.

REFERENCES

1. International Standard Organization. Information Technology - Open Systems Interconnection - Distributed Transaction Processing, IS 10026, May 1992.
2. The CPS Forum Technical Framework Specification for Conformance Testing SPAG S.A., 165, Avenue Louise Box 7, B-1050 Brussels, Belgium, 1991.
3. Advanced Testing Methods (ATM); Guide of the Implementation of an ISO 9646 Compliant Open Test Environment, European Telecommunications Standards Institute, Route de Lucioles, Sophia Antipolis, Valbonne, France, 1993.
4. International Standards Organization. Open Systems Interconnection - OSI Conformance Testing Methodology and Framework, Part 1-7 ISO/IEC JTC 1 International Standard 9646, December 1992.
5. International Standards Organization. Open Systems Interconnection - OSI Conformance Testing Methodology and Framework, Part 3: TTCN Extensions, ISO/IEC JTC 1 DAM-1, 1993.
6. International Standard Organization. Information Processing Systems, Open Systems Interconnection,. Estelle - A Formal Description Technique based on an Extended State Transition Model, IS-9074, 1989.
7. Sijelmassi, R.; Strausser B., The Distributed Implementation Generator: an Overview and User Guide, Report No. NCSL/SNA - 91/2, National Institute of Standards and Technology, Gaithersburg, MD 20899, U,S.A., January 1991.
8. Sijelmassi, R.; Strausser B., The Portable Estelle Translator: an Overview and User Guide, Report No. NCSL/SNA - 91/2, National Institute of Standards and Technology, Gaithersburg, MD 20899, U.S.A., January 1991.
9. Rose, T.M.; Onions, J.O., Robbins, C.J., The ISO Development Environment User's Manual, Volume 1-4. University of Pennsylvania, Department of Computer and Information Science, Moore School, Attn: David J. Farber, 200 South 33rd Street, Philadelphia, PA 19104-6314, U.S.A.