TTCN-3 和 TCL/TK 比较

一、TTCN-3 vs TCL

- TCL/TK
  – 一个简单的语言，最初设计并不是用于测试，但目前在测试应用很广泛。
- TTCN-3
  – 一个专门的标准测试语言，有专门的测试工具
  A specialized standard testing language with specialized testing tools

二、TTCN-3 vs TCL 概念比较

Concept	TCL/TK	TTCN-3
Test cases definition	no	yes
typing	no	yes
Invoke test cases	yes	yes
codec	Powerful regular expression feature	Yes (API and implementation languages)
Test results details	no	yes
Display test verdict	no	yes
Tracing of test events	no	yes

三、我们将从以下几个方面比较 TCL 和 TTCN-3

3.1、TCL 优点

- 快速的开发
- 跨平台应用
- 容易学习
- 扩展，嵌入和集成
- 免费

3.2、TTCN-3 优点

- 国际标准测试语言 Internationally standardized testing language
- 专门为测试和认证设计的（语言）
- 应用在各个不同应用领域和测试类型的一个测试技术
- 显著减少培训和维护的成本

3.3、TTCN-3 是如何不同的

- 丰富的类型系统，包括自有类型和对子类型的支持
- 体现了强大的内置匹配机制
- 语义快照，例如在访问过程中良好定义处理端口队列超时
- 判决的概念，以及判决的解决机制
- 支持并发测试行为的定义
- 支持计时器
- 允许在运行时测试配置
- 测试只关注在被测试的实现上
- 不依赖于特定的应用程序或者应用的接口
- 不依赖于特定的测试执行环境，编译器或者操作系统
- TTCN-3 代码不能被独立执行，需要一个编译器/解释器，适配器以及编解码实现

四、其它重要比较因素

- 匹配机制 Matching mechanism
- 结构化概念 Structuring concepts
- 工具结果检查功能 Tool results inspection features
- 复杂行为定义 Complex behaviors specification
- 类型 typing

4.1、匹配机制比较 Matching mechanism comparison

- **TCL 匹配机制 Matching mechanism TCL**
  - 常规的表达会变得过于复杂和难以管理，当这里有 Regular expressions can become overly complex and unmanageable when there are:
    – Alternate values (“href” | “HREF”)

– Alternate sequence patterns (see music categories list example. The list of categories could be in different order)

- **TTCN-3 匹配机制 Matching mechanism TTCN-3**
  - 和分配一个数据到一个数据结构一样容易
  - 不仅仅是分配一个数据，每个区的各种匹配规则可以被分别定义
  - TTCN-3 模板概念允许最大限度再利用能力，因此一个自然结构机制
    The TTCN-3 template concept allows maximum re-use capabilities, thus a natural structuring mechanism.

- **TTCN-3 匹配机制优势**
  - 复杂类型的透明匹配 Transparent matching of complex types
  - List 和 set 类型的透明匹配 Transparent matching of lists and sets
  - Alternative 类型匹配 Alternative values matching
  - 范围内匹配 Ranges matching

  - **复杂数据结构匹配**

```
type set of charstring myListType;
template myListType list_1 := {
        “john”, “mary”
}
template myListType list_2 := {
        “john”, “mary”
}
If(match(list_1, list_2)) {
   setverdict(pass)
}
else { setverdict(fail) }
```

4.2、结构化概念比较 Structuring concepts comparisons

- 结构化使代码可重用
- 结构化使代码清晰，因此使代码容易被修改和维护

结构化潜力

- 分解出可重复使用的代码
- 按功能分开代码

TCL 结构化概念

- 只有程序
- 没有测试例概念

TTCN-3 结构化概念

- 在抽象层内
  - 测试例
  - –功能
  - 模板
- Separation of concerns:
  - 在抽象层和测试适配层之间 Between abstract and adaptation layer
  - 在抽象层的行为和行为执行条件之间 Between behavior and conditions governing behavior in the abstract layer

测试例参数化

```
testcase webSystemTest(charstring theURL, WebResponseType theResponsePage)
                                runs on MTCType system SystemType {
timer responseTimer;
…
web_port.send(theURL);
responseTimer.start(10.0);
alt {
    [] web_port.receive(theResponsePage) {
       responseTimer.stop;
       setverdict(pass)
       }
    [] web_port.receive {
       responseTimer.stop;
       setverdict(fail)
       }
       [] responseTimer.timeout { setverdict(inconc)
       }
};
```

TTCN-3 模板概念的好处

TTCN-3 模板介于数据结构和功能之间，这允许:

- 静态再利用意味着巨大的结构化特质 Static re-use which implies great structuring qualities.
- 参数化允许模板的动态化再利用 Parametrization which allows dynamic re-use of

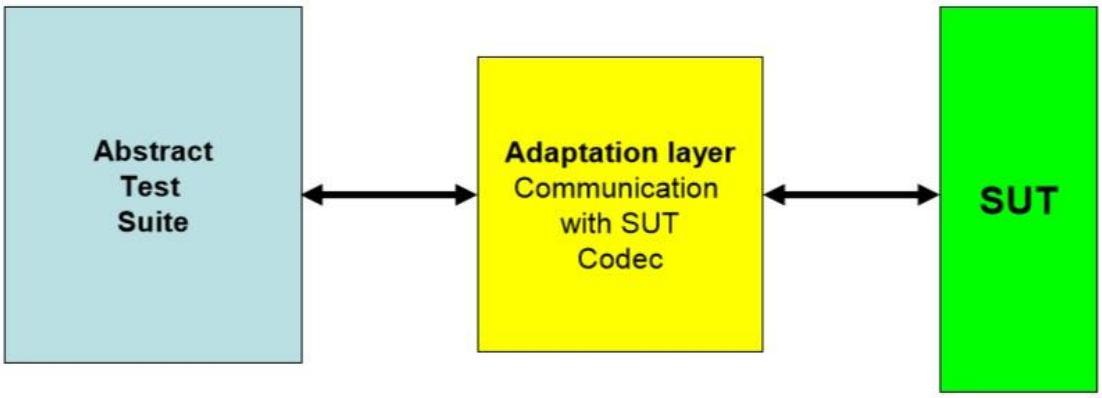
templates.

– 由于 TTCN-3 模板被强烈的类型化，这迫使测试者定义值或者规则 Because TTCN-3 templates are strongly typed, this forces the tester to specify a value or rule.

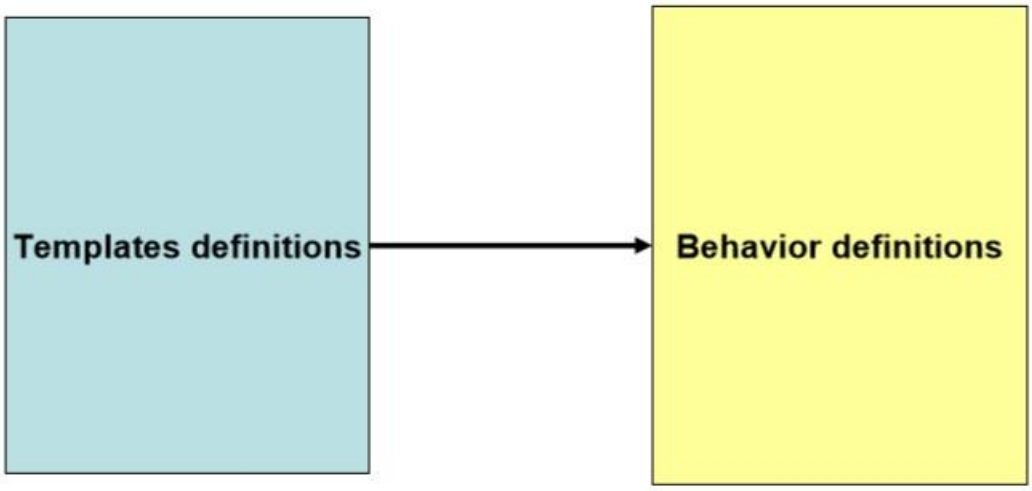
TTCN-3 separation of concern 的优势

- 在抽象层和实体层之间分开:
  - 改善测试行为的清晰度 Improve clarity of the test behavior
  - 再利用 Re-usability
  - 可重写 Re-writability
- 在行为和行为执行条件之间分开 Separation between behavior and conditions governing behavior
- 改善测试行为的清晰度 Improve clarity of the test behavior
- Provide overview qualities

在抽象层和实体层之间分开 Separation of concerns between abstract and concrete layers



在行为和行为执行条件之间分开 Separation between behavior and conditions governing behavior



结构化总结 Conclusions on structuring

- 在 TCL，结构化是测试执行者的自行决定自由
- 在 TTCN-3,结构是模型的一个组成部分，没有办法避免

4.3、工具结果检查功能对比

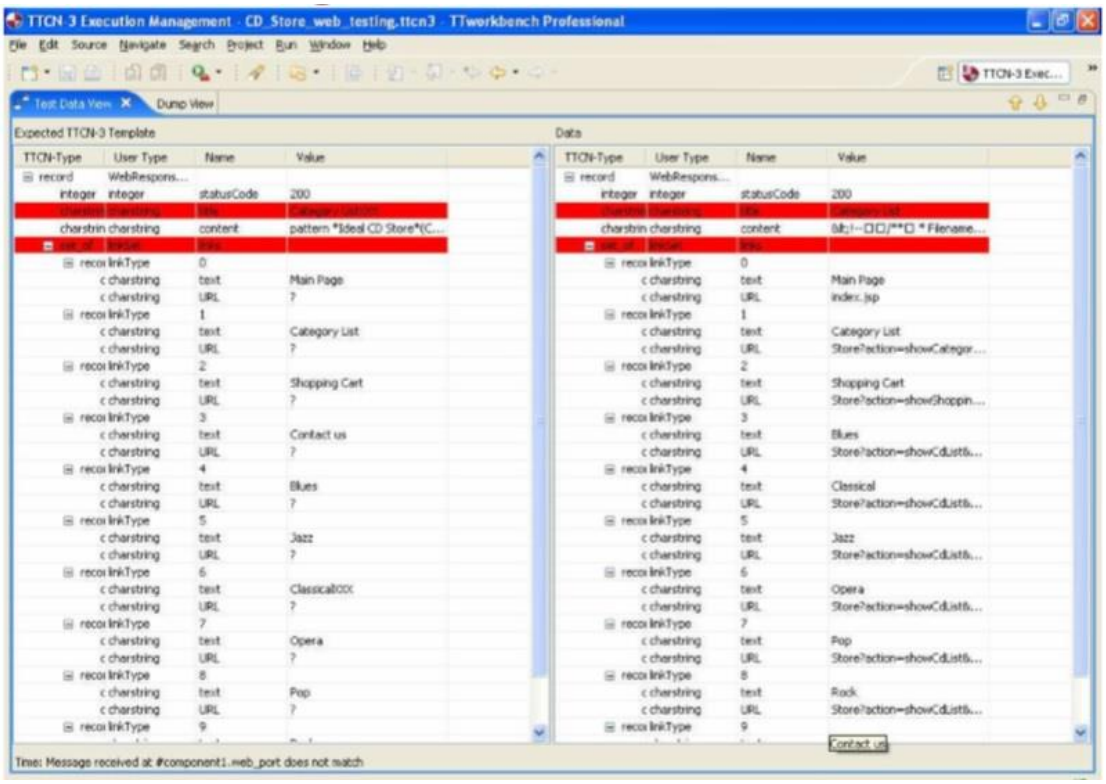
TCL/TK 结果分析功能

- 基本没有结果检查

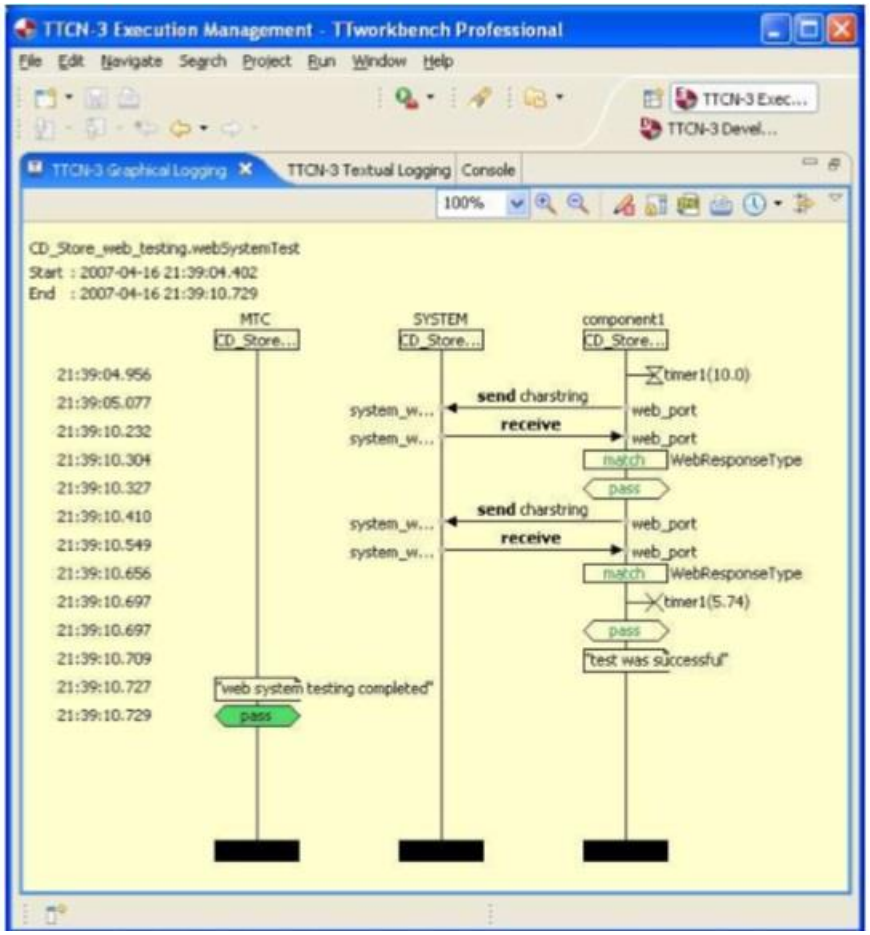
TTCN-3 工具功能

- **匹配机制概览 Matching mechanism overview**:
  in case of mismatch, the values of all the field that caused the mismatch can be viewed along with the correct values for other fields.
- **日志 Logging**: each event gets logged and thus the sequence of events can be thoroughly inspected. Thus tracing without the need of a classical debugger.
- **事件追踪 Event traceability**: Logs are not limited to display failures, they show successful events too. This improves traceability.

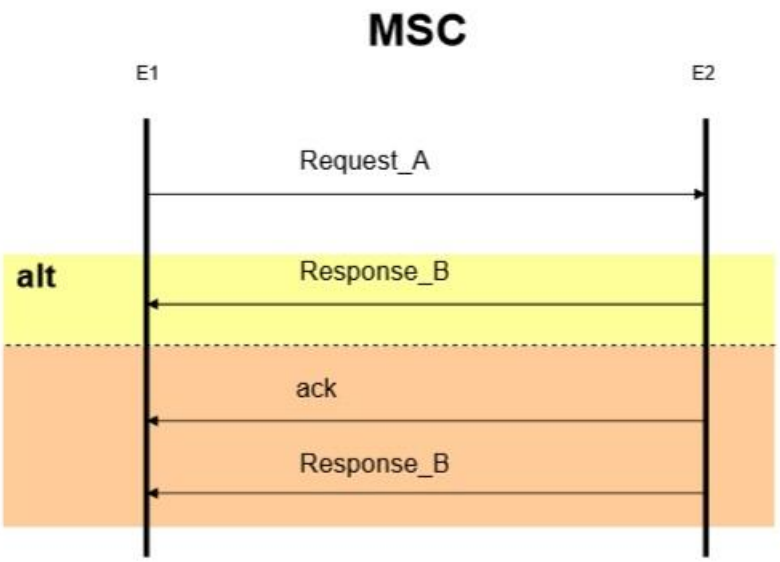
匹配机制概览 Matching mechanism overview



日志 Logging (graphical/textual)


TTCN-3 Execution Management - TTworkbench Professional
File Edit Navigate Search Project Run Window Help
TTCN-3 Exec...
TTCN-3 Devel...
TTCN-3 Graphical Logging
TTCN-3 Textual Logging
Console
100%
CD_Store_web_testing.webSystemTest
Start : 2007-04-16 21:39:04.402
End : 2007-04-16 21:39:10.729
MTC
CD_Store...
SYSTEM
CD_Store...
component1
CD_Store...
21:39:04.956
21:39:05.077
21:39:10.232
21:39:10.304
21:39:10.327
21:39:10.410
21:39:10.549
21:39:10.656
21:39:10.697
21:39:10.697
21:39:10.709
21:39:10.727
21:39:10.729
timer1(10.0)
send charstring
system_w...
web_port
receive
system_w...
web_port
match
WebResponseType
pass
send charstring
system_w...
web_port
receive
system_w...
web_port
match
WebResponseType
timer1(5.74)
pass
"test was successful"
"web system testing completed"
pass

4.4、复杂行为定义比较-Handling alternative behavior



在 TCL 中处理 alternatives

- TCL 无法应用 assert 方式处理 alternative,需要采用传统的 if-then-else 结构代替，当比较元素增加的情况下，这种 if-then-else 结构将非常难以理解。

在 TTCN-3 当中表达 alternate 行为

- TTCN-3 有自然的 alternate 结构。
- 从上到下尝试每一个 alternative，直到匹配
- 判定依据测试目的设定 Verdicts are set according to test purposes
- TTCN-3 alt 结构和模板概念的结合自然消除了行为表达的复杂度，特别是当比较元素复杂的情况下这种优势更为明显。
- TTCN-3 嵌套 alternatives 形成了自然的树形表达（tree representation）

Handling alternate behavior in TTCN-3

```
myport.send(request_A);
alt {
    [] myport.receive(response_B) { setverdict(pass)}
    [] myport.receive(ack) {
            alt {
                [] myport.receive(response_B) { setverdict(pass)}
                [] myport.receive { setverdict(fail) }
             }
  } [] myport.receive { setverdict(fail) }
}
```

- Hides the complex if-then-else constructs.

- Thus, improves clarity of behavior and gives overview qualities.
- Further structuring concept: the altstep

4.5、类型

- TCL 没有类型概念
- TTCN-3 有强大的类型系统，符合协议和软件测试对类型定义的需求
- TTCN-3 有操作语义（operational semantics）

4.5.1、TTCN-3 强大类型系统的好处

如果没有声明做为可选项，字段分配无法在模板中被省略 A field assignment can not be omitted in a template if it is not declared as optional:

```
template Car anotherCar := {
    color := "blue"
}
Above, we omitted the specification for the field named “engine”.
Compile time error message:
11:53:27:062: [ERROR]: '{ color := "blue" }' of type 'template mapping { charstring:length(4) color }' is not of type 'template Car'
11:53:27:062: [ERROR]: (reason) not optional: (field 'engine') '{ color := "blue" }.engine' must not be omitted
11:53:27:078: [ERROR]: compilation finished with errors
```

如果返回值类型错误，你不能调用一个功能来定义一个模板的字段匹配值 You can not invoke a function to specify a template field matching value if the return value type is wrong:

```
template Car aWrongCarTemplate := {
    color := "blue",
    engine := displayMyFavoriteEngine()
}
function displayMyFavoriteEngine() {
    log("I like diesel because it is cheaper");
    log("I like 8 pistons because it looks better");
}
Compile time error message:
12:04:04:515: [ERROR]: '{ color := "blue", engine := displayMyFavoriteEngine() }' of type 'template mapping { charstring:length(4) color; void engine }' is not of type 'template Car'
12:04:04:515: [ERROR]: (reason) 'displayMyFavoriteEngine()' of type 'void' would have to be of type 'record { charstring typeOfFuel; integer numberOfPistons }'
12:04:04:546: [ERROR]: compilation finished with errors
```

五、语义比较 Semantics

- TCL 没有语义概念
- TTCN-3 标准定义了静态语义和操作语义（static and operational semantics）
- 任何违反语义在编译时被检测到

六、编程风格

- TCL 的编程风格取决于开发者
- 在 TTCN-3 中，有强制执行的清晰模式，任何偏差在编译时都会自动、立即的被检测到
- 因此，TTCN-3 更为安全、便宜

七、测试语言标准化的好处

- 每个人都使用同样规范的语言
- 每个人遵循相同的模式
- 每个人因此理解公司内或者公司外第三方的方法和操作。Everybody will thus understand what another party within or outside the company means and does.
- 工具提供商将提供实用的插件如编解码器和测试适配器
- 标准组提供现成可用的测试套(ETSI,OMA,AUTOSAR、EUCONTROL、TETRA …)