

monadWS: A Monad-Based Testing Tool for Web Services

College of Computer, Nanjing University of Posts and
Telecommunications, Nanjing 210003, China

Yingzhou Zhang, Wei Fu, Changhai Nie

Email: zhangyz@njupt.edu.cn



Agenda

- **About *monadWS***
- **Monads Techniques**
- **The monads in *monadWS***
 - HSP monad
 - Gen monad
 - TestM monad
- **Tool Snapshots**
- **Conclusion**

About *monadWS*

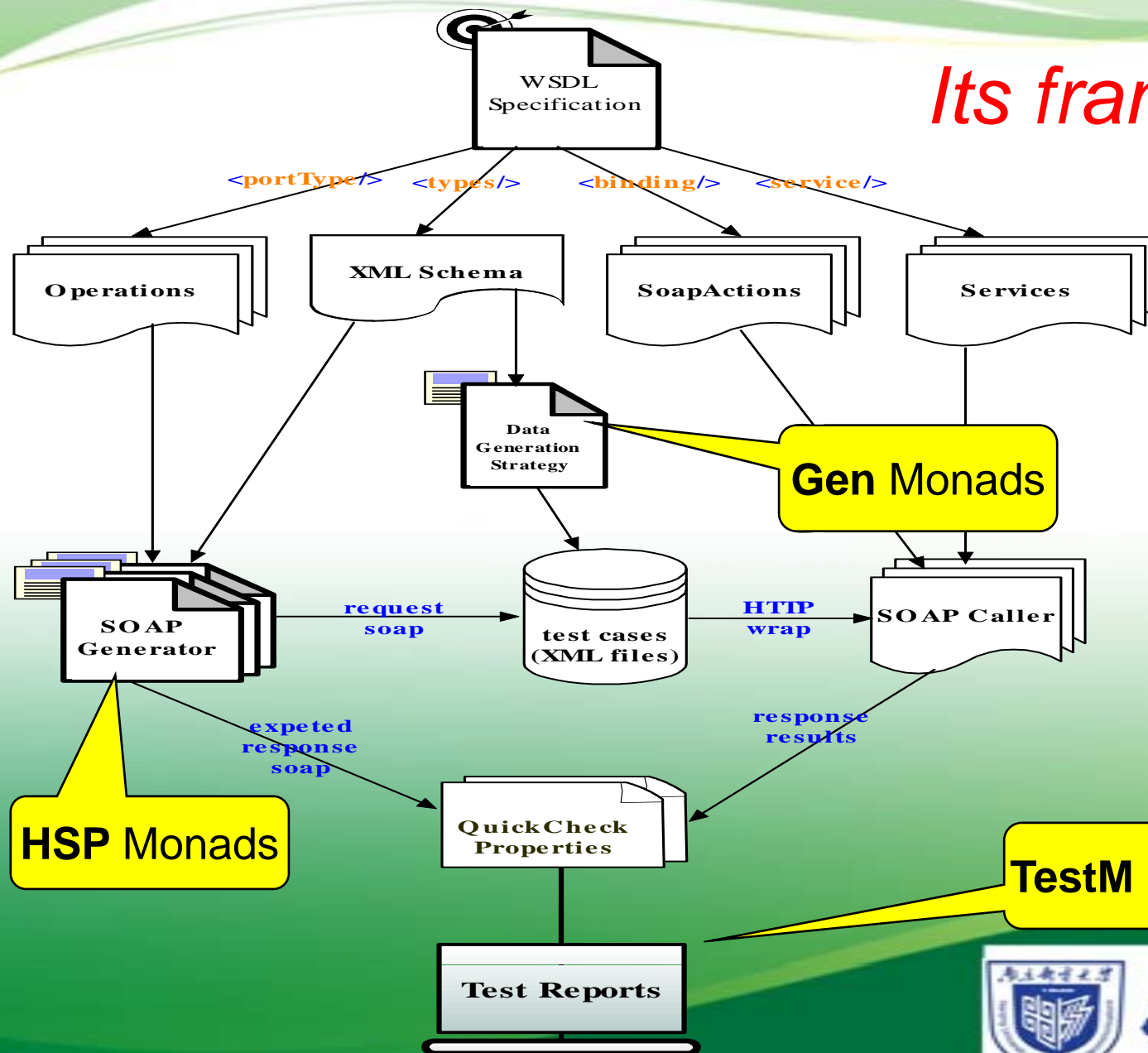
- An automatic WS testing tool based on monads techniques
- *monadWS* adopts
 - **HSP monad**^[1] for describing WS test cases
 - **Gen monad**^[2] for generating test data automatically, and
 - **TestM monad** for building the whole WS testing.

[1] HSP (Haskell Server Pages): <http://code.google.com/p/hsp/>

[2] Claessen K, Hughes J. QuickCheck: a lightweight tool for random testing of Haskell programs. In *Proceedings of the Fifth ACM SIGPLAN international Conference on Functional Programming*, New York, 2005: 268~279.



Its framework



Agenda

- About *monadWS*
- **Monads Techniques**
- The monads in *monadWS*
 - HSP monad
 - Gen monad
 - TestM monad
- Tool Snapshots
- Conclusion

Monads Intro.

- A general functional notation
- Its monadic version of this notion
- Operations on monad M

$f: a \rightarrow b$

$f: a \rightarrow M b$

$return :: a \rightarrow M a$

$bind :: M a \rightarrow (a \rightarrow M b) \rightarrow M b$

Therefore, a monad can be thought as
a **strategy** for combining computations
into more complex computations.

Agenda

- About *monadWS*
- Monads Techniques
- **The monads in *monadWS***
 - HSP monad
 - Gen monad
 - TestM monad
- Tool Snapshots
- Conclusion

HSP Monad: represent TC

- ◆ represents a WS test case as a SOAP message
- ◆ can encapsulate any side effects, and lets all XML expressions be computed safely

```
makeTC_serv :: InputType -> HSP XML
```

The input type of a service operation (say *serv*), such as String, Double ...



HSP Monad: represent TC

```
makeTC_serv :: InputType -> HSP XML
```

For example:

```
makeTC_getOffesetUTCTime :: Double -> HSP XML
makeTC_getOffesetUTCTime d =
  <soap:Envelope xmlns:soap=soapURI xmlns:xsd=xsdURI>
    <soap:Body>
      <getOffesetUTCTime xmlns=operURL>
        <hoursOffset> <% show d %> </hoursOffset>
      </getOffesetUTCTime>
    </soap:Body>
  </soap:Envelope>
where
  soapURI = "http://schemas.xmlsoap.org/soap/envelope/"
  xsdURI = "http://www.w3.org/2001/XMLSchema"
  operURL = "http://www.Nanonull.com/TimeService/"
```



HSP Monad: represent TC

```
makeTC_getOffesetUTCTime :: Double -> HSP XML
makeTC_getOffesetUTCTime d =
  <soap:Envelope xmlns:soap=soapURI xmlns:xsd=xsdURI>
    <soap:Body>
      <getOffesetUTCTime xmlns=operURL>
        <hoursOffset> <% show d %> </hoursOffset>
      </getOffesetUTCTime>
    </soap:Body>
  </soap:Envelope>
where
  soapURI = "http://schemas.xmlsoap.org/soap/envelope/"
  xsdURI = "http://www.w3.org/2001/XMLSchema"
  operURL = "http://www.NanjingUniversity.com/TimeService/"
```

HSP provides interfaces for other computations through the escapes (with “<%” and “%>”)



Gen Monad: generate data

- ◆ automatically generates data for simple types (such as Double, Int, String and so on)

```
class Arbitrary a where  
  arbitrary :: Gen a
```

instance Arbitrary String ...

instance Arbitrary Double ...



Gen Monad: generate data

```
class Arbitrary a where  
  arbitrary :: Gen a
```

```
makeTC_getOffesetUTCTime :: Double -> HSP XML
```

➤ *general testing method:*

Step 1. generate a test value

Step 2. generate the final test case with the value

```
testTCGen :: IO String  
testTCGen = do  
  d <- generate (arbitrary :: Gen Double)  
  return $ hsp2str (makeTC getOffesetUTCTime d)
```

Gen Monad: generate data

```
makeTC_getOffsetUTCTime :: Double -> HSP XML
```

➤ *general testing method:*

```
testTCGen :: IO String
testTCGen = do
  d <- generate (arbitrary :: Gen Double)
  return $ hsp2str (makeTC_getOffsetUTCTime d)
```

➤ *monadic method by liftM : on-the-fly generate*

```
testTCGen2 :: IO String
testTCGen2 =
  liftM (hsp2str . makeTC_getOffsetUTCTime) $
    generate (arbitrary :: Gen Double)
```



Gen Monad: generate data

- ◆ We can easily change the generation strategy through the instances of Arbitrary class.

```
class Arbitrary a where  
  arbitrary :: Gen a
```

```
instance Arbitrary Double where  
  arbitrary = myDoubleGen
```

```
myDoubleGen :: Gen Double  
myStringGen = suchThat arbitrary guard  
where  
  guard a = (a > 0) && (a < 1000)  
           || elem a [-1, 0, 9999]
```



TestM Monad: testing as a monad

- ◆ processes WS testing as the following *TestM* monad:

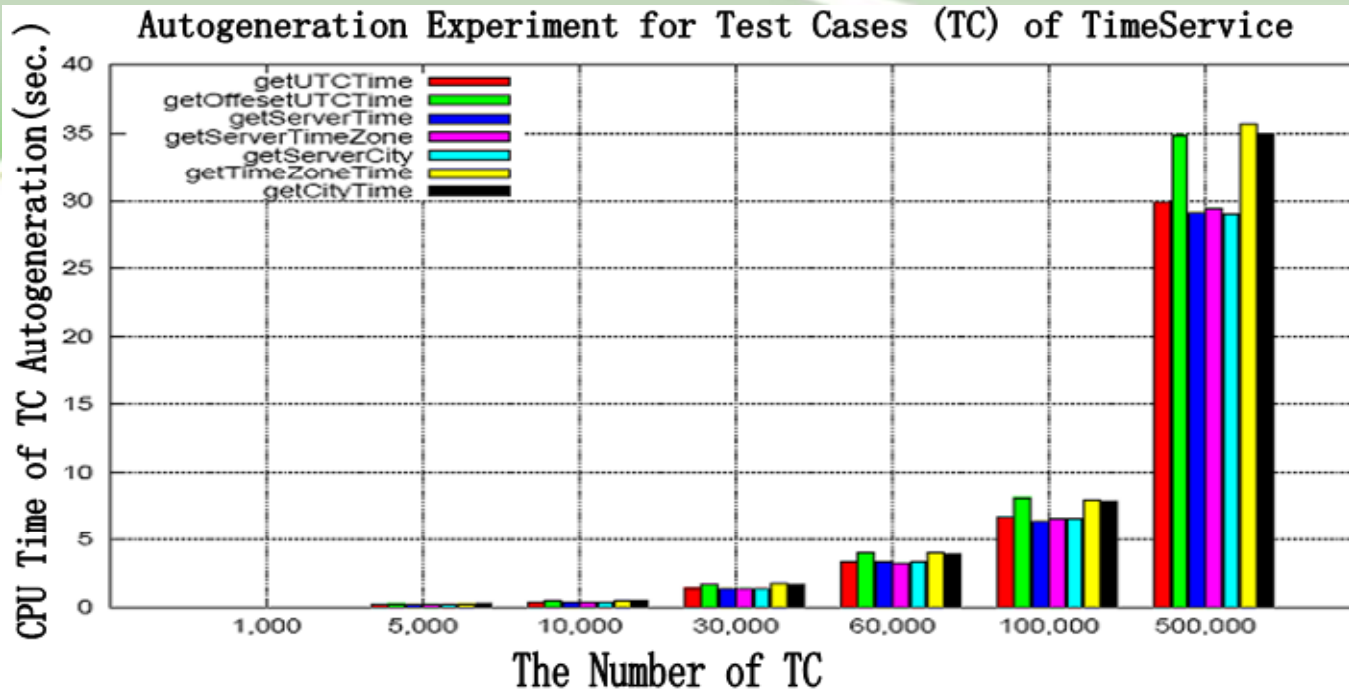
```
type TestM a
  = CoAlgMonadT
    (EnvT TestEnv
    (StateT TestState
    HSP
    )) a
```

Coalgebraic monad
for WS computing

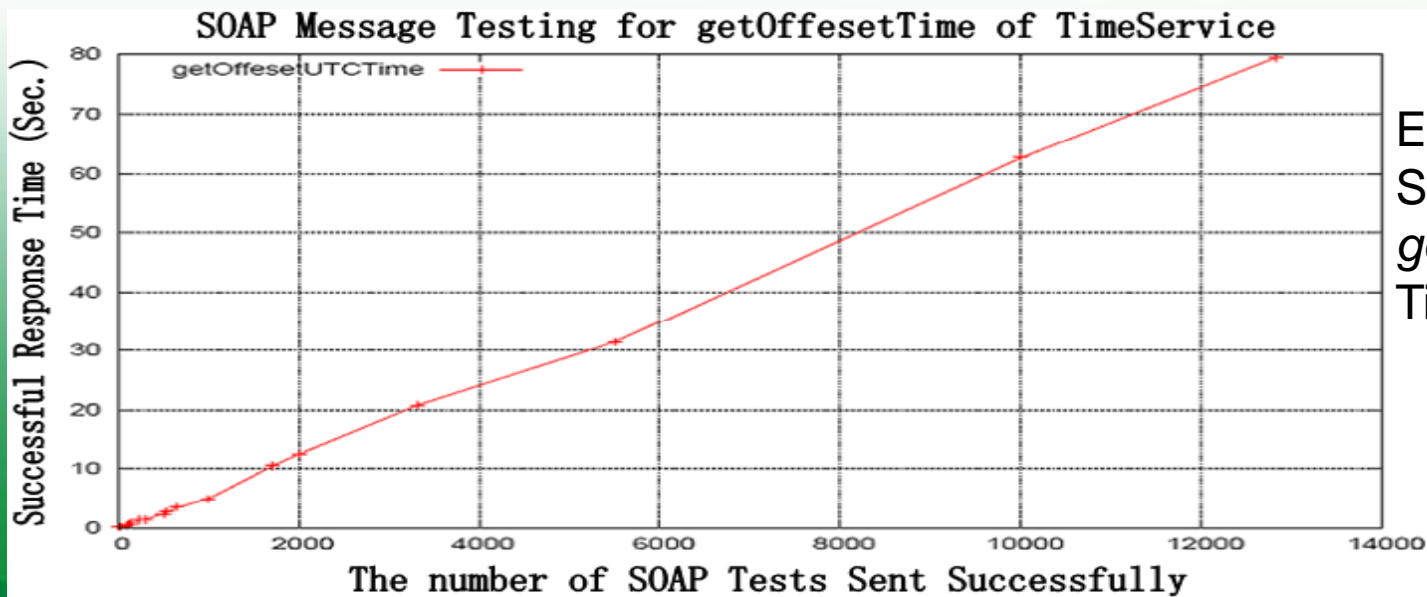
Environment monad for
testing environment
(eg. Schema type information,
test cases files)

State monad for testing states
(eg. executing time, testing results)





CPU Time Results of Autogeneration Test Cases for TimeService



Execution Time of the SOAP Messages of getOffsetUTCTime in TimeService

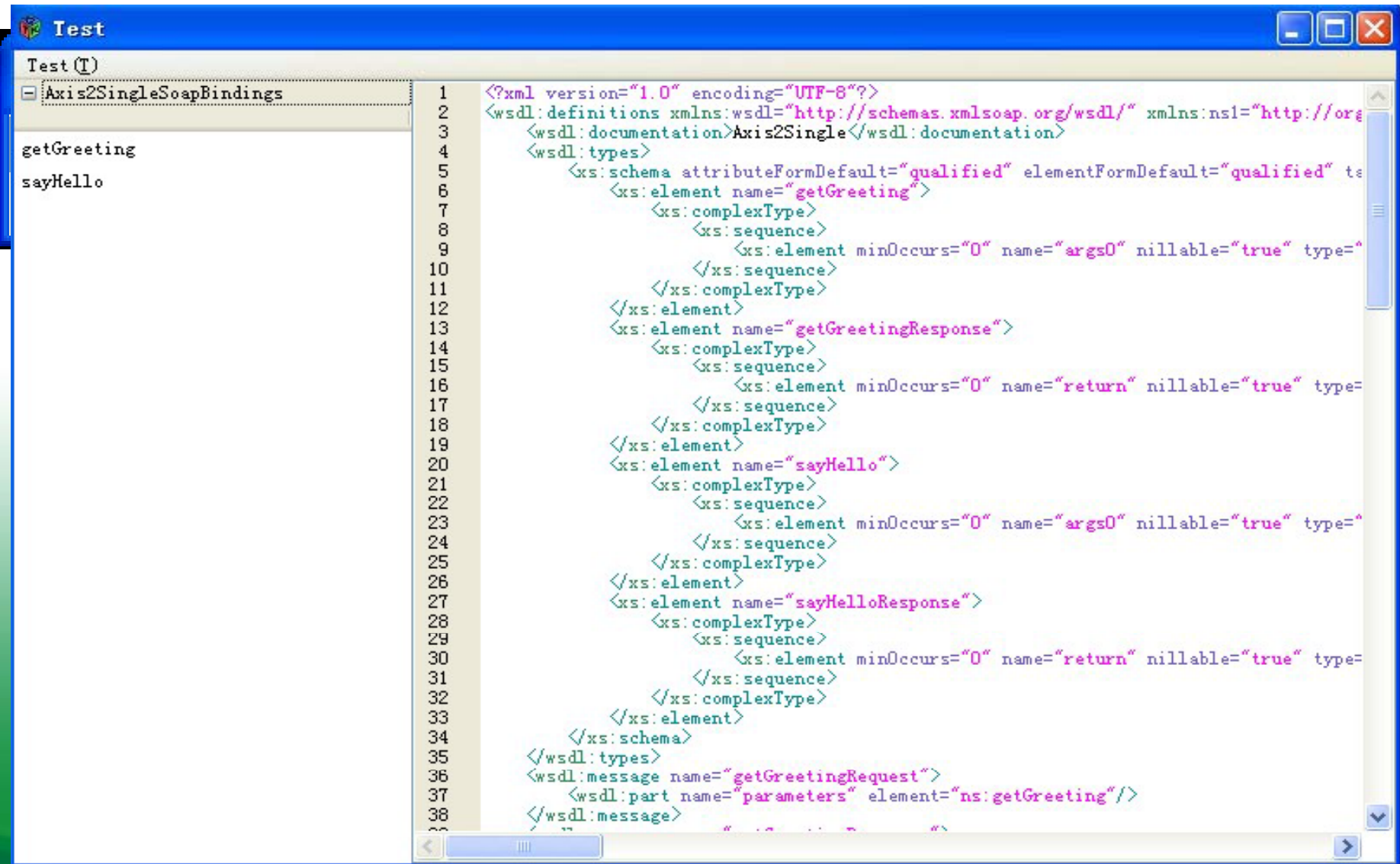
Comparison of WSTester with Other Test Tools for Web Services

web services		WSTester		soapUI		SoapTest	
websites	oper #	Time (sec.)	fail #	Time (sec.)	fail #	Time (sec.)	fail #
http://www.websvicex.net/country.aspx	10	0.406	0	2.665	0	3.219	2
http://www.websvicex.com/globalweather.aspx	2	0.078	0	1.054	0	0.632	0
http://www.websvicex.net/CurrencyConverter.aspx	1	0.078	0	0.196	0	0.239	0
http://www.websvicex.com/airport.aspx	4	0.109	0	0.814	0	0.978	0
http://www.websvicex.net/BibleWebService.aspx	4	0.265	3	44.26	0	1.823	1
http://www.websvicex.net/Statistics.aspx	1	0.125	0	0.976	0	0.312	0
http://www.xignite.com/xcompensation.aspx	10	0.203	0	7.612	0	2.800	0
http://www.webxml.com.cn/WebServices/WeatherWebService.aspx	5	0.172	0	0.745	0	4.297	2
http://www.websvicex.net/sendsmsworld.aspx	1	0.047	0	0.792	0	0.210	0
http://www.websvicex.net/stockquote.aspx	1	0.031	0	0.545	0	0.612	0
http://fy.webxml.com.cn/webservices/EnglishChinese.aspx	6	0.094	0	0.212	0	7.253	3
http://webservice.webxml.com.cn/WebServices/MobileCodeWS.aspx	2	0.047	0	0.149	0	1.484	1
http://webservice.webxml.com.cn/WebServices/WeatherWS.aspx	6	0.140	0	0.147	0	5.320	3
http://webservice.webxml.com.cn/WebServices/StockInfoWS.aspx	1	0.078	0	0.019	0	4.055	0
http://webservice.webxml.com.cn/WebServices/ChinaOpenFundWS.aspx	6	0.328	0	0.107	0	6.301	2
http://webservice.webxml.com.cn/webservices/DomesticAirline.aspx	2	0.109	0	0.523	0	15.05	1

Agenda

- About *monadWS*
- Monads Techniques
- The monads in *monadWS*
 - HSP monad
 - Gen monad
 - TestM monad
- Tool Snapshots
- Conclusion

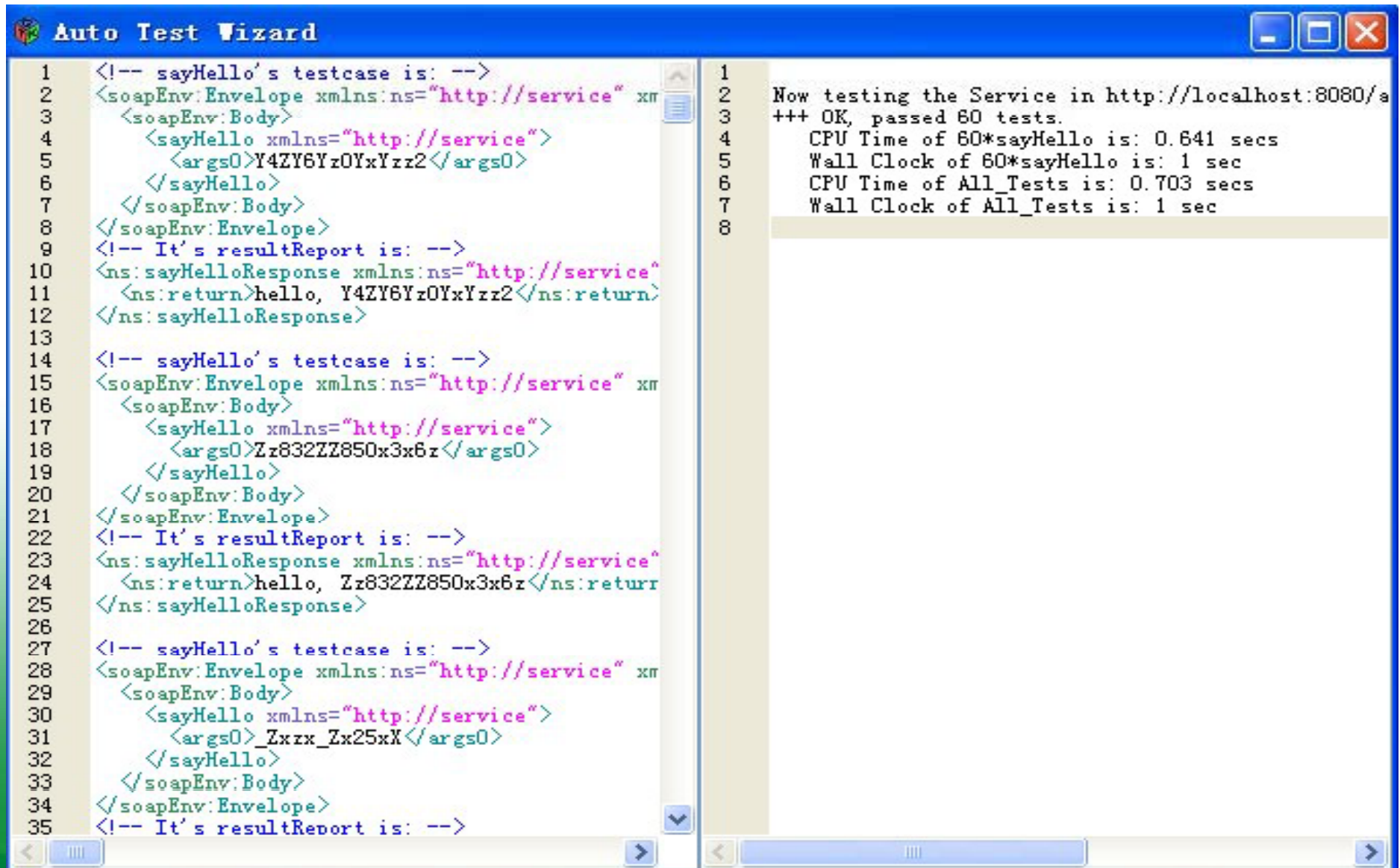
Snapshots of our tool *monadWS*



The screenshot displays a window titled "Test" with a tree view on the left and a code editor on the right. The tree view shows a folder "Axis2SingleSoapBindings" containing two items: "getGreeting" and "sayHello". The code editor shows the following WSDL schema definition:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:ns1="http://org
3 <wsdl:documentation>Axis2Single</wsdl:documentation>
4 <wsdl:types>
5 <xs:schema attributeFormDefault="qualified" elementFormDefault="qualified" te
6 <xs:element name="getGreeting">
7 <xs:complexType>
8 <xs:sequence>
9 <xs:element minOccurs="0" name="args0" nillable="true" type="
10 </xs:sequence>
11 </xs:complexType>
12 </xs:element>
13 <xs:element name="getGreetingResponse">
14 <xs:complexType>
15 <xs:sequence>
16 <xs:element minOccurs="0" name="return" nillable="true" type=
17 </xs:sequence>
18 </xs:complexType>
19 </xs:element>
20 <xs:element name="sayHello">
21 <xs:complexType>
22 <xs:sequence>
23 <xs:element minOccurs="0" name="args0" nillable="true" type="
24 </xs:sequence>
25 </xs:complexType>
26 </xs:element>
27 <xs:element name="sayHelloResponse">
28 <xs:complexType>
29 <xs:sequence>
30 <xs:element minOccurs="0" name="return" nillable="true" type=
31 </xs:sequence>
32 </xs:complexType>
33 </xs:element>
34 </xs:schema>
35 </wsdl:types>
36 <wsdl:message name="getGreetingRequest">
37 <wsdl:part name="parameters" element="ns:getGreeting"/>
38 </wsdl:message>
```

Snapshots of our tool *monadWS*



The screenshot shows a window titled "Auto Test Wizard" with two panes. The left pane displays XML test cases for a "sayHello" service, and the right pane shows the test results.

```
1 <!-- sayHello's testcase is: -->
2 <soapEnv:Envelope xmlns:ns="http://service" xm
3   <soapEnv:Body>
4     <sayHello xmlns="http://service">
5       <args0>Y4ZY6Yz0YxYzz2</args0>
6     </sayHello>
7   </soapEnv:Body>
8 </soapEnv:Envelope>
9 <!-- It's resultReport is: -->
10 <ns:sayHelloResponse xmlns:ns="http://service"
11   <ns:return>hello, Y4ZY6Yz0YxYzz2</ns:return>
12 </ns:sayHelloResponse>
13
14 <!-- sayHello's testcase is: -->
15 <soapEnv:Envelope xmlns:ns="http://service" xm
16   <soapEnv:Body>
17     <sayHello xmlns="http://service">
18       <args0>Zz832ZZ850x3x6z</args0>
19     </sayHello>
20   </soapEnv:Body>
21 </soapEnv:Envelope>
22 <!-- It's resultReport is: -->
23 <ns:sayHelloResponse xmlns:ns="http://service"
24   <ns:return>hello, Zz832ZZ850x3x6z</ns:returr
25 </ns:sayHelloResponse>
26
27 <!-- sayHello's testcase is: -->
28 <soapEnv:Envelope xmlns:ns="http://service" xm
29   <soapEnv:Body>
30     <sayHello xmlns="http://service">
31       <args0>_Zxrx_Zx25xX</args0>
32     </sayHello>
33   </soapEnv:Body>
34 </soapEnv:Envelope>
35 <!-- It's resultReport is: -->
```

```
1
2 Now testing the Service in http://localhost:8080/a
3 +++ OK, passed 60 tests.
4 CPU Time of 60*sayHello is: 0.641 secs
5 Wall Clock of 60*sayHello is: 1 sec
6 CPU Time of All_Tests is: 0.703 secs
7 Wall Clock of All_Tests is: 1 sec
8
```

Snapshots of our tool *monadWS*

The screenshot displays the 'Manual Test Wizard' window. It features a 'Begin Test' button at the bottom left. The main area is divided into two panes. The left pane shows the SOAP request XML with line numbers 1 through 7 on the left margin. The right pane shows the SOAP response XML. A small table at the top left of the wizard lists 'Available Operations' with line numbers 1, 2, and 3.

Available Operations	1	2	3
	<soapEnv:Envelope xmlns:ns="http://ser	<soapEnv:Body>	

```
1 <soapEnv:Envelope xmlns:ns="http://service" xmlns:ser="http://service" >
2   <soapEnv:Body>
3     <getGreeting xmlns="http://service" >
4       <args0>Wei Fu</args0>
5     </getGreeting>
6   </soapEnv:Body>
7 </soapEnv:Envelope>
```

```
<ns:getGreetingResponse xmlns:ns="http://service">
  <ns:return>hi, Wei Fu</ns:return>
</ns:getGreetingResponse>
```



Agenda

- About *monadWS*
- Monads Techniques
- The monads in *monadWS*
 - HSP monad
 - Gen monad
 - TestM monad
- Tool Snapshots
- **Conclusion**

Conclusion

- briefly introduce our monad-based testing tool, *monadWS*
- To automatically test web services.
- *monadWS* can use monads related to help
 - WS test case representation,
 - test data autogeneration, and
 - test auto-execution.

Conclusion

- *monadWS* can use monads related to help
 - WS test case representation,
 - test data autogeneration, and
 - test auto-execution.
- Future work
 - More comparisons with existing tools
 - Use more data-generation strategy
 - Apply to WS composition (with BPEL)



Thank you!

