Natural Language Requirement Specification for Web Service Testing

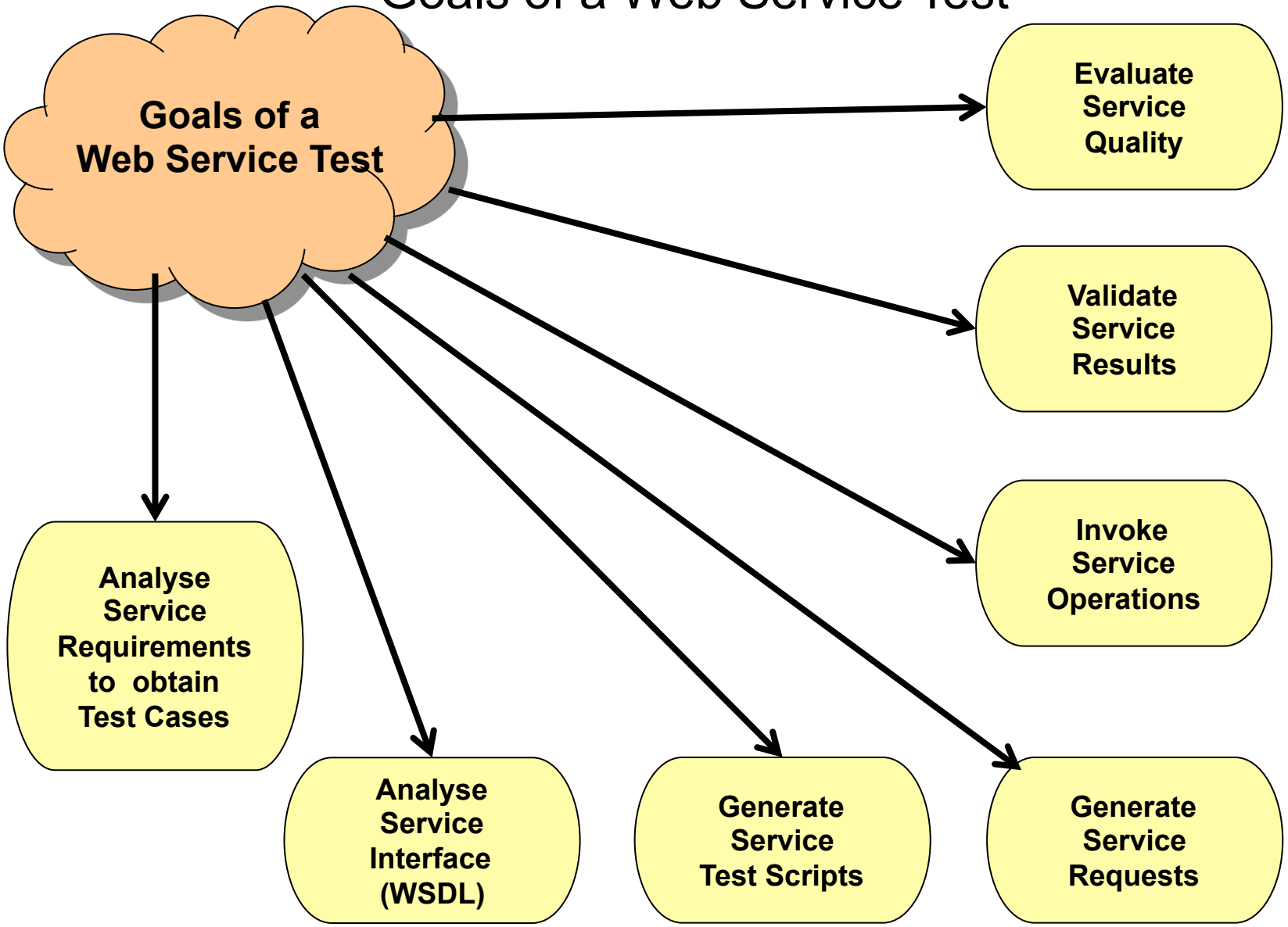
Harry M. Sneed &
Chris Verhoef

for

WSE-2013

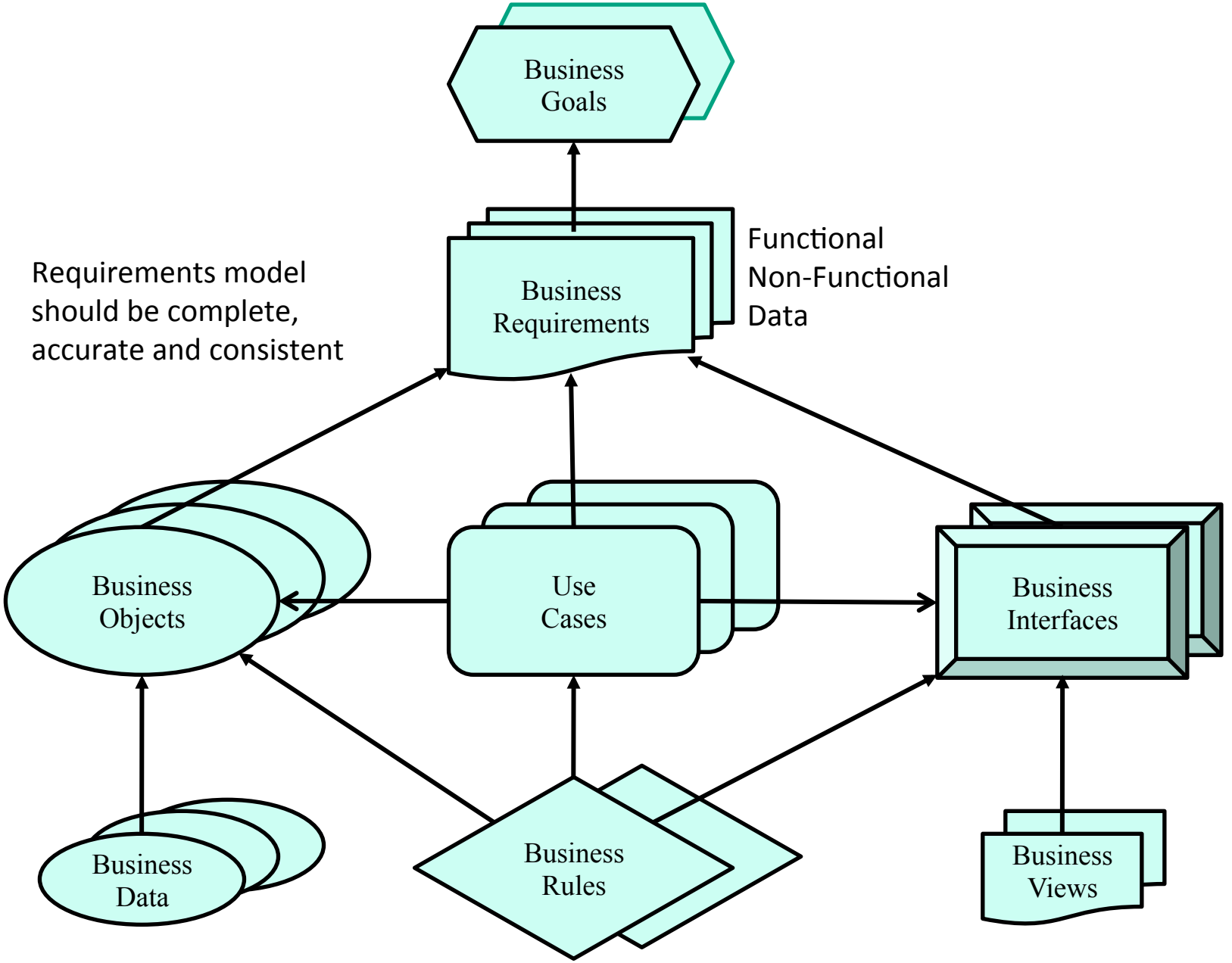
Eindhoven

Goals of a Web Service Test

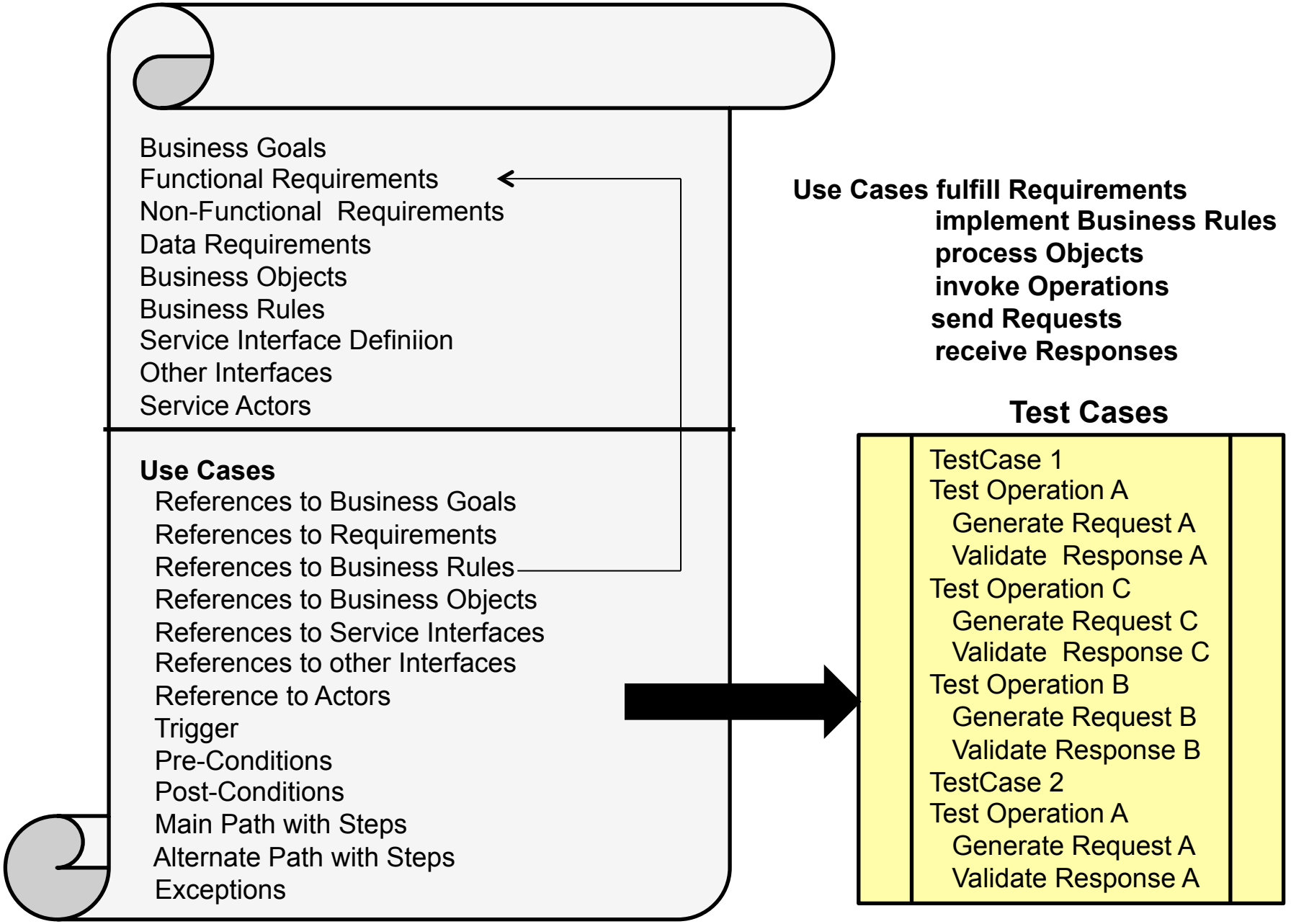


The overall Goal is to select the best possible service for the task at hand

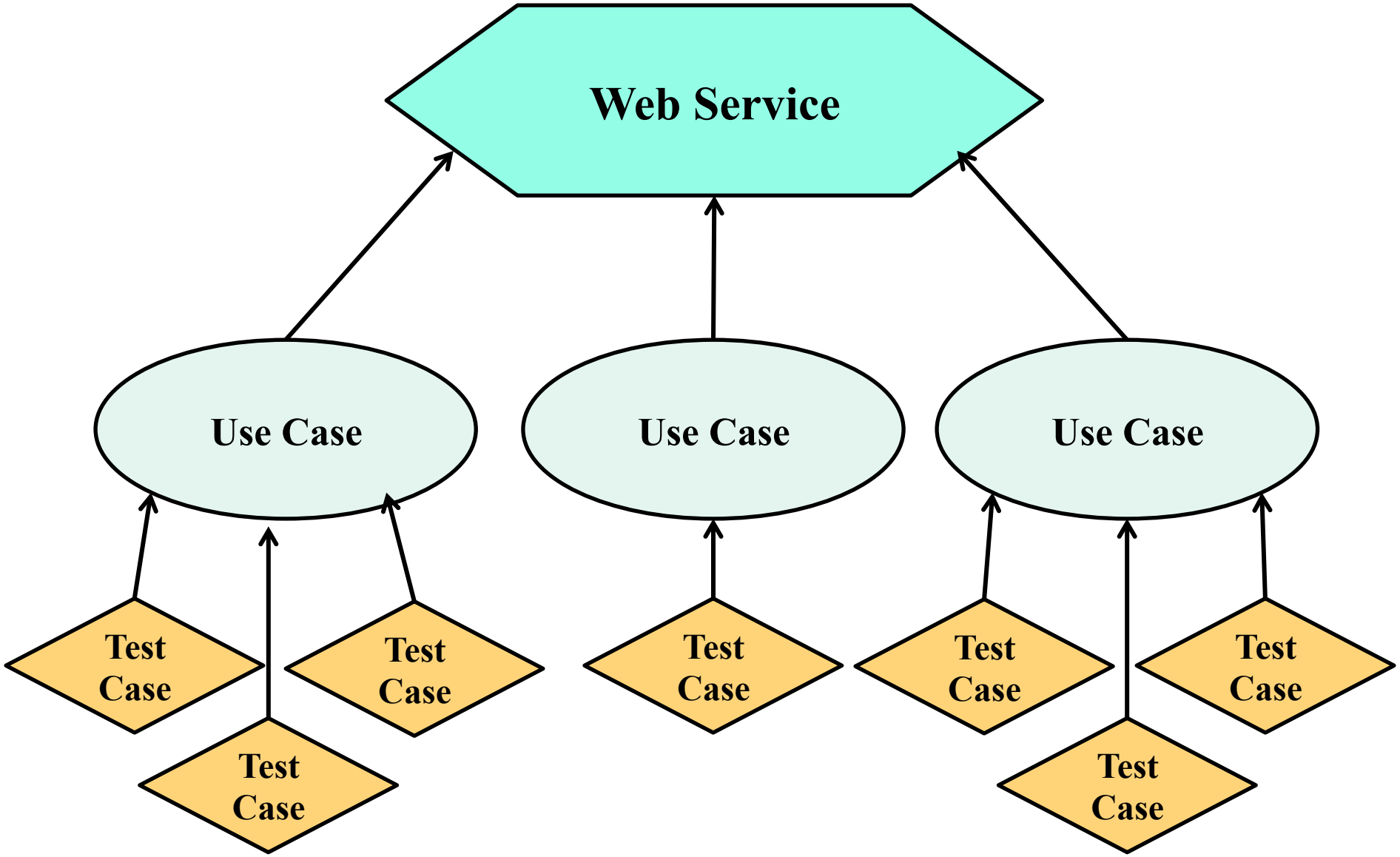
Structure of a Service Requirement Specification (SRS)



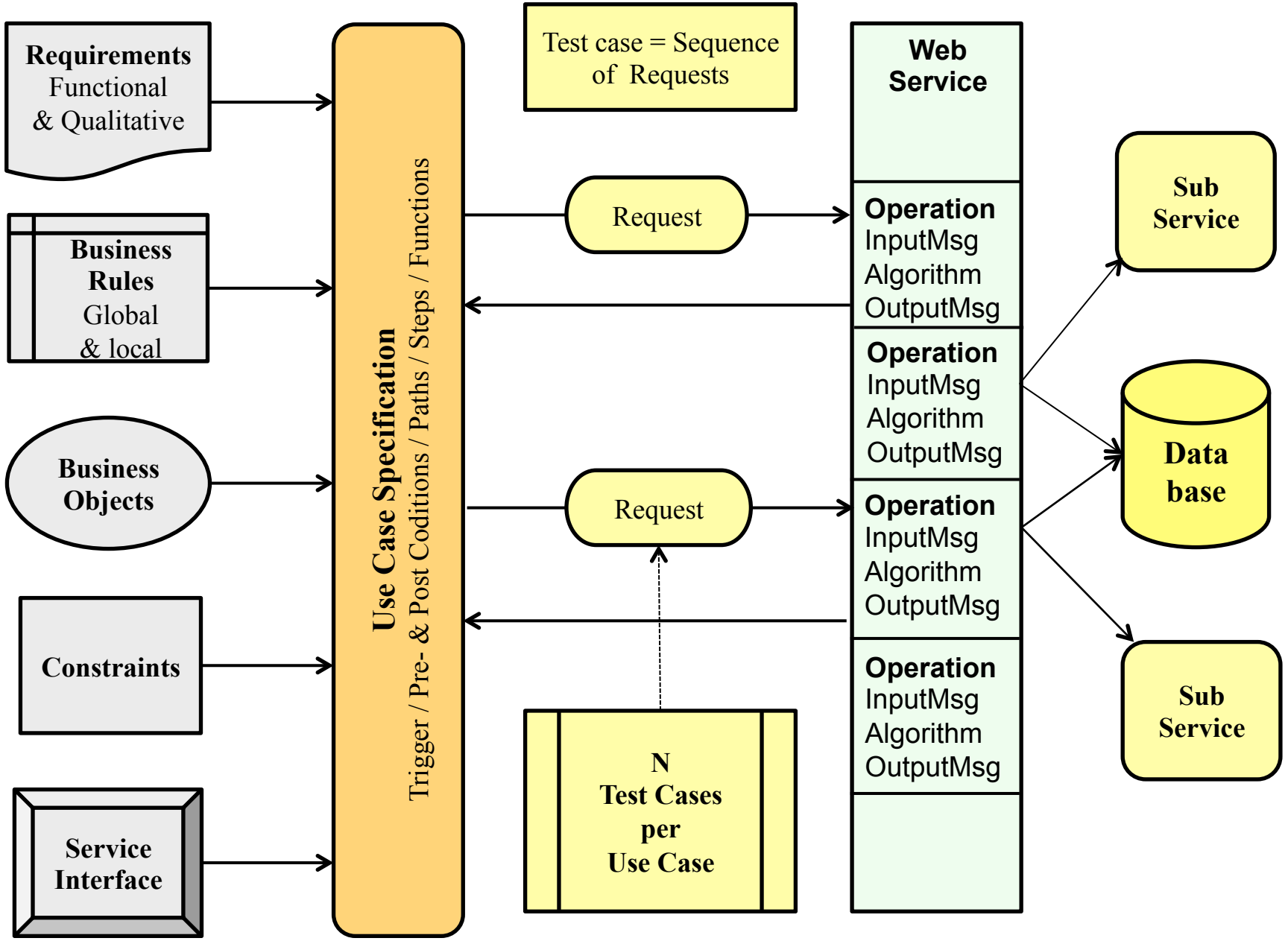
Contents of a Service Requirement Specification

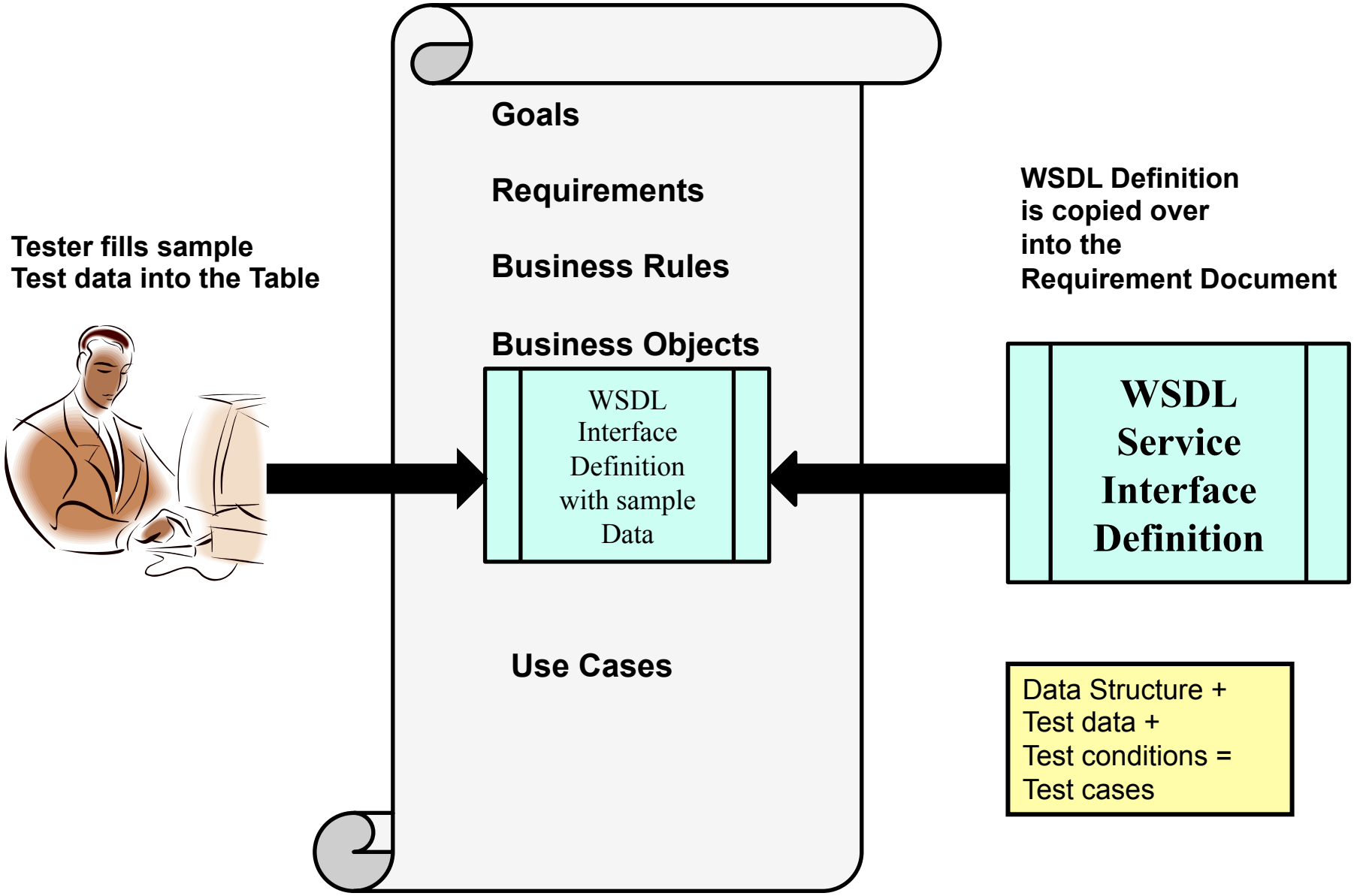


Role of Use Cases in Connecting Test Cases to the Service



From Requirements to Web Services via Use Cases




Goals
Requirements
Business Rules
Business Objects
Use Cases
Tester fills sample
Test data into the Table
WSDL
Interface
Definition
with sample
Data
WSDL Definition
is copied over
into the
Requirement Document
WSDL
Service
Interface
Definition
Data Structure +
Test data +
Test conditions =
Test cases

Service Interface Definition Table

FUNC-04:	**0**	**QueryAvailableProductsString**	**struc**	
INPUT-04:	**1**	**QueryAvailableProductsString3Request**	**struc**	
PARM:	**2**	**CustNo**	**string**	**“099999” ;**
PARM:	**2**	**ArtType**	**string**	**“Buch”;**
OUTPUT-04:	**1**	**QueryAvailableProductsString3Response**	**struc**	
RESULT:	**2**	**return**	**array**	
RESULT:	**3**	**item[1]**	**string**	**“Endstation Wien”;**
RESULT:	**3**	**item[2]**	**string**	**“SoftwareEvolution”;**
FUNC-05:	**0**	**BuySomething**	**struc**	
INPUT-05:	**1**	**BuySomething4Request**	**struc**	
PARM:	**2**	**AnOrder**	**struc**	
PARM:	**3**	**CustNo**	**string**	**“099999”;**
PARM:	**3**	**OrderNo**	**int**	**“05054”;**
PARM:	**3**	**DeliveryMode**	**string**	**“Post”;**
PARM:	**3**	**PaymentMode**	**string**	**“Cash”;**
PARM:	**3**	**Orders**	**array**	
PARM:	**4**	**item[1]**	**struc**	
PARM:	**5**	**OrderArtNo[1]**	**string**	**“01”;**
PARM:	**5**	**OrderArtType[1]**	**string**	**“Buch”;**
PARM:	**5**	**OrderArtName[1]**	**string**	**“Endstation Wien”;**
PARM:	**5**	**OrderArtAmount[1]**	**int**	**“5”;**
PARM:	**4**	**item[2]**	**struc**	
PARM:	**5**	**OrderArtNo[2]**	**string**	**“02”;**
PARM:	**5**	**OrderArtType[2]**	**string**	**“Buch”;**
PARM:	**5**	**OrderArtName[2]**	**string**	**“SoftwareEvolution”;**
PARM:	**5**	**OrderArtAmount[2]**	**int**	**“11”;**

Specification of a Use Case

Attribute	Description
Title	Billing
fulfills	*Func-Req-06
Implements	*BR-12, *BR-13, *BR-14, *BR-15.
Invokes	*FUNC-06.
Receives	*INPUT-06.
Sends	*OUTPUT-06, *REPORT-03.
Processes	*BO-01, *BO-10, *BO-11.
Trigger	GUI-Menu.
Actor	Accountant
PreConditions	Billing item file must be available.
PostConditions	Bills are printed.
Main Path	1) Accountant starts Billing job. 2) Service sorts Bills by Customer. 3) Service collects billing items for each customer. 4) Service fetches the customer data. 5) Service sums up the prices. 6) Service computes the VAT. 7) Service prints out a bill for each customer.
Alternate Paths	None
Exceptions	Service finds no billing items. Service finds no customer data for the billing items. Service finds no VAT Table.
Response Time	3000 MS
Inherits	Standard-Printing Use Case
Uses	Batch-Process
Extends	Customer Order processing
Comment	Bill recipient is the customer.

Specifying Individual Requirements

Functional Requirements

FREQ-02 ArticleSelection
The customer should have the possibility to view all the articles of a selected type with their names and prices.
The articles should be ordered alphabetically.
FREQ-03 OrderProcessing
The customer should have the possibility of selecting one or more articles to be ordered. He needs only to select an article and give in the amount. The service will confirm his identity and check his credibility. The credibility rate should be over 2. If they are both ok, the processing may continue, otherwise it should be terminated with a message to the user.
If an ordered article is available on stock and in sufficient quantity, the order is to be accepted. If the article is not on stock the order is to be rejected. If the quantity on stock is too low, a back order is to be created.

Non-Functional Requirements

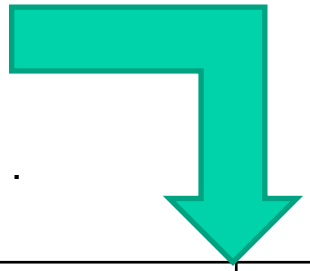
QREQ-01 ResponseTime
The response time for a customer query should be <= 2 seconds. The response time for a customer order should be <= 3 seconds. The response time for the background jobs should be <= 10 seconds.
QREQ-03 Availability
The system should be available 24 hours a day, 7 days a week, for at least 95% of the time.

Specifying a Service Use Case

Use Case Specification;
Label: processCustomerOrder
Requirements: FREQ-01, FREQ-02, FREQ-03.
Rules: BR-01, BR-02, BR-03, BR-04, BR-05, BR-06.
Functions: FUNC-01, FUNC-02, FUNC-03, FUNC-04.
Inputs: INPUT-01, INPUT-02, INPUT-03, INPUT-04.
Outputs: OUTPUT-01, OUTPUT-02, OUTPUT-03, OUTPUT-04.
Objects: BO-01, BO-02, BO-05, BO-06, BO-07, BO-09, BO-11, BO-13.
Trigger: Menu_Selection
Actor: Customer
Frequency: Continuous
ResponseTime 3 Sec
PreConditions: Ordered article must be on stock. Customer must be authorized to purchase articles.
Customer must be credible.
PostConditions: Article amount is reduced by the order and Dispatch order exits for each fulfilled order item and
Billing item exits for each fulfilled order item or back order exits for not fulfilled order item.
A supply order item exits if the article amount falls below the minimum order amount required.
MainPath:
1) GetArticleTypes Customer requests article types.
Service returns a list of article types.
2) ArticleQuery Customer selects an article type.
Service returns a list of articles and prices.
3) CustomerOrder Customer orders an article.
Service checks customer credit.
Service checks if article on stock.
Service checks if amount on stock is sufficient.
If not sufficient, service creates a back order.
If sufficient, service subtracts amount ordered from amount on stock, creates a dispatch order item and creates a billing item .
If amount on stock falls below the minimum amount, the service creates a resupply order.

Generating a Service Test Case Table

FUNC-02: QueryAvailableProducts.
INPUT-02: QueryAvailableProducts2Request.
The article query request should contain:
CustNo = “009999”.
ArtType = “BOOK”.
OUTPUT-02: QueryAvailableProducts2Response.
The article query response should contain:
item[1].
ResponseArtNo[1] = “004711”.
ResponseArtType[1] = “BOOK”.
ResponseArtName[1] = “MeinKampf” .
ResponseArtPrice[1] = “40.50”.



TestCase	Operation	Parameter	Type	Ind	Value
Orders02	GetTypes	GetTypes1Request	Parameter		?
	GetTypes	item	Return	1	**MAGA**
	GetTypes	item	Return	2	**NEWS**
	GetTypes	item	Return	3	**BOOK**
	QueryAvail	CustNo	Parameter		**009999**
	QueryAvail	ArtType	Parameter		**BOOK**
	QueryAvail	Item	Return	1	**Struc**
	QueryAvail	ResponseArtNo	Return	1	**4711**
	QueryAvail	ResponseArtType	Return	1	**BOOK**
	QueryAvail	ResponseArtName	Return	1	**MeinKampf**
	QueryAvail	ResponseArtPrice	Return	1	**40.50**

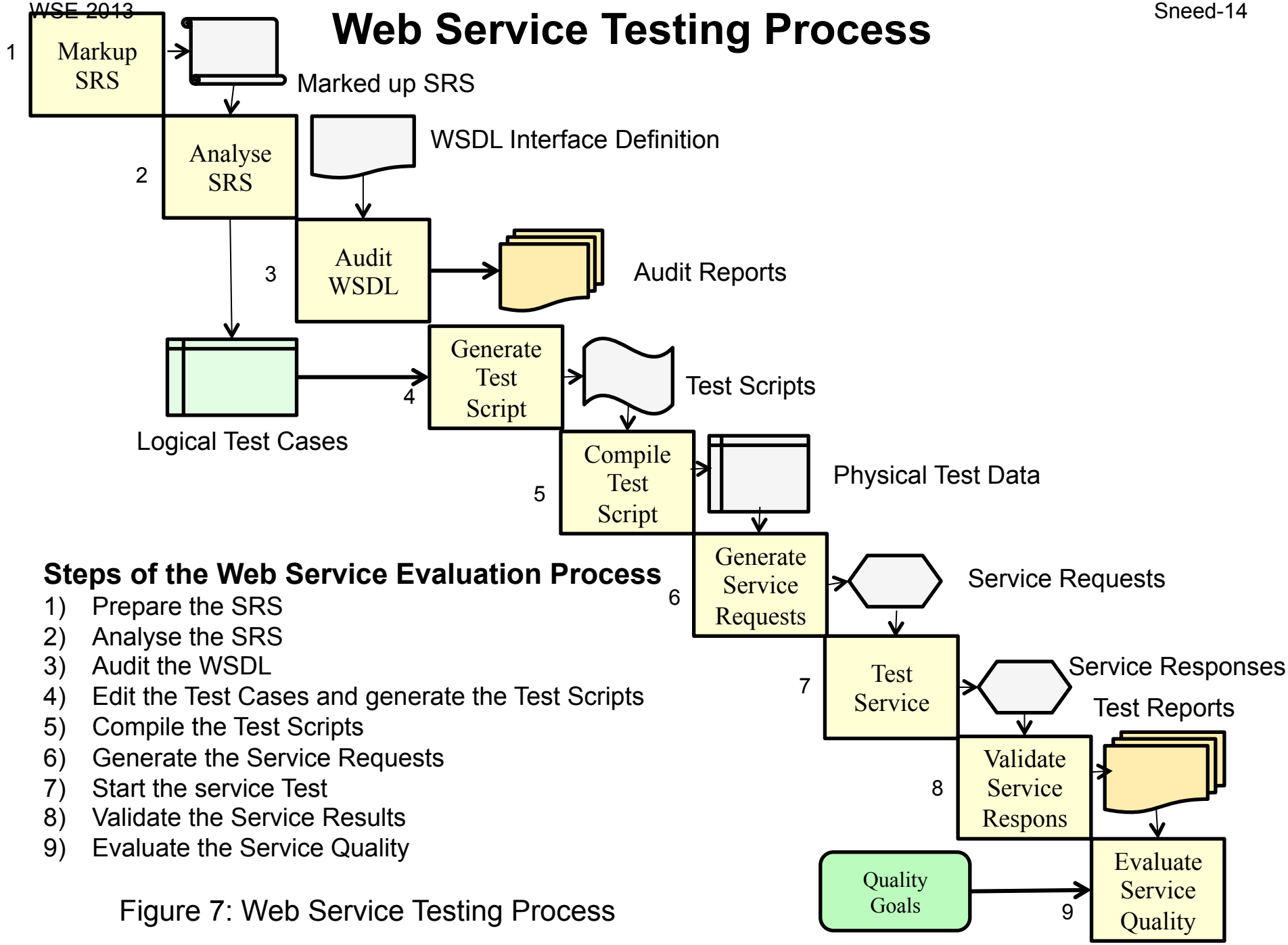
Generating a Web Service Test Script


```
//  First Request to Frontend to order Articles
      if ( operation =   "GetTypes");
         if ( request  =   "GetTypes1Request");
            assert inp.GetTypes1Request_DummyParam  = "?";
         endRequest ;
         if ( response  =   "GetTypes1Response");
            assert out.$ResponseTime < "1000";
            if ( object = "return" occurs = "2");
               assert out.item[1]  = old.item[1];
           endObject;
         endResponse ;
      endOperation;
      if ( operation =   "QueryAvailableProducts");
         if ( request  =   "QueryAvailableProducts2Request");
            assert inp.CustNo  = "009999";
            assert inp.ArtType  = "BOOK";
         endRequest ;
         if ( response  =   "QueryAvailableProducts2Response");
           assert out.$ResponseTime < "1000";
           if ( object = "return" occurs = "1");
               if ( object = "item[1]");
                  assert out.ResponseArtNo  = "4711";
                  assert out.ResponseArtType  = "BOOK";
                  assert out.ResponseArtName  = "MeinKampf";
           endObject;
```

Validating a Web Service Response

WSDL Response Validation Report		
Tester: IWareHouseWebService		Date: 04.09.12
TestName: WSDL-Response	TestCase: 001	System: Orders
Non-Matching Params	**Non-Matching Values**	
Resp-Id: GetTypes1Response_001_return		
lst: item[2]	BOOK	
Soll:Asserted_Value	=NEWS	
lst: item[3]	0	
Soll:Asserted_Value	=BOOK	
lst: item[1]	_____	
Soll:Asserted_Value	=MAGA	
lst: item[2]	BOOK	
Soll:Asserted_Value	=NEWS	
Resp-Id: GetTypes1Response_001	missing from the old Response	
Total Number of old Responses:	08	
Number of old Responses matched to new ones:	06	
Total Number of Params checked:	18	
Total Number of non-Matching Params:	16	
Percentage of matching Params:	11 %	
Percentage of matching Responses:	80 %	

Web Service Testing Process



Steps of the Web Service Evaluation Process

1) Prepare the SRS
2) Analyse the SRS
3) Audit the WSDL
4) Edit the Test Cases and generate the Test Scripts
5) Compile the Test Scripts
6) Generate the Service Requests
7) Start the service Test
8) Validate the Service Results
9) Evaluate the Service Quality

Figure 7: Web Service Testing Process

Summary

- A structured semi-formal Requirement Specification can be used as a basis for web service testing
- The requirement document must follow a predefined structure and be marked with key words
- It is more costly to create but the costs are saved later.
- The logical test cases can be automatically generated from the natural language text.
- The test cases can lead to a compileable test script.
- The biggest remaining effort is for selecting meaningful test data values which fit the purpose of the test cases.