Service-Oriented Computing: Architecture, Programming, and Applications

A TUTORIAL PRESENTATION
by Dr. Yinong Chen
Department of Computer Science and Engineering

Roadmap

- Basic Concepts
- Ontology and Semantic Web
- Web Services
- Service Registry
- Application Building
- Case Studies

High level discussion
With full technical details

SEA 2005 Tutorial
Roadmap

- **Basic Concepts**
- Ontology and Semantic Web
- Web Services
- Service Registry
- Application Building
- Case Studies with technical details

Paradigms of Computing

![Diagram showing the evolution of programming paradigms over time](image-url)
Evolution and Revolution

Service-Oriented Programming

Object-Oriented Programming
- Abstract data types
- Encapsulation
- Inheritance
- Dynamic binding

Functional Programming
- Stateless
- Side-effect free
- Enforced modular design

Logic Programming
- Database
- Relations
- Query and matching
- Reasoning

Autonomous services
Remote objects
Component-based composition
Remote invocation and binding

Separation of development
Open standards and protocols
Platform-independent
Repository of reusable services
Automatic discovering and binding
Collaboration negotiation
Dynamic reconfiguration
Dynamic re-composition
Ontology-based reasoning

Impact of SOC Paradigm

- Programmers do not have to understand application logic
- Application builders do not have to write program
- Tougher but equal competition
- Shorter development cycle
- Better and more reliable software
- Different skill requirements:
  - Few programmers will be needed
  - More application builders will be needed
Definitions and Terminologies

- **Service-Oriented Architecture** (SOA): A software system consisting of a collection of loosely coupled services (components) that communicate with each other through standard interfaces and via standard message-exchanging protocols.
- **Service-Oriented Computing** (SOC) refers to the paradigm that represents computation in SOA.
- **Web Services** implement a Web-based SOA and a set of enabling technologies, including:
  - XML (eXtensible Markup Language) for data representation;
  - SOAP (Simple Object Access Protocol), enable remote invocation;
  - WSDL (Web Services Description Language);
  - OWL-S (Web Ontology Language for Services);
  - UDDI (Universal Description Discovery and Integration) for WS publication;
  - ebXML for service directory and service repository
- **Ontology**: A framework for database with semantic features: Classification, taxonomy, relation, constraints, etc.

Separation of Development

The Three-Party Model of Web Services

![Diagram: Three-Party Model of Web Services]

- Application builder
  - Application development platforms: .Net, WebSphere, WSFL, BPEL, PSML for composition, code generation
- Service brokers
  - Registry
  - Directory services: UDDI, WSDL, SOAP, ebXML, CPP
  - Ontology
  - Found
  - Find
  - Internet
  - Service providers
  - Service agents
  - SOAP call
  - Results
- White page
  - Yellow page
  - Green page
- Programming languages: C++, C#, Java
- Computing service development: .Net (Microsoft), WebSphere (IBM)
- Web and data service development: XML, RDF, OWL, WSDL
**SOC Technologies and Their Relations**

![Diagram showing relationships between service requesters, service directory, other web services, service agents, global ontology, local ontology, WSDL interface, IDE (.Net or WebSphere), programming languages, OWL, RDF and RDF Schema, XML and XML Schema, SOAP, and service directory (UDDI or ebXML).]

**Service-Oriented Enterprise (SOE)**

![Diagram showing composite electronic business applications, Web services, service-oriented enterprise (SOE), configurable business logic, ebSOA (E-Business SOA), SOM (Service-Oriented Management), SOI (Service-Oriented Infrastructure), hardware, and virtualized computing, memory, networking resources.]

---

*SEA 2005 Tutorial Y. Chen*
Roadmap

- Basic Concepts
- **Ontology and Semantic Web**
- Web Services
- Service Registry
- Application Building
- Case Studies

XML

XML (Extensible Markup Language) is a Meta language that can be used:

- To define other languages. Most languages and protocols in SOC paradigm are based on XML
  - RDF and RDF Scheme
  - WSDL
  - SOAP
- To describe data and attach semantic meanings to the data. Data transferred between Web services are encoded in XML
**HTML versus XML**

```html
<html>
<IASTED International Conference on Software Engineering and applications, November 14-16, 2005, Phoenix, AZ, USA</h3>
<li>Software design and development</li>
<li>Software engineering applications</li>
<li>Software security</li>
</html>

```xml
<?xml version="1.0">
<conference>
<IASTED International Conference on Software Engineering and applications</title>
<date>November 14-16, 2005</date>
<location>Phoenix , AZ, USA </location>
<keyword>Software design and development</keyword>
<keyword>Software engineering applications</keyword>
<keyword>Software security</keyword>
</conference>

```

### Differences between HTML and XML languages

<table>
<thead>
<tr>
<th></th>
<th>HTML</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose of tags</td>
<td>Formatting data for display.</td>
<td>Define attributes on data that can be interpreted by the users.</td>
</tr>
<tr>
<td>Syntax of tags</td>
<td>Tags may be left open, e.g., <code>&lt;br&gt;</code>.</td>
<td>All tags must be in pair.</td>
</tr>
<tr>
<td>Semantics of tags</td>
<td>Tags are predefined with given formatting meaning. Tags are limited.</td>
<td>The user can choose any tags and the meanings can be defined separately. Assigning meanings to a set of tags defines a vocabulary and thus a new markup language.</td>
</tr>
<tr>
<td>Semantics of data</td>
<td>It is difficult for machine to understand. For example, is Sedona a part of the title?</td>
<td>Once the tags are defined, it is easy for machine to understand. Sedona is a location, not a part of the title.</td>
</tr>
</tbody>
</table>
RDF: Resource Description Framework

RDF is used to describe resources on the Web. The key concepts of RDF are:

- A **resource** on the Web is identified by a URI (Universal Resource Identifier). URL (Universal Resource Locator) or Web address is an example of a URI.
- A **predicate** or **property** defines an attribute or a relation that describes a resource.
- A **statement** is a triple of (subject, predicate, and object).
- RDF is used to define ontologies or semantic Web

Semantic Web and Ontology

- **Semantic Web** is a vision for the future of the Web
  - Information is given explicit meaning
  - It is possible for Web services to automatically process and integrate information available on the Web.
- **Ontology** is a method of implementing semantic Web, which defines:
  - a vocabulary of terms (words)
  - their meanings (semantics),
  - their interconnections (synonym and antonym), and
  - rules of inference
Example of an Ontology

In RDFS (upper part):
- subClassOf
- range
- domain

In RDF and from RDF to RDFS:
- edge (property instance)
- type
- instance of class

Roadmap

- Basic Concepts
- Ontology and Semantic Web
- Web Services
- Service Registry
- Application Building
- Case Studies
Web Service (WS)

- Web Services are building blocks of SOC software
- Every WS has an standard interface, e.g., in WSDL
- Every WS can be remotely invoked by a standard protocol, e.g., SOAP – remote procedure call
- Web Services are platform-independent, it can be written in any languages: Java, C#, C++, etc.
- Every piece of program can be wrapped as a WS
- Every WS is placed in an internet-searchable repository
- In the near future, most services required will be available.
- There is no need to write new services in most cases
- The programming languages likes Java, C#, C++, etc will be less frequently used.

WSDL: Web Service Description Language

- WSDL is used to describe Web services, including four critical aspects of Web services:
  - Functionality of the services in standard taxonomy;
  - Data types for parameter values and return type of the function (service) calls;
  - Binding information about the transport protocol to be used, usually, SOAP;
  - Address information for locating the specified service.
- The last three aspects can be automatically generated.
- Web services described in WSDL can be searched, matched with the requirement.
- Web services described in WSDL provides the remote call detail.
SOAP: Simple Object Access Protocol

- SOAP protocol is used to transport messages between Web services
- A SOAP message is an XML document
- Like any communication protocol, a SOAP message consists of wrapper information and payload;
- A SOAP message is often wrapped in a HTTP (Hypertext Transfer Protocol) message and sent as an HTTP packet over the internet;
- The format of SOAP message is as follows:

```xml
<eve:Envelop>  
  <eve:header>  . . . </eve:header>  
  <eve:body>     .   .   .    </eve:body> </eve:Envelop>
```

Main Web Service Authoring Tools (1)

**C# Web Services on Visual Studio .Net**

- Define an ordinary class with data members and methods in C#;
- Deploy the class as Web services using .Net: each method will be a service;
- WSDL file will be generated automatically;
- URL of the Web services will be generated automatically;
- SOAP call interface will be generated automatically;
- There is little difference with writing a C# class
Main Web Service Authoring Tools (2)

Java Web Services

- Define an ordinary class with data members and methods in Java;
- Sun Microsystems does not directly offer the tool to covert Java class to Web services; There are different methods to wrap a Java class into a set Web services (generate WSDL and SOAP files)
  - Manually extract the required information and wrap them into WSDL and SOAP syntax
  - Community projects and third parties, e.g., Apache Tomcat tool, IBM WebSphere, etc.
  

Web Services are Wrapped Classes

- Service requester
  - Remote invocation in XML/SOAP
    - Function call
    - Return value in XML/SOAP

- Service agent
  - Functions written in C# or Java
    - Manually write the functions in C# or in Java
    - Automatically generate the WSDL file

Web service interfaces

To UDDI server

WSDL Description for UDDI publication

SEA 2005 Tutorial

Y. Chen
Roadmap

- Basic Concepts
- Ontology and Semantic Web
- Web Services
- **Service Registry**
- Application Building
- Case Studies

UDDI Service Registry

**Universal Description, Discovery, and Integration**

UDDI registry information is organized in three groups:

- **White page** includes service provider's name, identify, e.g., the DUNS number, contact information.
- **Yellow page** includes industry type, product and service type, and geographical location.
- **Green page** includes binding information associated with services, references to the technical models those services implement, and pointers to various file and URL-based discovery mechanisms. The information can be searched and interpreted by programs.
Search and Discover a Service

![Diagram of service discovery process]

Microsoft Enterprise UDDI Services

![Diagram of Microsoft Enterprise UDDI Services]

SEA 2005 Tutorial  
Y. Chen
IBM WSFL: Web Services Flow Language

WSFL uses the following models to specify applications:

- **Global Model** describes overall interaction among Web services and applications that use Web services, for example, describes the interaction between a client, a travel agency, airlines, and hotels.

- **Flow Model** describes a composite Web service or an application by describing the execution flow among the functions of the constituent Web services.

- **Nested Composition**: Usually, the global model is used to describe the behaviors between composite Web services, each of which is described by a flow model. However, WSFL allows a composite Web service created by a global model to be used as a constituent service in the composition of another Web service, leading to nested composition.
WSFL Global and Flow Model Examples

Example of a Global Model

Example of a Flow Model
ASU PSML-S
Process Modeling and Specification Language for Services

- Used to define static structure – the program
- Used to define dynamic behaviors – the processes
- Used to define other packages
- Used to define constraints in both structure and behavior packages
PSML-S Constructs in Structure Package

- **Actor**: is a model element that represents a system or a component. An actor is implemented as a service agent that can reside locally or remotely.
- **Action**: is a model element that represents an operational process of changing the internal status of an actor.
- **Attribute**: is defined to provide security, safety, performance, timing or reliability information. An attribute can be associated with any model element.
- **Condition**: is a predicate on data elements used to determine the course of a process taken by actors.
- **Data**: is an information carrier that represents the internal status of an actor.
- **Event**: is a model element that represents an observable occurrence with no time duration of input to an actor or output from an actor.
- **Relation**: is defined to provide relationship information between any model elements.

PSML-S Modeling Process

- Original Requirements
- Basic requirement analysis
  - Categorize non-functional and functional requirements
  - Itemize requirements

- Requirement formulation
  - Requirement decomposition
    - Structure model
      - Collaboration specification (functional)
      - Policy specification (non-functional)
      - Process specification (functional)

- PSML
  - Analyses
    - Static analysis
    - Dynamic analysis
Dynamic Analyses on PSML-S Models (Specifications)

PSML-S functional & policy models

Code generation
Reliability modeling
Policy enforcement
Reconfiguration and recomposition
Simulation

Model checking
Completeness & consistency
Dynamic testing
Collaboration negotiation

PSML-S based SOC Development Environment

Model checking
Completeness & consistency
Test case generation
PSML-S functional model
PSML-S policy model
Policy database
Policy revision

Complete feedback

Test cases
Testing
Simulation

Code generation
No
Yes
Pass?

Requirement revision

Automatic
Reconfiguration
Re-composition
Manual

Policy enforcement
Execution / Simulation
Data collection
Data mining

SEA 2005 Tutorial
Y. Chen
Roadmap

- Basic Concepts
- Ontology and Semantic Web
- Web Services
- Service Registry
- Application Building

**Case Studies and Examples**
- Getting Started with Microsoft ASP .Net and UDDI
- Examples

Visual Studio .Net-Based Development

**System Requirements**

1. Windows XP Professional, including IIS (Internet Information Service) Component
2. UDDI: Download at http://uddi.microsoft.com/

OR

1. Windows Server 2003 or later, including UDDI and IIS
2. Visual Studio .Net 2003 or later version
Development Scope and Overview

1. Local development/experiment on Windows XP. Install following software on your machine
   1) Use Windows XP CD to add IIS component
   2) Install Visual Studio .Net
   3) Install UDDI server software (standalone)

2. Intranet Development on Windows Server. Install following software on your machine
   • Install Windows Server 2003, including IIS and UDDI
   • Install Visual Studio .Net
   • Register services to the your own (private) UDDI

3. Global Development: Register your services to global UDDI

As a Service Provider

Develop Web Services Using .Net
Develop Web Services Using .Net

Add Your Functions into the Template

```csharp
/// Summary description for Service1.
///
/// The HelloWorld() example service returns the string Hello World.
///
/// To build, uncomment the following lines that are comments and build the
data.

[WebMethod] // Make the following function accessible over Web
public string HelloWorld() //
{ return "Hello World!"; }

[WebMethod] public double PiValue() { return 3.14159265358979; }

[WebMethod] public int f(int x) { if (x > 0) return x; else return -x; }
```
Compile and Execute the Program

The Web Service is Created

- 

This web service is using http://tempuri.org as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services in the Web. http://tempuri.org is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company’s Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not refer to actual resources on the Web. The namespace prefix used (ns1, ns2) is only local to the service namespace (ns1.org).

For XML web services created using ASP.NET, the default namespace can be changed using the WebService attribute’s namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to http://tempuri.org/WebServices/:

```csharp
WebService [Namespace="http://tempuri.org/WebServices/"]
public class MyWebService1 : // Implementation
```
You entered the service function

Service1
Click here for a complete list of operations.

abs
Text
To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter | Value
--- | ---
x | 1802

Invoke

WSDL Service Description

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the WSDL Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is suitable for WSDL Web services that are under development, but published WSDL Web services should use a more permanent namespace.

Your WSDL Web service should be identified by a namespace that you control. For example, you can use your company’s Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not refer to actual resources on the Internet. The namespace used here is "http://tempuri.org/"

For XML Web services created using ADF BPEL, the default namespace can be changed using the WebService attribute’s namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to http://tempuri.org/WebService:

class myWebService

public class MyWebService
{
    // Implementation
}
Make Your Services Available on Internet

- So far you are developing your Web services on your local machine.
- Repeat what you did on a server, e.g., a machine running Windows Server 2003: All you need to do is to create a project, paste the program into the program, compile and execute the program.
- The Web services will automatically launched and be given a URL. For example:

  http://webstrar.dhcp.asu.edu/webstrar/service1.asmx

  where webstrar is the domain name we choose for the Web services.
- You can register the Web service’s URL and its WSDL file to a service directory to make the services known to the public.

Please note that the code snippet shown is an example of a Web service definition, which is not directly related to the text described. The actual implementation details such as the URLs and service names should be handled according to the specific requirements and environments.
Sample SOAP/HTTP Call Code for Accessing your Web Service from a Program

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
             xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <absResponse xmlns="http://tempuri.org/">
      <absResult>int</absResult>
    </absResponse>
  </soap:Body>
</soap:Envelope>

Service Registry and Directory

Publish your Services

Find services published by other service providers
Free UDDI at: http://uddi.microsoft.com/

Create a new account
This is a manual process using a GUI

1. Publish a Service Provider
This is a manual process using a GUI

2. Publish your Services
This can be a manual process using a GUI or automatic process using a API

3. Search for Services
This can be a manual process using a GUI or automatic process using a API
Register a new account: A manual process

Publish a Provider and then Services
Publish a Service Provider

Make providers easy to locate by adding details, identifiers, and categorizations. Publish support contacts, links to additional information, or relationships with other providers by adding contacts, discovery URLs, and relationships.

<table>
<thead>
<tr>
<th>Details</th>
<th>Services</th>
<th>Contacts</th>
<th>Identifiers</th>
<th>Categories</th>
<th>Discovery URLs</th>
<th>Relationships</th>
</tr>
</thead>
</table>

A categorization scheme is a predefined set of categories, derived from an internal or external hierarchy, that can be used to classify a provider. Add one or more categories by selecting from available schemes. If an appropriate categorization is not available, contact a UDDI Services Coordinator.

Categorization Schemes:
- udml.org: schemes
- udml.org: relationships
- nist-gov: wss-1.0
- nsdl.org: wss-1.0
- nsdl.org: wss-1.0
- VS Web Service Search Categorization
- nist-gov: names:2002
- nsdl.org: wss-1.0
- nsdl.org: wss-1.0

1 record(s) found.

Publish a Service

A model usually represents a description of an interface. Make models easy to locate and reference by adding details, categories, and identifiers. Publish technical information, such as an interface specification or WSDL file, by pointing to one or more overview documents.

**Details**

- Name and briefly describe the interface (or other data) that this model represents. (The model key is unique and is intended for use in programmatic queries only.)
- Owner: cececo
- Model Key: uid:00216944-9c99-4799-a5c6-1dec206cde54

**Descriptions**

- Language: en-US
- Description: Provide text string
  - Compute A value
  - Compute absolute value

(*Maximum 200 characters; text only)

1 record(s) found.
Search for Services

This can be a manual process using a GUI or automatic process using a API

As an Application Builder

Develop Windows Applications or Web Applications Using .Net
Develop Applications Using .Net (on another computer)

Right-Click on the “Reference” and Choose “Add Web Reference”
Project Stack

Service1

The following operations are supported. For a formal definition, please review the Service Description:

- add
- PrintValue
- HelloWorld

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are being used on a development machine.

Web Form for Application Building
GUI Design for Application Building

Combinatorial Models
using edu.asu.dhcp.webstrar;

private void Button1_Click(object sender, System.EventArgs e) {
    Service1 hw = new Service1();
    this.Label1.Text = hw.HelloWorld();
}

private void Button2_Click(object sender, System.EventArgs e) {
    Service1 pivar = new Service1();
    this.Label2.Text = pivar.PiValue().ToString();
}

private void Button3_Click(object sender, System.EventArgs e) {
    Service1 absvar = new Service1();
    int number = Convert.ToInt32(this.TextBox1.Text);
    int result = absvar.abs(number);
    this.Label3.Text = result.ToString();
}

private void Button4_Click(object sender, System.EventArgs e) {
    int number = Convert.ToInt32(this.TextBox2.Text);
    Service1 absvar = new Service1();
    number = absvar.abs(number);
    double result = number + absvar.PiValue();
    this.Label4.Text = result.ToString();
}
Example 1: Recomposable Embedded Systems

Component Requirement

- Forward(n);
- Backward(n);
- TurnLeft(\theta); where, \theta = 45, 90, 135 degree
- TurnRight(\theta); where, \theta = 45, 90, 135 degree
- TurnBack(); turn 180 degree
- boolean Obstacle(); return true if there is an obstacle
Application Building Based on the Components (1)

A and B cooperate to try to move through the gate.
C tries to block them.
A and B will test C’s intelligence, and send the information back to the RMC.
RMC will construct programs for A and B.
A and B will follow the new program to bypass C.
Example 2: Service-Oriented Teaching Assistance

The Web Services-based application “Teaching Assistance” has the following features:

- Service providers to register new Web services;
- Instructors to enter test questions and solutions into an ontology to form a semantic Web;
- Instructors to build test papers based on the existing Web services;
- Give a test online and have the test automatically graded.

Teaching Assistance Ontology

A → B: To test B, A must be immediately before B
A → B: A must be fore B, if both A and B are selected
Web Services Manipulating the Ontology

- addTreeNode(subRoot, name);
- removeTreeNode(nodeName);
- addTreeLeave(subRoot, name);
- removeTreeLeave(leaveName);
- selectLeave(leaveName);
- takeTest(testName);
- gradeTest(testName, grade);
- enterGrade(roster, testName, grade);
- sort(roster);
- display(roster, range);
- login(userName, pwd);
- logout();

Application Building

- testPaper(subRoot, name);
  - login(userName, pwd);
  - selectLeaf(name1) … selectLeave(namen);
  - buildTest(testName);
  - logout();
- testGiving(testName);
  - login(userName, pwd);
  - takeTest(testName);
  - gradeTest(testName, grade);
  - enterGrade(roster, testName, grade);
  - logout();
- reportGrade(roster, key1, key2, key3);
  - login(userName, pwd);
  - sort(roster);
  - display(roster, range);
  - logout();
SUMMARY

- Basic Concepts
- Ontology and Semantic Web: XML, RDF, ontology example
- Web Services: WSDL, SOAP, Java Web services, C# Web services
- Service Registry: Principles and UDDI
- Application Building: IBM WSFL, Microsoft Whitehorse, ASU PSML-S

Case Studies and Examples
- Getting Started with ASP.Net and UDDI
- Use ASP.Net to develop C# based Web Services
- Use UDDI to publish service providers and Web services
- Use ASP.Net to develop Web services based applications
- Examples: Recomposable Embedded Systems and SOA-based Teaching Assistance

References

### Part I: Fundamentals of Service-Oriented Computing
- **Introduction to Service-Oriented Computing**
  - Chapter 3
- **Essentials of Web Services**
  - Chapter 2
- **Building Applications based on Web Services**
  - Chapter 3
- **Infrastructure for Collaborative WS Development**
  - Chapter 4

### Part II: Dynamic Design and Simulation of Service-Oriented Systems
- **Dynamic Recomposition Service**
  - Chapter 5
- **Service-Oriented Simulation**
  - Chapter 6
- **Policy Specification and Enforcement**
  - Chapter 7
- **Cooperation & Interoperability**
  - Chapter 8

### Part III: Service-Oriented Testing and Evaluation Techniques
- **Service-Oriented Testing**
  - Chapter 9
- **Model Checking & Test Generation**
  - Chapter 10
- **Web Services Group Testing**
  - Chapter 11
- **Faults, Failures and Reliability Modeling**
  - Chapter 12

### Part IV: Applications
- **Embedded System**
  - Chapter 13
- **Process Control**
  - Chapter 14
- **C2 System**
  - Chapter 15
- **E-Commerce**
  - Chapter 16
- **Enterprise Computing**
  - Chapter 17
- **Virtual Ontology**
  - Chapter 18

---

**About the Tutorial Presenter**

- PhD from the University of Karlsruhe, Germany, in 1993.
- Postdoctoral research at LAAS-CNRS, France in 1995 and 1996.
- Lecturer and senior lecturer at Wits University at Johannesburg from 1994 to 2000.
- Lecturer at Arizona State University from 2001 to 2004
- Senior research scientist at Arizona State University from 2004

Dr. Chen has taught a wide range of computer science and engineering courses in the past decade. He is the author of the text, Introduction to Programming Languages, published by Kendall/Hunt in 2003 and a coauthor of the text, Service-Oriented Computing and System Engineering, to be published in 2006.

Dr. Chen's research areas include dependable computing, software engineering, and service-oriented computing. He has coauthored five books and over 80 research papers, 20 of which are on service-oriented computing and were published in 2005. More information about the instructor can be found at the Web page: [http://www.public.asu.edu/~ychen10/](http://www.public.asu.edu/~ychen10/)