CSC301: Introduction to Software Engineering

Lecture 10

Wael Aboulsaadat

Revisiting OOD: cohesion and coupling

Increase cohesion where possible

- A subsystem or module has high cohesion if it keeps together things that are related to each other, and keeps out other things
  - This makes the system as a whole easier to understand and change
  - Type of cohesion:
    - Functional, Layer, Communicational, Sequential, Procedural, Temporal, Utility

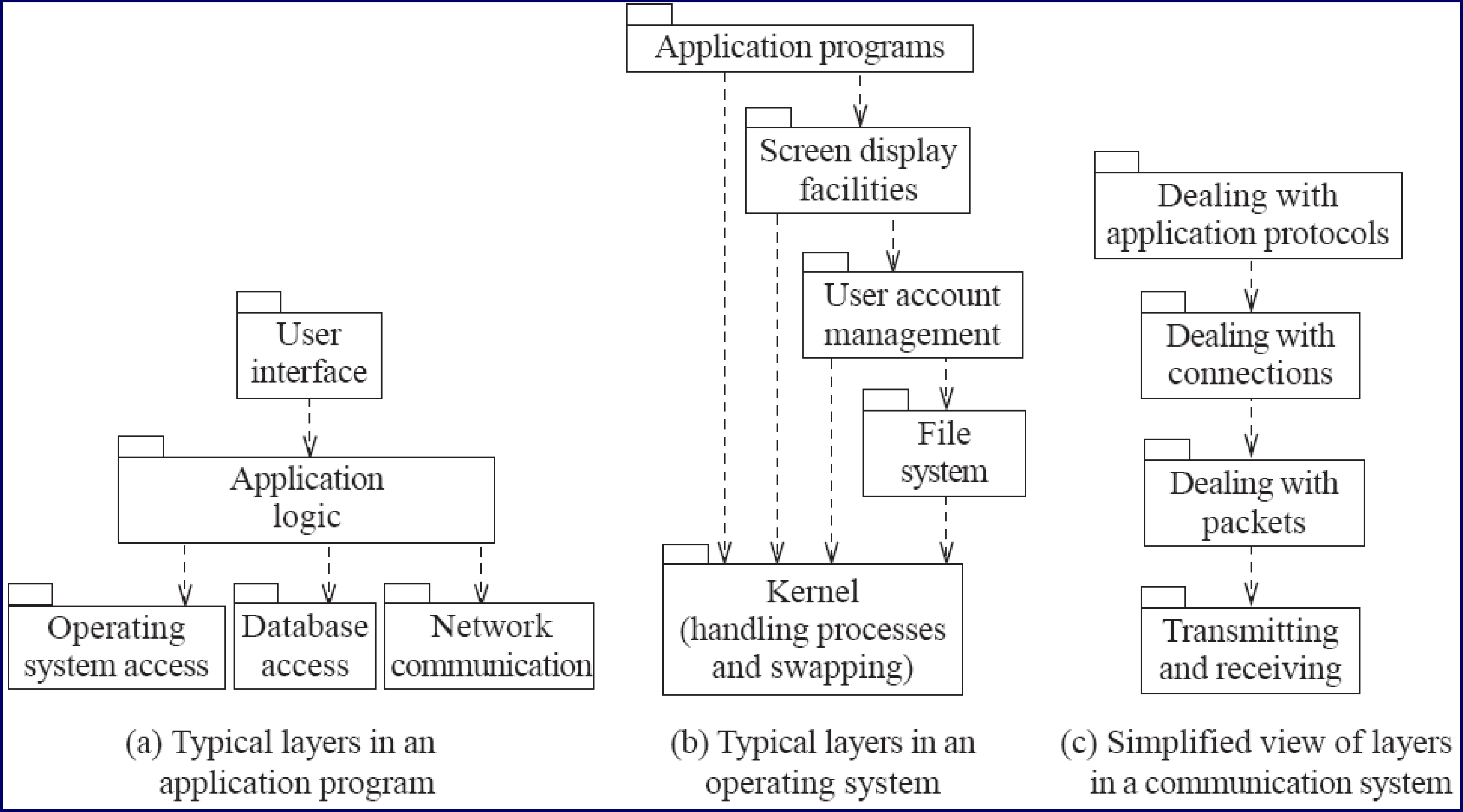
Functional cohesion

- This is achieved when *all the code that computes a particular result* is kept together - and everything else is kept out
  - i.e. when a module only performs a *single* computation, and returns a result, *without having side-effects*.
  - Benefits to the system:
    - Easier to understand
    - More reusable
    - Easier to replace
  - Modules that update a database, create a new file or interact with the user are not functionally cohesive

Layer cohesion

- All the *facilities for providing or accessing a set of related services* are kept together, and everything else is kept out
  - The layers should form a hierarchy
    - Higher layers can access services of lower layers,
    - Lower layers do not access higher layers
  - The set of procedures through which a layer provides its services is the *application programming interface (API)*
  - You can replace a layer without having any impact on the other layers
    - You just replicate the API

Example of the use of layers


User interface
Application logic
Operating system access
Database access
Network communication
(a) Typical layers in an application program
Application programs
Screen display facilities
User account management
File system
Kernel (handling processes and swapping)
(b) Typical layers in an operating system
Dealing with application protocols
Dealing with connections
Dealing with packets
Transmitting and receiving
(c) Simplified view of layers in a communication system

Communicational cohesion

- All the *modules that access or manipulate certain data* are kept together (e.g. in the same class) - and everything else is kept out
  - A class would have good communicational cohesion
    - if all the system's facilities for storing and manipulating its data are contained in this class.
    - if the class does not do anything other than manage its data.
  - Main advantage: When you need to make changes to the data, you find all the code in one place

Sequential cohesion

- *Procedures, in which one procedure provides input to the next*, are kept together – and everything else is kept out
  - You should achieve sequential cohesion, only once you have already achieved the preceding types of cohesion.

Procedural cohesion

- *Procedures that are used one after another* are kept together
  - Even if one does not necessarily provide input to the next.
  - Weaker than sequential cohesion.

Temporal Cohesion

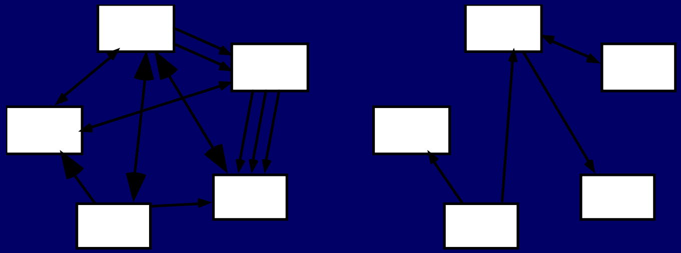
- *Operations that are performed during the same phase of the execution* of the program are kept together, and everything else is kept out
  - For example, placing together the code used during system start-up or initialization.
  - Weaker than procedural cohesion.

Utility cohesion

- When *related utilities which cannot be logically placed in other cohesive units* are kept together
  - A utility is a procedure or class that has wide applicability to many different subsystems and is designed to be reusable.
  - For example, the **java.lang.Math** class.

Reduce coupling where possible

- *Coupling* occurs when there are *interdependencies* between one module and another
  - When interdependencies exist, changes in one place will require changes somewhere else.
  - A network of interdependencies makes it hard to see at a glance how some component works.
  - Type of coupling:
    - Content, Common, Control, Stamp, Data, Routine Call, Type use, Inclusion/Import, External



Content coupling:

- Occurs when one component *surreptitiously* modifies data that is *internal* to another component
  - To reduce content coupling you should therefore *encapsulate* all instance variables
    - declare them private
    - and provide get and set methods
  - A worse form of content coupling occurs when you directly modify an instance variable *of* an instance variable

Example of content coupling

```
public class Line
{
  private Point start, end;
  ...
  public Point getStart() { return start; }
  public Point getEnd()  { return end; }
}

public class Arch
{
  private Line baseline;
  ...
  void slant(int newY)
  {
    Point theEnd = baseline.getEnd();
    theEnd.setLocation(theEnd.getX(),newY);
  }
}
```

Common coupling

- Occurs whenever you use a *global variable*
  - All the components using the global variable become coupled to each other
  - A weaker form of common coupling is when a variable can be accessed by a *subset* of the system’s classes
    - e.g. a Java package
  - Can be acceptable for creating global variables that represent system-wide default values
  - The Singleton pattern provides encapsulated global access to an object

Control coupling

- Occurs when one procedure calls another using *a 'flag' or 'command'* that explicitly controls what the second procedure does
  - To make a change you have to change both the calling and called method
  - The use of polymorphic operations is normally the best way to avoid control coupling
  - One way to reduce the control coupling could be to have a *look-up table*
    - commands are then mapped to a method that should be called when that command is issued

Example of control coupling

```
public routineX(String command)
{
  if (command.equals("drawCircle")
  {
    drawCircle();
  }
  else
  {
    drawRectangle();
  }
}
```

Stamp coupling:

- Occurs whenever one of your application classes is declared as the *type* of a method argument
  - Since one class now uses the other, changing the system becomes harder
    - Reusing one class requires reusing the other
  - Two ways to reduce stamp coupling,
    - using an interface as the argument type
    - passing simple variables

Example of stamp coupling

```
public class Emailer
{
  public void sendEmail(Employee e, String text)
  {...}
  ...
}
```

Using simple data types to avoid it:

```
public class Emailer
{
  public void sendEmail(String name, String email, String text)
  {...}
  ...
}
```

Example of stamp coupling

Using an interface to avoid it:

```
public interface Addressee
{
  public abstract String getName();
  public abstract String getEmail();
}

public class Employee implements Addressee {…}

public class Emailer
{
  public void sendEmail(Addressee e, String text)
  {...}
  ...
}
```

Data coupling

- Occurs whenever the types of method arguments are either primitive or else simple library classes
  - The more arguments a method has, the higher the coupling
    - All methods that use the method must pass all the arguments
  - You should reduce coupling by not giving methods unnecessary arguments
  - There is a trade-off between data coupling and stamp coupling
    - Increasing one often decreases the other

Routine call coupling

- Occurs when one routine (or method in an object oriented system) calls another
  - The routines are coupled because they depend on each other’s behaviour
  - Routine call coupling is always present in any system.
  - If you repetitively use a sequence of two or more methods to compute something
    - then you can reduce routine call coupling by writing a single routine that encapsulates the sequence.

Type use coupling

- Occurs when a module uses a data type defined in another module
  - It occurs any time a class declares an instance variable or a local variable as having another class for its type.
  - The consequence of type use coupling is that if the type definition changes, then the users of the type may have to change
  - Always declare the type of a variable to be the most general possible class or interface that contains the required operations

Inclusion or import coupling

- Occurs when one component imports a package
  - (as in Java)
- or when one component includes another
  - (as in C++).
  - The including or importing component is now exposed to everything in the included or imported component.
  - If the included/imported component changes something or adds something.
    - This may raises a conflict with something in the includer, forcing the includer to change.
  - An item in an imported component might have the same name as something you have already defined.

External coupling

- When a module has a dependency on such things as the operating system, shared libraries or the hardware
  - It is best to reduce the number of places in the code where such dependencies exist.
  - The Façade design pattern can reduce external coupling

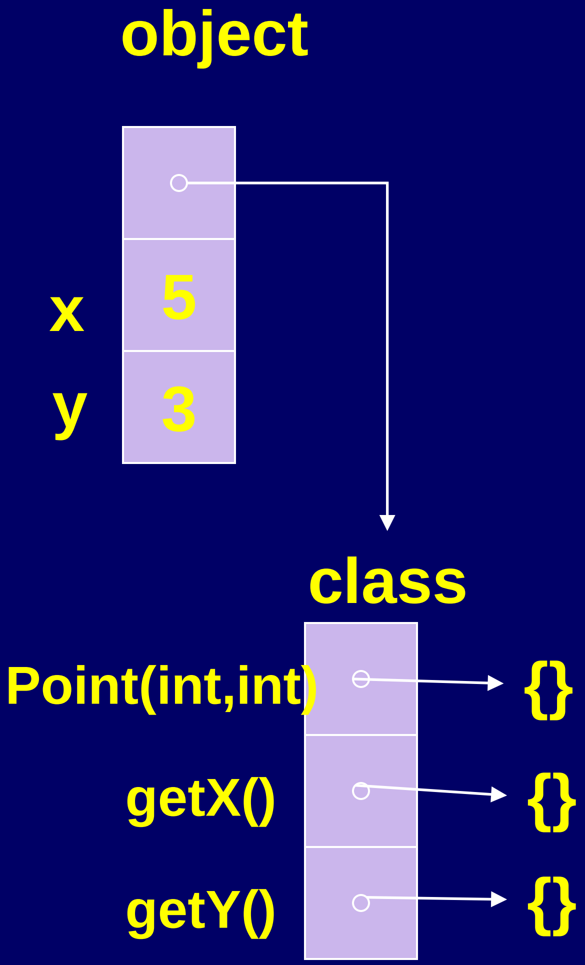
Revisiting OOD: cohesion and coupling and the Use of Reflection

Background

- Turing's great insight: programs are just another kind of data
  - Source code is text
  - Manipulate it line by line, or by parsing expressions
- Compiled programs are data, too
  - Integers and strings are bytes in memory that you interpret a certain way
  - Instructions in methods are just bytes too
- No reason why a program can't inspect itself

How objects work

```
class Point {
  public Point(int x, int y) {
    x = 5; Y = 10;
  }
  public int getX() {
         return x;
  }
  public int getY() {
         return y;
   }
  protected int x, y;
}
```



The class Class

- Instances of the class Class store information about classes
  - Class name
  - Inheritance
  - Interfaces implemented
  - Methods, members, etc.

- Can look up instances:
  - By name
  - From an object

http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Class.html

Showing a type

```
public static void showType(PrintStream out,
                                                    String className)
            throws ClassNotFoundException
{
  Class thisClass = Class.forName(className);
  String flavour    =  thisClass.isInterface() ? "interface" : "class";
  out.println(flavour + " " + className);
  Class parentClass = thisClass.getSuperclass();
  if (parentClass != null) {
    out.println(" extends " + parentClass.getName());
  }
  Class[] interfaces = thisClass.getInterfaces();
  for (Class interf : interfaces) {
    out.println("  implements " + interf.getName());
  }
}
```

Output for type example

```
class java.lang.Object

class java.util.HashMap
 extends java.util.AbstractMap
  implements java.util.Map
  implements java.lang.Cloneable
  implements java.io.Serializable

class Point
 extends java.lang.Object
```

Examining class contents

```
public static void showContents(PrintStream out,
                                boolean hideObject,
                                String name)
    throws ClassNotFoundException {

  Class cls = Class.forName(name);

  out.println(name);

  showMembers(out, hideObject, name + " fields",    cls.getFields());

  showMembers(out, hideObject, name + " constructors",cls.getConstructors());

  showMembers(out, hideObject, name + " methods",     cls.getMethods());
}
```

Examining class contents

```
public static void showMembers( PrintStream out,
                                boolean hideObject,
                                String title,
                                Member[] members) {
  out.println("  " + title);
  for (Member mem : members) {
    if (mem.getDeclaringClass() == Object.class) {
      if (hideObject) {
        continue;
      }
    }
    out.println("\t" + mem);
  }
}
```

Point *(somewhat edited)*

```
 Point fields
 Point constructors
     public Point(java.lang.String,int,int)
    public Point(int,int)

Point methods
    public java.lang.String Point.toString()
    public java.lang.String Point.getName()
    public void Point.setName(java.lang.String)
    public int Point.getX()
    public void Point.setX(int)
    public int Point.getY()
    public void Point.setY(int)
```

Getting at members

- How to access members of a specific object?
  - Without making raw pointers into memory part of the language
    - (Raw pointers are a rich source of errors in C/C++)

- Introduce a class Field
  - Encapsulates access to a particular field of instances of a class
  - Knows "where the field is" in objects of that class
  - Use its get() and set() methods to inspect and modify the object

Examining fields

```
public static void main(String[] args) {
    PublicPoint p = new PublicPoint("center", 3, 3);

    showField(System.out, p, "fName");
    showField(System.out, p, "fX");
    showField(System.out, p, "fY");
    showField(System.out, p, "fZ");
}
```

```
public static void showField(PrintStream out,
           Object obj, String fieldName) {
    try {
        Class cls    = obj.getClass();
        Field field  = cls.getField(fieldName);
        Object value = field.get(obj);
        out.println(fieldName + ": " + value);
    }
    catch (NoSuchFieldException e) {
        System.err.println(e);
    }
    catch (IllegalAccessException e) {
        System.err.println(e);
    }
}
```

Output

- fName: center
- fX: 3
- fY: 3
- java.lang.NoSuchFieldException: fZ

```
public static void showMethods(
                                PrintStream out, Object obj)
          throws NoSuchMethodException,
             IllegalAccessException,
                  InvocationTargetException {

  Class cls = obj.getClass();
  out.println(cls.getName());
  for (Method meth : cls.getMethods()) {
    if (meth.getDeclaringClass() == cls) {
      showMethod(out, meth);
    }
  }
}
```

Calling methods

1. Look up a method based on its signature: the name and list of parameter types

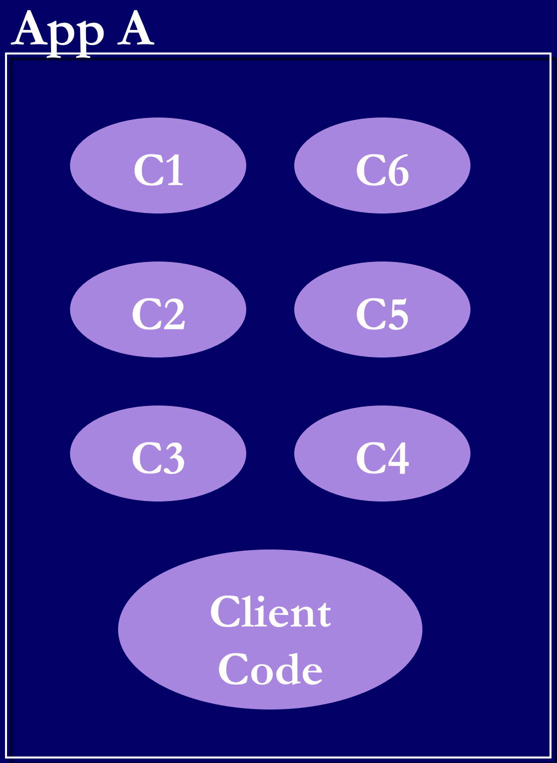
2. Specify signature as a comma-separated list of Class objects
   - Specifies the types of arguments
   - Special values for types like int and boolean

3. Call the method, passing in parameters and capturing return value

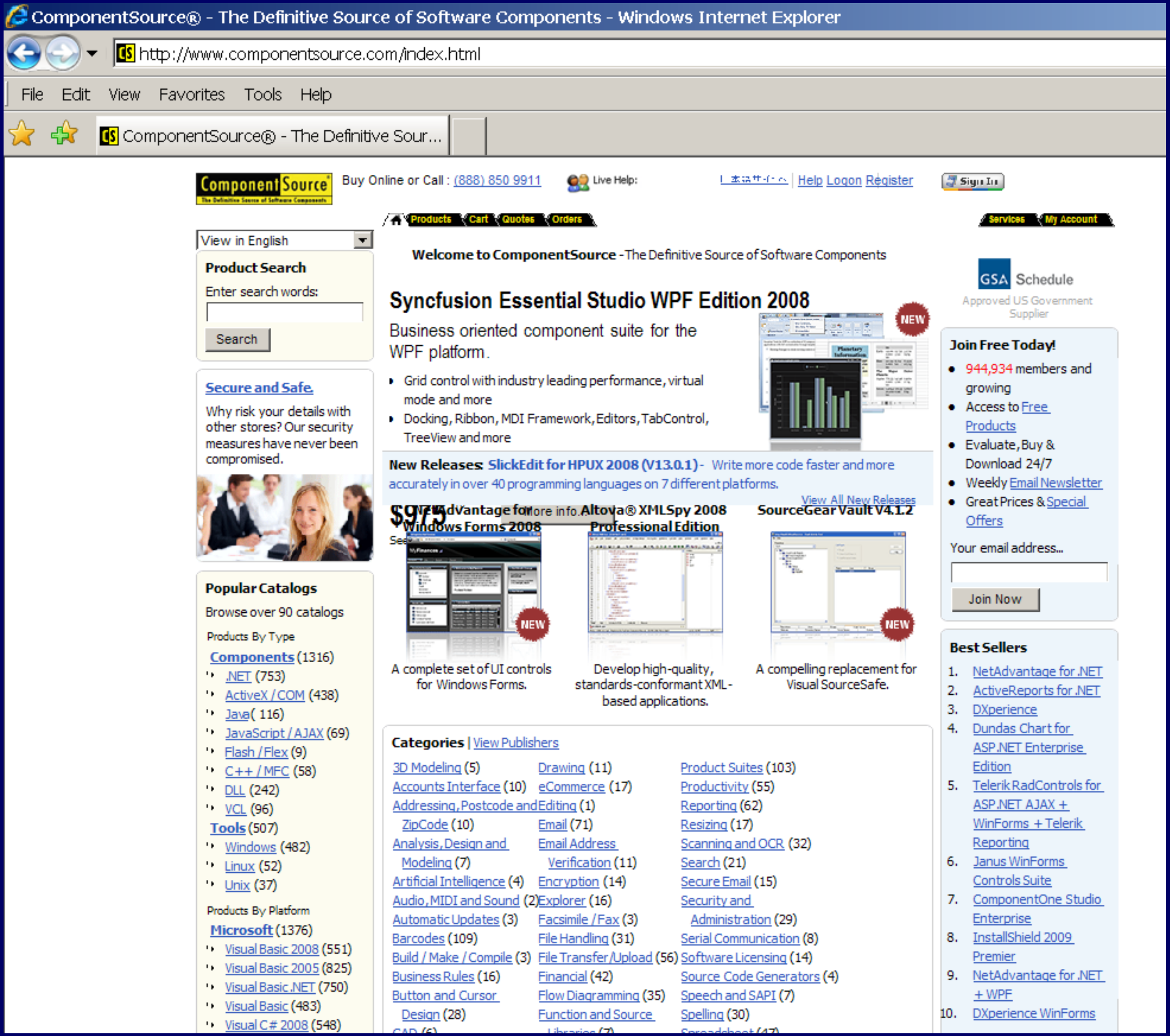
```
Class mysClass = Class.forName("Mystery");
Object o = mysClass.newInstance();

Method m = mysClass.getMethod("euclidean",
  Double.TYPE, Double.TYPE);
double result = (Double) m.invoke(o,
 new Double(5.0), 12.0);

Method m2 = mysClass.getMethod("play",
    Class.forName("java.lang.String"), String.class);

m2.invoke(o, "Che", "Karl");
```

Revisiting OOD: cohesion and coupling and the Use of Components

Component-based Software



CSC301: Introduction to Software Engineering



ComponentSource® - The Definitive Source of Software Components - Windows Internet Explorer

http://www.componentsource.com/index.html

File Edit View Favorites Tools Help

ComponentSource® - The Definitive Sour...

Buy Online or Call : (888) 850 9911 Live Help: Help Logon Register Sign In

Products Cart Quotes Orders Services My Account

View in English

Product Search

Enter search words:

Search

Secure and Safe

Why risk your details with other stores? Our security measures have never been compromised.

Popular Catalogs

Browse over 90 catalogs

Products By Type

- Components (1316)
  - .NET (753)
  - ActiveX / COM (438)
  - Java (116)
  - JavaScript / AJAX (69)
  - Flash / Flex (9)
  - C++ / MFC (58)
  - DLL (242)
  - VCL (96)
- Tools (507)
  - Windows (482)
  - Linux (52)
  - Unix (37)

Products By Platform

- Microsoft (1376)
  - Visual Basic 2008 (551)
  - Visual Basic 2005 (825)
  - Visual Basic .NET (750)
  - Visual Basic (483)
  - Visual C# 2008 (548)

Welcome to ComponentSource - The Definitive Source of Software Components

Syncfusion Essential Studio WPF Edition 2008 NEW

Business oriented component suite for the WPF platform.

- Grid control with industry leading performance, virtual mode and more
- Docking, Ribbon, MDI Framework, Editors, TabControl, TreeView and more

New Releases: SlickEdit for HPUX 2008 (V13.0.1) - Write more code faster and more accurately in over 40 programming languages on 7 different platforms.

View All New Releases

$975 NetAdvantage for Windows Forms 2008 — More info — NEW — A complete set of UI controls for Windows Forms.

Altova® XMLSpy 2008 Professional Edition — Develop high-quality, standards-conformant XML-based applications.

SourceGear Vault V4.1.2 — NEW — A compelling replacement for Visual SourceSafe.

Categories | View Publishers

3D Modeling (5)	Drawing (11)	Product Suites (103)
Accounts Interface (10)	eCommerce (17)	Productivity (55)
Addressing, Postcode and ZipCode (10)	Editing (1)	Reporting (62)
	Email (71)	Resizing (17)
Analysis, Design and Modeling (7)	Email Address Verification (11)	Scanning and OCR (32)
		Search (21)
Artificial Intelligence (4)	Encryption (14)	Secure Email (15)
Audio, MIDI and Sound (2)	Explorer (16)	Security and Administration (29)
Automatic Updates (3)	Facsimile / Fax (3)	
Barcodes (109)	File Handling (31)	Serial Communication (8)
Build / Make / Compile (3)	File Transfer /Upload (56)	Software Licensing (14)
Business Rules (16)	Financial (42)	Source Code Generators (4)
Button and Cursor Design (28)	Flow Diagramming (35)	Speech and SAPI (7)
	Function and Source	Spelling (30)

GSA Schedule

Approved US Government Supplier

Join Free Today!

- 944,934 members and growing
- Access to Free Products
- Evaluate, Buy & Download 24/7
- Weekly Email Newsletter
- Great Prices & Special Offers

Your email address...

Join Now

Best Sellers

1. NetAdvantage for .NET
2. ActiveReports for .NET
3. DXperience
4. Dundas Chart for ASP.NET Enterprise Edition
5. Telerik RadControls for ASP.NET AJAX + WinForms + Telerik Reporting
6. Janus WinForms Controls Suite
7. ComponentOne Studio Enterprise
8. InstallShield 2009 Premier
9. NetAdvantage for .NET + WPF
10. DXperience WinForms

Component standards

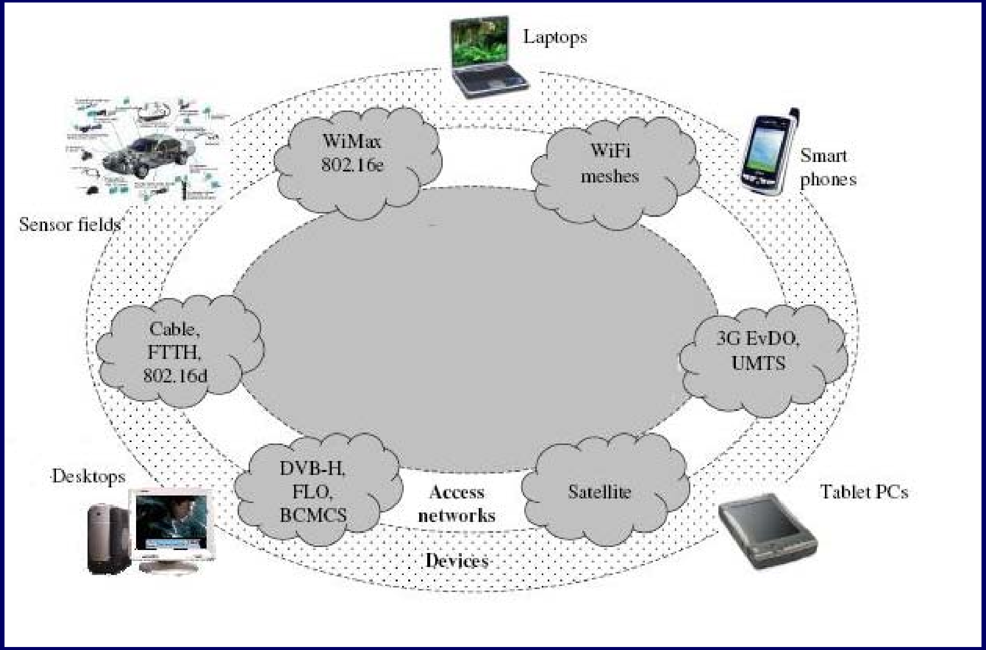
- Sun: JavaBeans
- Microsoft: ActiveX, COM, DCOM

Java Beans

- Reusable components for visual programming
  - Standard constructor
    - No arguments
  - Standard ways of editing parameters
    - Set methods
    - Property editors
    - Customizers
  - Standardized event handling
    - Including property change notification and control

Revisiting OOD: cohesion and coupling and the Use of XML

How can we exchange data between heterogeneous systems?



Message in the Bottle (or: towards the Digital Rosetta Stone)

Degree of "self-description":

not quite

```
^@Some Quotations from the Universal Library^M1
Famous Quotes^M1.1 By William I^M[2,
Sonnet XVIII]^MShall I compare thee to a
summer's day?^MThou art more lovely and
more temperate.^MRough winds do shake the
darling buds of May,^MAnd summer's lease
hath all too short a date.^MSometime too hot
the eye of heaven shines,^MAnd often is his
gold complexion dimmed.^MAnd every fair
from fair some declines,^MBy chance or
nature's changing course untrimmed.^MBut thy
eternal summer shall not fade,^MNor lose
possession of that fair thou owest,^MNor shall
Death brag thou wander'st in his
shade^MWhile in eternal lines to time thou
growest.^MSo long as men can breathe, or
eyes can see,^MSo long live this, and this
gives life to thee.^M1.2 By William II^M[1,
p.265]^M\223The obvious mathematical
breakthrough would be development of^Man
easy way to factor large prime
numbers."^MReferences^M[1] W. H. Gates.
The Road Ahead. Viking Penguin, 1995.^M[2]
W. Shakespeare. The Sonnets of
Shakespeare.609.^M^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@
```

Two Important Ideas: (1) Markup?

- Information added to a text to make its structure comprehensible
- Pre-computer markup (punctuational and presentational)

Two Important Ideas: (2) declarative

- Names and structure
- Finer level of detail (most human-legible signals are overloaded)
- Independent of presentation (abstract)
- People often call this "semantic"

Message in the Bottle (or: towards the Digital Rosetta Stone)

Degree of "self-description"

not quite

```
^@Some Quotations from the Universal Library^M1 Famous Quotes^M1.1 By William I^M[2, Sonnet XVIII]^MShall I compare thee to a summer's day?^MThou art more lovely and more temperate.^MRough winds do shake the darling buds of May,^MAnd summer's lease hath all too short a date.^MSometime too hot the eye of heaven shines,^MAnd often is his gold complexion dimmed.^MAnd every fair from fair some declines,^MBy chance or nature's changing course untrimmed.^MBut thy eternal summer shall not fade,^MNor lose possession of that fair thou owest,^MNor shall Death brag thou wander'st in his shade^MWhile in eternal lines to time thou growest.^MSo long as men can breathe, or eyes can see,^MSo long live this, and this gives life to thee.^M1.2 By William II^M[1, p.265]^M\223The obvious mathematical breakthrough would be development of^Man easy way to factor large prime numbers."^MReferences^M[1] W. H. Gates. The Road Ahead. Viking Penguin, 1995.^M[2] W. Shakespeare. The Sonnets of Shakespeare.609.^M^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
```

not bad

```
\documentclass{article}
 \begin{document}
 \title{Some Quotations from the
        Universal Library}
...
\section{Famous Quotes}
 \subsection{By William I}
 \textbf{\cite[Sonnet
        XVIII]{shakespeare-sonnets-
        1609}}
   \begin{verse}
   Shall I compare thee to a summer's
        day?\\
   Thou art more lovely and more
        temperate. \\
   Rough winds do shake the darling buds
        of May, \\
   And summer's lease hath all too short a
        date. \\
   Sometime too hot the eye of heaven
        shines, \\
   And often is his gold complexion
        dimmed. \\
  ...
    \qquad So long as men can breathe, or
        eyes can see,\\
    \qquad So long live this, and this gives
        life to thee.   \\
  \end{verse}
...
 \bibliographystyle{abbrv}
\bibliography{msg}

\end{document}
```

XML: Basic format

1) *Element*: **`<tag>`**`content`**`</tag>`**
   - basic unit
   - tag name defines what the content is
   - opening and closing tags enclose content

2) *Attribute*: Information about the data
   - Attribute names are usually adjectives
   - Stored as `attribute="value"` pairs:
     - `<tag` **`attribute="value">`**
     - `      content`
     - `</tag>`

Rules for well-formed XML

- Elements that contain data must have <start> and </end> tags!
- Empty tags must be closed `<some-tag/>`
- Elements should not overlap

  Bad Nesting:

  `<trunk> <branch> </trunk> </branch>`
- All attribute values must be wrapped in quotes

  `<a href="newpage.html">`
- XML is case sensitive: `<TAG>` and `<Tag>` are treated differently. (Standard: use lower case.)

More XML Rules

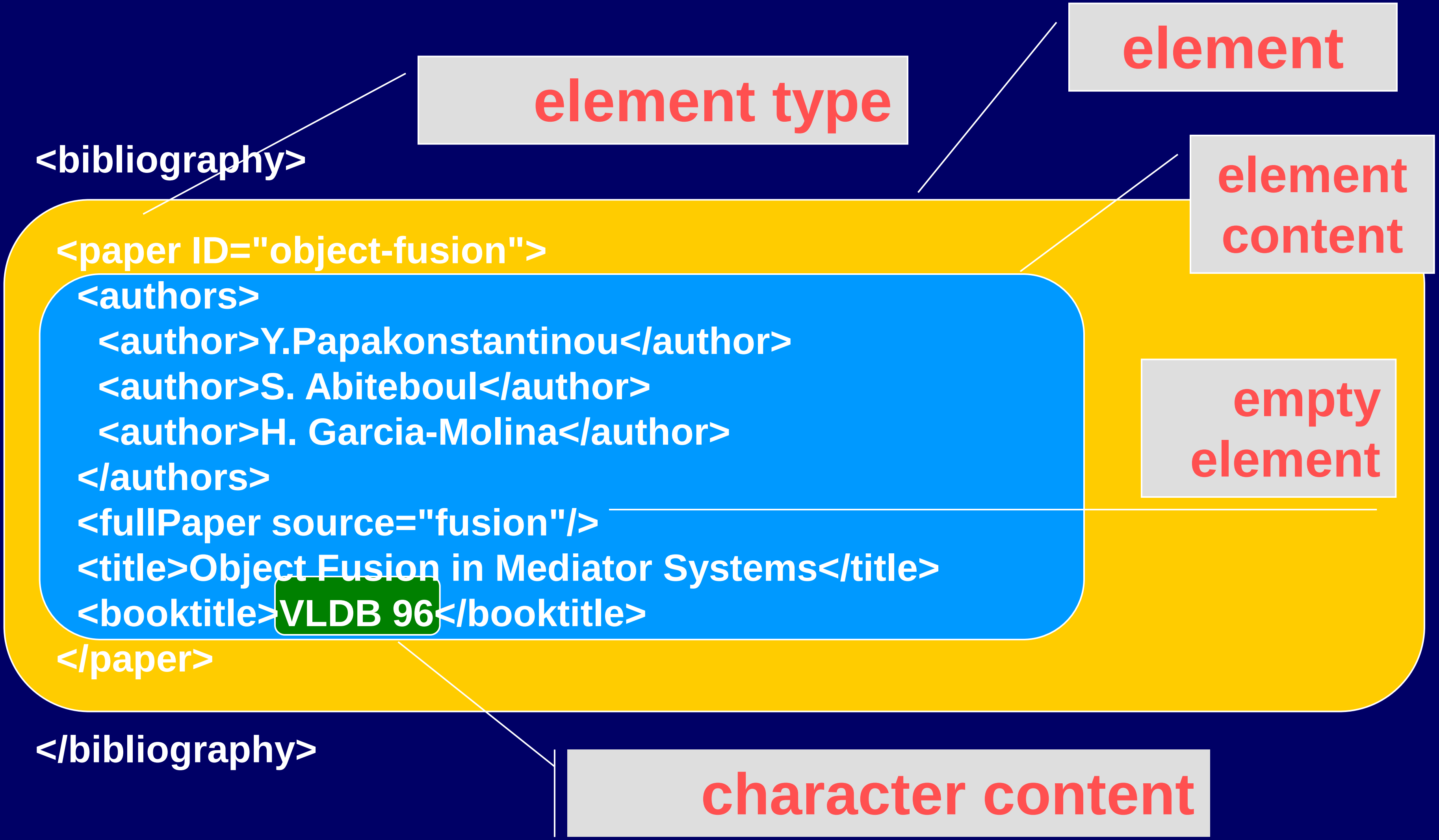
- A document beginning with:
  - an ***XML Declaration***

    ```
    <?xml version="1.0" encoding="UTF-8"?>
    ```

- ***Root element*** immediately follows; encloses entire content of the document.

  ```
      <book>
          everything else
      </book>
  ```

Elements and their Content



```
<bibliography>
  <paper ID="object-fusion">
    <authors>
      <author>Y.Papakonstantinou</author>
      <author>S. Abiteboul</author>
      <author>H. Garcia-Molina</author>
    </authors>
    <fullPaper source="fusion"/>
    <title>Object Fusion in Mediator Systems</title>
    <booktitle>VLDB 96</booktitle>
  </paper>
</bibliography>
```

XML Example: content objects in a book

- Book
  - FrontMatter
    - BookTitle
    - Author(s)
    - PubInfo
  - Chapter(s)
    - ChapterTitle
    - Paragraph(s)
  - BackMatter
    - References
    - Index

A simple XML fragment

```
<Book>
  <FrontMatter>
    <BookTitle>XML Is Easy</BookTitle>
    <Author>Tim Cole</Author>
    <Author>Tom Habing</Author>
    <PubInfo>CDP Press, 2002</PubInfo>
  </FrontMatter>
  <Chapter>
    <ChapterTitle>First Was SGML</ChapterTitle>
    <Paragraph>Once upon a time …</Paragraph>
  </Chapter>
</Book>
```

This is NOT XML, why?

```
<PoemFragment>
  <Stanza>
    <Line><Sentence>It was six men of Indostan</Line>
    <Line>To learning much inclined,</Line>
    <Line>Who went to see the Elephant</Line>
    <Line>(Though all of them were blind),</Line>
    <Line>That each by observation</Line>
    <Line>Might satisfy his mind </Sentence></Line>
  </Stanza>
</PoemFragment>
```

Message in the Bottle (or: towards the Digital Rosetta Stone)

Degree of "self-description":

not quite — *not bad* — *pretty good*

not quite

```
^@Some Quotations from the Universal Library^M1 Famous Quotes^M1.1 By William I^M[2, Sonnet XVIII]^MShall I compare thee to a summer's day?^MThou art more lovely and more temperate.^MRough winds do shake the darling buds of May,^MAnd summer's lease hath all too short a date.^MSometime too hot the eye of heaven shines,^MAnd often is his gold complexion dimmed.^MAnd every fair from fair some declines,^MBy chance or nature's changing course untrimmed.^MBut thy eternal summer shall not fade,^MNor lose possession of that fair thou owest,^MNor shall Death brag thou wander'st in his shade^MWhile in eternal lines to time thou growest.^MSo long as men can breathe, or eyes can see,^MSo long live this, and this gives life to thee.^M1.2 By William II^M[1, p.265]^M\223The obvious mathematical breakthrough would be development of^Man easy way to factor large prime numbers."^MReferences^M[1] W. H. Gates. The Road Ahead. Viking Penguin, 1995.^M[2] W. Shakespeare. The Sonnets of Shakespeare.609.^M^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
```

not bad

```
\documentclass{article}
 \begin{document}
 \title{Some Quotations from the
        Universal Library}
...
\section{Famous Quotes}
 \subsection{By William I}
 \textbf{\cite[Sonnet
        XVIII]{shakespeare-sonnets-
        1609}}
   \begin{verse}
    Shall I compare thee to a summer's
        day?\\
    Thou art more lovely and more
        temperate. \\
    Rough winds do shake the darling buds
        of May, \\
    And summer's lease hath all too short a
        date. \\
    Sometime too hot the eye of heaven
        shines, \\
    And often is his gold complexion
        dimmed. \\
   ...
     \qquad So long as men can breathe, or
        eyes can see,\\
     \qquad So long live this, and this gives
        life to thee.   \\
  \end{verse}
...
 \bibliographystyle{abbrv}
\bibliography{msg}

\end{document}
```

pretty good

```
<?xml version="1.0"?>
<universal_library>
  <books>
    <book> <title>Some Quotations from the Universal
         Library</title>
      <section> <title>Famous Quotes</title>
        <subsection>  <title>By William I</title>
          <quote bibref="shakespeare-sonnets-1609">
          <title>Sonnet XVIII</title>
          <verse>
            <line>Shall I compare thee to a summer's
           day?</line>
            <line>Thou art more lovely and more
           temperate. </line>
            <line>Rough winds do shake the darling buds of
           May, </line>
          </verse>
  ...
      <subsection> <title>By William II</title>
        <quote bibref="gates-road-ahead-1995">
          <title>Page 265</title>
          <line>``The obvious mathematical breakthrough
           would be development of an easy way to factor
           large prime numbers.''</line>
        </quote>
      </subsection>
      </section>
</book>
...
</books>
</universal_library>
```

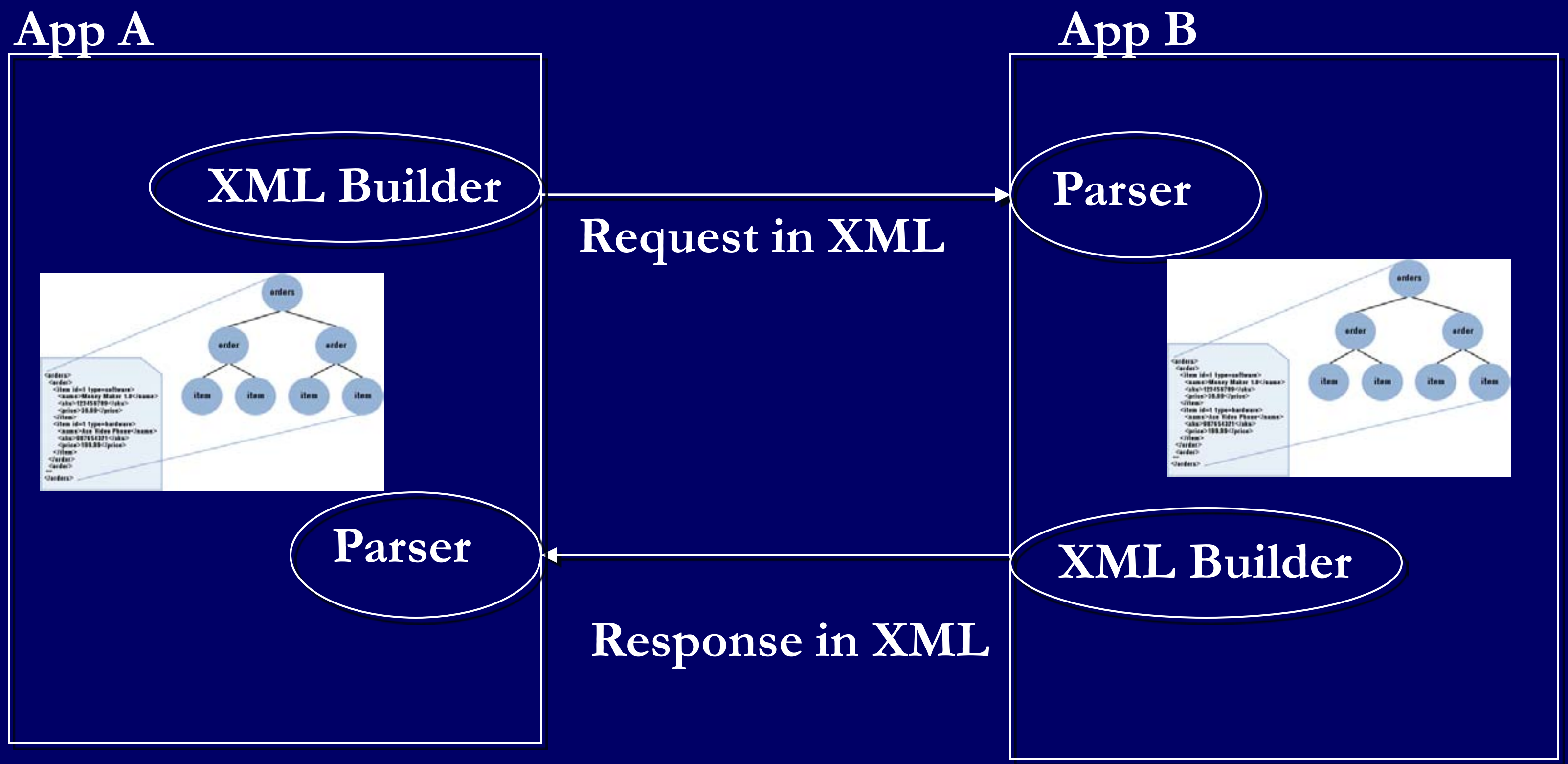
XML Industry Initiatives

- Every community is building it's own XML protocols, for example:

- Advertising: adXML place an ad onto an ad network or to a single vendor
- Literature: Gutenberg convert the world's great literature into XML
- Web Servers: apacheXML parsers, XSL, web publishing
- Travel: openTravel information for airlines, hotels, and car rental places
- News: NewsML creation, transfer and delivery of news
- Voice: VoxML markup language for voice applications
- Wireless: WAP (Wireless Application Protocol) wireless devices
- Weather: OMF Weather Observation Markup Format (simulation)
- Geospatial: ANZMETA distributed national directory for land information
- Banking: MBA Mortgage Bankers Association of America --> credit report, loan file, underwriting…
- Healthcare: HL7 DTDs for prescriptions, policies & procedures, clinical trials
- Math: MathML (Mathematical Markup Language)
- Surveys: DDI (Data Documentation Initiative) "codebooks" in the social and behavioral sciences

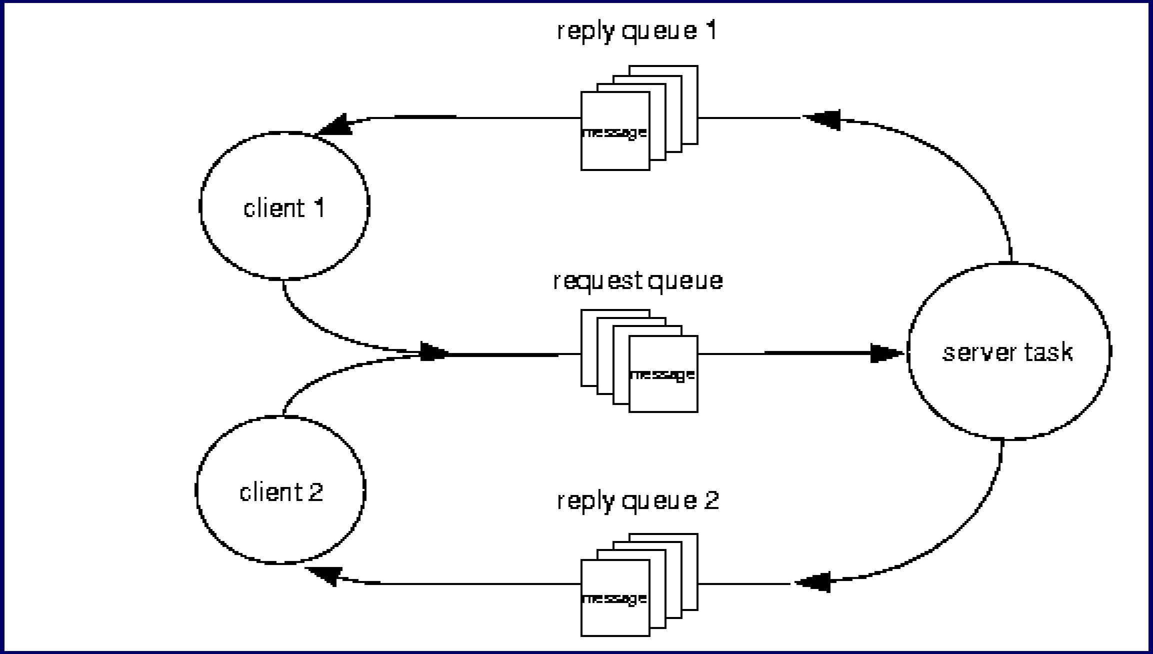
Http://www.oasis-open.org/cover/xml.html#applications

Two Applications Communicating



Revisiting OOD: cohesion and coupling and the Use of Message Queues

Two Client Applications communicating with a server


reply queue 1
message
client 1
request queue
message
server task
client 2
reply queue 2
message