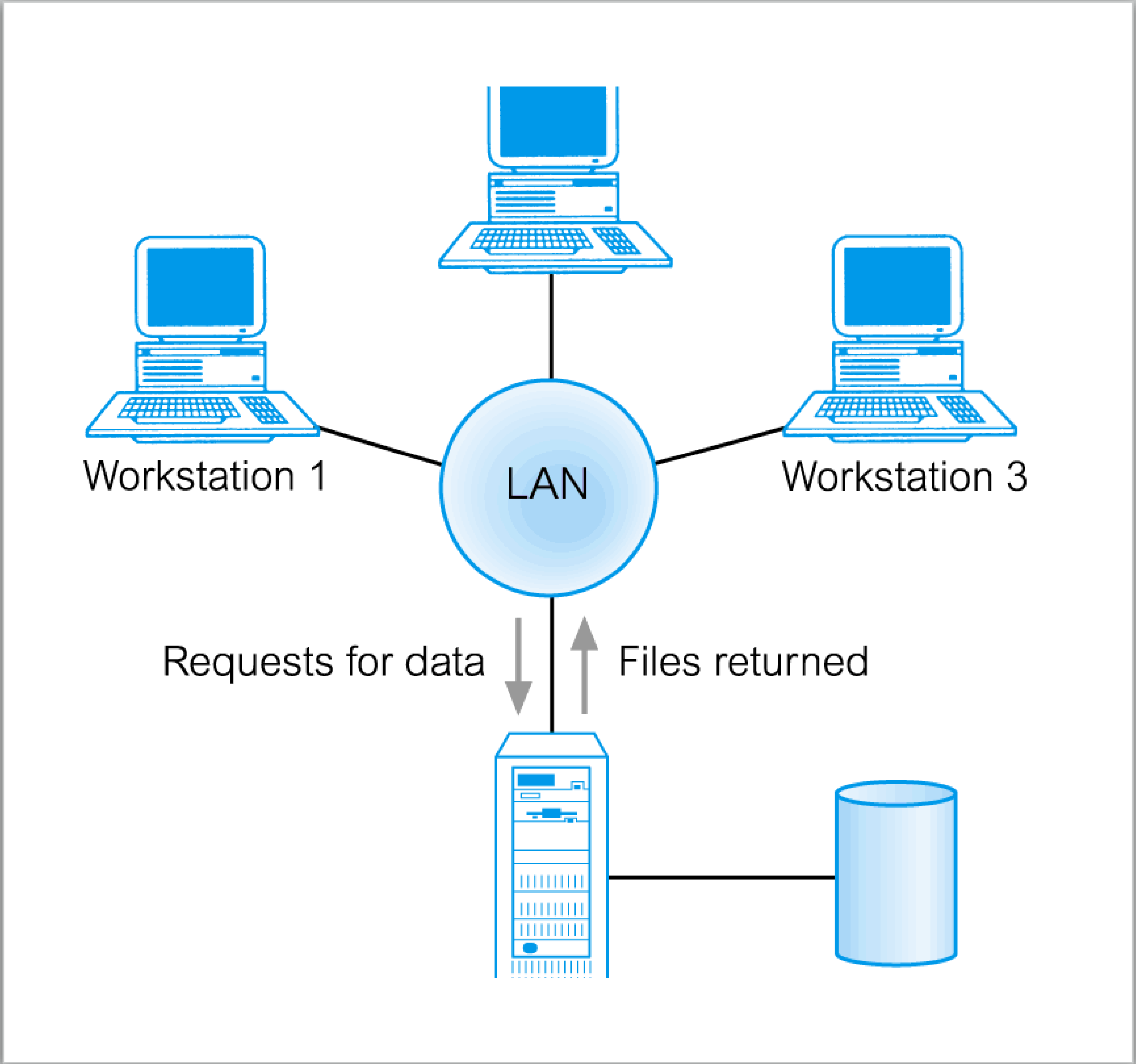
Database Architectures

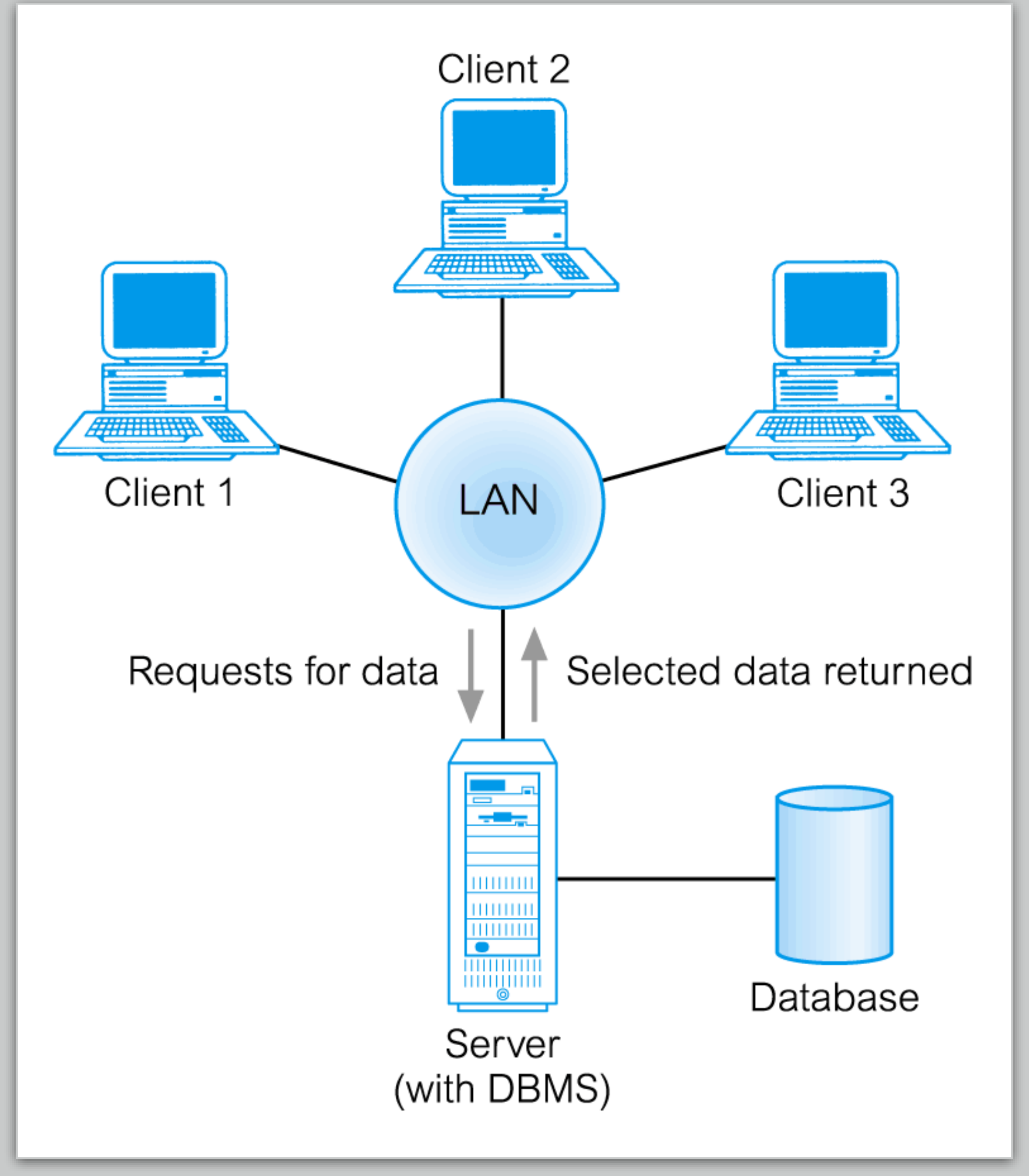
Updated 01/01/2023

File-Server Architectu re



Traditional Two-Tier Client-Server

- Client (tier 1) manages user interface and runs applications.
- Server (tier 2) holds database and DBMS.



Summary of Client-Server Functions

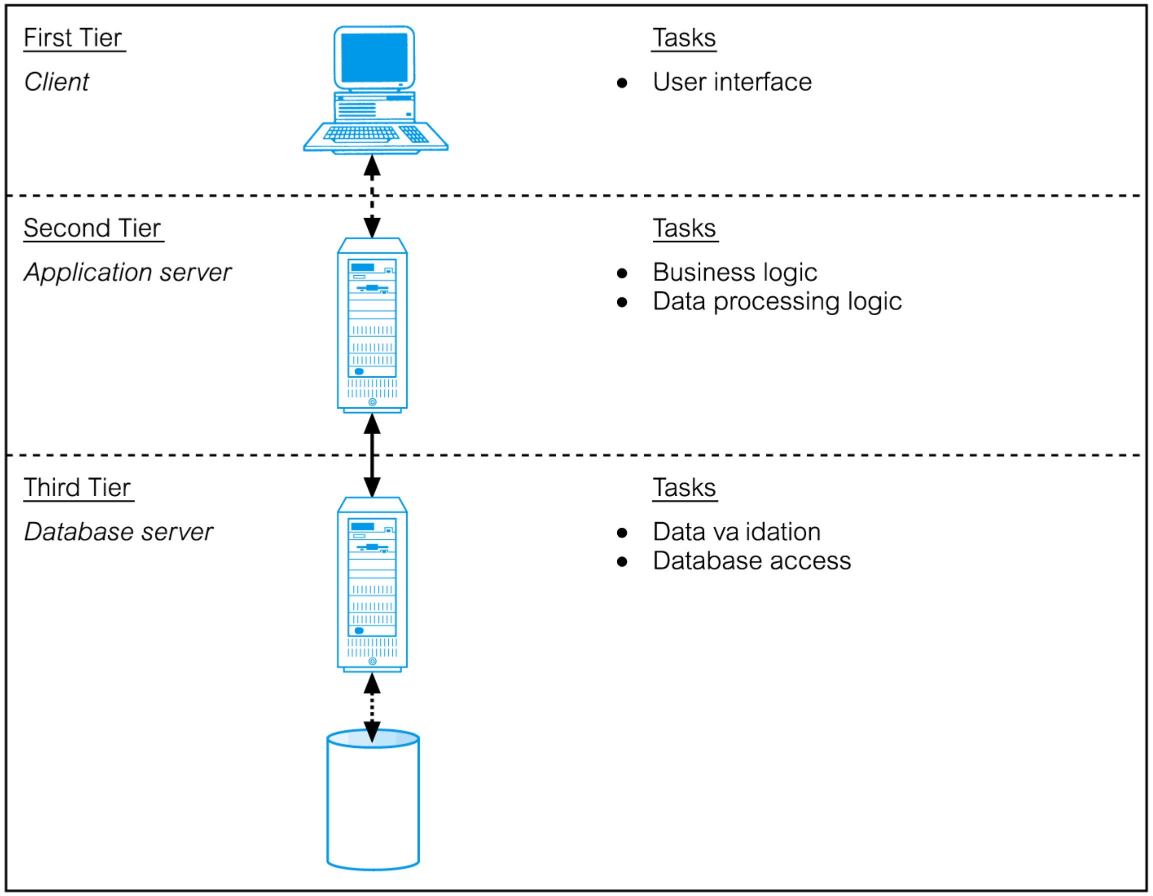
Client

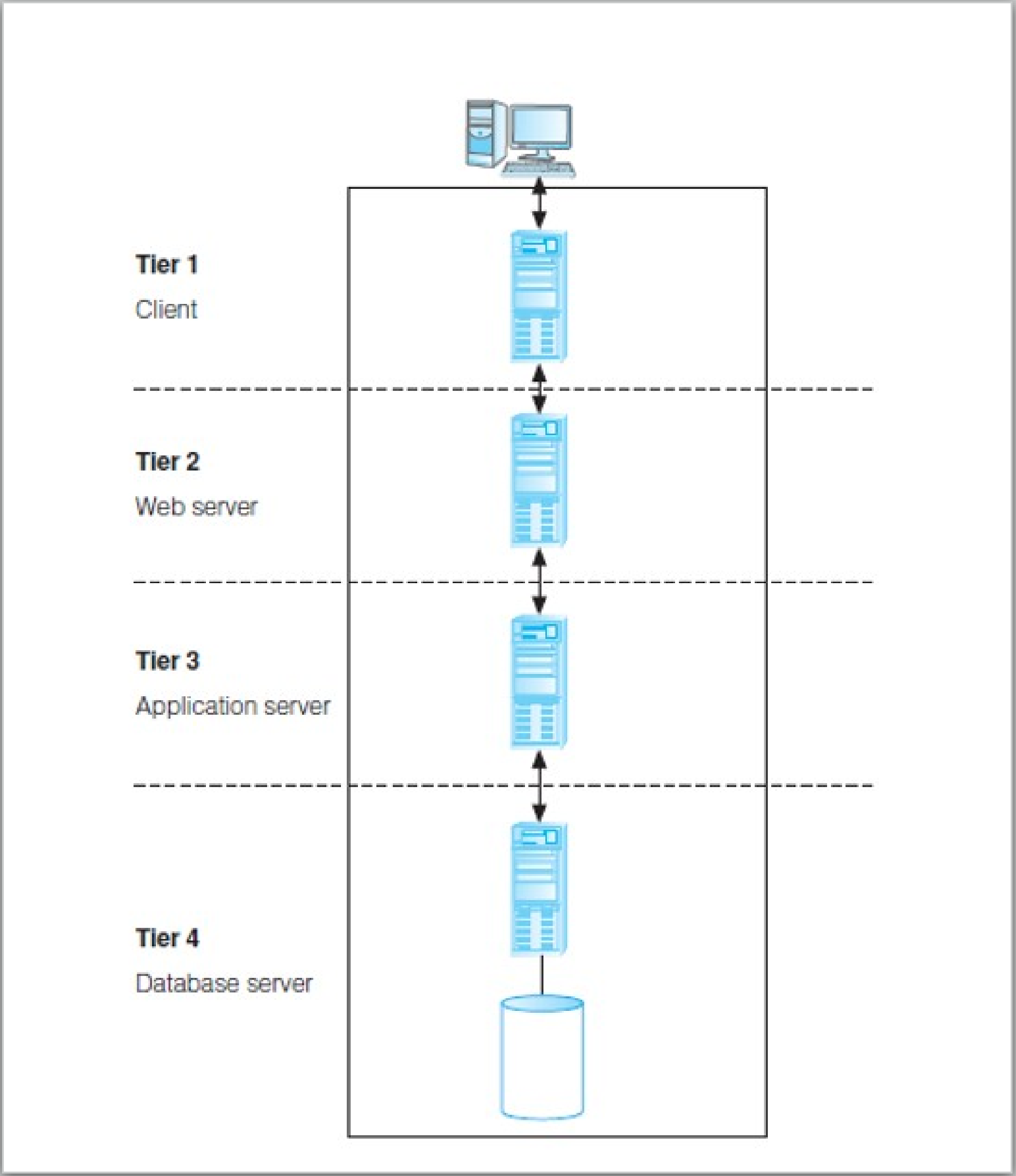
- Manages user interface
- Accepts and verifies user input
- Processes application logic
- Generates database requests and transmits to server
- Displays response to user

Server

- Accepts and processes database requests from clients
- Check authorization
- Ensures integrity constraints are not violated
- Performs query/update processing and transmits response to client
- Maintains system catalog
- Provides concurrent databases access
- Provides recovery control

Three-Tier Client-Server

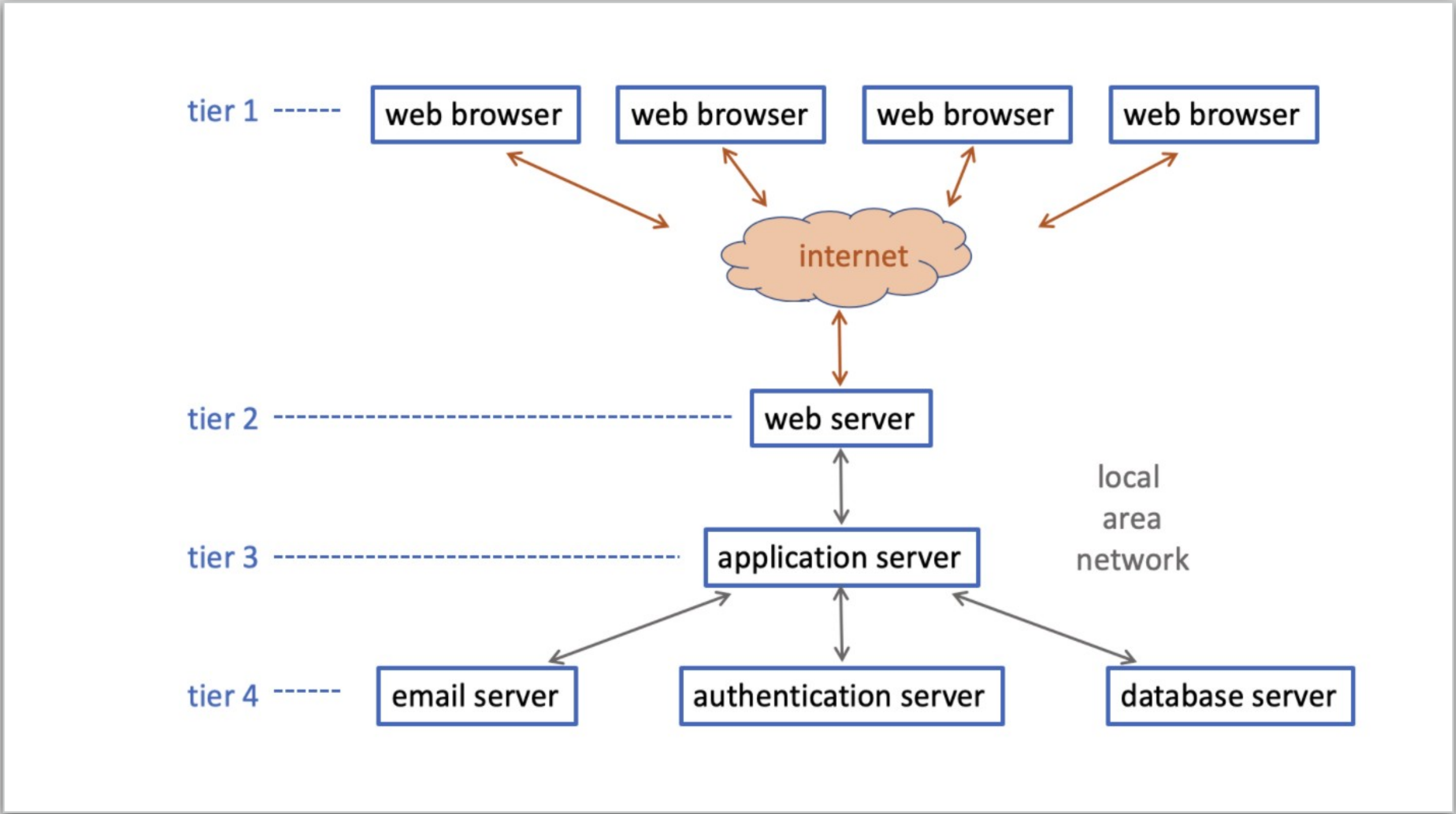




n-Tier Client-Server

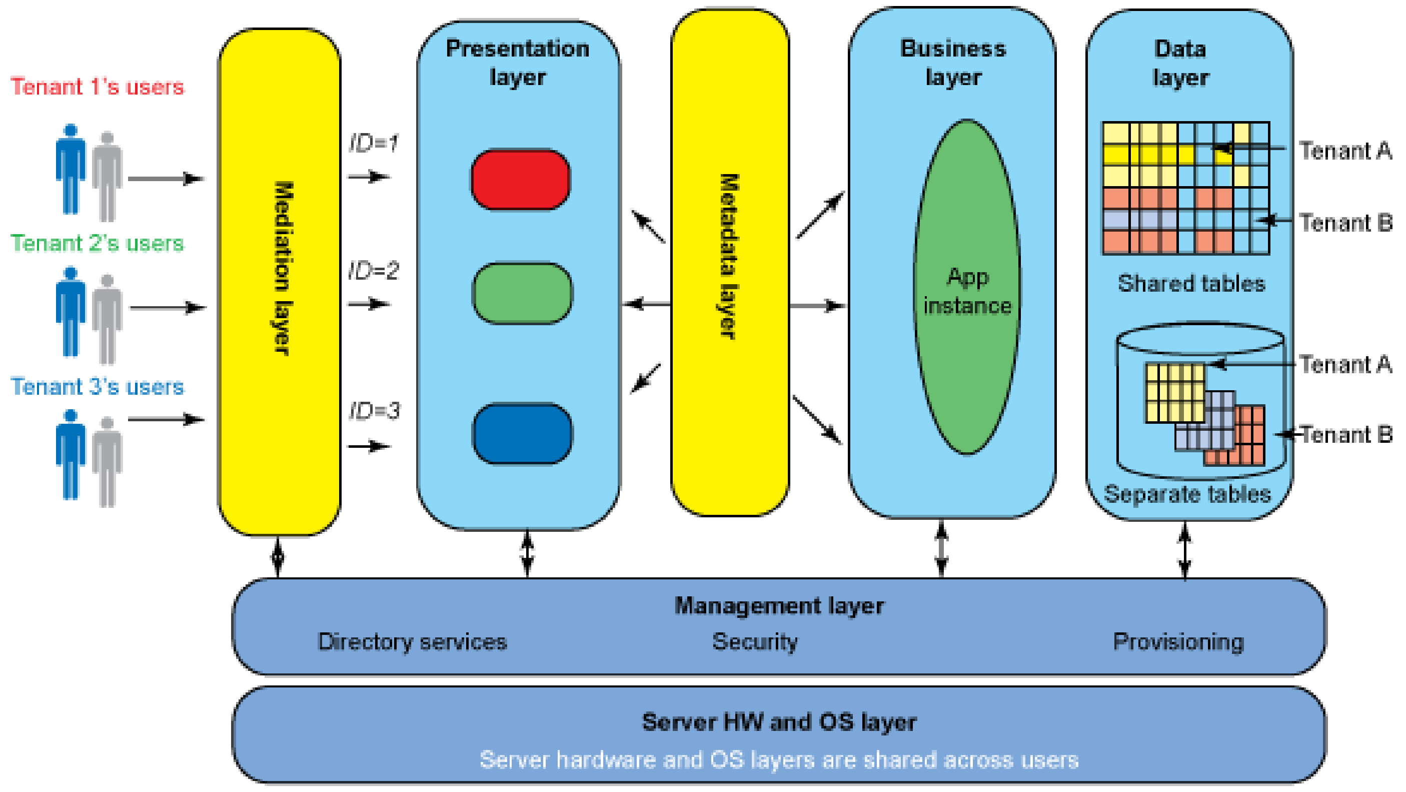
- The three-tier architecture can be expanded to n tiers, with additional tiers providing more flexibility and scalability.
- Applications servers host API to expose business logic and business processes for use by other applications.
- Number of tiers depend on layers between user and DBMS
- Imagine if OSINT was in between tiers 2 and 3, what would it do?

Cloud Databases



Web Architecture

Cloud Multi-Tenancy Design



From IBM Developer

Key Characteristics of Cloud Computing

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Benefits of Cloud Computing

- Cost-Reduction
- Scalability/Agility
- Improved Security
- Improved Reliability
- Access to new technologies
- Faster development
- Large scale prototyping/load testing
- More flexible working practices.
- Increased competitiveness

Risks of Cloud Computing

is he disconnected what my move didn't go through
what the f what the fridge oh what the heck
0:10 / 1:35:41 • Intro
@GMHikaru | GM Hikaru Nakamura
Chess.com

A Comcast Gambit During Titled Tuesday? Why not??

- Network Dependency
- System Dependency
- Cloud Provider Dependency
- Lack of control
- Lack of information on processing transparency

Cloud Computing – Service Models

- Software as a Service (SaaS)
  - A single application
- Platform as a Service (PaaS)
  - A group of applications working together
- Infrastructure as a Service (IaaS)
  - An entire IT system, VM’s, web portal

Distributed Databases

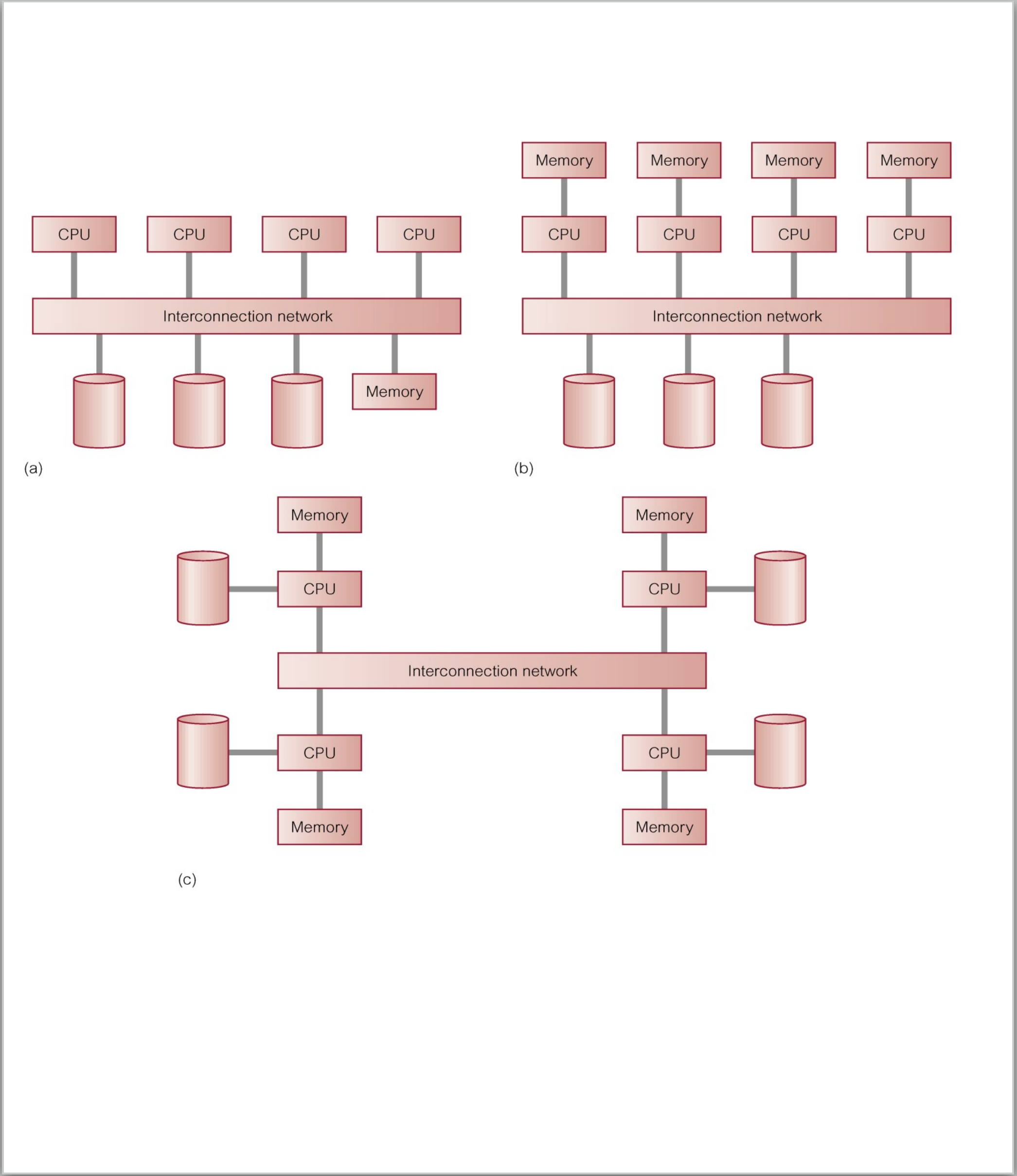
Parallel DBMS

- A DBMS running across multiple processors and disks designed to execute operations in parallel, whenever possible, to improve performance.
- Based on premise that single processor systems can no longer meet requirements for cost-effective scalability, reliability, and performance.
- Parallel DBMSs link multiple, smaller machines to achieve same throughput as single, larger machine, with greater scalability and reliability.

Parallel DBMS

Main architectures for parallel DBMSs are:

- Shared memory
- Shared disk
- Shared nothing



Distribute d DBMSs

- A distributed database is physically distributed over a computer network.
- A distributed DBMS is the software system that permits the management of the distributed database and makes the distribution transparent to users.
- A DDBMS consists of a single logical database split into a number of fragments.
- Each site is capable of independently processing user requests that require access to local and of processing data stored on other computers in the network.

Advantages of DDBMSs

- Reflects organizational structure
- Improved shareability and local autonomy
- Improved availability
- Improved reliability
- Improved performance
- Economics
- Modular growth

Disadvantages of DDBMSs

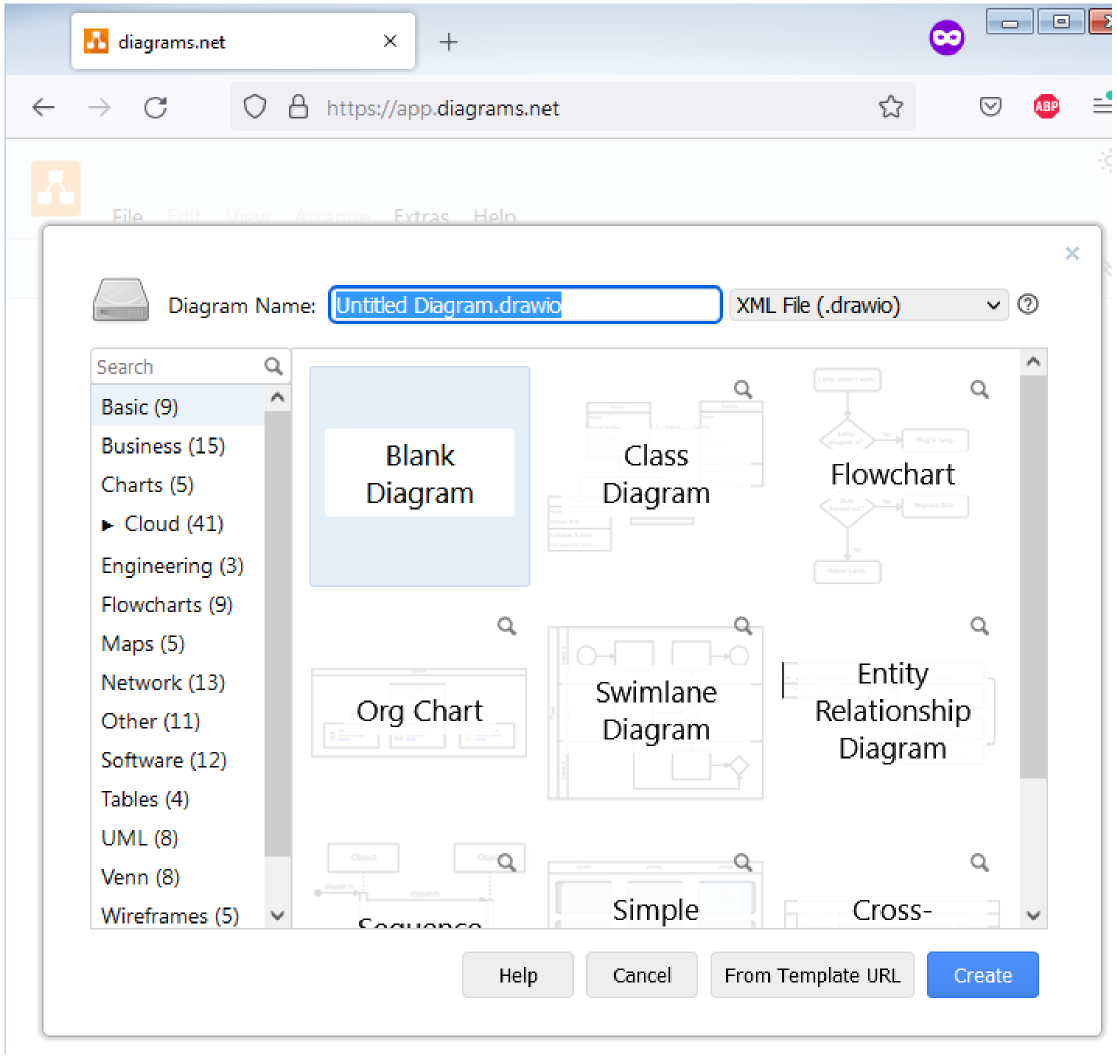
- Complexity
- Cost
- Security
- Integrity control more difficult
- Lack of standards
- Lack of experience
- Database design more complex

Draw.io

Drawing Models

- Effective communication of technical elements often requires images that executives can quickly absorb and understand
- Leaders communicate ideas effectively

Practice Drawing with Draw.io



Drawing tips provided in separate handout

Middleware: Types and protocols

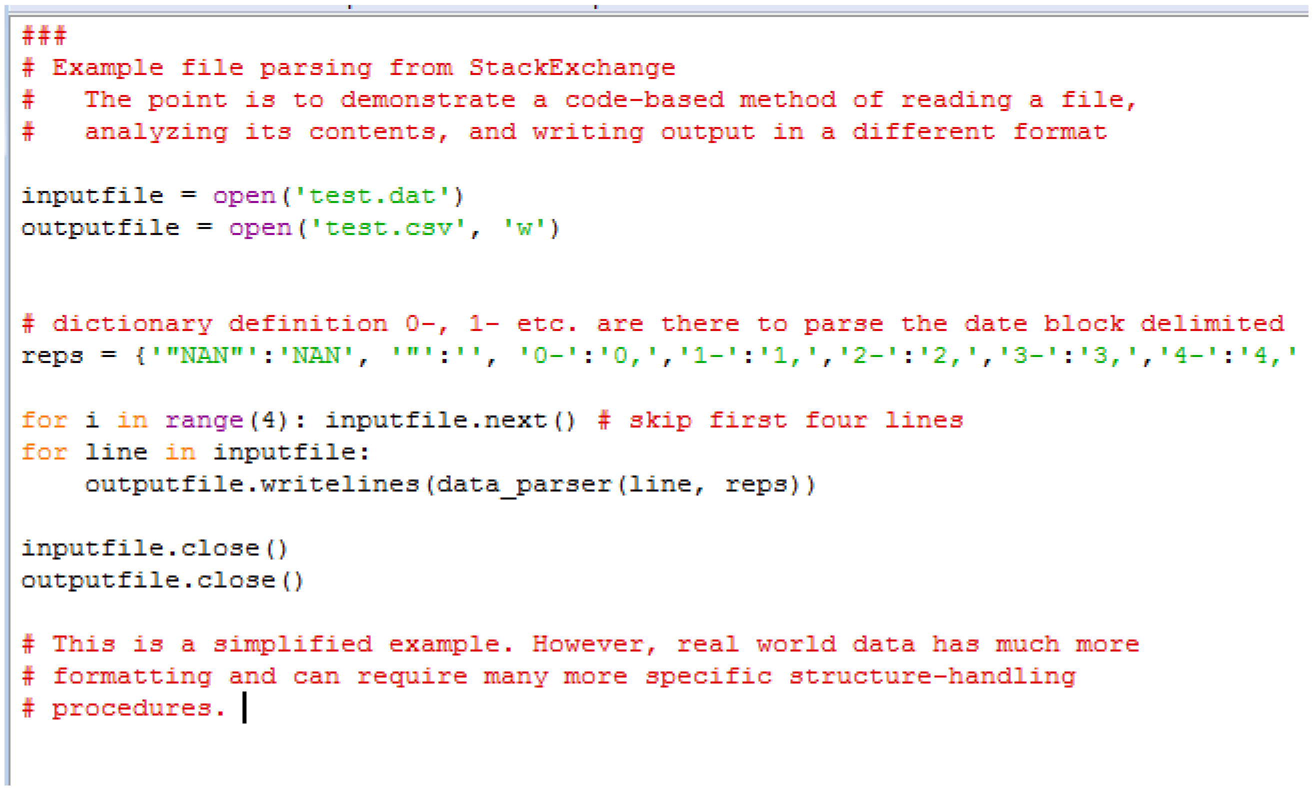
Middleware

Middleware allows for communication between different applications

The need for middleware arises when distributed systems become too complex to manage efficiently without a common interface.

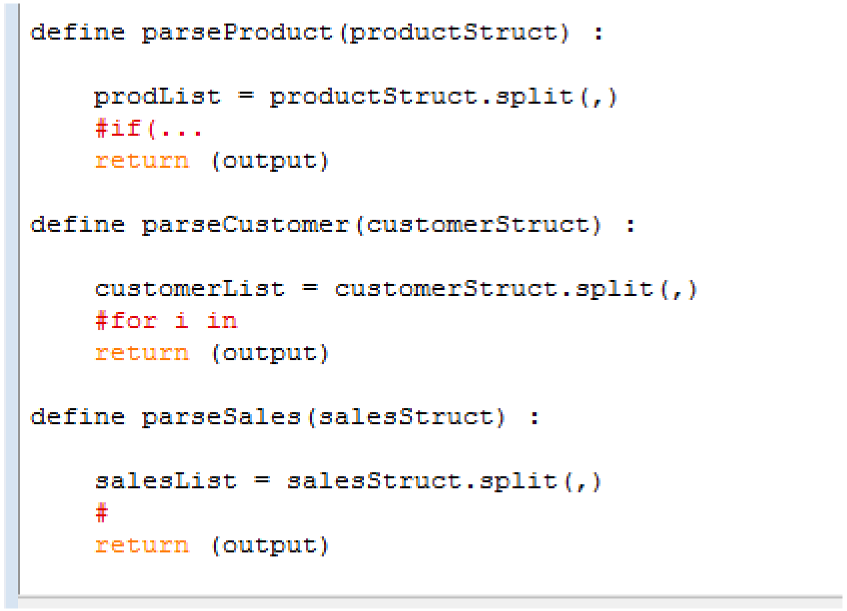
Allows abstraction between an outsider and a complex system

Middleware as Custom-Scripting



```
###
# Example file parsing from StackExchange
#   The point is to demonstrate a code-based method of reading a file,
#   analyzing its contents, and writing output in a different format

inputfile = open('test.dat')
outputfile = open('test.csv', 'w')


# dictionary definition 0-, 1- etc. are there to parse the date block delimited
reps = {'"NAN"':'NAN', '"':'', '0-':'0,','1-':'1,','2-':'2,','3-':'3,','4-':'4,'

for i in range(4): inputfile.next() # skip first four lines
for line in inputfile:
    outputfile.writelines(data_parser(line, reps))

inputfile.close()
outputfile.close()

# This is a simplified example. However, real world data has much more
# formatting and can require many more specific structure-handling
# procedures.
```

Scripts Get Complicated



```
define parseProduct(productStruct) :

    prodList = productStruct.split(,)
    #if(...
    return (output)

define parseCustomer(customerStruct) :

    customerList = customerStruct.split(,)
    #for i in
    return (output)

define parseSales(salesStruct) :

    salesList = salesStruct.split(,)
    #
    return (output)
```

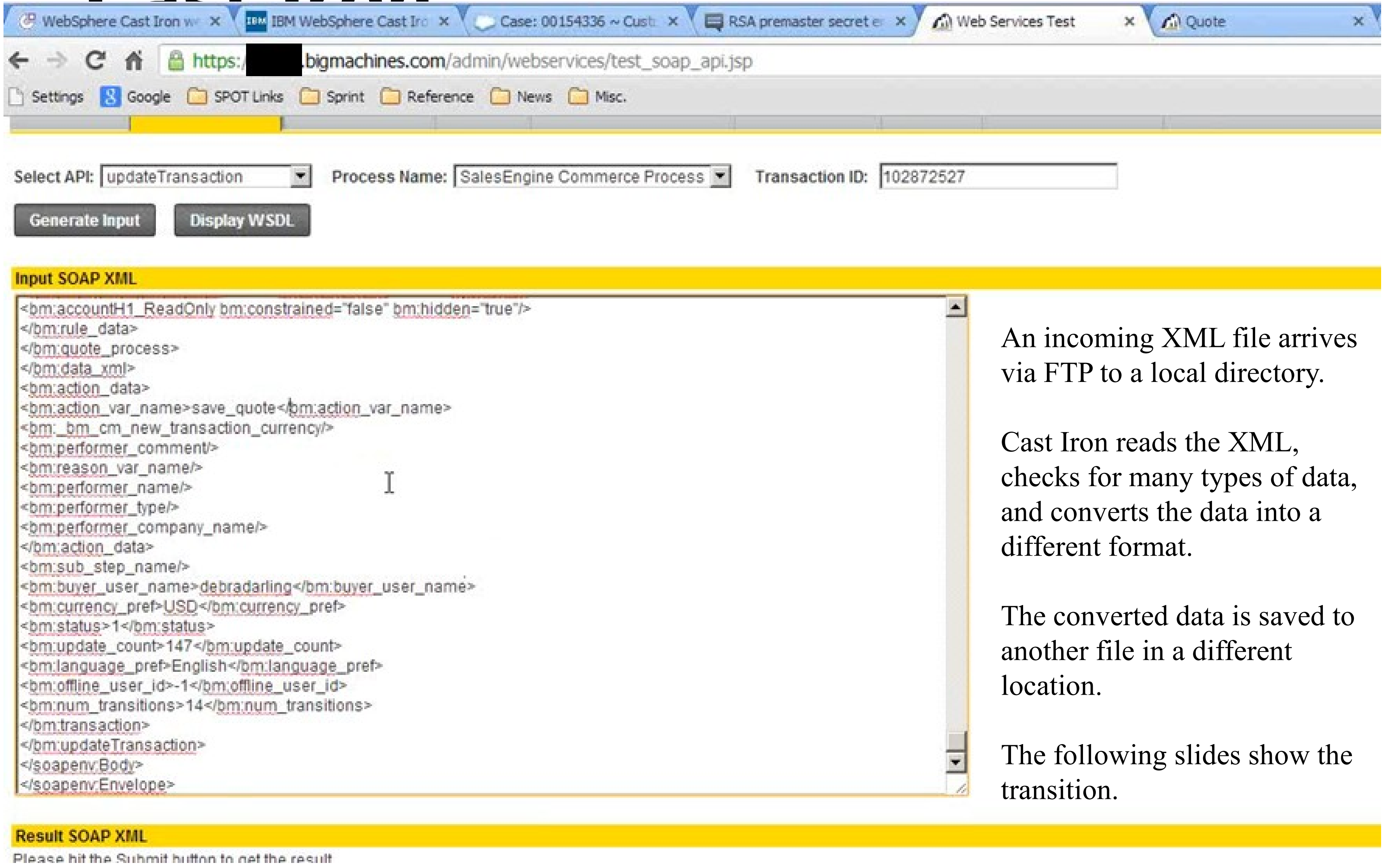
Imagine the maintenance and debugging!

Software vendors found automated solutions to replace custom scripts.

Examples of Middleware

- Message-Oriented-Middleware (traditional) – converts data transmissions from one type to another
- Object Middleware – handles object and service requests
- RPC Middleware – handles remote procedure calls
- Database Middleware
- Portals
- API’s
- Content Integration

Message-Oriented-Middleware Example:
Cast Iron



Select API: updateTransaction

Process Name: SalesEngine Commerce Process

Transaction ID: 102872527

Input SOAP XML

```
<bm:accountH1_ReadOnly bm:constrained="false" bm:hidden="true"/>
</bm:rule_data>
</bm:quote_process>
</bm:data_xml>
<bm:action_data>
<bm:action_var_name>save_quote</bm:action_var_name>
<bm:_bm_cm_new_transaction_currency/>
<bm:performer_comment/>
<bm:reason_var_name/>
<bm:performer_name/>
<bm:performer_type/>
<bm:performer_company_name/>
</bm:action_data>
<bm:sub_step_name/>
<bm:buyer_user_name>debradarling</bm:buyer_user_name>
<bm:currency_pref>USD</bm:currency_pref>
<bm:status>1</bm:status>
<bm:update_count>147</bm:update_count>
<bm:language_pref>English</bm:language_pref>
<bm:offline_user_id>-1</bm:offline_user_id>
<bm:num_transitions>14</bm:num_transitions>
</bm:transaction>
</bm:updateTransaction>
</soapenv:Body>
</soapenv:Envelope>
```

Result SOAP XML

Please hit the Submit button to get the result

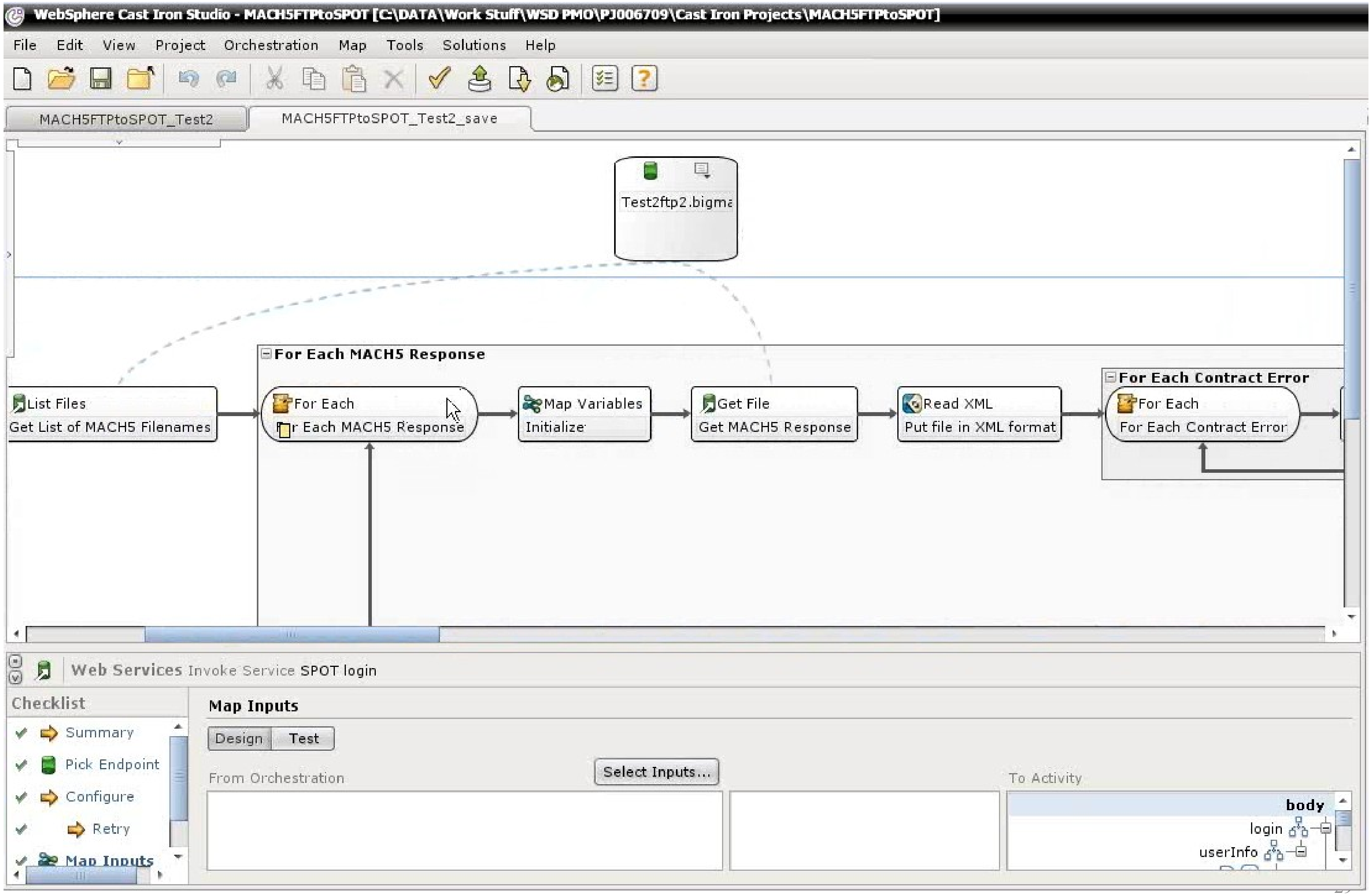
An incoming XML file arrives via FTP to a local directory.

Cast Iron reads the XML, checks for many types of data, and converts the data into a different format.

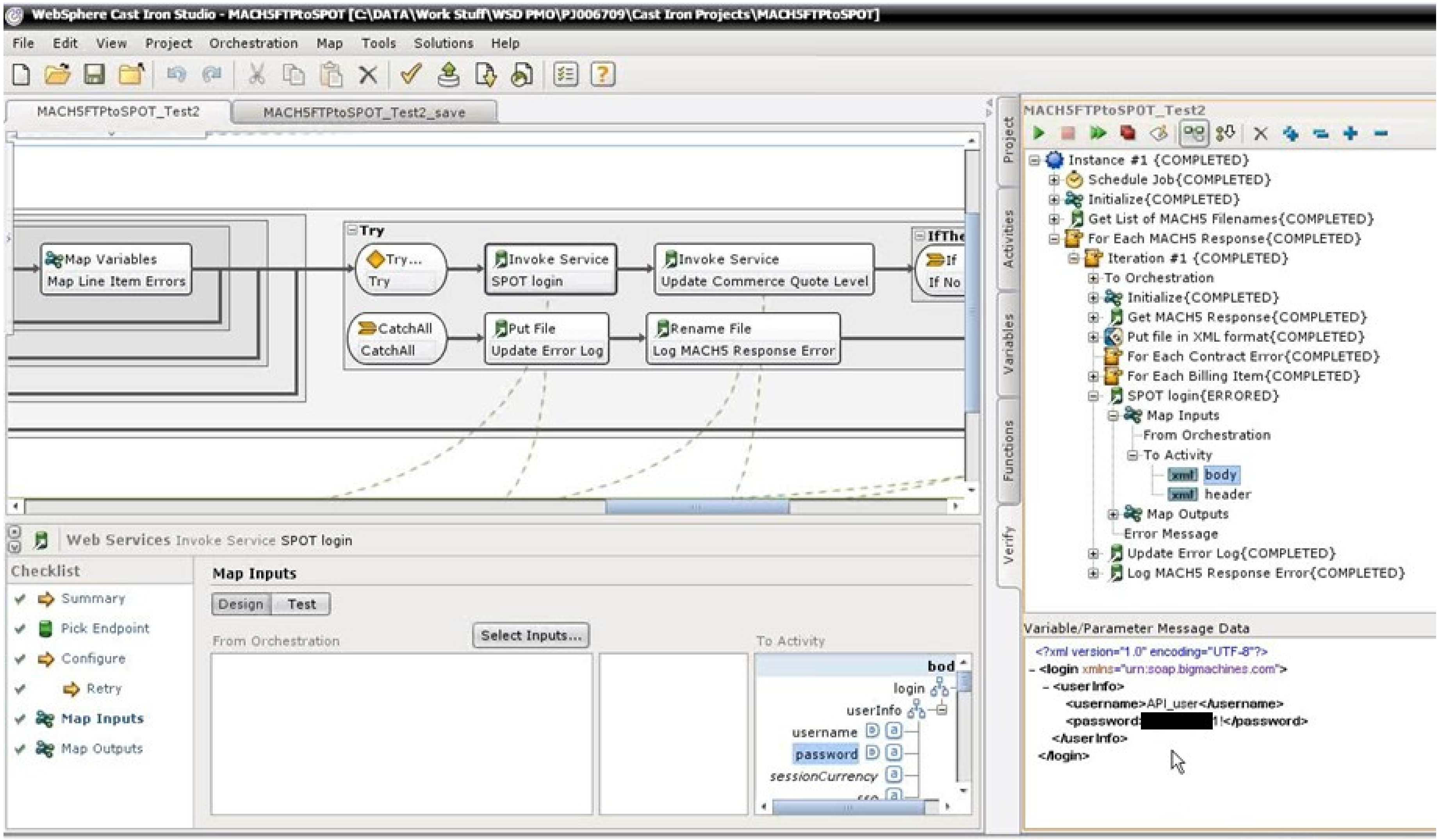
The converted data is saved to another file in a different location.

The following slides show the transition.

Cast Iron Example, p2 – Check for files, check format for accuracy, map input fields to output fields



Cast Iron Example, p3 – Next steps of the conversion process



WebSphere Cast Iron Studio - MACH5FTPtoSPOT [C:\DATA\Work Stuff\WSD PMO\PJ006709\Cast Iron Projects\MACH5FTPtoSPOT]

Web Services Invoke Service SPOT login

MACH5FTPtoSPOT_Test2

```
Instance #1 {COMPLETED}
  Schedule Job{COMPLETED}
  Initialize{COMPLETED}
  Get List of MACH5 Filenames{COMPLETED}
  For Each MACH5 Response{COMPLETED}
    Iteration #1 {COMPLETED}
      To Orchestration
      Initialize{COMPLETED}
      Get MACH5 Response{COMPLETED}
      Put file in XML format{COMPLETED}
      For Each Contract Error{COMPLETED}
      For Each Billing Item{COMPLETED}
      SPOT login{ERRORED}
        Map Inputs
          From Orchestration
          To Activity
            xml body
            xml header
        Map Outputs
        Error Message
      Update Error Log{COMPLETED}
      Log MACH5 Response Error{COMPLETED}
```

Variable/Parameter Message Data

```
<?xml version="1.0" encoding="UTF-8"?>
- <login xmlns="urn:soap.bigmachines.com">
  - <userInfo>
      <username>API_user</username>
      <password>[redacted]!!</password>
    </userInfo>
  </login>
```

Web Services

- Web services approach uses accepted technologies and standards, such as:
  - XML (extensible Markup Language).
  - SOAP (Simple Object Access Protocol) is a communication protocol for exchanging structured information over the Internet and uses a message format based on XML. It is both platform- and language-independent.
- WSDL (Web Services Description Language) protocol, again based on XML, is used to describe and locate a Web service.

SoapUI Resource – www.soapui.org

- https://www.soapui.org/resources/tutorials/soap-sample-project/

- Open source (free)

SOAP Example – Used with Salesforce

- <xsl:template match="/">
- <!-- Begin SOAP XML -->
- <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <soap:Header>
- <QueryOptions xmlns="urn:partner.soap.sforce.com">
- <batchSize>
- <xsl:value-of select="/query/page_size"/>
- </batchSize>
- </QueryOptions>
- <SessionHeader xmlns="urn:partner.soap.sforce.com">
- <sessionId>
- <xsl:value-of select="/query/user_info/session_id"/>
- </sessionId>
- </SessionHeader>
- <CallOptions>

RESTful Webservices

- Based on HTTP
- Often use JSON to present the data
- Very commonly used by large companies such as Twitter, Google and Netflix
- Example:

https://api.nytimes.com/svc/books/v3/lists.json?api-key=42ff06dcd8c04a4cae037a10a43ffd4c&list=hardcover-fiction

Above source from NYT Bestsellers

Using REST

- https://www.guru99.com/restful-web-services.html#1

- Uses “CRUD” type operations:

  POST
  GET
  PUT
  DELETE

- API’s are specific to different languages

Input File Format Prototypes

- Many standards exist
- Can all be described as one of the following prototypes:
  - Delimited text
  - XML
  - JSON

Delimited Text File (1 of 3)

- Delimited text file
  - Text file containing data separated by a special character or sequence of characters, or patterns
  - Tab is a common delimiter
  - Excel has a ‘comma separated value’ format
  - Can be formatted text to look like code

JSON (2 of 3)

- Text file with simple or nested array structures
- https://json.org/example.html
- { “Example”:[{“First Category”: “thing1}, {“Second Category”: “thing2”}]}

XML (3 of 3)

- Custom HTML tags – can use anything, but receiver must know what to expect
- <Name>”MyName” </Name>
- <Age>25</Age>
- <Fun>”Dancing Like a Fool”</Fun>

XML Example

- <?xml version="1.0"?>
- <quote_process document_number="1" data_type="0" buyer_user_name="schan" bs_id="12254735">
- <_billTo_name> </_billTo_name>
- <_billTo_company_name>BUY BUY FUNITURE</_billTo_company_name>
- <_billTo_address>1513 MYRTLE AVE</_billTo_address>
- <_billTo_address_2></_billTo_address_2>
- <_billTo_city>BROOKLYN</_billTo_city>
- <_billTo_state>New York</_billTo_state>
- <_billTo_zip>11237</_billTo_zip>
- <_billTo_country></_billTo_country>
- <_billTo_phone>(347) 555-1234</_billTo_phone>

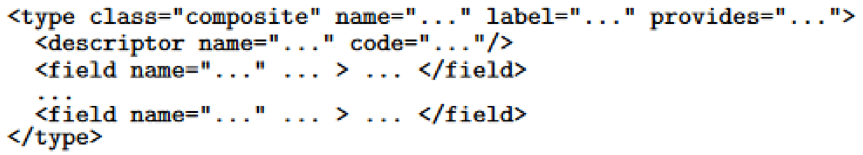
Examples of Standards Using Text or XML

- https://pubs.opengroup.org/onlinepubs/009649399/toc.pdf
  - C and Cobol standards for developing XATMI API’s
  - Delimited text/XML hybrid

- https://www.amqp.org
  - Open standard business messaging

   (advanced message queuing protocol)

  XML format



```
<type class="composite" name="..." label="..." provides="...">
  <descriptor name="..." code="..."/>
  <field name="..." ... > ... </field>
  ...
  <field name="..." ... > ... </field>
</type>
```

- https://standards.ieee.org/ieee/1516/3744/
  - Proprietary standards

- https://xmpp.org/
  - XML-based standard, open and free

Questions?

SUMMARY

- Architectures, Tiers
- Cloud
- Distributed Systems
- Modeling, Drawing
- Middleware