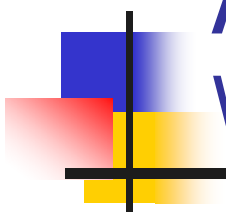


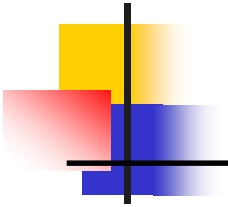
An Evaluation of Web Services

What are they good for? What not?

Dan Johnsson

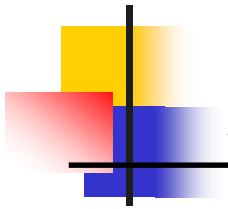
Omegapoint AB; Sweden

dan.johnsson@omegapoint.se



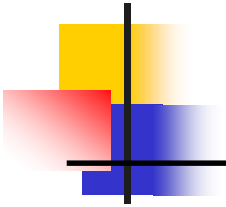
Scope of This Session

- Using WS today or near future
- Related to Java/J2EE space
- Reality Check
- Evaluating, not describing
  - No WS basics – assumed familiar
  - SOAP, WSDL, UDDI
  - Java Web Services
  - [Denise Hatzidakis] Web Services… Where Do You Start?



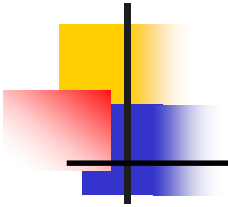
Agenda

- Evaluate? Could you make that clear?
- What kind of creature is Web Services?
- Alternatives
- Evaluation (*aka* "the meat")
  - Integration scenarios
- Some high-level WS-specs
- Conclusions/Guidelines



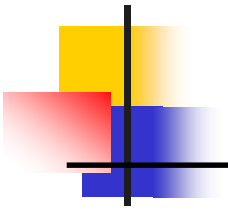
Disclaimer

- Not comprehensive
  - Based on my experience
- Young fast-moving technology
  - Might have missed an update

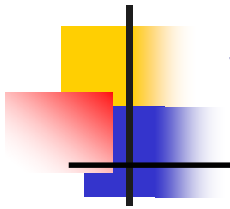


Silly Example

- Hungry?
- See what the snack bar can offer …
- Hmm,
  - What's on the menu
  - Order just a simple drink
  - … or a complex meal
- Better talk to the Snack-Bar-Tender
- Time to integrate!

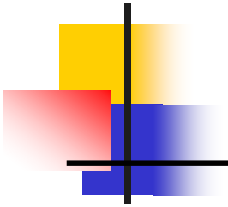


How to Evaluate



What We Are Evaluating

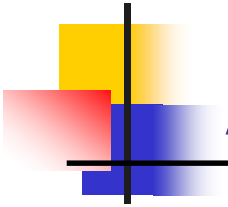
- Web Services is technology
  - Communication between processes
- Not enterprise model
  - Enables enterprises
- Not architectural model
  - Used in architecture



… but SOA?

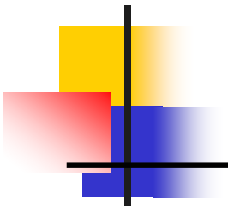
- SOA = Service Oriented Architecture
- SOA ≠ WS
- Orthogonal concepts
- WS *recommends* SOA

	SOA	Not SOA
WS	possible	possible
Not WS	possible	possible



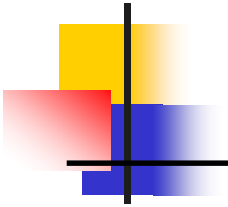
Aspects of Business Integration

- Functionality
  - How powerful is the business process
- Static / Dynamic
  - Fixed? Register? Unannounced discovery?
- Extern
  - B2B – integrating partners
- Intern
  - Way to glue systems together



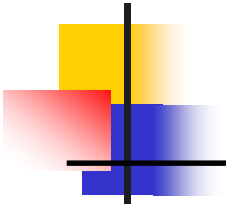
Evaluation Criteria

- Assume always get functionality in place
  - Hack around if we have to
  - All techs are "Turing complete"
- Performance
  - Stop the clock!
- Capacity
  - Simultaneous requests, sessions, whatever
- Extensibility
  - I'll stare at this code till … it … makes … sense!
- Reliability
  - It does what it should, doesn't it …
  - … and nothing else … ?



Good or Bad?

- Good if fulfills NFR
- If not: apply transformation
  - has trade-off
- Clarifying trade-offs – our task!

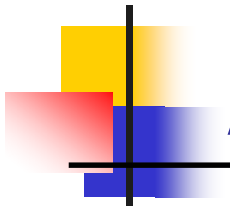


Origin of NFRs

- NFRs reflect business risk

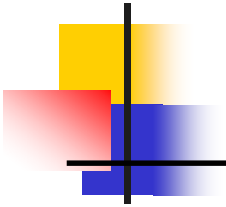
- Good or Bad – business decision
- Good or Bad – evaluation of trade-offs

- Good if eliminates business risk
- eBay lose a house a day
- My bank do not lose an account a day …



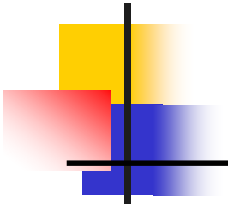
Analysing Web Services

- What kind of creature is WS
- What role does it fulfil



Web Presentations

- Web for humans
  - Viewing system state
  - HTTP transfer, HTML data format
  - State queries, and changes
- Web for machines
  - Getting system state
  - HTTP transfer, XML data
  - State queries, and changes
- Same, same; but different

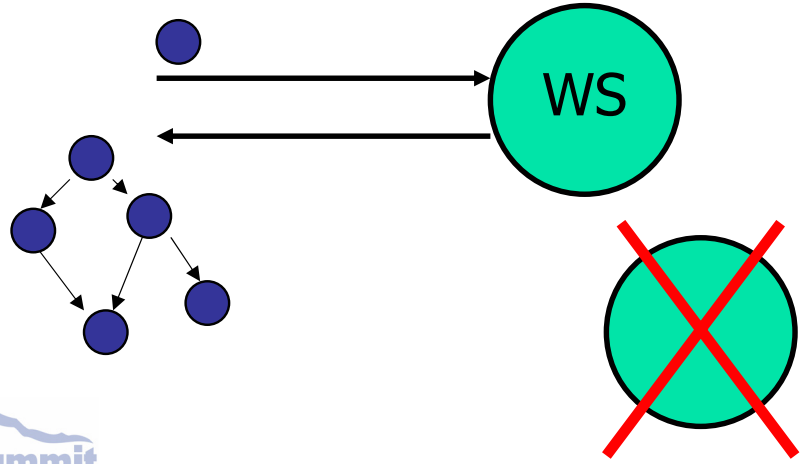


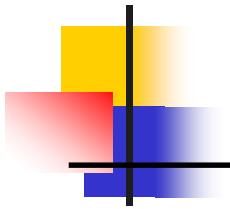
Remote Procedure Calls

- Structured data in, structured data out
- XML in XML out => WS is RPC
- WS = several RPCs in one place
- Several procedures acting on one state ...
- [oo googles on]… an object!
- *A Web Service is an object*
  - RPC = method on object

What Kind of Object?

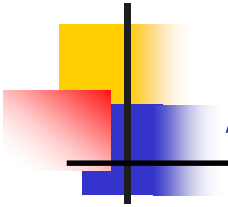
- Simple object model
  - Data in, data out
  - Even complex data, *e.g.* composite trees
  - No references to other objects ("spawn")





Simple Object Model

- Protocol for Access to Simple Objects?
- Protocol for Simple Access to Objects?
- SOAP
  - Coding: xml/text
  - Often carried by HTTP/TCP/IP
  - Note, from SOAP 1.2 not an acronym

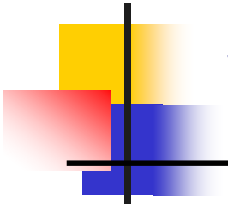


About the Object

- Describing the object
- Define all methods
  - Data type in, and out
- WSDL

name	XML schema in	XML schema out
search	Query schema	ResultSet schema
book	…	…
pay	…	…

- Finding the object
- Lookup in registry
  - By name
  - By type
  - By attribute
- UDDI

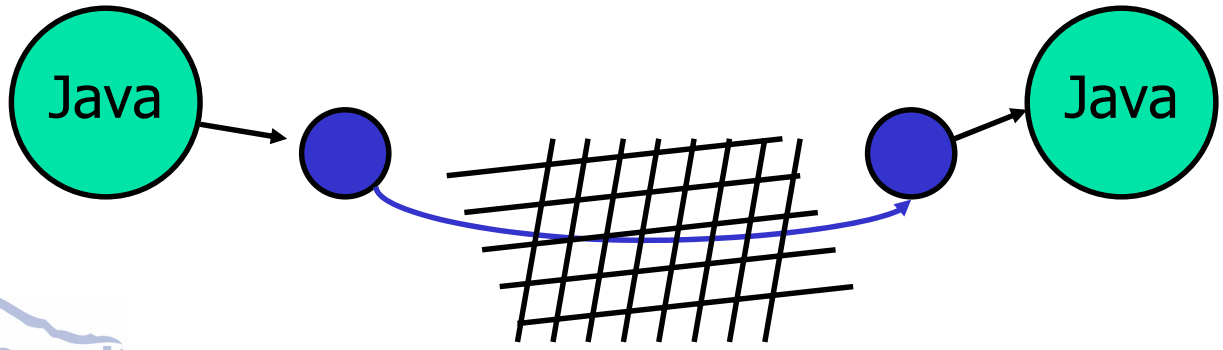


What Is Web Services?

- Communication protocol
  - SOAP
- Type description
  - WSDL
- Lookup mechanism
  - UDDI
- *Web Services are distributed objects*

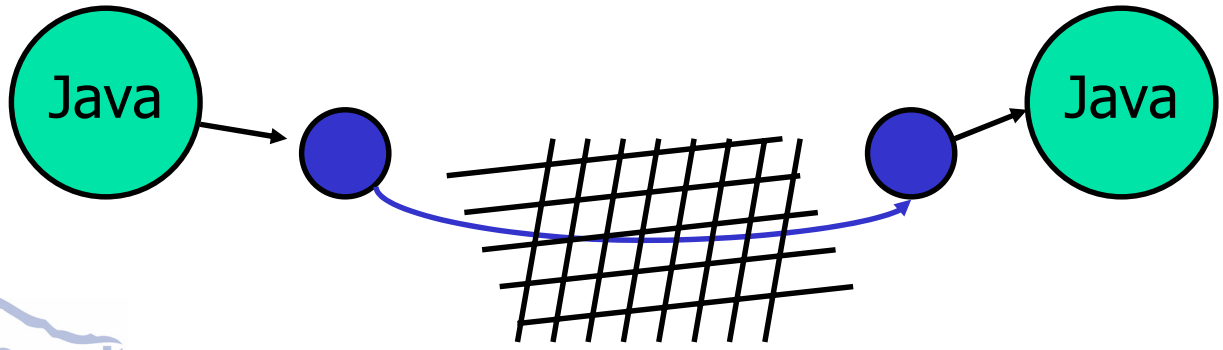
WS in Java Context

- WS is communication channel
- Programmes on both sides
  - Caller of functionality
  - Provider of functionality
- Intermediaries
  - Encoding/decoding



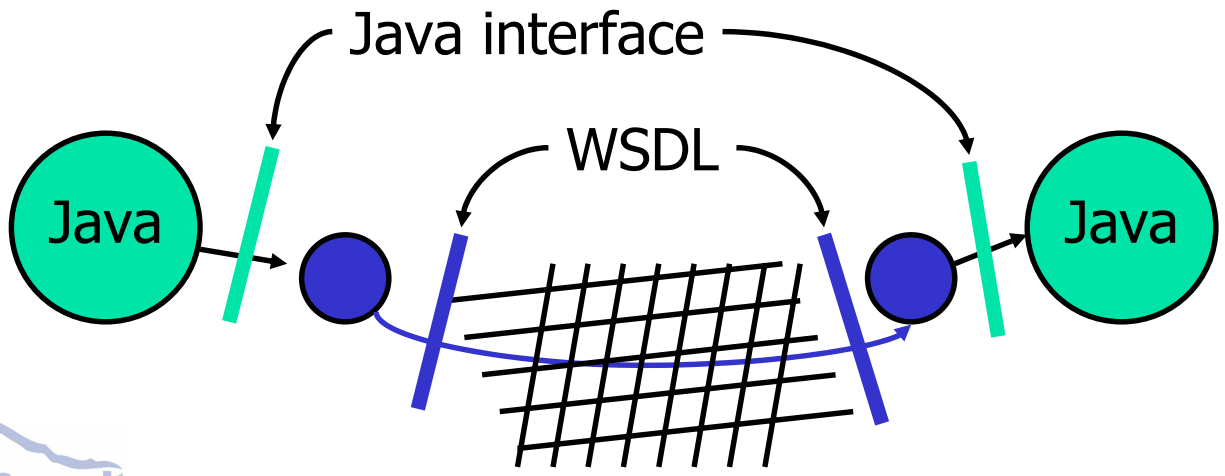
Java Standards

- Standardised encoding/decoding
  - Java-WSDL map (JAX-RPC) – JSR 101
- Support for generating stubs
  - Client programming model – JSR 109
- Support for generating skeletons
  - Deploy as Web Service – JSR 109 (5.3.2)
    - Java class in web container
    - Stateless Session EJB



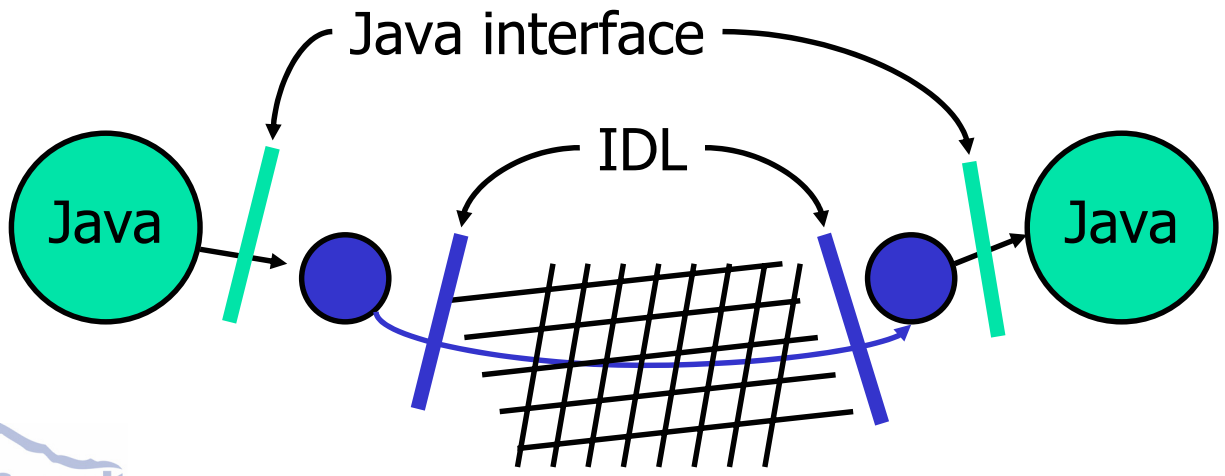
Web Services in Java/J2EE

- Technology for remote calls to distributed objects
- Easy to set up / deploy
- Easy to write client code to
- Programmers never see the Web Service
- Code against Java interface



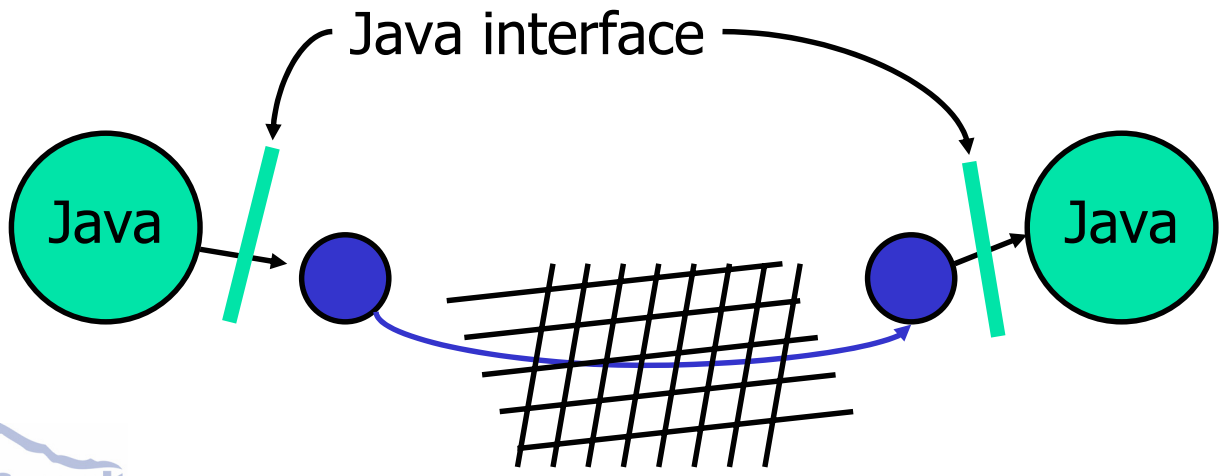
CORBA in Java/J2EE

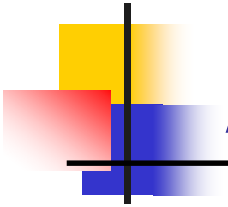
- Technology for remote calls to distributed objects
- Interface described in language-neutral IDL
- Exist standard Java-IDL mapping
- Code to the Java interface



RMI in Java/J2EE

- Technology for remote calls to distributed objects
- Interface described as Java interfaces
- Can be carried by JRMP, IIOP, or prop prot
- Code to the Java interface

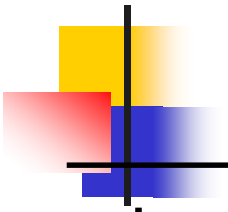




Alternatives to Web Services

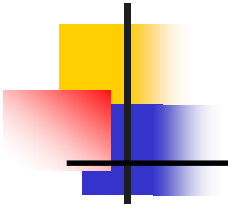
Technology	Web Service	CORBA	RMI	Jini
Data format	text/xml	binary	binary	binary
Protocol	SOAP	IIOP	IIOP or Prop	RMI-compliant
Interface	WSDL	IDL	Java	Java
Discovery	UDDI	COS Naming	JNDI	Registries
Object model	Simple	Remote	Complex	Full

Table not perfectly strict, but gives the scent



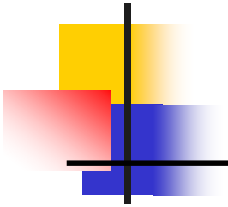
WS in J2EE – Characteristics: Downside

- Low extensibility
  - Only support for simple object models
  - Need to "hack around" limitations
    - *e.g.* push conversational state elsewhere
- Poor performance
  - XML text
    - Poor info/bit (<name>Dan Johnsson</name>)
    - String parsing
  - SOAP/HTTP/TCP/IP
    - Not really well-designed protocol stack ☺
  - Some remedies possible
    - [Peter Haggar: Using Data Compression…]



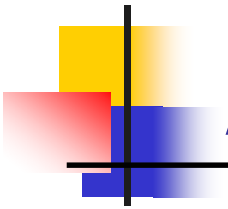
WS in J2EE – Characteristics: Upside

- High capacity
  - No conversational state – low server load
  - HTTPS can put load on web server
- Extreme manageability
  - Web and app servers in place
  - Auditing/logging probably fixed already
  - Simple deploy process
  - FW already accept HTTP

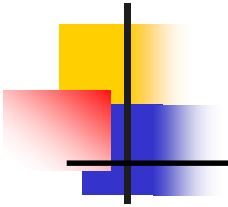


Guidelines – Rough Draft

- Good situations
- Up and running fast
  - Manageability
- Many users
  - High capacity if anonymous
- Trivial business process
  - OK with just global state
- Bad situations
- Lots of data
  - Parse a few gig string?
- Latency limits
  - Protocol stack overhead
- Nontrivial business process
  - Must tweak interface



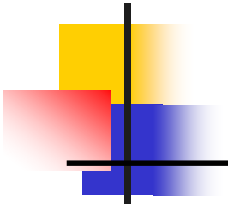
Anything Interesting This Far?



Evaluating Web Services

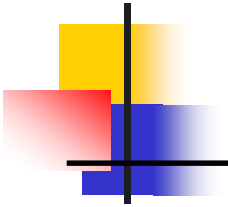
- We've had a look at the car
- Now take it out for a spin

- Integration situation
- Snack bar example
- Compare to Java/RMI-IIOP
- Reality Check



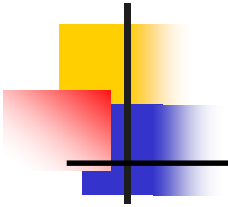
Scenarios

- Open Information
- Simple Orders
- Complex Order
- Critical Suborders
- Dynamic Dream



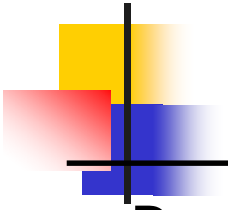
Open Information

- Publish read-only info
- No change of state



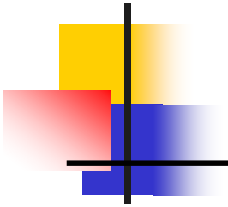
Open Info: Snack Bar

- Check the price list of drinks
- String[] drinkList()
- int drinkPrice(String drink)
- Web Service: No problem
  - JAX-RPC map
    - Primitives
    - Simple Classes (e g String)
    - Arrays thereof
    - Composite Classes



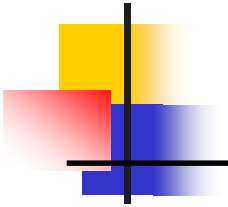
Open Info: Capabilities

- Performance
  - OK, as long as small amount of data
- Capacity
  - Really good
  - HTTPS + very many users = might be problem
- Extensibility
  - Good, follows business logic
- Reliability
  - No problem, just observing global state
- Security
  - HTTP BASIC will probably do



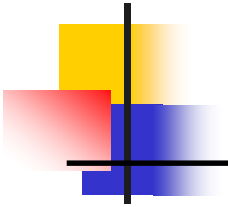
Open Info: Reality check

- Realistic to Real
- Lots of real publications – ready for WS
  - Time tables, gym schedules, stock quotes
- *e.g.* Time tables
  - Cross transport route search
  - Multi channel support (J2ME: JSR 172)



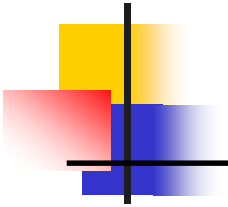
Simple Order

- Changing system state
  - Typically save order
- Trivial business process
  - One shot
- Change of public state
  - No private state
- Just change this system
  - No other systems involved



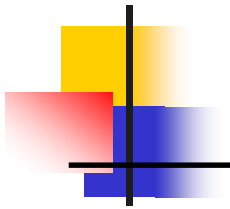
Simple Order: Snack Bar

- void buyDrink(String drink, boolean ice, int custID)
- Decrease stock of drink and ice
- Put cost on bar list
  - Global state affected
- OK or not



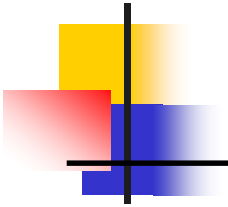
Simple Order: Capabilities

- Performance
  - Probably small amount of data: OK
- Capacity
  - Will probably need SSL – limits web server scalability
- Extensibility
  - OK, as long as we just extend order data format
- Reliability
  - OK, all updates are within system
- Security
  - BASIC or CLIENT-CERT enough for non-repudiation



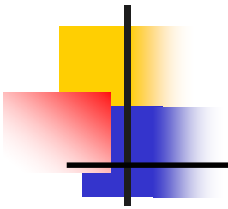
Simple Order: Reality Check

- Realistic to Real
  - On-line shopping exists already
- Enough for non-critical business processes
  - Public transport ticket
  - What if booking goes away?



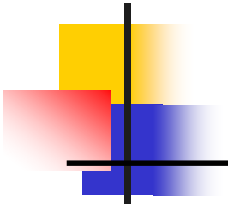
Complex Order

- Non-trivial business flow
- Building composite/complex order
- Complex search tree
- Fast fail wanted



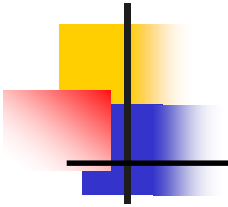
Complex Order: Snack Bar

- Complete meal
  - startOrder(custid)
  - addBurger(type)
  - addDrink(drink)
  - addSideOrder(type)
  - buy()
- Constraints
  - Max energy
  - Max cost
  - Never milk-and-meat
- Fast fail



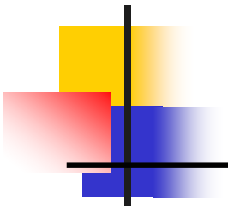
Complex Order: Snack Bar Tweak

- Cannot keep conversational state
- Push state elsewhere
  - Methods for this: [CSS2003:State is Not Evil]
  - "cookie model"
    - State in cookie
    - ID in client, state on server
- Rewrite interface



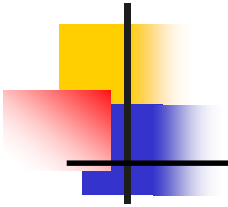
Complex Order: Snack Bar Again

- int startOrder(custid)
- void addBurger(orderid, type)
- void addDrink(orderid, drink)
- void addSideOrder(orderid, type)
- void buy(orderid)
- void endOrder(orderid)



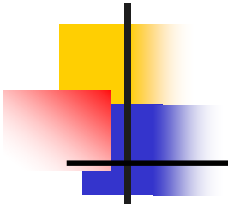
Complex Order: Capabilities

- Performance – OK
- Capacity – OK. Probably few clients
- Extensibility
  - Poor, hard to integrate against, hard to extend
- Reliability
  - (cookie + server-state) OK
  - (just cookie) Poor, takes extra effort to ensure no client manipulations
- Security
  - Not good, vulnerable to stolen cookie style attacks



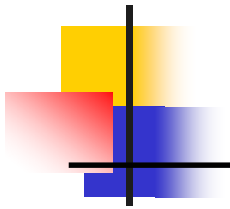
Complex Order: Reality Check

- Borderline realistic
- Lost extensibility – reason for integration
- Too much work to get it running
  - Expensive development
- RMI-IIOP probably better alternative
  - FW + NAT + etc might tilt trade-off



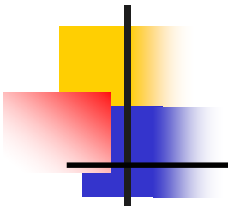
Critical Suborders

- Calling other systems for critical parts
- Cannot complete order w/o suborders
- Technical assistance: trans-action



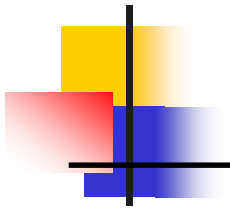
Critical Suborder: Snack Bar

- Burger = bread + meat + salad
- No part missing
- Internal subsystems for
  - Bread
  - Meat
  - Salad
- Integrate against these, *e.g.* bread



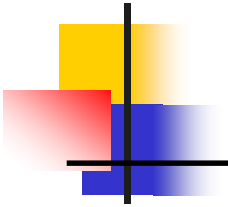
Critical Suborders: Burger

- orderBurger()
  - bakery.orderBread
  - grill.orderMeat
  - garden.orderSalad
- No transaction tracing
- Code around – implement two-phase-commit
- All interfaces become messier



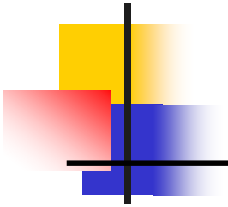
Critical Suborders: Bakery

- startOrder
- orderBread
- cancelOrder
- completeOrder



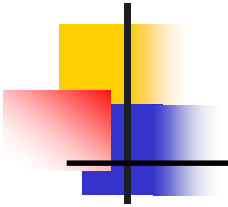
Critical Suborders: Capabilities

- Performance – OK
- Capacity – OK
- Extensibility
  - Catastrophe – everything is hard to understand
- Reliability
  - Really poor – we probably end up w/o quality transactions
- Security – OK



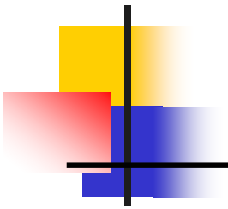
Critical Suborders: Reality Check

- Unrealistic
- Go for something that carries transactions
  - IIOP?
- Note: travel agent example often critical suborders



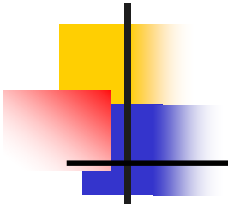
Dynamic Dream

- Find suppliers
  - On the fly
  - Plug and play
  - Hurray



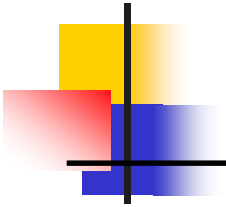
Dynamic Dream: Snack Bar

- Outsourced grill
- Check UDDI for new grill suppliers
- Easy to change supplier
- Pressing costs



Dynamic Dream: Reality Check

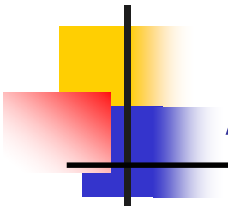
- Start buying meat from unknown supplier?
- Accept hotel room in Bangalore at hotel unknown to travel agent?
- Quality of Business?
- Humans handle dynamic phase
- Technology is often after deal is closed
  - *i.e.* static phase
- However: for internal use …



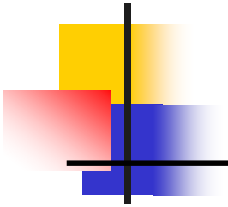
Scenarios

Realistic *Border* *Not realistic*

- Open Info
- Simple Order
- Complex Order
- Critical Suborders
- Dynamic Dream

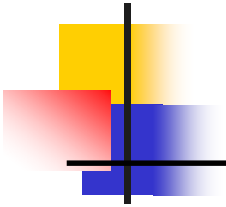


Anything Interesting This Far?



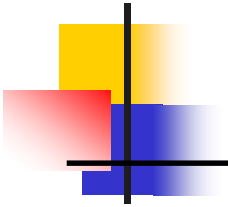
Note on Travel Agency Example

- Common WS "killer app" example
- Travel agents dynamically find travel parts
  - Airline tickets
  - Hotel nights
  - Car rentals
- Composite of
  - (Complex order)
  - Critical suborders (missing flight leg?)
  - Dynamic Dream (unknown hotel in Bangkok)
- Good example of WS capabilities?



High Level WS-specs

- WS-Security
  - Lots of neat stuff
  - Encryption/signatures of XML parts
- WS-Resources
  - Associate state with client
- WS-Coordination/WS-Transaction
  - High level "transactions"
  - Even slow business processes

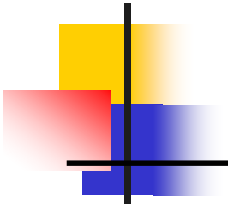


Use It?

Thompson's Rule for first-time telescope makers:

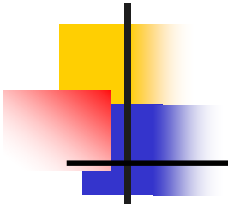
It is faster to make a four-inch mirror then a six-inch mirror, than to make a six-inch mirror.

Do the simplest thing that will work [YAGNI].



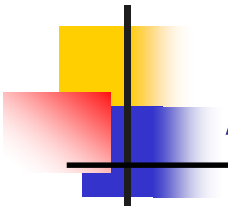
Guidelines: WS Nice Alternative

- Trivial business processes
  - open information
  - simple orders
- Up and running fast
  - FW config not a problem



Guidelines: Reconsider WS

- Large amounts of data
- Tight business integration
  - non-trivial process
  - part of critical superprocess
- Internal use
  - have control over network and deploy



Anything Interesting at All?