

Service Oriented Computing: Opportunities and Challenges

Boualem Benatallah

School of Computer Science and Engineering,
University of New South Wales, Sydney

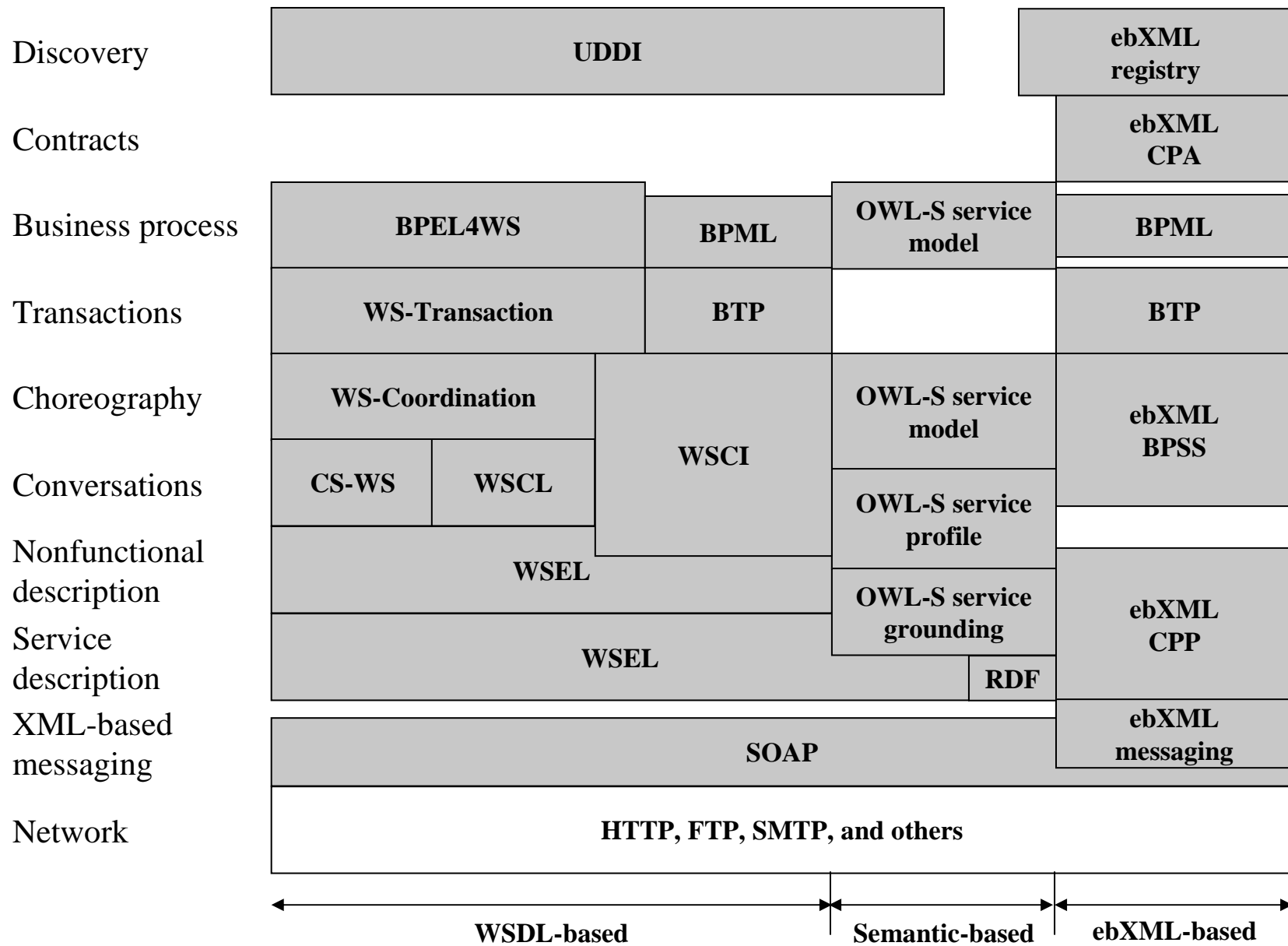
Agenda

- **Service Oriented Computing: Overview and Enabling Technologies**
- **Issues and Directions**

Vision and Principles

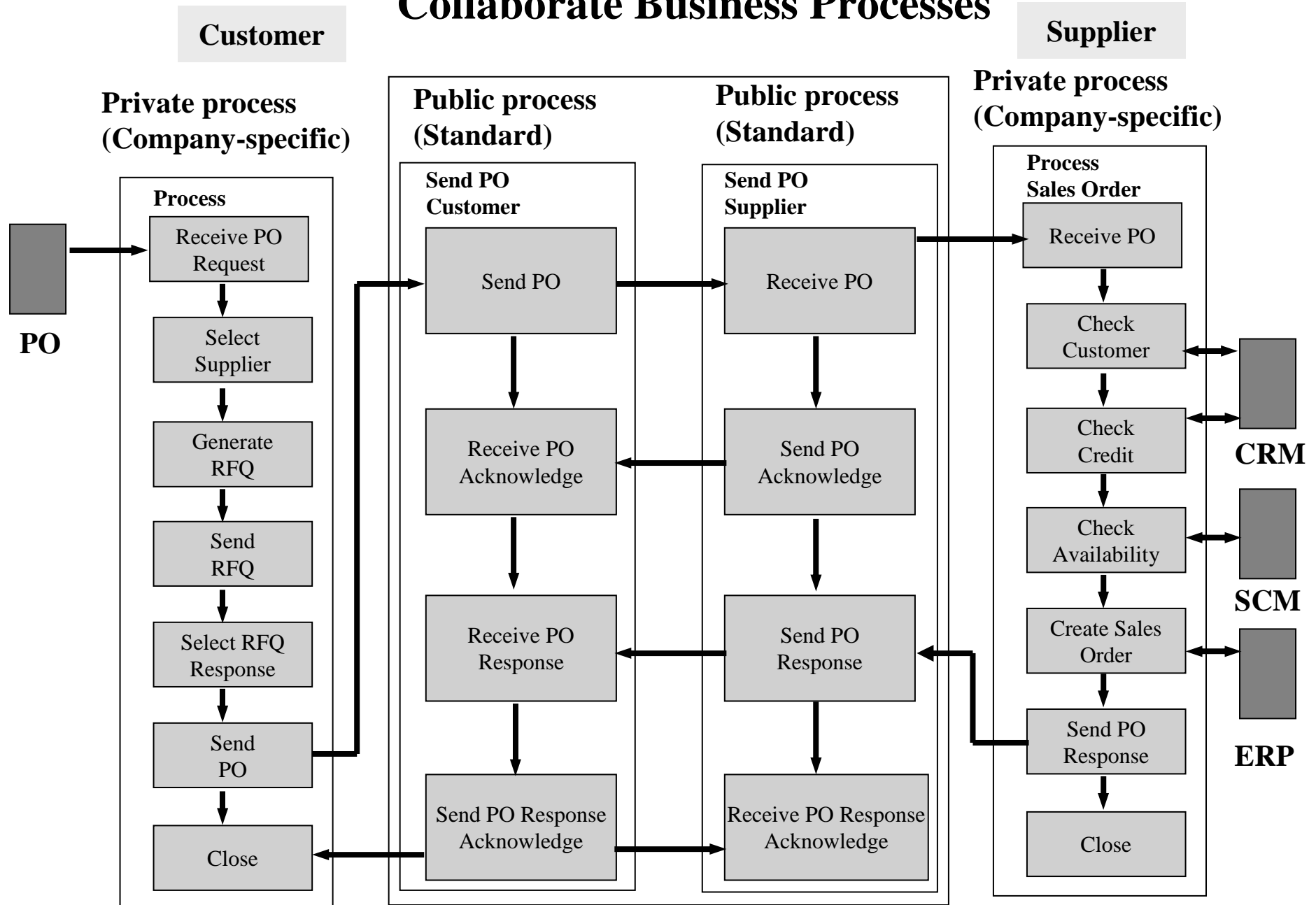
- Allow autonomous partners to advertise their terms and capabilities, and engage in peer-to-peer interactions with any other partners
- Web services: self-described and autonomous software entities, can be published, discovered, and invoked over the Internet (using XML-based standard languages and protocols)
- Resource virtualization, infrastructure simplification and consolidation
- On demand computing through composition and outsourcing

Service Oriented Technologies - Layers



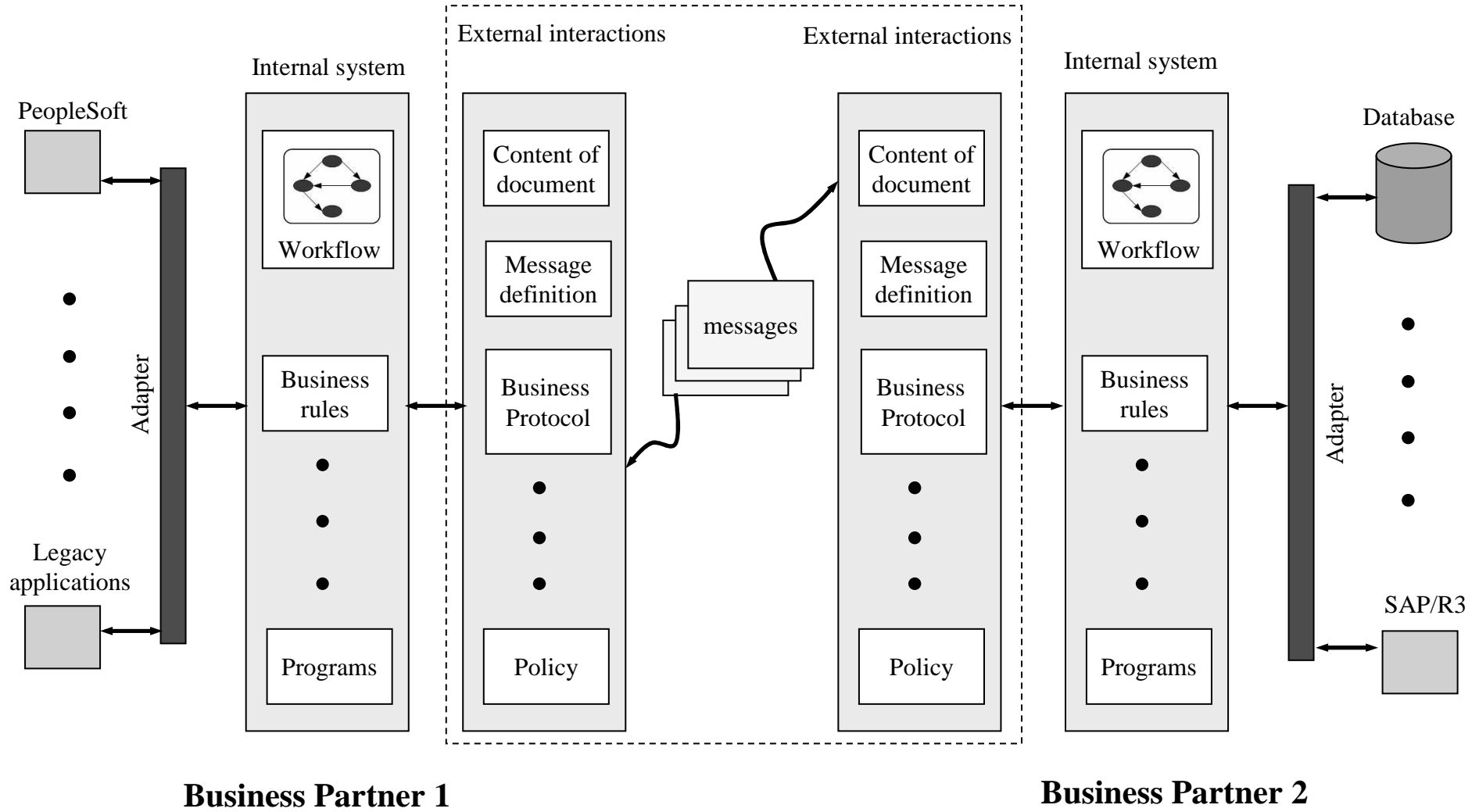
Source (Turning Software to a Service, M. Turner et al., IEEE Computer, Vol 36(10), 03)

Collaborate Business Processes

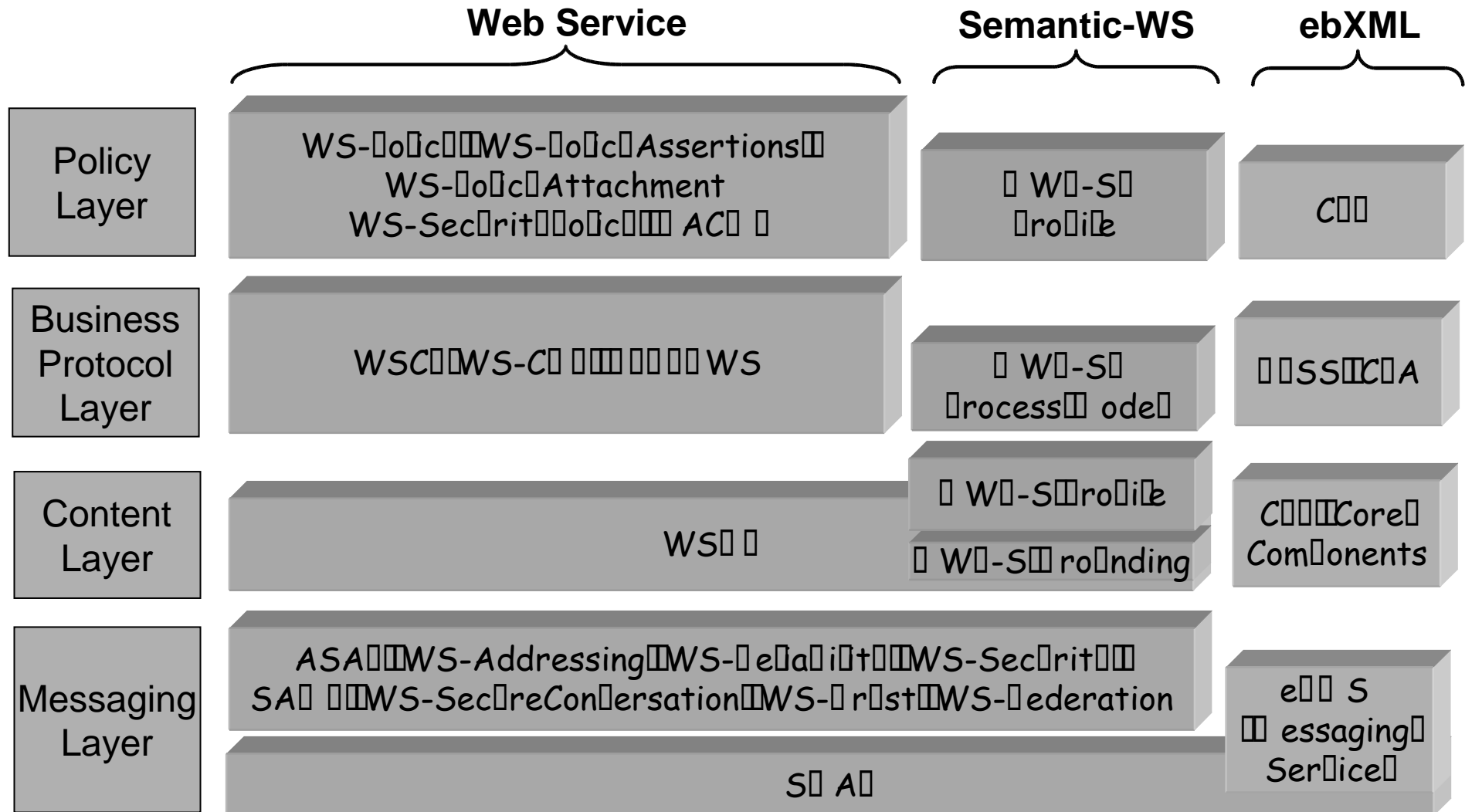


(Source: e-business Architectures and Standards, Anil L. Nori, Tutorial, VLDB'2002, HongKong, China)

Architecture First



Usefulness and Usability Challenges



Interoperability Layers (B2B integration Example)

- **B2B Application**
 - Company A purchasing a product from a company B
 - Agree to collaborate, define collaborative process, and provide means to implement the collaborative process
- **Business process layer**
 - After discovering a match (e.g., using a public or a private registry), A and B need to agree on the joint business process (activities, delivery mode, etc.) and interaction contracts (security, privacy policies, QoS, etc.)
- **Content layer**
 - A needs to know and understand of the product to buy and send a purchase order to B (e.g., product description, order)
- **Communication layer**
 - There must be a way to communicate the messages that contain requests/business documents between A and B.

Content Layer: Message structure and semantics

- Partners must understand the structure and semantics of messages
- E.g., does a document represents a purchase order? A request for quote? A production description?
- Structure: diverse information formats, e.g., XML schemas, text (e.g., different structures for a purchase order), services may provide same functionality but with different operation structures (e.g., different names, different signatures)
- Semantics: Does a service provides a required functionality? does *Price* means *Price* including *tax*?
- Service Retrieval : similarity based on inputs/outputs of operations (Woogle project, University of Washington)

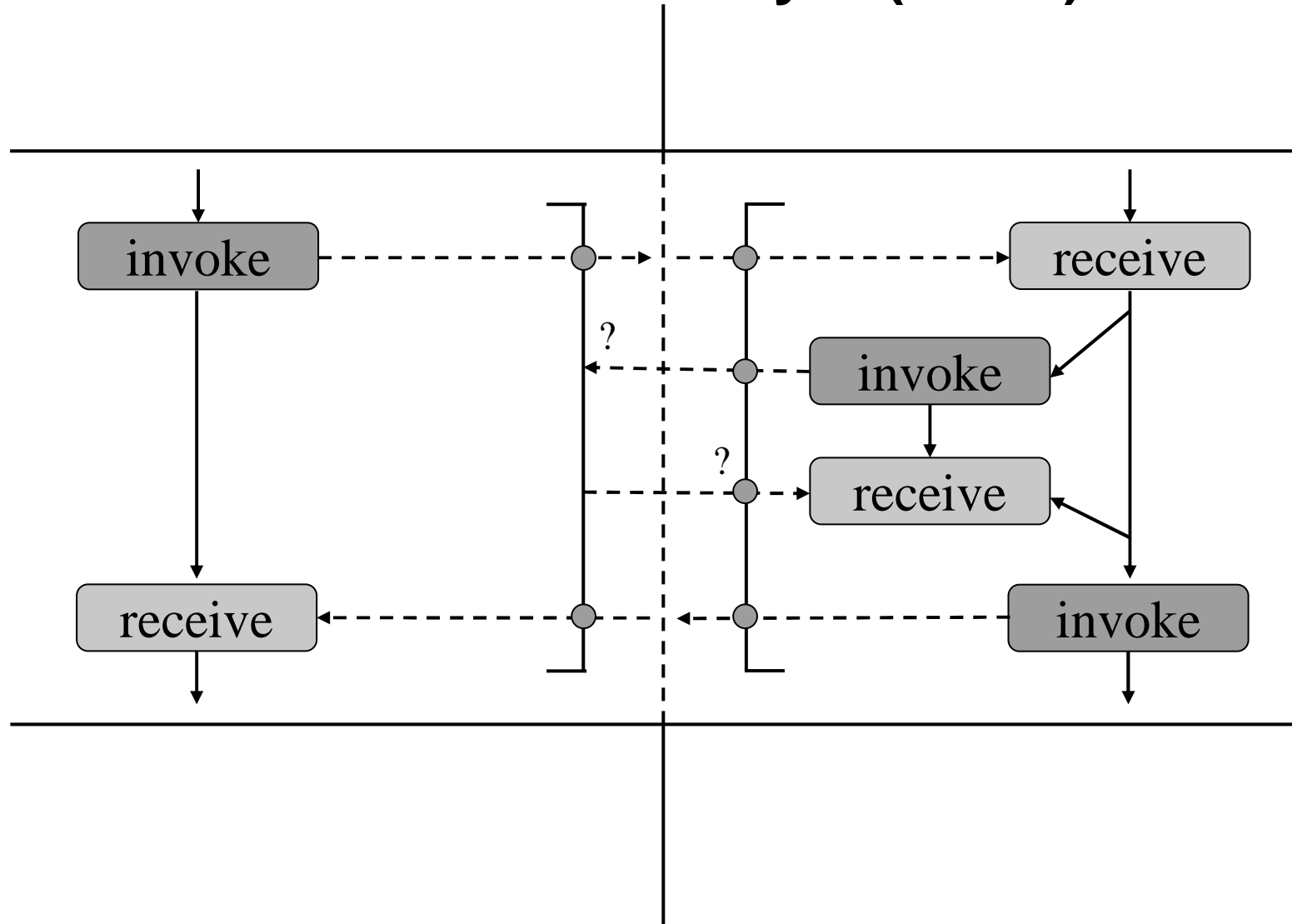
Message structure and semantics (Cont.)

- Existing solutions (e.g, schemas mapping in DBs) use linguistic / structural / ontology analysis, service descriptions (WSDL) are usually very compact (may not benefit directly from previous solutions)
- Large banks may have as many as 1300 trade counter-parties, other businesses may have over 2,000 partners (source: ContextMetrics, Forward Look Corporation), one-to-one mappings between partner systems do not scale
- If used, ontologies will help
- Message standards (e.g., EDI) usage may not be followed because of concerns such as complexity and volatility of business requirements
- *Fully automated and on-demand integration is a very far reaching objective*
- *What seems realistic is to provide tools support and methodologies for error-prone and time consuming integration activities (e.g., discovering description mappings)*

Business Protocol Layer

- Semantics of interactions (joint business process)
- Partners must agree on the choreography of interactions and meaning of messages
- E.g, steps (send order, process order, deliver product), deals (a purchase is refundable after 2 days)
- Semantics of interactions must be well defined, such that there is no ambiguity as to:
 - What a message may mean? What actions are allowed? What responses are expected?
- For example, if a company A requires an acknowledgement of purchase orders from its partners, then partner processes must have a corresponding activity

Business Protocol Layer (Cont.)



Business Protocol Layer (cont.)

- Interoperability at this layer requires the understanding of the behavior of partner public processes (called external conversations, business protocols)
- Traditional EAI middleware
 - component interface describes very little semantics (e.g., message formats)
 - business process is usually agreed upon off-line.
 - Examples: CORBA-based solutions
- Service oriented architectures: richer interface descriptions are needed (services should be self-describing, whatever this means)
- Automation requires rich description models but a balance between expression power and simplicity is important for the success of the technology (expressive: useful and usable)

Service Protocols and Policies: New Interoperability Dimension

- Making implicit information (as in closed environments) explicit (essential in autonomous environments)
- Messages order (e.g., buy after login)
- Transactional implications (e.g., can I cancel a purchase?, if yes at what cost)
- Temporal aspects (e.g, can I cancel a purchase any time? After a fixed time period?)
- Security (will the results be digitally signed?)
- Privacy (How do you know if partners have compatible policies?)
- Quality of service (e.g., performance/reliability)
- Exception Handling (e.g., support for transaction protocols)

Service Protocols and Policies (cont.)

- Compliance and Monitoring: Does a service implementation / composition satisfies its policies? How can this be monitored? Does a privacy policy complies to law?
- Compatibility: Does the assurances of provider satisfies the requirements of a client? Static Vs Dynamic Compatibility Checking ?
- Changes management: Can I safely use a new version of a service (replace-ability) ? What impacts a change will have on agreed upon interactions?
- Federated Identity Management: How is authentication and authorization handled in composite services ? How is information regarding identities, attributes and authorizations transferred between services, and what issues does this raise in regards to privacy?

Service Protocols and Policies: Some Principles for Solutions

- Identify abstractions (vocabularies and models): useful and usable machine-enforceable descriptions
- Preferably modelling should use same underlying core constructs + specifics
- What languages to use? WS-* languages ? Process specification languages (e.g., pi-calculus) ? OWL-S?
- Low level languages are intractable to be usable for automating integration activities
- Existing high level process specification languages do not explicitly take important service abstractions (e.g., privacy and transactional implications) into account
- Identify intra and inter- abstractions relationships (e.g., compatibility, replace-ability, refinement)
- Operators to manipulate abstractions at a high level (e.g., identifying similarities and differences, transformations, change management)
- Data Analysis for auditing, compliance, and improvement.
- Methodology and tool support for the above

What do we need to get there?

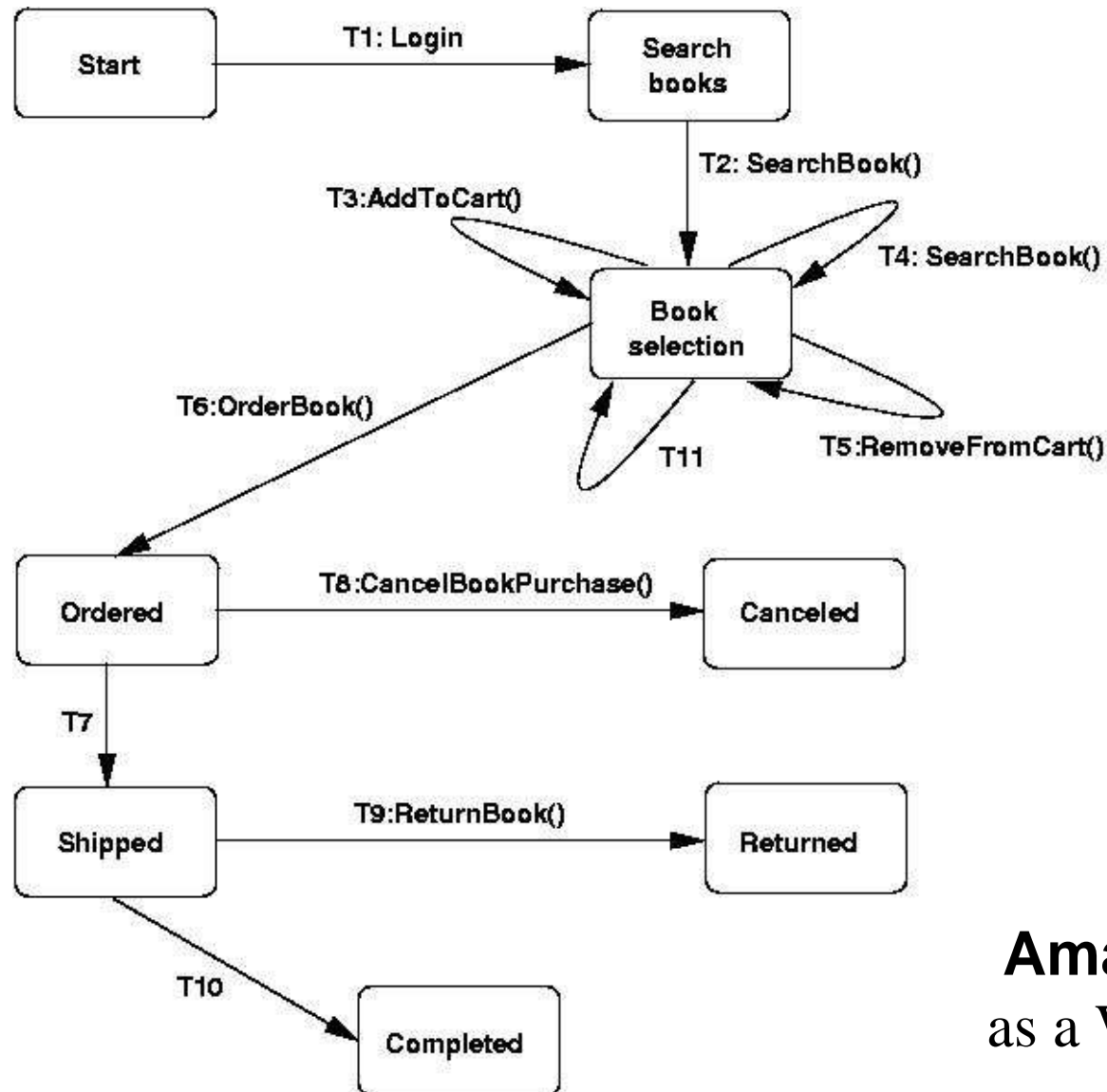
- Models, languages
 - hopefully standard
- Protocol analysis and protocol algebra
 - comparisons, query,.....
 - also in connection with service composition
- Development and runtime tools

Transactional implications and temporal aspects of business protocols

WEB PORTAL	SERVICE	BEGIN	COMMIT	ROLLBACK	LOCK/HOLD	COMPENSATE
Amazon.com	Sell Books	At checkout time when customers finalize the purchase of one or more items	At book shipping time	Cancel via phone, web, or email before product shipped	None	Return within 30 days for refund if in good condition (or in ANY condition for books recommended by Amazon). MANUAL PROCESS
Travelocity.com	Rent cars	At checkout	At end of checkout process	N/A	None	Rental may be compensated within a time T from the pick up time, otherwise a fee F needs to be paid. T and F vary by company, car size, agreed rate, etc.
Expedia.com	Sell flight tickets	At checkout	At end of checkout process	Session expires (no explicit cancel required) before ticket issued	Locks seats on hold until 12am next day (available only for certain flights). Does not guarantee fare	Return: depends on agreement with airlines, fare selected, etc. It is therefore on a case-by-case basis. E-PROCESS

(Source: Conceptual Modeling of Web service Conversations, Benatallah, Casati, and Toumani, Hamadi, CAiSE'04)

Business Protocols Modeling: Embryonic Conversation Model



Amazon.com
as a Web service

Identified Business Protocols Abstractions

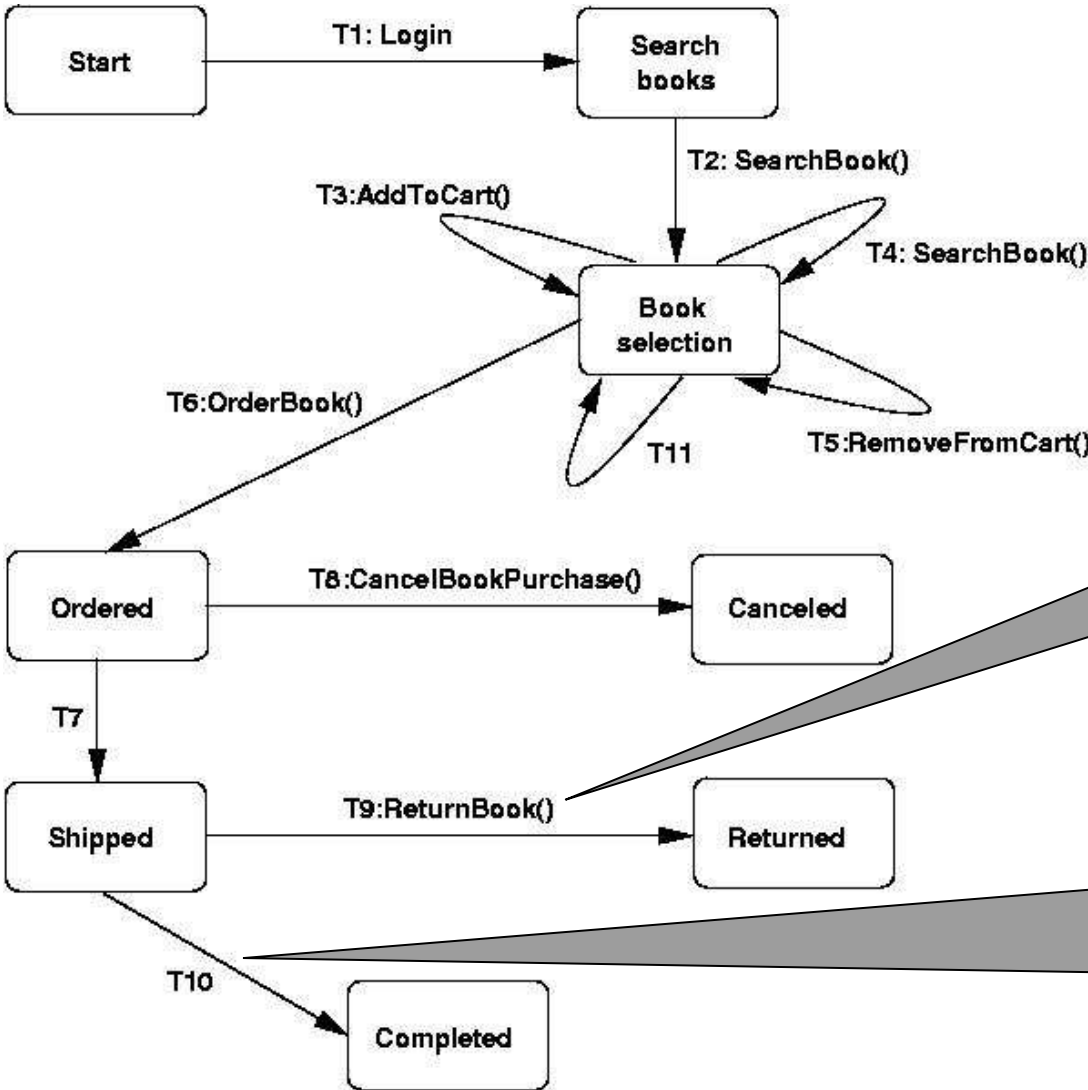
Activation Abstractions

- Implicit & Explicit operations
- Timed operations

Completion Abstractions

- Implications and effects of operations from the requester perspective
- Compensating operations (temporal conditions and cost)
- Resource locking operations

Modeling Activation Abstractions



- Examples:

mode := "user"
event := "ReturnBook"
O-condition := True
U-condition := membership = 'gold'
T-condition := "within 30 days after end(T7)"

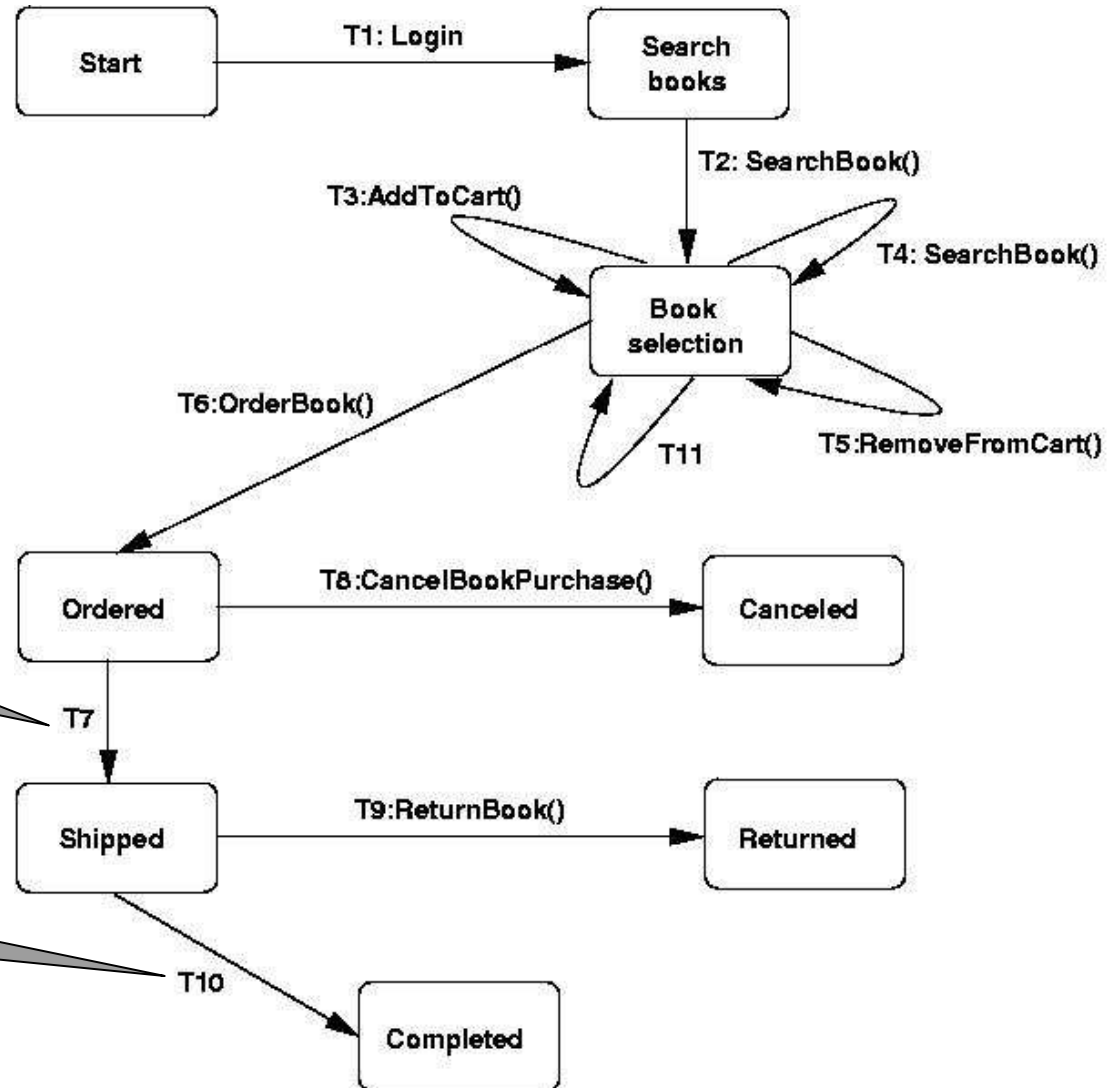
mode := "provider"
event := "30 days after end(T7)"
O-condition := True
U-condition := True
T-condition := True

Modeling Completion Abstractions

- Specifies the effect and implications of a transition on the requester side:
 - **Effect-less**: no effect on the requester
 - **Credential-disclosure**: the requester is required to reveal certain credentials but no effect on the requester
 - **Definite**: the effect is permanent and cannot be compensated
 - **Compensatable**: some effect on the requester that can be undone
 -

Compensation Property

- Examples:



name := "T7"
type := "Compensatable"
compensation-transition := "T9"
cost := 10% * books_price

name := "T10"
type := "Definite"

Resource Locking Property

- Specifies temporary reservation of service provider resources for a requester when invoking a transition
- Two types of resource locking:
 - **Lock (L)**
 - **Tentative-lock (TL)**

Trust Negotiation

- Goal: Identify *abstractions* to model trust negotiations, and *operators* to manage trust negotiation policies
- Defined simple model for trust negotiation policies based on state machines
- States represent level of trust, transitions used to enforce policies
- Resources (credentials and service operations) mapped to roles, roles mapped to states
- Authorization abstractions: Attach conditions to transitions. Include credential disclosure, provisions, obligations, timeouts

Summary

- Service oriented computing offers tremendous opportunities for agile and Internet scale integration
- XML-based standardization is a key to interoperability
- But critical interoperability challenges still need to be addressed (service protocols and policies)
- Tools support and methodologies for error-prone and time consuming integration activities (model management and analysis for web service protocols)

References

Web Service Conversation Modeling (IEEE Internet Computing, Jan 04, CAiSE'03, Benatallah, Casati, Toumani, Hamadi)

Generation of Composition Skeletons from Conversation Models (CAiSE'04, Baina, Benatallah, Casati, Toumani)

Trust Negotiation Management (Modeling, Changes Management, IEEE Internet Computing, Nov. 03, WWW'04, Skogsrud, Benatallah, Casati)

Service Protocols Analysis and Management (ER'04, Benatallah, Casati, Toumani)