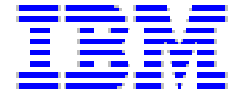




e-business

ICC 2001, Session G50: Service Management Challenges



Determining Service Dependencies in Distributed Systems

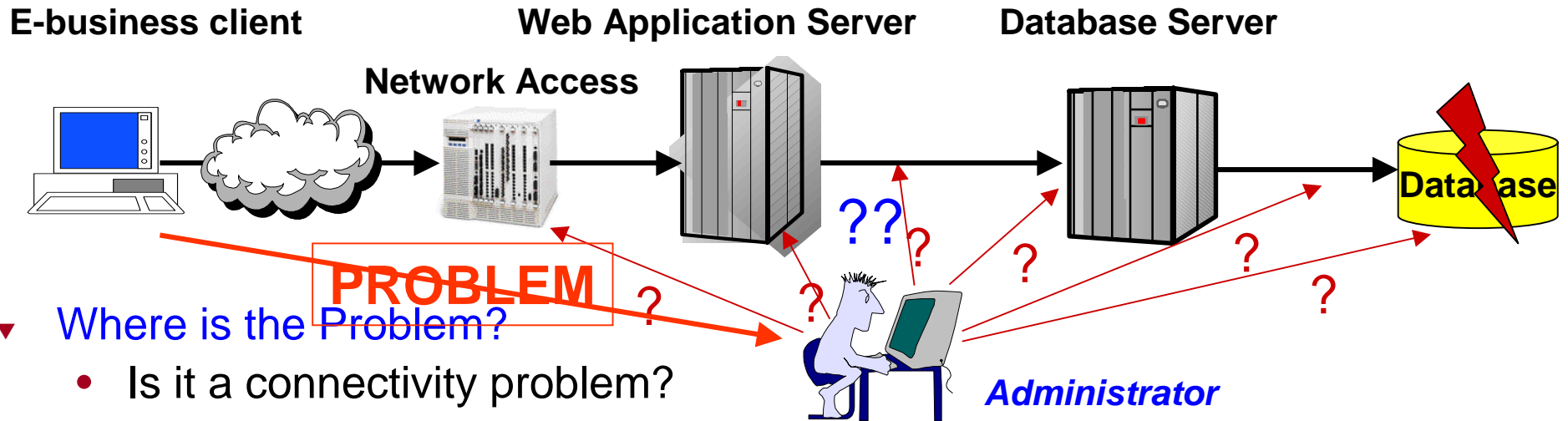
Alexander Keller, Gautam Kar

IBM T.J. Watson Research Center

Yorktown Heights, NY, USA

<http://www.research.ibm.com/sysman>

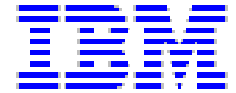
Motivation: Managing a Web Storefront



- ▼ Where is the Problem?
 - Is it a connectivity problem?
 - ... a Web Application Server Problem?
 - ... a Database Server Problem?
 - ... an internal Database Error?
 - How are the different Components related?
- ▼ Mapping "Service ▷ Implementation (Product) ▷ Running Instance" essential:
 - obtaining the three Parts individually is tractable
 - combining the three Parts into a uniform Model is challenging
 - finding an efficiently computable Dependency Model is hard



e-business



Dependencies: Management Issues & Requirements

▼ Root Cause Analysis

- Determine and isolate the Source of a faulty Service
- **Today:** manual Test of Services by Operator, Service Dependencies not explicitly specified
- **Needed:** “drill-down”, (automated) traversal of Layers to find Root Cause

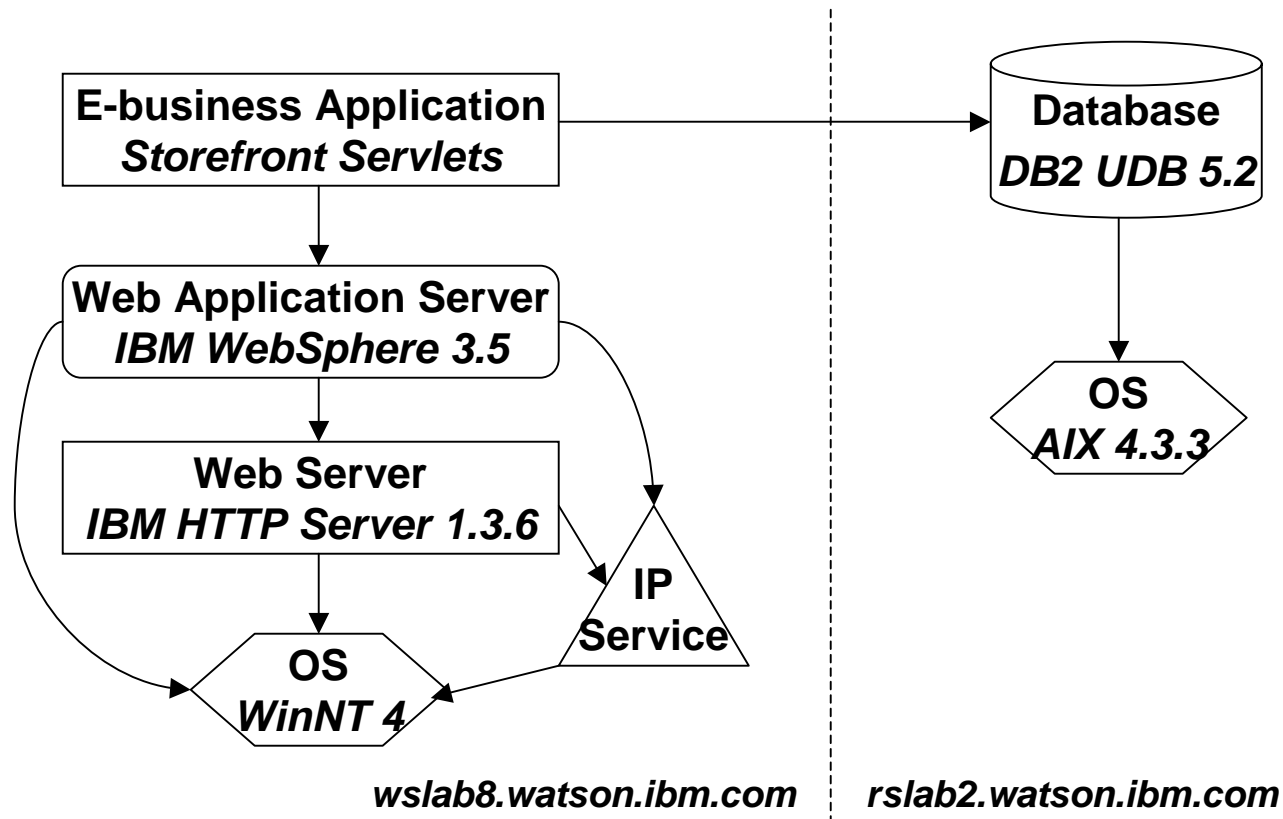
▼ Impact Analysis

- Determine which Services/Customers are affected by a Problem
- **Today:** Planned Outages: Proactive Warnings, Accidental Outages: N/A
- **Needed:** “drill-up”, determine potentially affected Services/Customers

➔ Requirements

- Make Service Dependencies explicit, available & efficiently computable
- Define an open Format for interchanging Dependency Information
- Dependency Model must allow (recursive) Queries

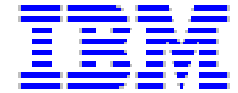
Dependency Graph of a Web Storefront



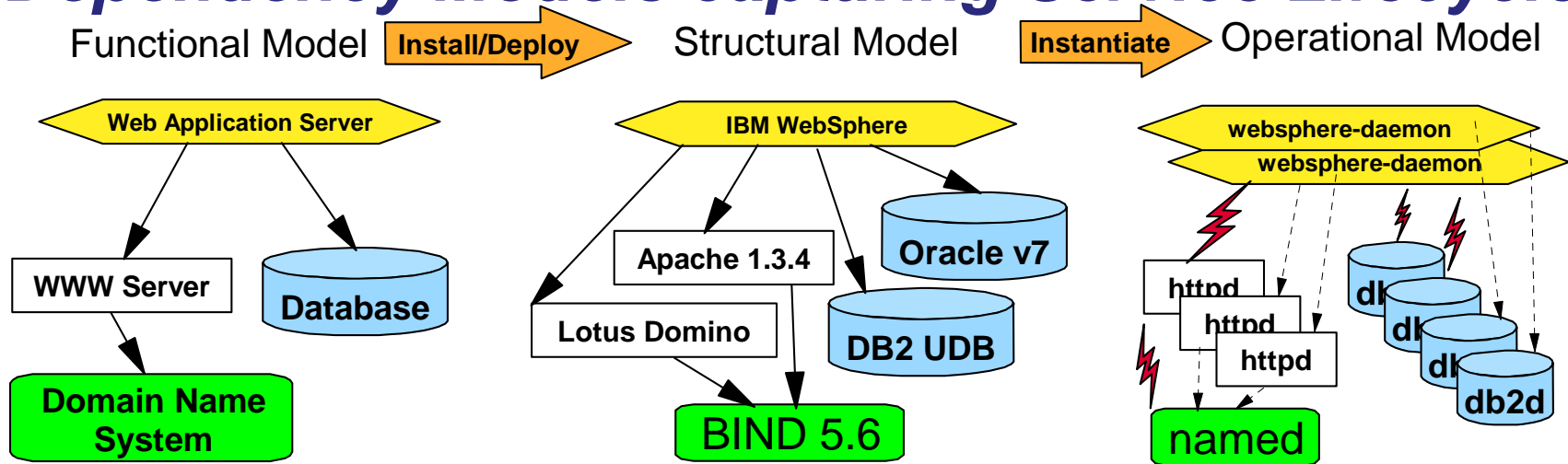
- ▼ Dependencies usually span multiple Systems and Domains (SLAs)
- ▼ Representation of Dependencies as directed, acyclic Graph
- ▼ Nodes represent Services, Edges represent Dependencies



e-business



Dependency Models capturing Service Lifecycle



▼ Functional Model

- defines dependencies between generic services (not: within)
- further models are based on functional model and refine it

▼ Structural Model (Manageability)

- detailed description of SW components based on system inventory
- typically captured during installation/deployment

▼ Operational Model

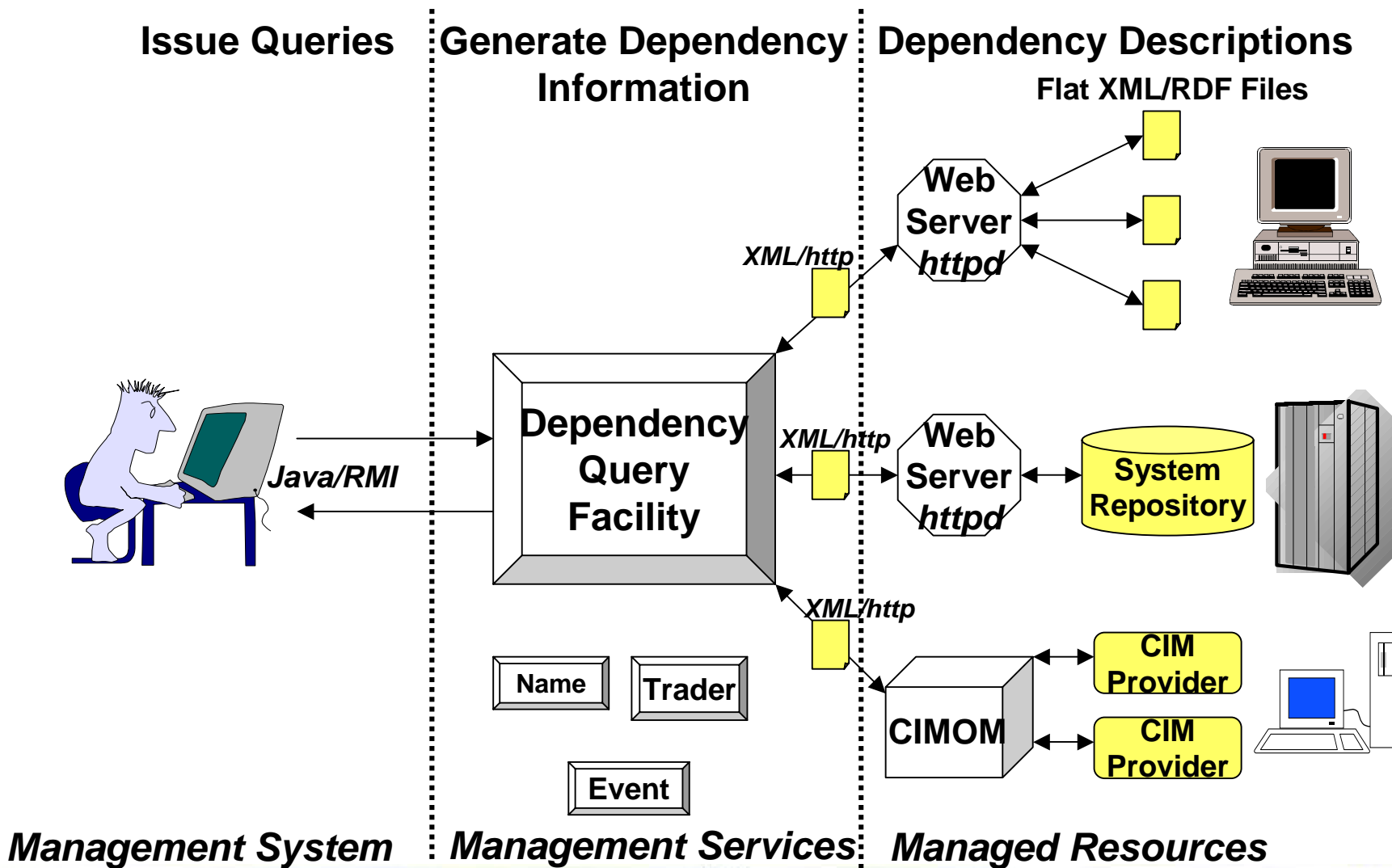
- created when bindings between services are established
- highly dynamic - not stored but computed stepwise as needed



e-business

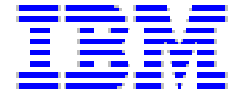


Distributed Dependency Processing: Architecture





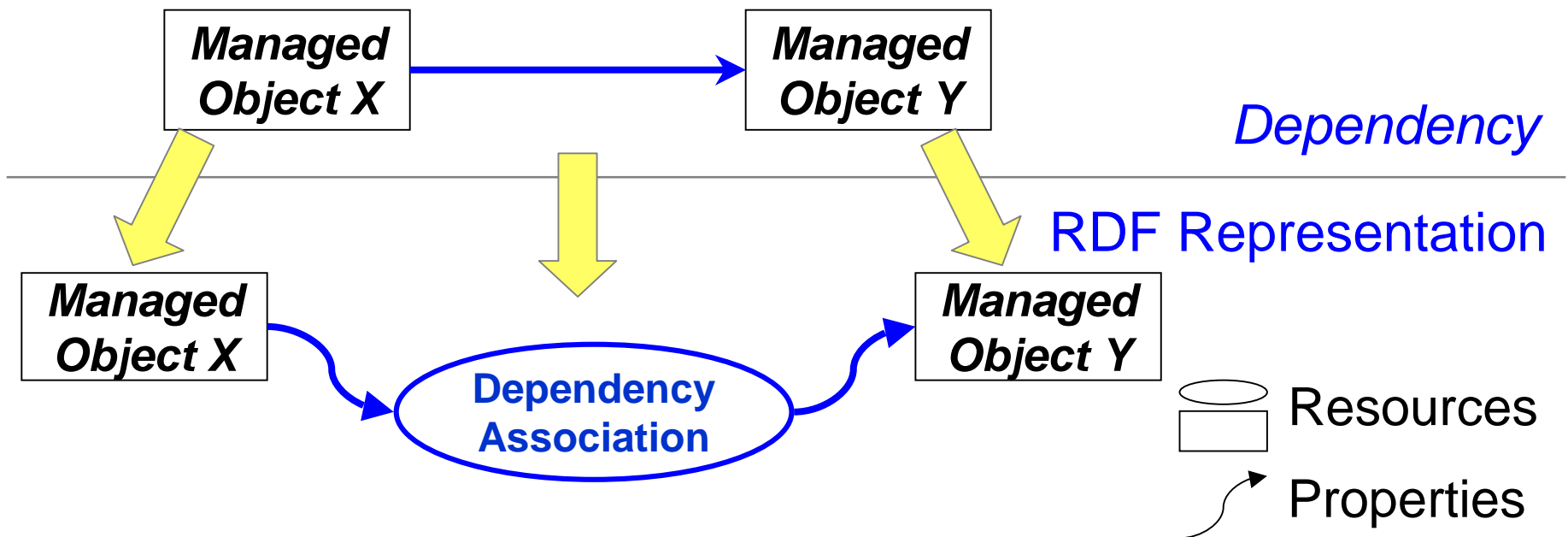
e-business



Why use the Resource Description Framework?

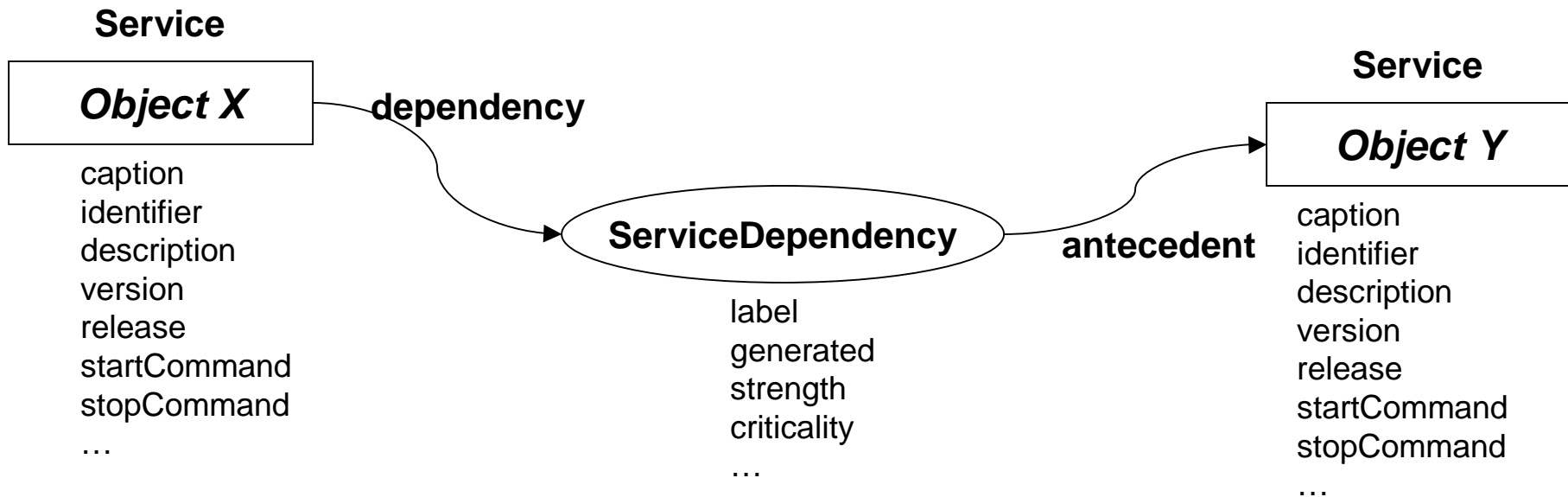
- ▼ **Open Standard (W3C Candidate Recommendation)**
 - Defines Metadata for XML Tags
 - Initially conceived for content classification of Documents
 - RDF Documents are XML Documents
 - ▼ can be parsed with off-the-shelf XML Parsers
 - ▼ can be queried with XML Path Language (XPath)
 - Type system: Distinction between MOs and Dependency Objects
- ▼ **Applicability to our Work:**
 - RDF is able to represent DAGs in (different) XML Documents
 - Nodes in a DAG can have attributes and reference other nodes
 - Dependencies can be qualified with attributes for classification
 - ▼ E.g., “start”, “stop”, “failover”, “strength”, “criticality”, “generated” etc.
 - ▼ Allows searching for MOCs, dependency types and their properties

Expressing Dependencies in RDF



- ▼ Straightforward 1:1 Mapping for Managed Objects
- ▼ Dependency Association is an Object on its own (can have Attributes)
- ▼ Managed Objects and Associations are *RDF Resources*
- ▼ Links between MOs and Association are *RDF Properties* of the resp. Objects

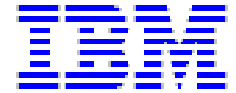
Elements of a Dependency Description



- ▼ Service: Managed Object with various Attributes
- ▼ dependency: RDF property of Service
- ▼ ServiceDependency: Attributes help classify Dependency
- ▼ antecedent: RDF property of ServiceDependency
- ▼ Dependencies modeled as unidirectional Links



e-business

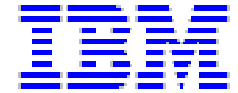


Querying Dependency Graphs with XPath

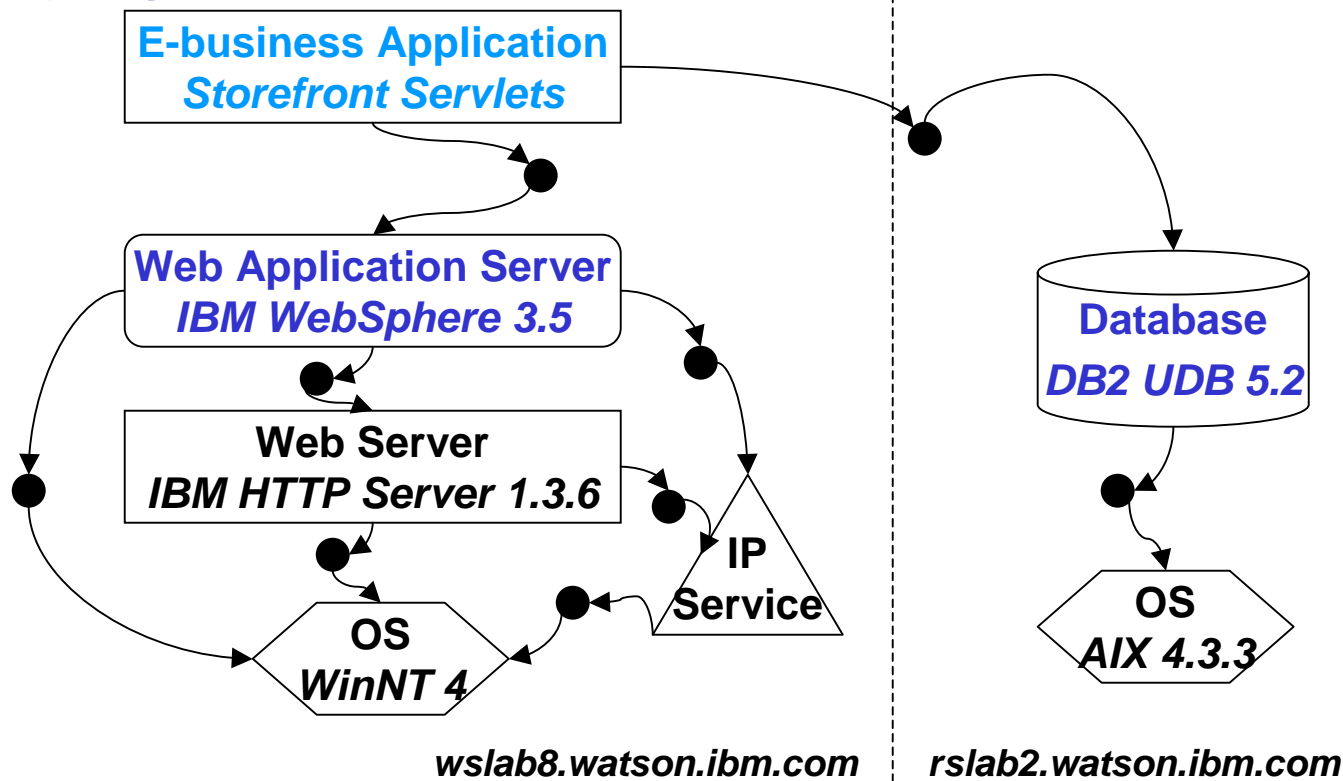
- ▼ Open Standard (W3C Recommendation)
- ▼ Combine various XML Documents into a new Document
- ▼ Compact, reusable Query Expressions
- ▼ Used here for:
 - Navigating Dependency Graph and selecting Nodes
 - Applying filtering Rules to the Nodes
 - Building an XML Document containing the Results
 - cf. OSI “Scoping and Filtering”
- ▼ **Dependency Graphs can be navigated in both Directions**
 - From top to bottom: finding Root Cause
 - From bottom to top: performing Impact Analysis
 - Either step-by-step or recursively (“Drill-down”, “Drill-up”)



e-business



Applying XPath: Antecedents of E-business Application



```

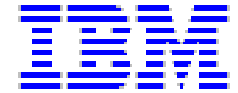
/descendant::[(self::ds:Service)]/child::ds:dependency/
*[(self::ds:ServiceDependency)]/child::ds:antecedent/
@rdf:resource

```

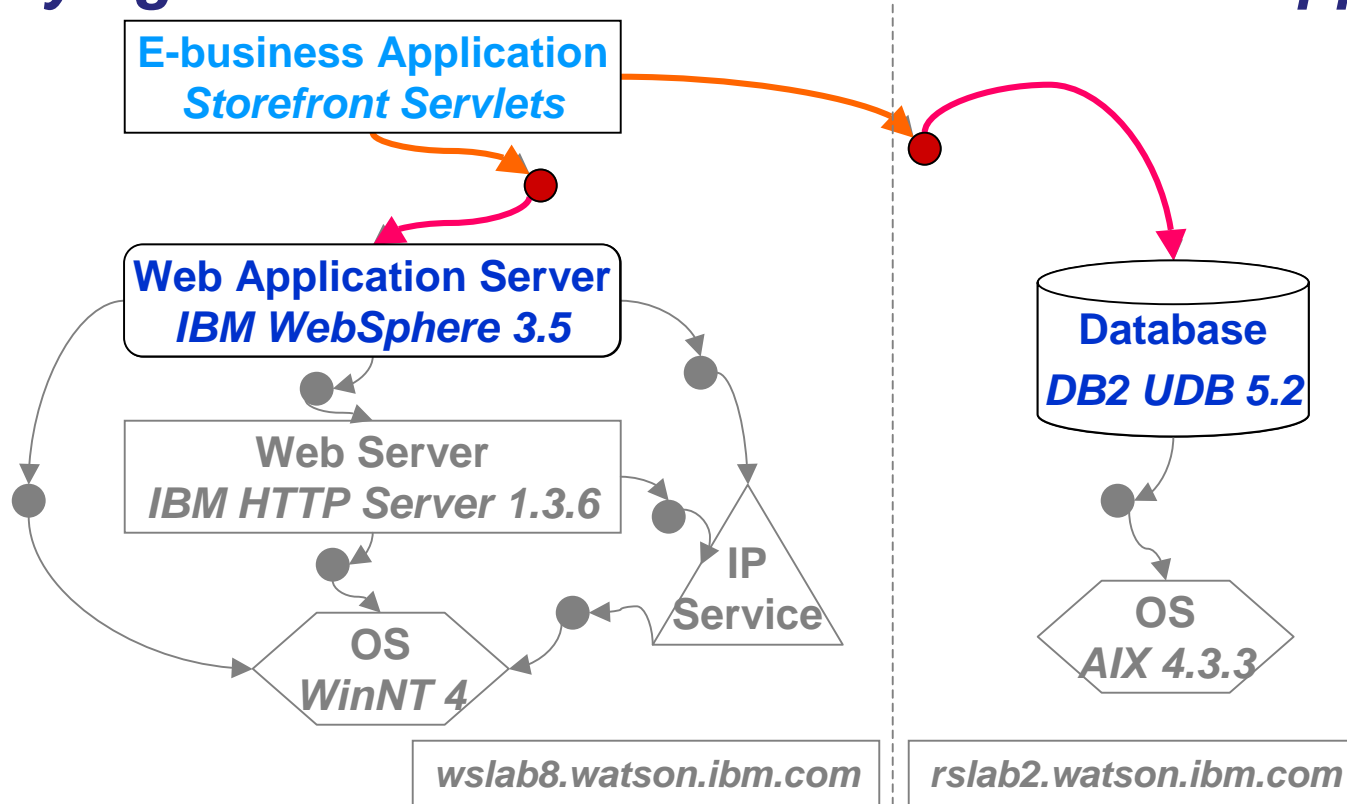
➤ Yields the URIs of **Web Application Server** and **Database**



e-business



Applying XPath: Antecedents of E-business Application

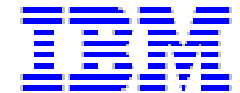


```

/ descendant::[(self::ds:Service)] / child::ds:dependency /
* [(self::ds:ServiceDependency)] / child::ds:antecedent /
@rdf:resource

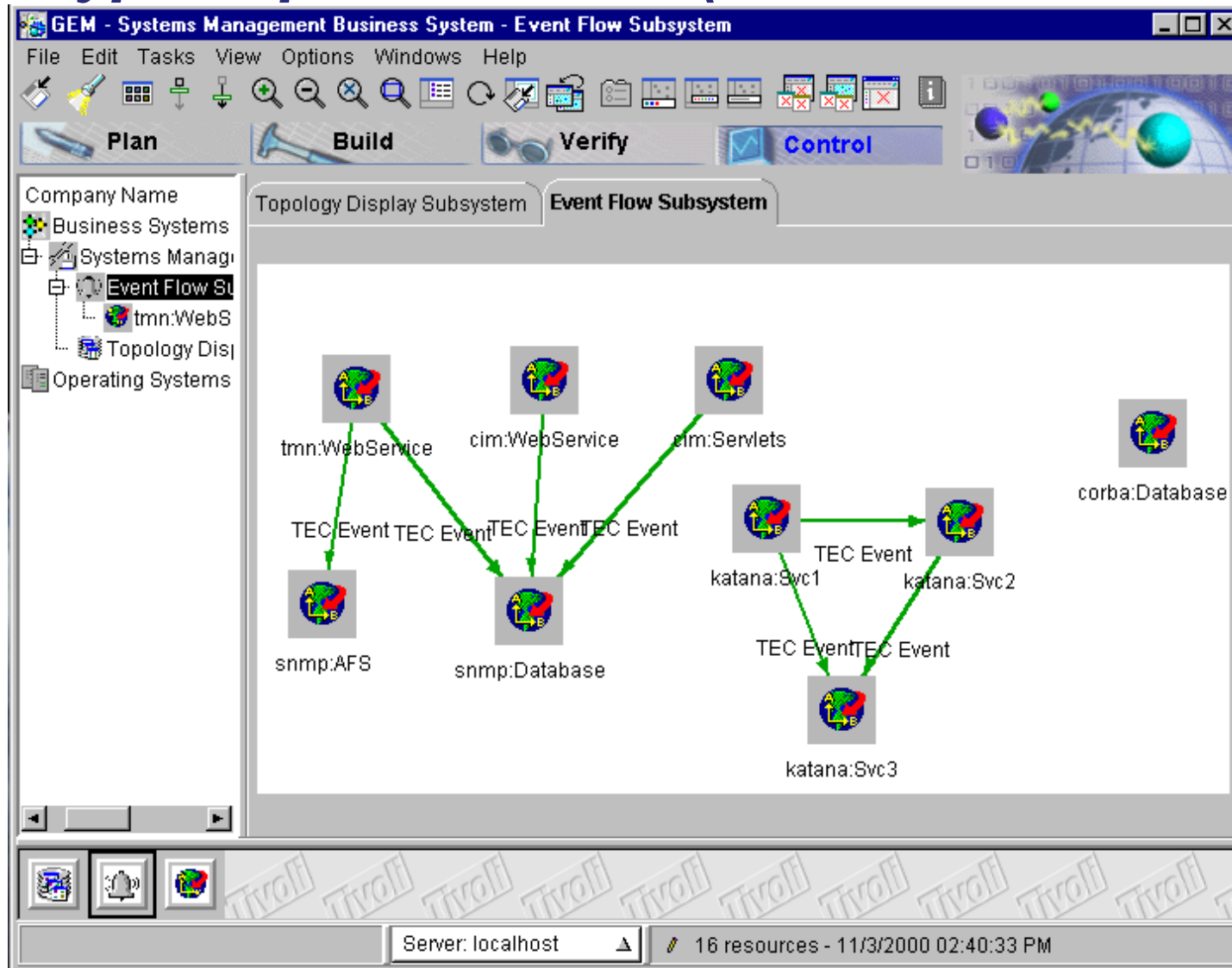
```

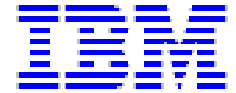
➔ Yields the URIs of **Web Application Server** and **Database**



e-business

Prototype Implementation (based on Tivoli GEM)





Conclusions and Outlook

▼ Results:

- RDF for representing DAGs in XML, hierarchical Type System
- Exploit inherent Strengths of widespread Representation Format
- Minimum overhead by delegating the bulk of work to XML tools
- XPath as a straightforward means to:
 - ▼ Navigate Dependency Data
 - ▼ Issue (recursive) Queries against multiple distributed XML Documents
 - ▼ Combine Results into one XML Document

▼ Current work:

- Distinguishing between Instances of hosted Applications requires further Instrumentation
- Optimizing memory consumption by SAX or efficient DOM Parsers
- Integration with available CIM Object Manager
- Propose Extensions to CIM RunTime Application Working Group