E6998-04: Web Application Servers – Architecture and Design

Dr. Donald F. Ferguson,
IBM Fellow
Chief Architect, IBM Software Group
dff@us.ibm.com, or donff2@aol.com
BLOG:
http://www.ibm.com/developerworks/blogs/page/donferguson

Agenda

- Session I: 11:00 – 11:50
  - About me
  - Class overview and process
  - The evolution of the Web
  - Role of Web Application Server in applications
  - Discussion
- Break/Discussion
- Session II: 11:55 – 12:50
  - Workload and Availability Management
  - The Process Model and Thread Model
  - Protocol Support
  - Infrastructure Services
  - Caching
  - What is a Container? Containers
  - Some Containers
  - Assignments

Agenda

- Session I: 11:00 – 11:50
  - About me
  - Class overview and process
  - The evolution of the Web
  - Role of Web Application Server in applications
  - Discussion
- Break/Discussion
- Session II: 11:55 – 12:50
  - Workload and Availability Management
  - The Process Model and Thread Model
  - Protocol Support
  - Infrastructure Services
  - Caching
  - What is a Container? Containers
  - Some Containers
  - Assignments

Qualifications

- PhD. From Columbia University in 1989
- IBM Research from 1987 until 1998
- IBM Software Group from 1998 until present
  - 30,000 employees
  - $16B in revenue
- Roles
  - Chief Architect for WebSphere from initiation until 2001
  - Chief Architect for all SWG products from 2001
- Some contributions
  - About 10 patents
  - About 25 publications
  - Co-Authorship of several standards in Java 2 Enterprise Edition and Web services.
  - Lot’s of lectures and keynotes

Class Overview and Process

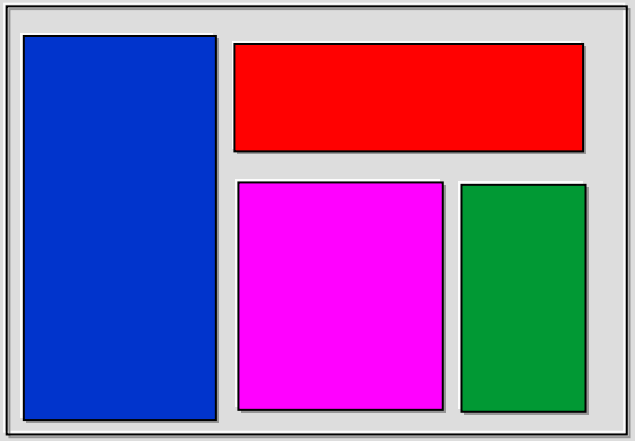
- Encourage interaction
  - There are no stupid questions, so please ask.
  - I will ask questions of the class
- Refine the class based on your feedback
- We will run the class like a product development architecture team
  - Logically design a Web application server, at a high level.
  - Produce and review
    - Concept Design Documents
    - (Small) System/Subsystem Architecture and Design Documents
  - Student presentations and peer reviews
- Grade
  - Class participation
  - Presentations, and CDDs/SDDs
    - Authoring
    - Review
  - The same way we “grade” and “promote” in the technical teams
- There are no finals and midterms in real life

Web 1.0 – HTML and HTTP

- The Web was originally a Web of documents
  - Rich text {Heading, ordered lists, bold face, etc}
  - Imbedded images, figures, etc.
  - HTML links between pages for navigation.
  - URLs/URI identified pages, embedded content and links
- The “application server” was basically a “file server”
  - URI identified the file.
  - URL
    - www.columbia.edu identified the file server
    - …/foo/bar.html identified the file
  - Basic set of protocol verbs: GET, POST, DELETE, … …
  - Simple functions, e.g. security
- Simple functions, like
  - Authentication and authorization
  - Request spraying

Web 1.5 – Dynamic Content

Role based navigation menu



New products/changes since last logon

My orders and status

My account status

- Page (document) becomes specific to user and task
- Cannot “pre-render” all pages, unlike previous paper based approaches. (Why?)
- Content depends on
  - Information in databases, applications, etc
  - Changed through other UIs and processes, e.g. teller, EFT, …

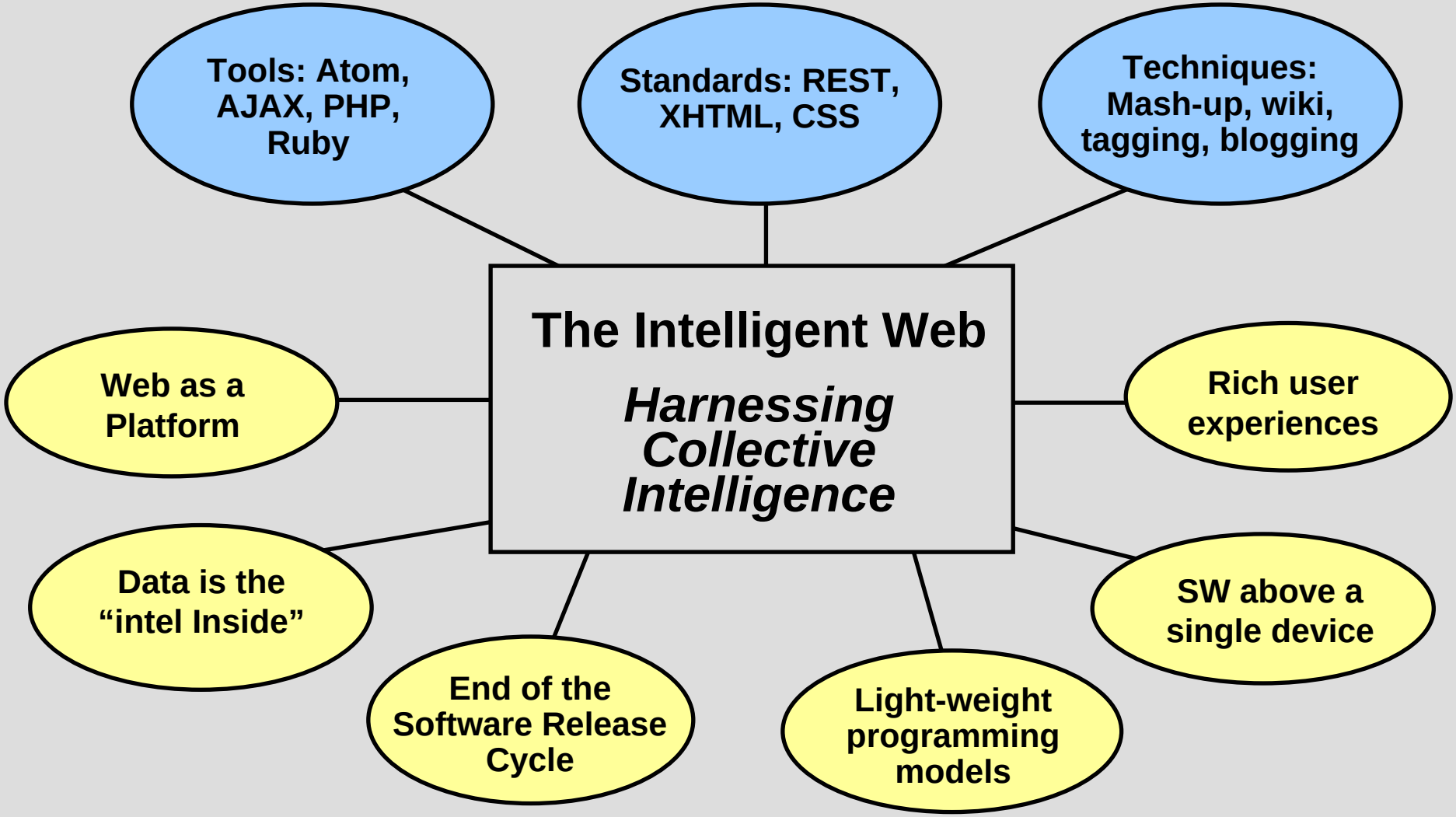
Some Additional Concepts

- Dynamic, data driven content technology was diverse
  - CGI, C, C Lib
  - Proprietary extensions to HTML for access to data
- Evolved to standards (Why?)
  - Java, JavaServer Pages
  - PHP
  - etc
- Moving beyond file systems and databases
  - Hierarchical database of content, with links
  - Content format and render templates
  - Metadata tags and properties
- Personalization, e.g. Amazon
  - Tailor “helpful” content on the page based on CM, business goals, experience with user, etc.
  - Based on simple rules
- Customization, e.g. My Yahoo
  - User selects from a portfolio of content (portlets)
  - What the user wants to see, unlike personalization
- Multiple Devices: PDAs, voiceML, …

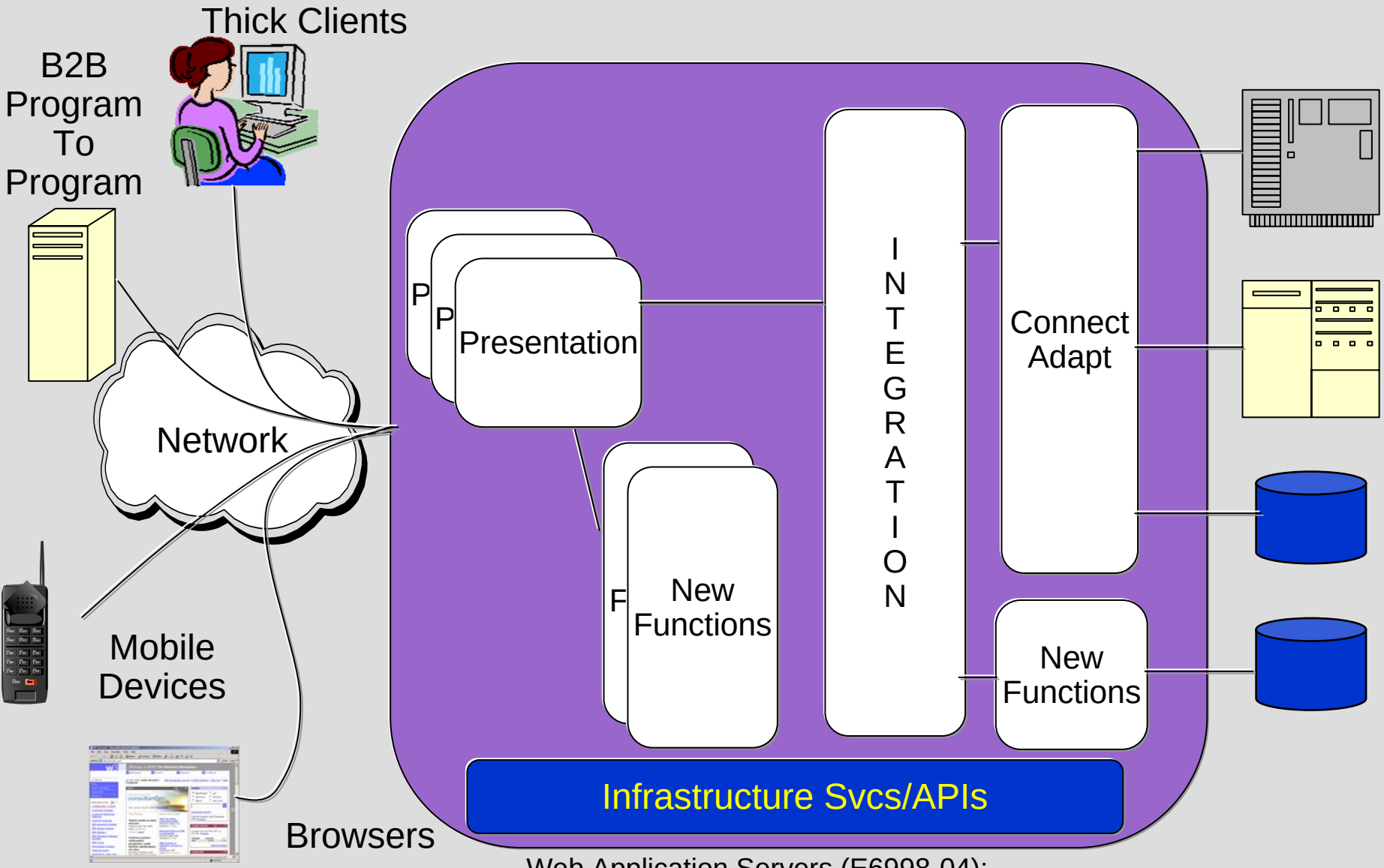
Web 1.8 – Web Services

- A Simple Business Problem: Think about three corps.
  - IBM buying office supplies
  - Staples providing supplies
  - Ups providing shipping
- Phase I – Mail, Phone, Fax
  - Companies communicate via mail, paper forms, fax, etc
  - Employees interacted with intra-enterprise applications through terminals, based on data in forms.
- Phase II – Web Documents and Forms
  - IBM employee could directly “enter data” into Staples apps.
  - Replaced mail/email, fax, etc.
- Phase III – Linked Business Process and Applications
  - Directly link the applications.
  - Avoid {employee, app} → Web site → {employee, app}
- Web Services built on dynamic Web pages
  - Web Services Description Language for application interfaces
  - Basic protocols on HTTP, e.g. SOAP
  - Advanced protocols, e.g. security, reliable messaging, …
- Most scenarios are a mix of Web services and dynamic Web pages.

Web 2.0 Themes

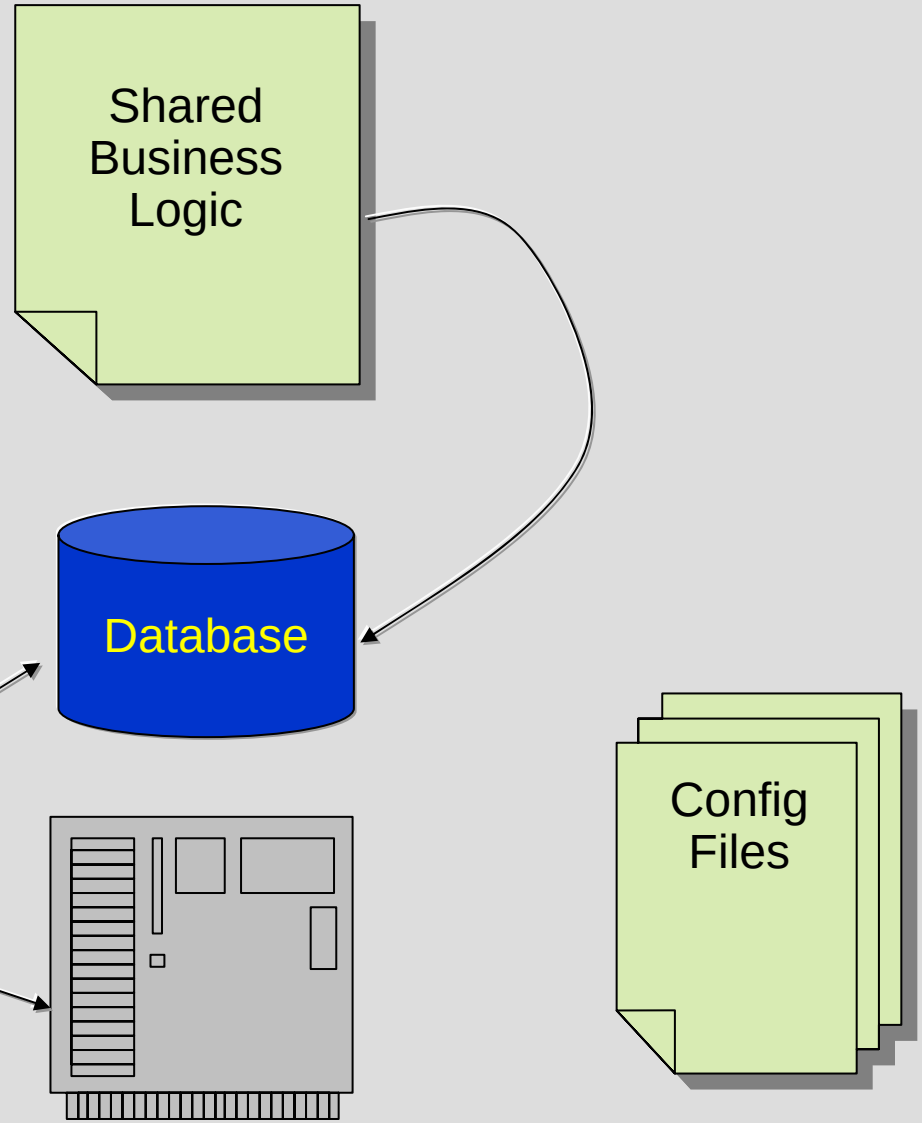


Web Application Server -- Role

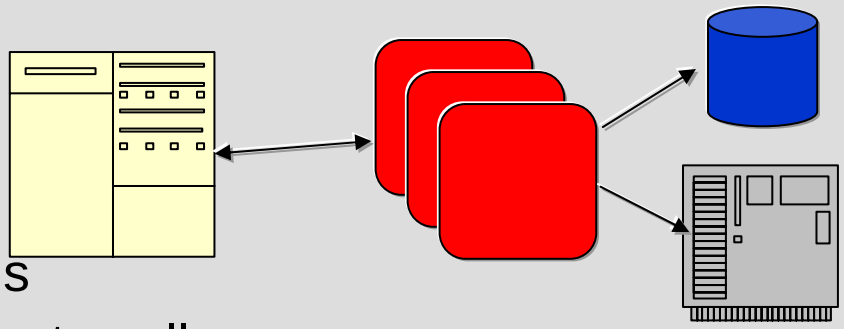


Application Structure


```
<…>
    <….>
    <…>
    <script>
        Some code
    </script>
    <…>
    <…>
    <script>
        Some code
    </script>
</…>
```



Key Concepts



- Process Pools
  - CGI was fork-run-end
  - Evolved to pre-warmed pools
  - Web server used IPC/TCP/… to call process
  - Process pre-loaded programs
  - Why?
- Thread management
  - Each process was multi-threaded (why?)
  - Conditioned process
    - Signal handlers
    - Map protocol elements to process context (why?)
- Application isolation
- Connection pools and threads
- Session management

Key Concepts

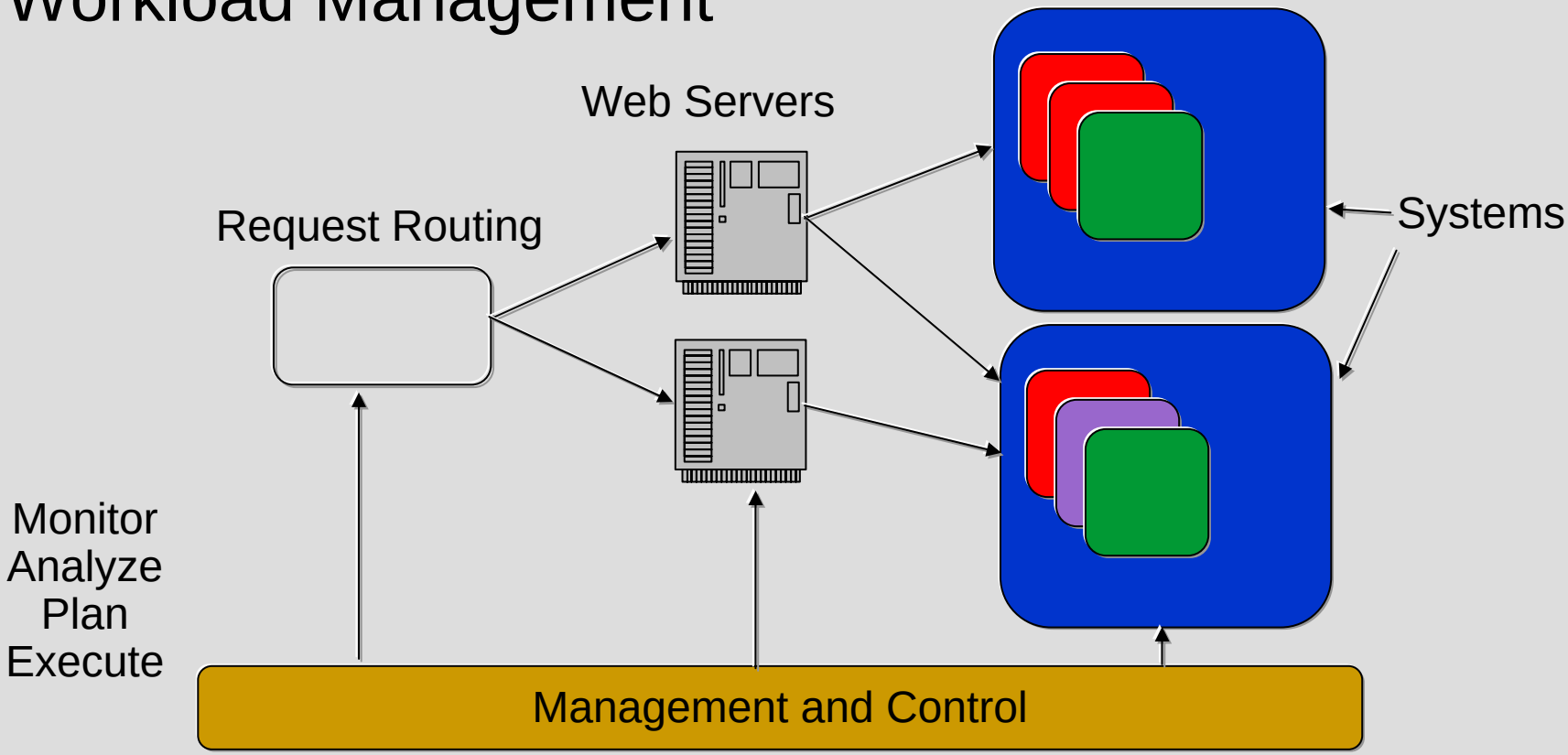
- Workload management
  - Request routing
  - Priorities
  - Session affinity, e.g. handling multiple requests in a user conversational interaction
- Availability
  - Detect failures
  - Route around failures
- Security
  - Map Basic-Auth → UID/PW directories
  - Authorization
  - Audit
- Transactions
- Connection management and pools
- Systems and application management
  - Monitoring
  - Configure servers and applications
  - Problem determination

Discussion and Break

Agenda

- Session I: 11:00 – 11:50
  - About me
  - Class overview and process
  - The evolution of the Web
  - Role of Web Application Server in applications
  - Discussion
- Break/Discussion
- Session II: 11:55 – 12:50
  - Workload and Availability Management
  - The Process Model and Thread Model
  - Protocol Support
  - Infrastructure Services
  - Caching
  - What is a Container? Containers
  - Some Containers
  - Assignments

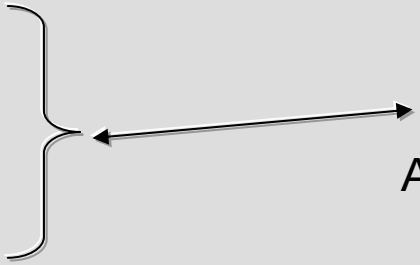
Workload Management



- Request routing
  - Which applications and servers are where
  - Load balancing
  - Failed system bypass
- Control
  - No of nodes allocated to applications
  - No application servers per node (Why more than 1?)
  - Threads per application

Some Key Concepts

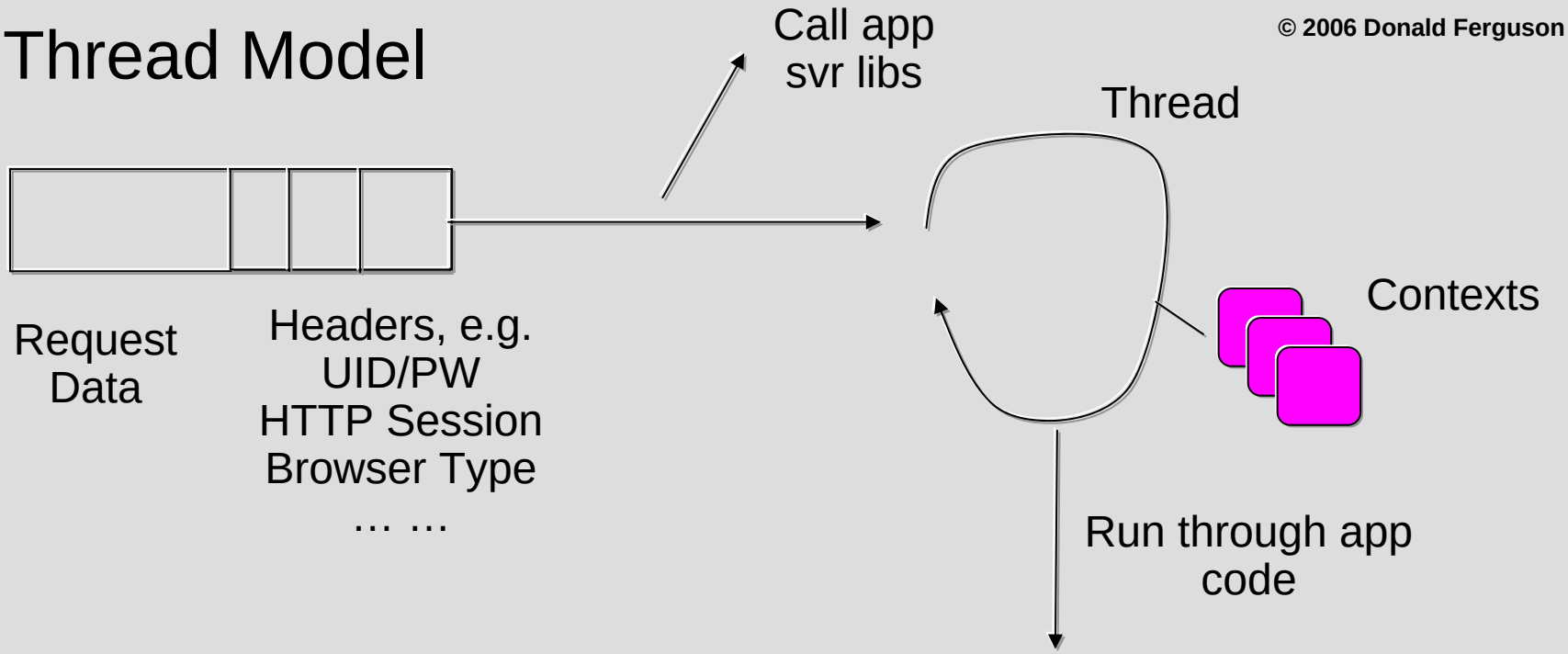
- Affinity
  - Should not load balance requests in isolation
  - There are “sessions” that “group” sequences of requests
    - User logon/logoff
    - HTTP Session
      - Begin
      - R1
      - R2
      - …
      - End



Context information
User profile
Already retrieved data
… …

  - Route requests to same server within logon or session (why?)
- Availability
  - Monitor node and server failures and restarts
  - Update routing tables and configuration
  - Rely on application probes
    - Advisor applications residing in servers
    - WLM/Availability systems calls advisors (why?)
- Performance management for administrators
  - Report on resource consumption, throughput, response time
  - Automatically manage to goals by adjusting controls

Thread Model



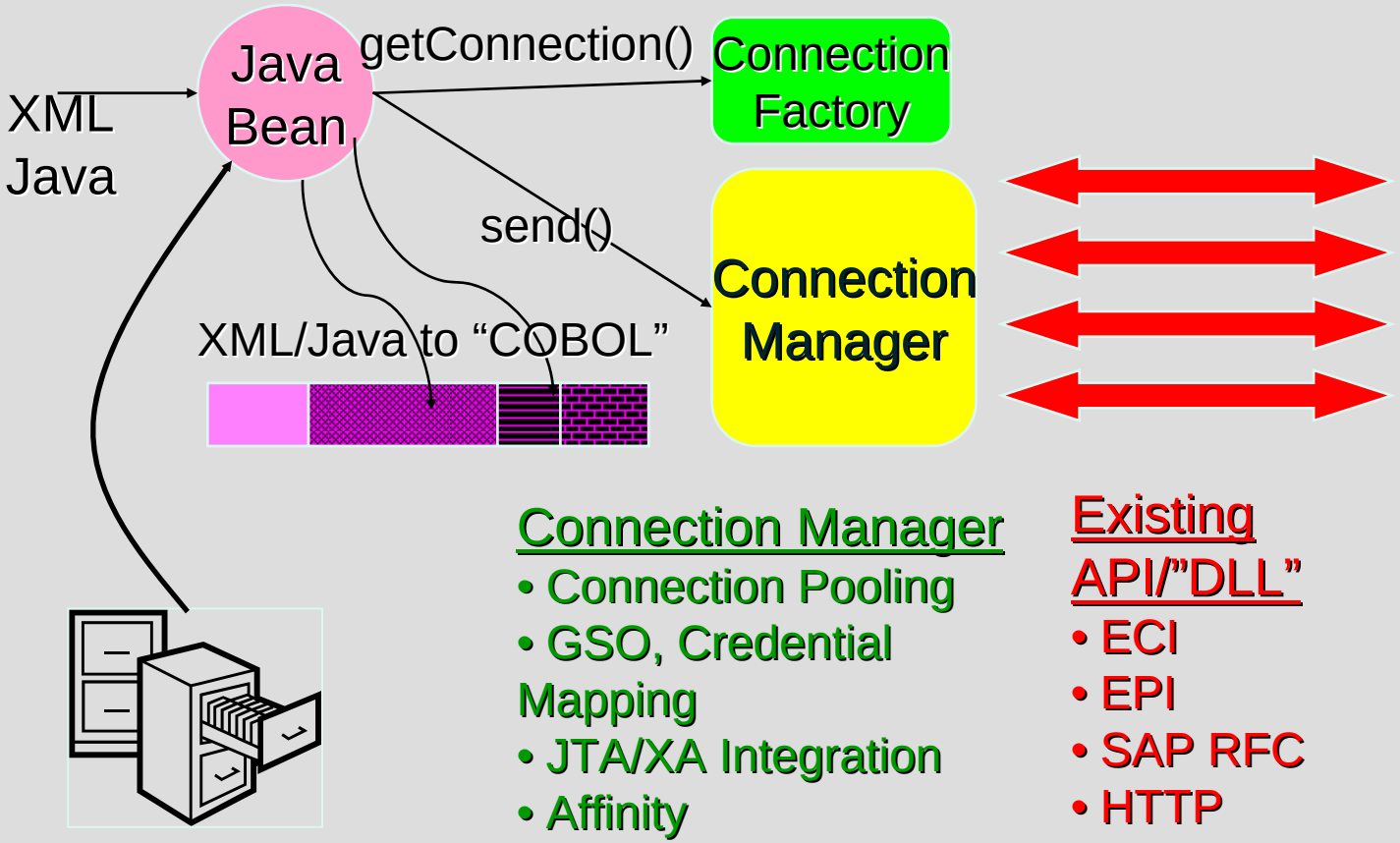
- Condition threads
  - App svr provided signal handlers associated with threads (why?)
  - Map request headers into thread context, e.g. security
- Thread enters an applications “main” and flows through application artifacts

Thread Management

- Multiple thread pools
  - Incoming protocol handlers for requests
  - Pools per application
  - Application server needs pools itself
- Some examples of contexts
  - Security
  - HTTP Session
  - Debug and trace
  - Transaction
- Thread management
  - Has handlers for requests
  - Processes header
    - Validates
    - Transforms and assigns to threads
    - Removes at request return

Connection Management

Consider some Java code calling customer database, and ERP system



Connection Manager

- Connection Pooling
- GSO, Credential Mapping
- JTA/XA Integration
- Affinity

Calls out to App Server

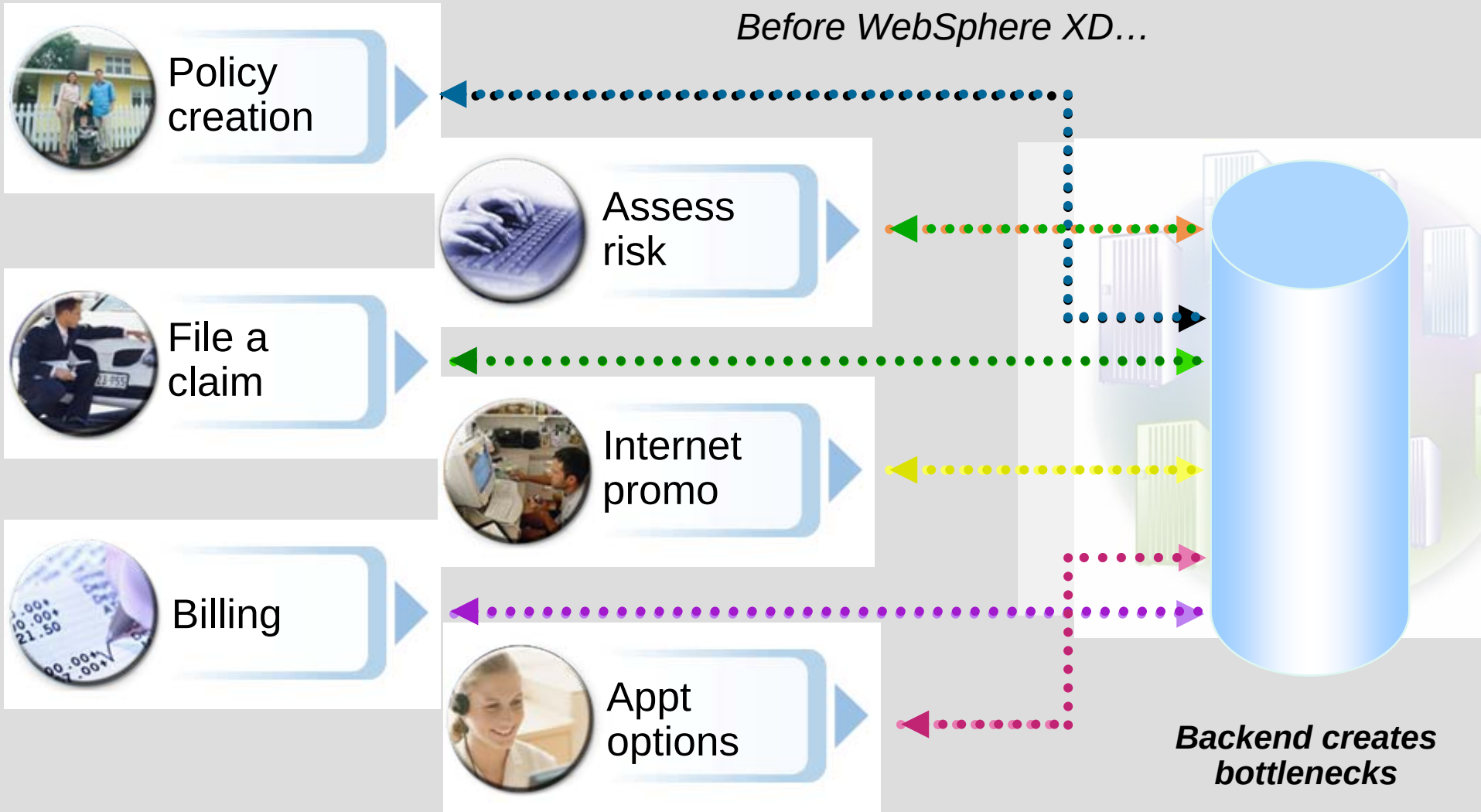
Existing API/"DLL"

- ECI
- EPI
- SAP RFC
- HTTP
- etc

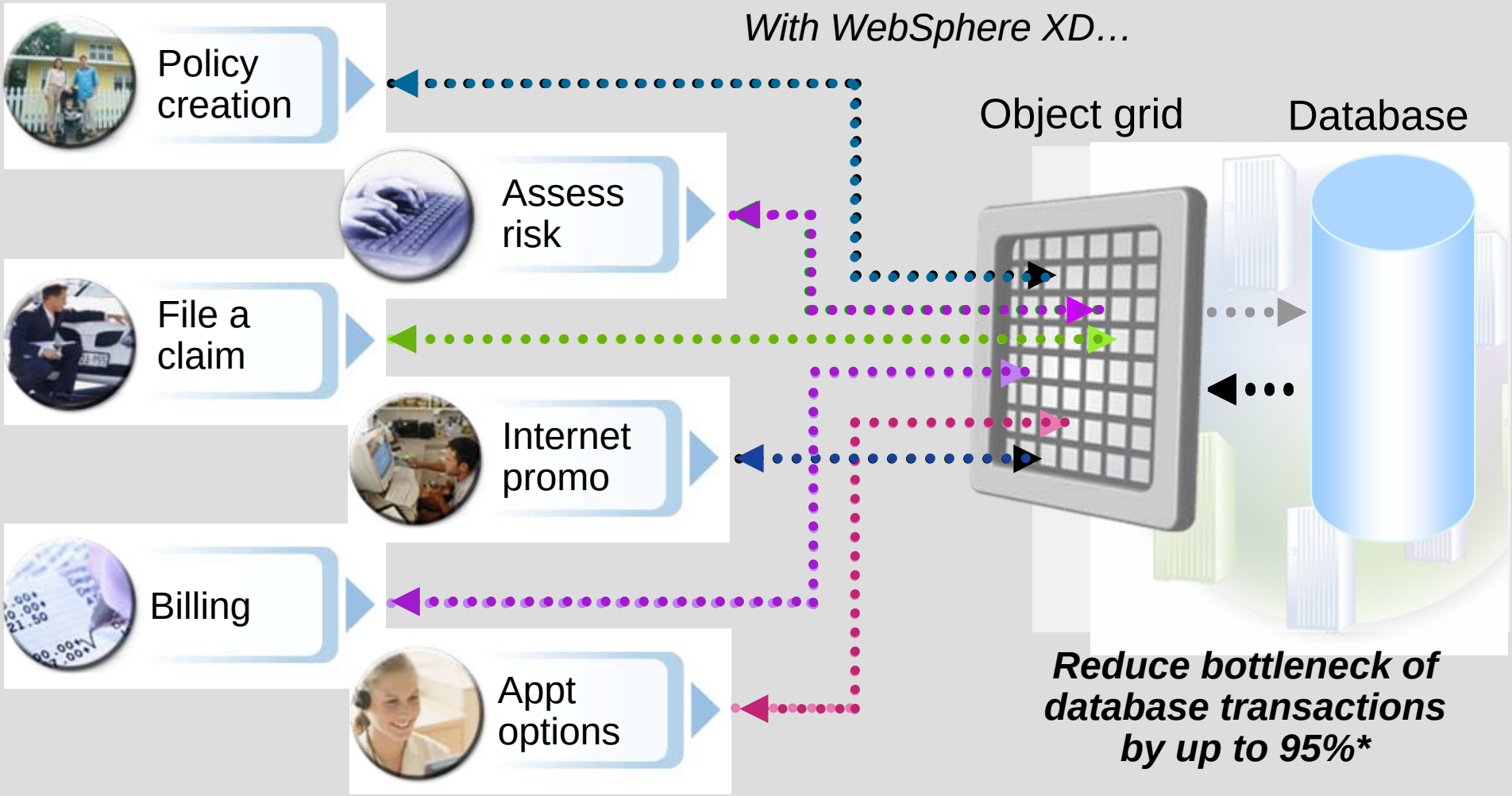
Connection Management

- An application server handles many in and out protocols
  - Database client, e.g. CLI
  - Message oriented middleware
  - HTTP
  - SIP
  - Client server protocols for SAP, CICS, … …
- Need an abstraction that makes them appear as similar as possible (why?)
- Configuration management, tracing, etc.
- Critical element of performance and availability
  - Connection affinity improve request performace
  - Avoid “overloading” existing systems with millions of request
  - Map “Internet” contexts into existing security, transaction, …
  - Error handling and reporting

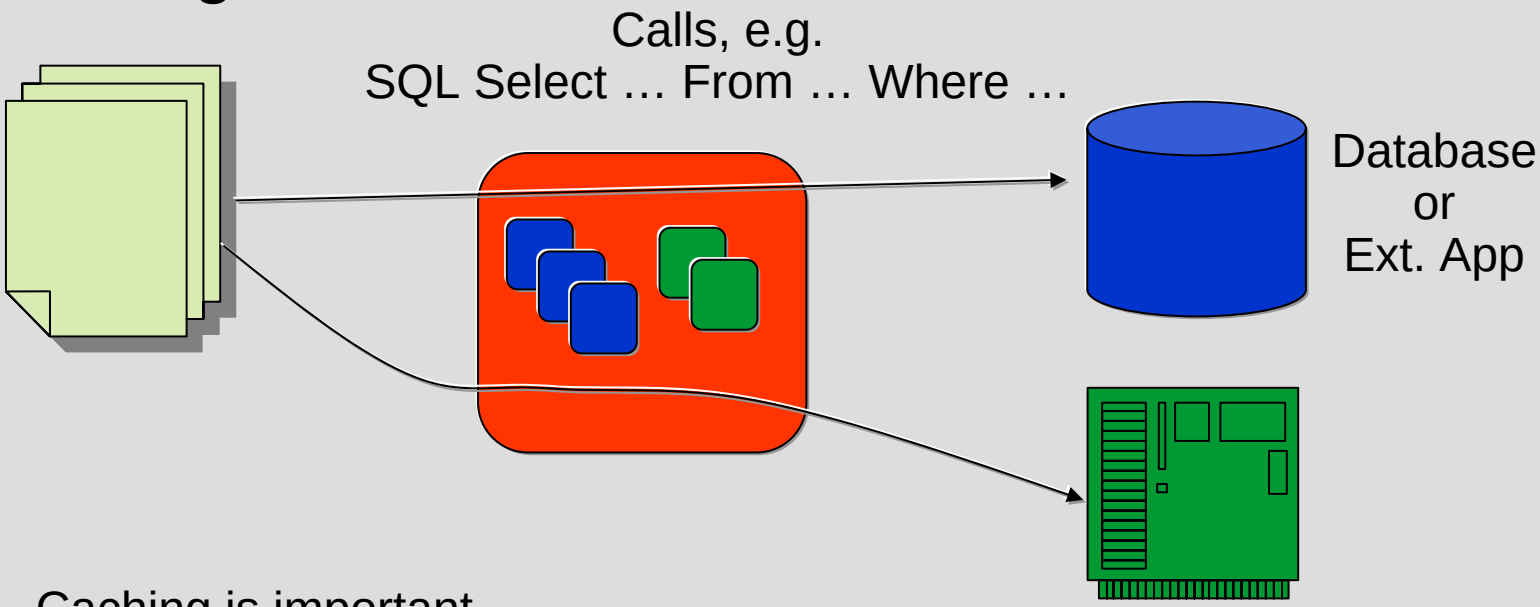
Same data is accessed by multiple applications



Maximizing transaction throughput, reliability, and performance



Caching

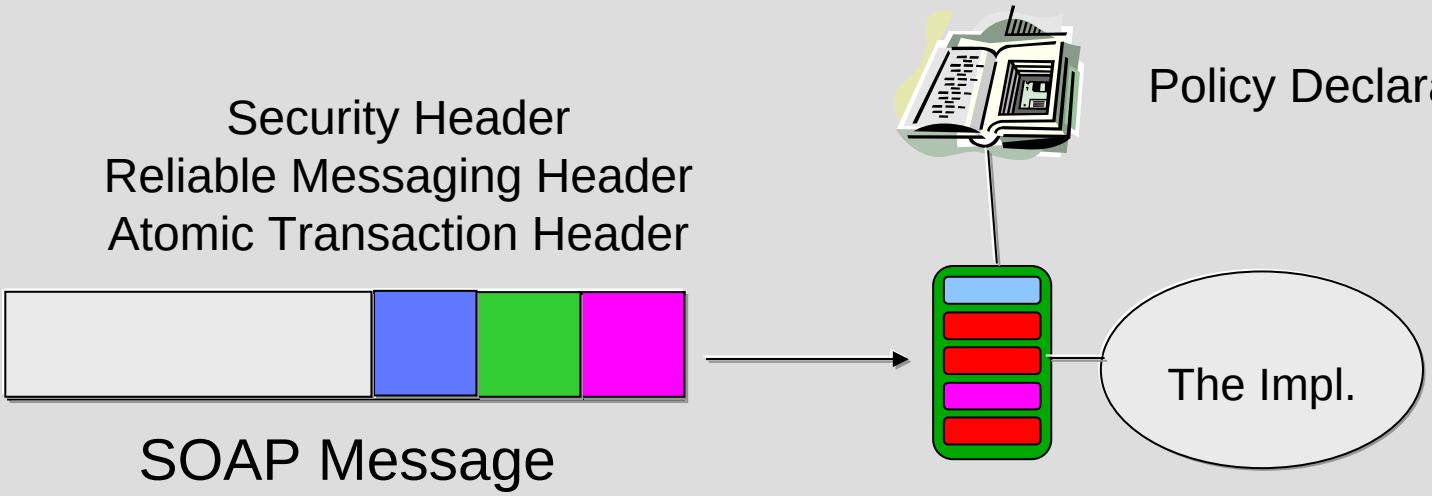


- Caching is important
  - Performance: RT and throughput (how?)
  - Protection of backend systems (how? why?)
- Cache many things: Web service calls, DB calls, partially rendered pages, …
- Surprisingly complicated to identify data and cache elements
  - Data key/identity
  - Transaction scope
  - User/context/…
- Management of pool size, lifetime model, spill, … …

APIs and Infrastructure APIs

- Much of the value of a Web application server is the integrated suite of APIs
  - Database access
  - Message queuing
  - HTTP Session
  - User profile
  - Content subsystem
- Application servers also provide infrastructure APIs
  - Transactions
  - Logging
  - Request context
  - Connection management
  - Security
  - Caching

The Role of the Container



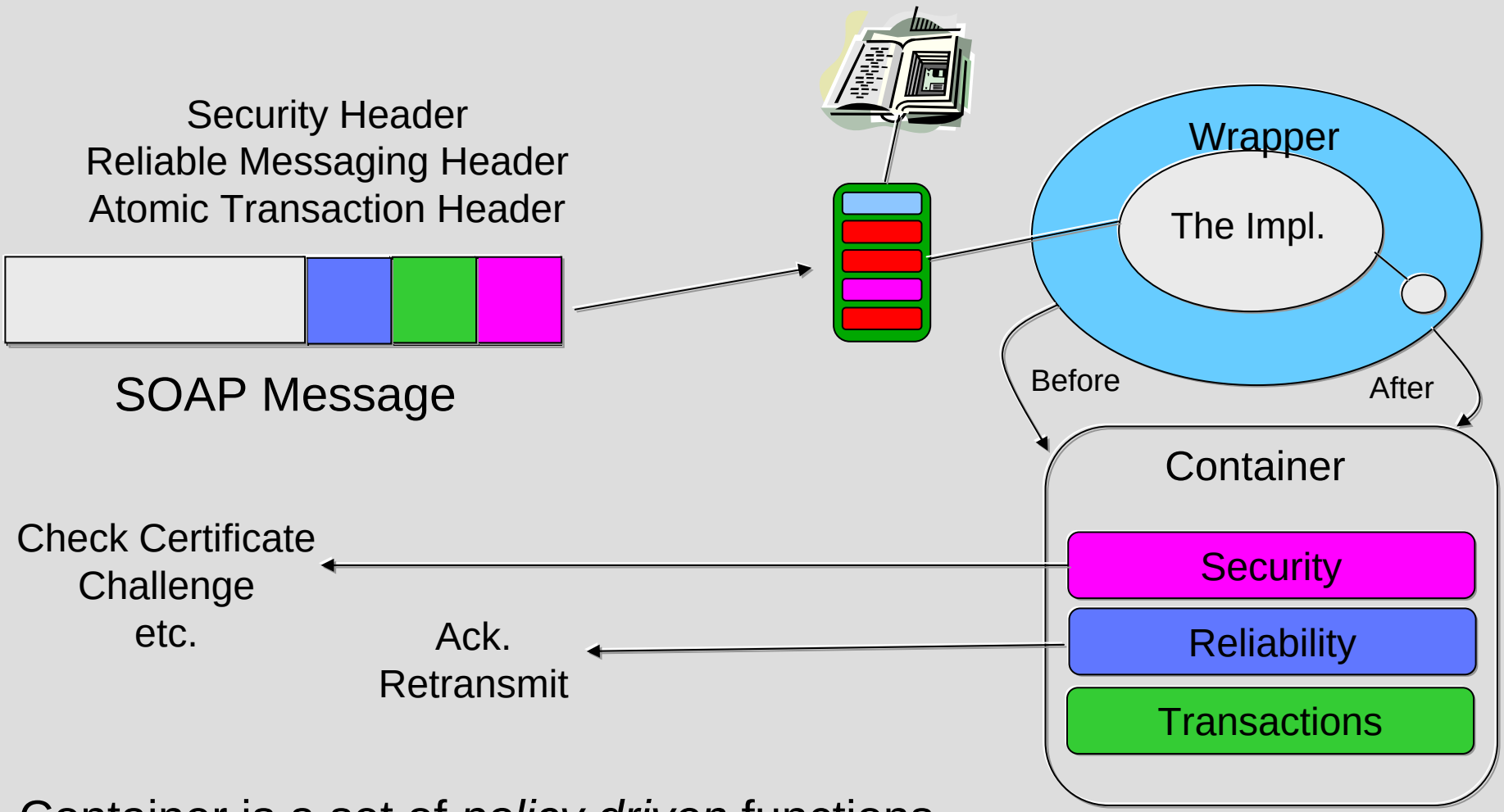
This is fragile,
changes over time,
complex for business programmers,
error prone,
etc.

```
double          deposit(Message m) {
                  checkForDuplicate(m.seqNo);
                  registerForTransaction(m.context);
                  isCAValid(m);
                  checkSignature(m);
                  updatePerformanceInfo();

                  balance += m.amount;

                  // … …
                  updatePerformanceInfo();
                }
```

The Role of the Container



Container is a set of *policy driven* functions.
Interceptor pattern for business logic and "stubs."
Before and After factoring of code.

Containers

- In some sense, a Web application server is a set of containers that run application artifact types
- Some examples
  - Web Container: Servlets, JSPs, Java classes
  - Business Logic Container
    - EJBs
    - Java classes
    - Persistence manager
  - Process Container
    - Workflow processes, e.g. BPEL4WS
    - Long running transactions
  - Message Container
    - Queues, destinations
    - Event brokers
  - Portal Container
  - Etc.

Assignments

- Gather some documents on Web Application Servers
  - Apache
  - LAMP stacks
  - .NET
  - Others
- Think about application → application server
  - Examine some of the sample applications
  - Write a simple application, or modify an example
  - Read APIs docs and configuration docs
- Be prepared to talk about
  - APIs
  - Functions configured through file or SM APIs
  - What are some interesting design problems? Performance, availability, frameworks, etc
- Begin thinking about component design you will lead.