Advanced Enterprise Portal (AEP)

Architecture and Problems

Walter Kriha

Portal Definition

- Combines several legacy backend data sources and applications at request-time into one page
- Provides Single-Sign-On (SSI)
- Content ist highly dynamic, personalized, integrated and secured
- >12000 concurrent sessions, >500 conc. Requests
- Runs on Web Cluster (load-balanced)

Dynamic and
personalized
homepage

Common: customize, filter, contact etc.

Portfolio: Siemens,
Swisskom, Esso,

Welcome Mrs. Rich,
e would like to point you to our
New Instrument X that fits nicely
To your current investment strategy.

Messages: 3 new
From foo: hi Mrs. Rich

Common: Banner

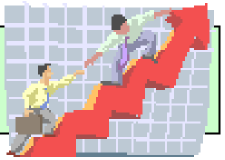
Quotes: UBS 500,
ARBA 200

News: IBM invests in company Y

Links: myweather.com,
UBS glossary etc.

Research: asian equity update

Charts: Sony



Common: customize, filter, contact etc.

Dynamic,
personalized and
INTEGRATED
homepage

Welcome Mrs. Rich,
e would like to point you to our
ew Instrument X that fits nicely
To your current investment strategy.

Portfolio: Siem
add **X**?

Messages: 3 new
From advisor: about **X** inv.

Common: Banner about **X**

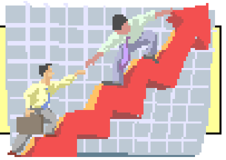
Quotes: UBS 500,
X 100

News: IBM invests in company **X**,
X now listed on NASDAQ

Links: **X** homepage
myweather.com,.

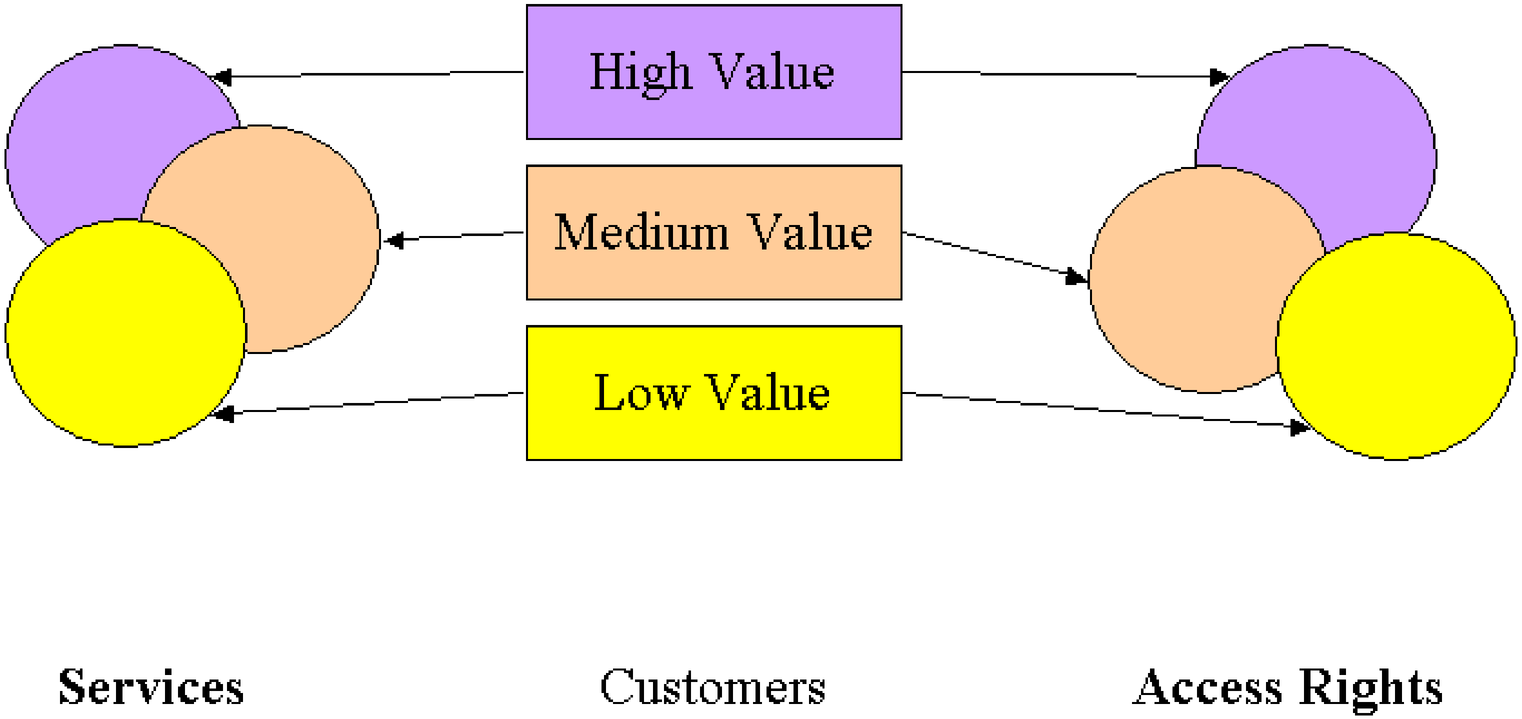
Research: **X** future prospects
asian equity update

Charts: **X**



Personalization

Who sees what? Customer Segmentation



High Value

Medium Value

Low Value

Services

Customers

Access Rights

Business defines the segmentation (at least initially)

Bad (hard-coded) Segmentation

GUI: select background color

```
type = userObject.getUserType();
If (type == LOWVALUE)
        backgroundColor=“yellow”;
If (type == HIGHVALUE)

        backgroundColor=“purple”;
```

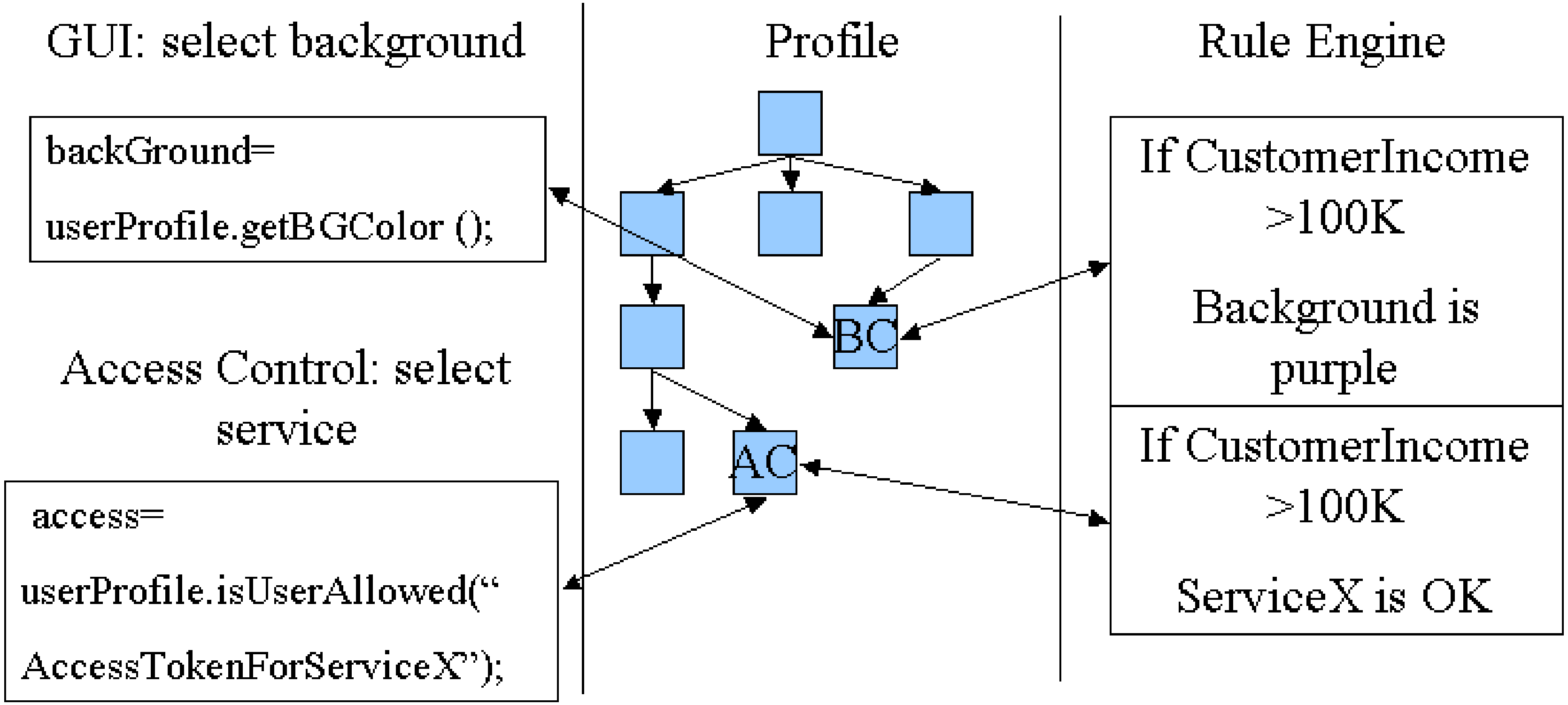
Access Control: select service

```
type = userObject.getUserType();
If (type == LOWVALUE)
        access=NO;
If (type == HIGHVALUE)

        access=YES;
```

If the customer segmentation changes all this code needs to change!

Good (dynamic) Segmentation



GUI: select background

```
backGround=
userProfile.getBGColor ();
```

Access Control: select service

```
 access=
userProfile.isUserAllowed(“
AccessTokenForServiceX”);
```

Profile

Rule Engine

If CustomerIncome >100K

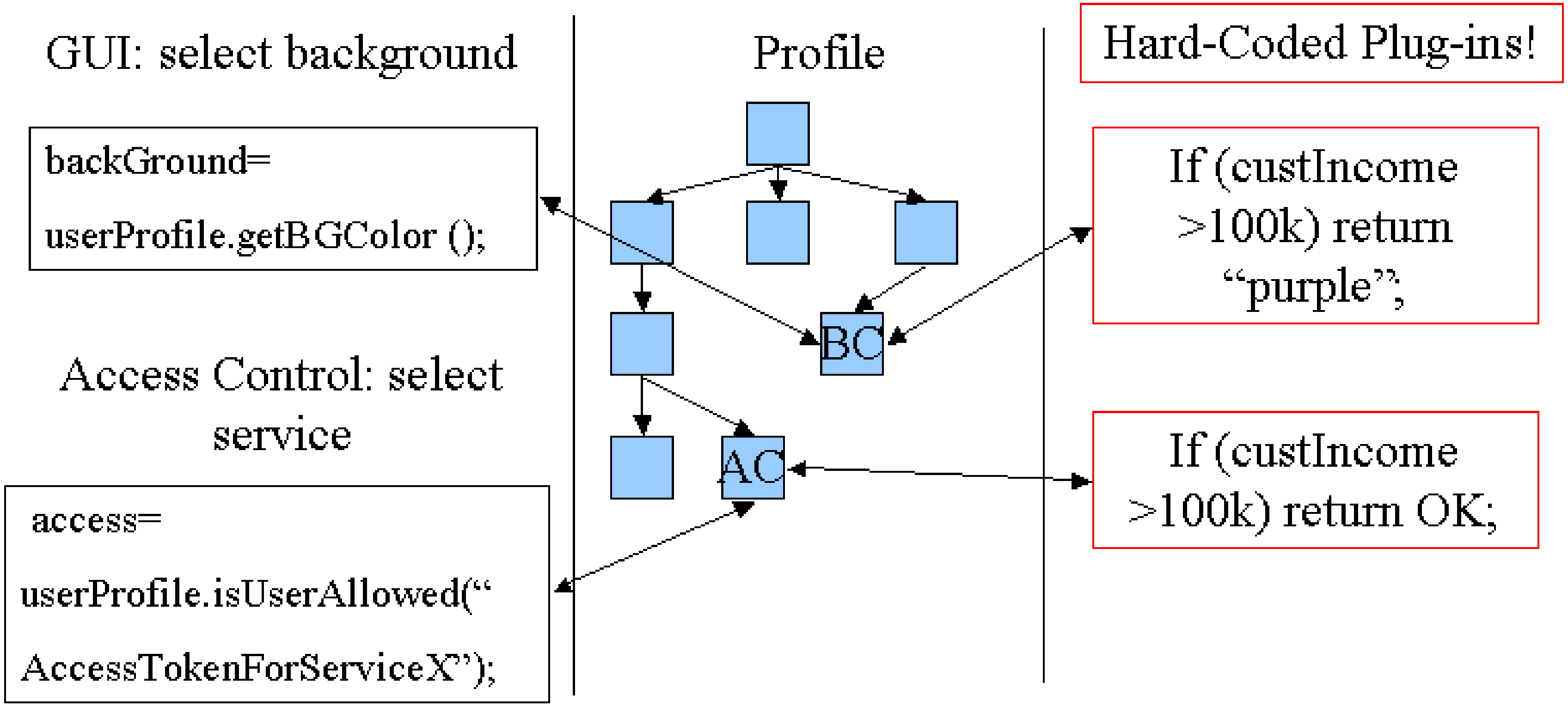
Background is purple

If CustomerIncome >100K

ServiceX is OK

Simple value interface to profile. Profile elements are adapters and hide rule engine. No open calls to rule engine. Easy to change segmentation

But Performance?



GUI: select background

backGround=

userProfile.getBGColor ();

Access Control: select service

access=

userProfile.isUserAllowed("

AccessTokenForServiceX");

Profile

Hard-Coded Plug-ins!

If (custIncome >100k) return "purple";

If (custIncome >100k) return OK;

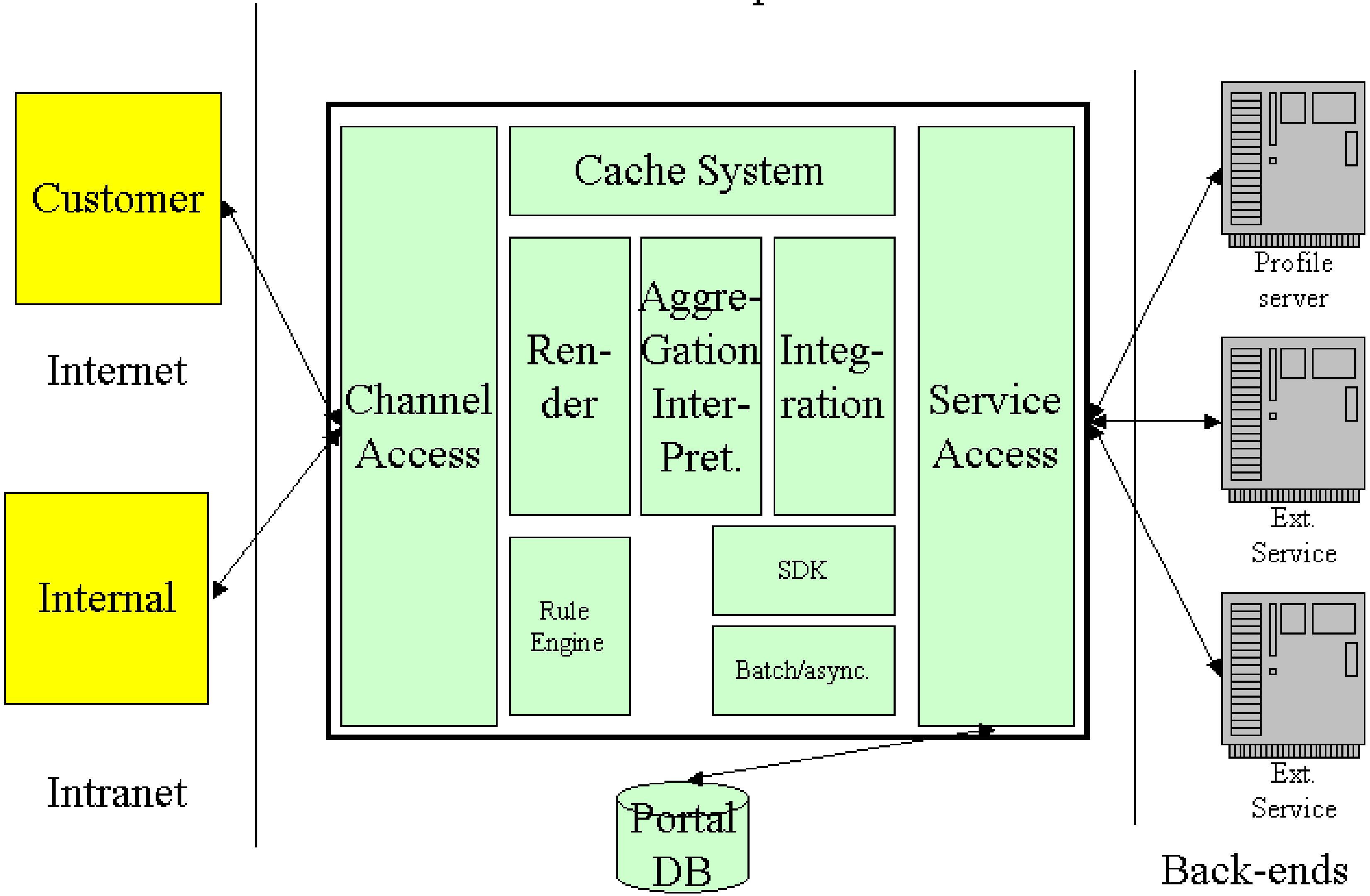
"Role" as a set of access rights is a concept known only within the business (engine) part of the portal. Services themselves do not know about it – and therefore need not change!

Portal Problems

- High implementation costs, permanent re-designs > 15 Mio.
- Hardware costs per user extremely high
- Low performance, no scalability,
- Low stability due to new technology used
- Integration problems with existing systems
- Wrong management strategies and expectations

Failed or troubled projects (Vontobels “You”, UBS e-services, Schweizer Post “yellowworld” and many others.

Portal Conceptual Model



Simple Model2 Architecture

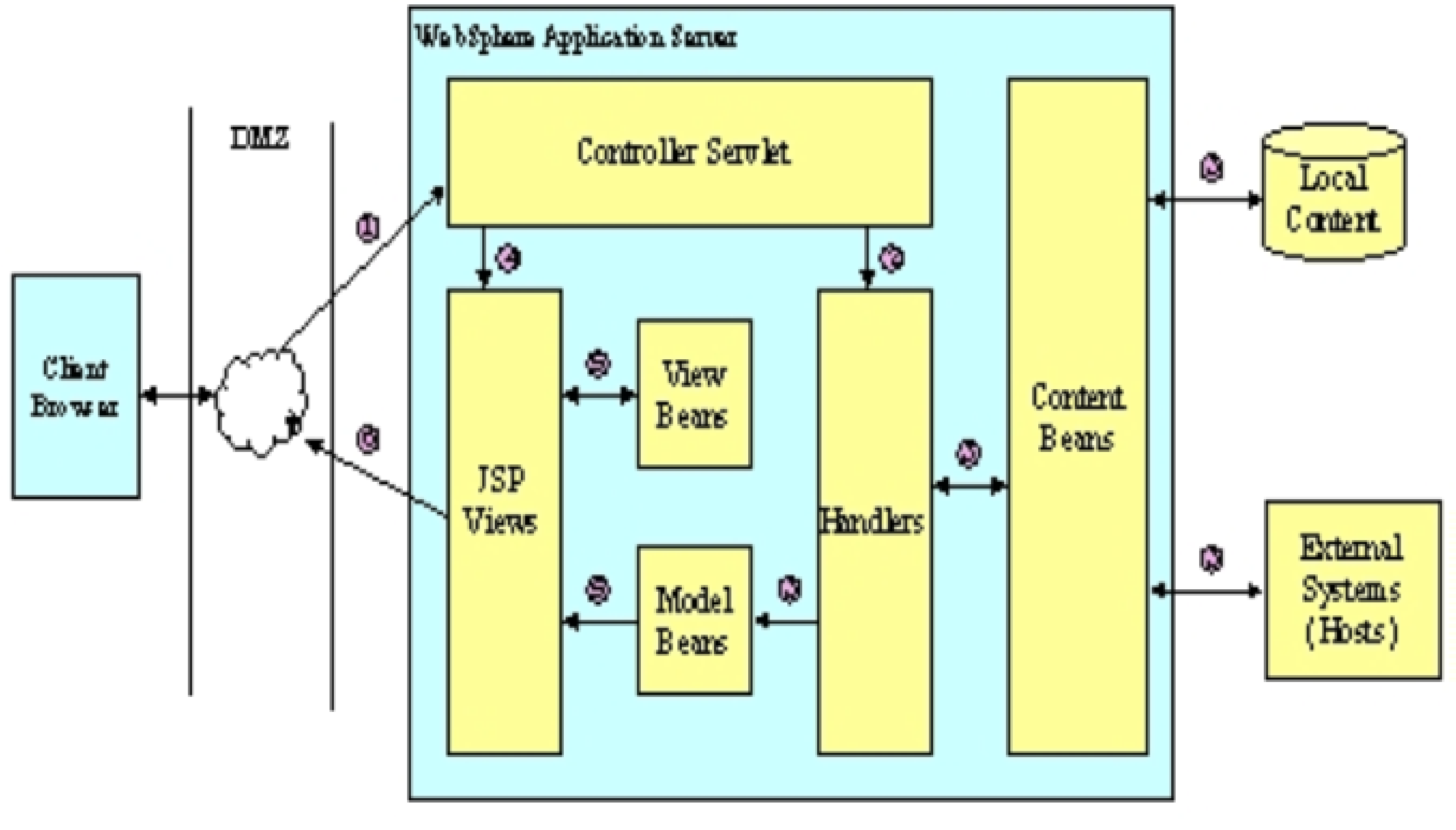


Figure 1, Technical Architecture

Critical Aspects of a Model2-arch.

- Only one servlet – container load-balancing, threading and security not usable
- Push model: JSPs and handlers are tightly coupled. JSPs cannot freely assemble content
- Servlet stream model not so good for information gathering, integration and final formatting
- Browser side syndication possible?

layout

Services: customize, filter, contact etc.

Portfolio

Portfolio: Siem

add **X**?

1 handler
(command obj.)
Per service

welcome

Welcome Mrs. Rich,
would like to point you to our
ew Instrument X that fits nicely
To your current investment strategy.

Messages: 3 new

ShowMessages

dvisor: about **X** inv.

Services: Banner about **X**

ShowBanner

Quotes: UBS 500,
X 100

ShowQuotes

News: IBM invests in company **X**,
X now listed on NASDAQ

ShowNews

Links: **X** homepage
myweather.com,.

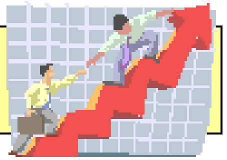
ShowLinks

Research: **X** future prospects
asian equity update

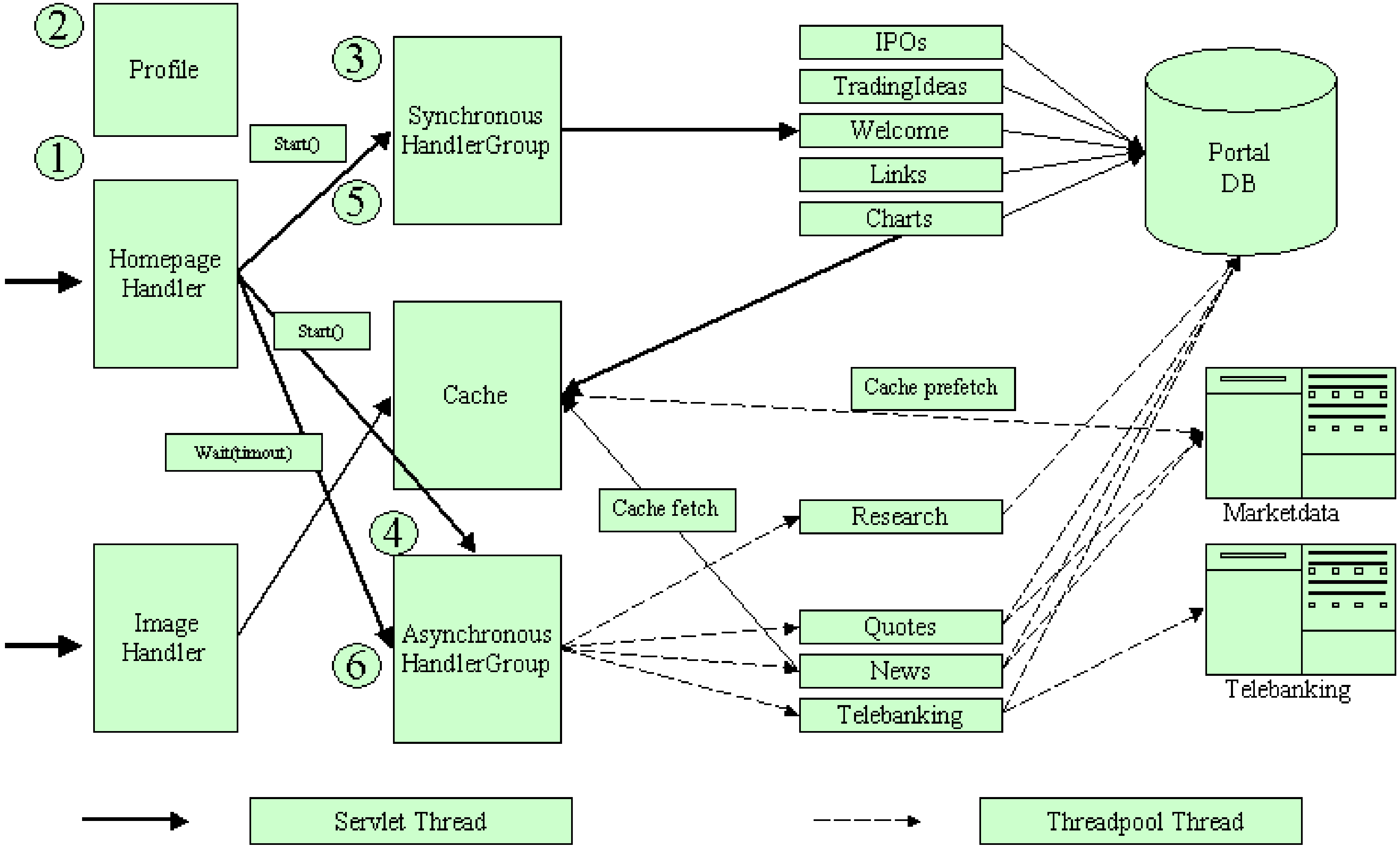
ShowResearch

Charts:

ShowCharts



PortalPage Request Flow and Assembly



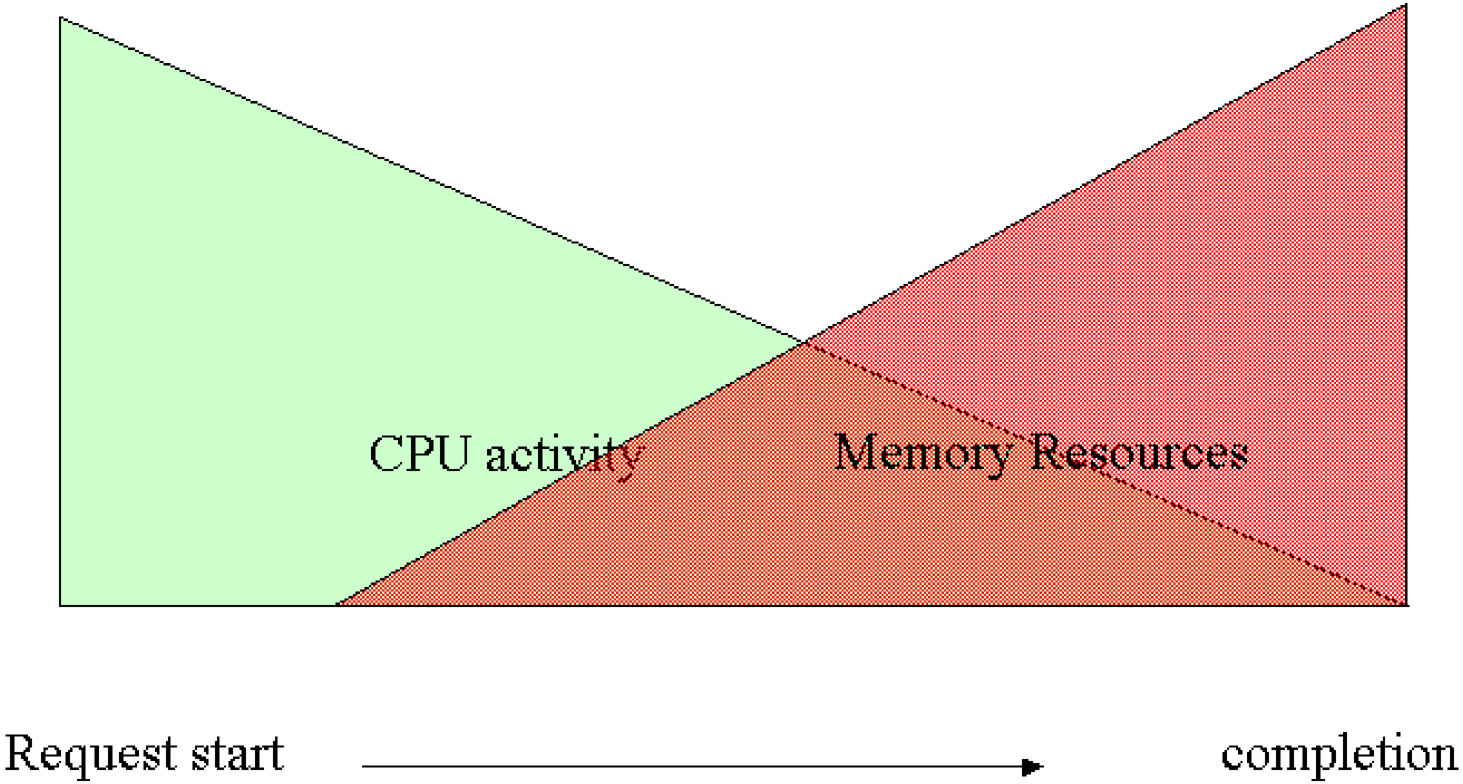
Portal Problem Analysis

- Reliability
- Performance, Caching and Architecture
- GUI design
- Implementation
- Infrastructure
- Maintenance
- Management

Reliability Problems

- Java VM blows up in case of stalled backend requests
- No service access layer to control availability of backend systems
- Side-effects of internal threading

Java VM memory consumption during complex homepage request

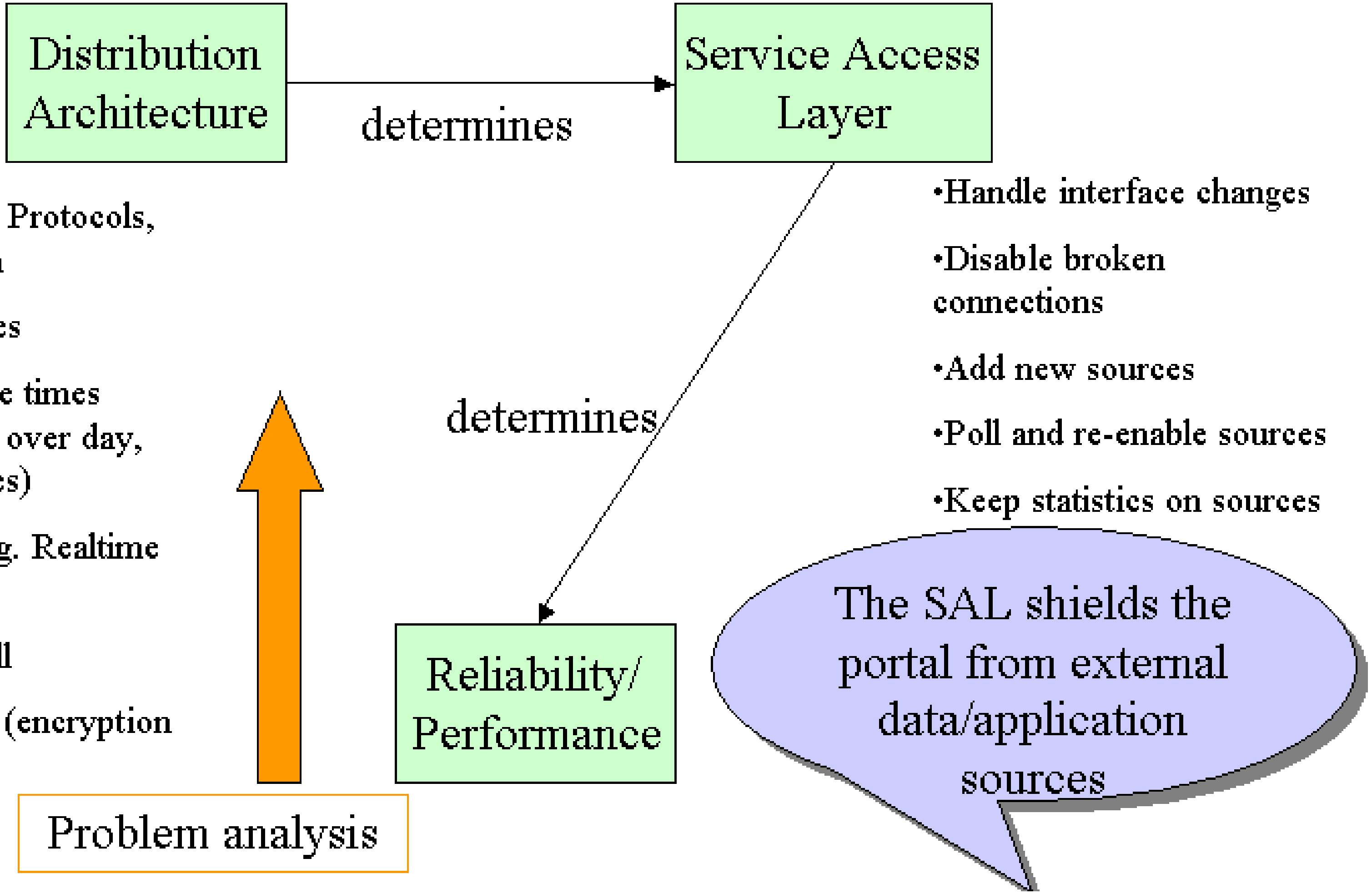


CPU activity

Memory Resources

Request start → completion

Data Aggregation: What, Where and How?



Distribution Architecture

determines

Service Access Layer

- Sources, Protocols, Schemata
- Data rates
- Response times (average, over day, downtimes)
- QOS (e.g. Realtime quotes)
- Push/Pull
- Security (encryption etc.)

Problem analysis

determines

Reliability/ Performance

- Handle interface changes
- Disable broken connections
- Add new sources
- Poll and re-enable sources
- Keep statistics on sources

The SAL shields the portal from external data/application sources

Distribution Architecture

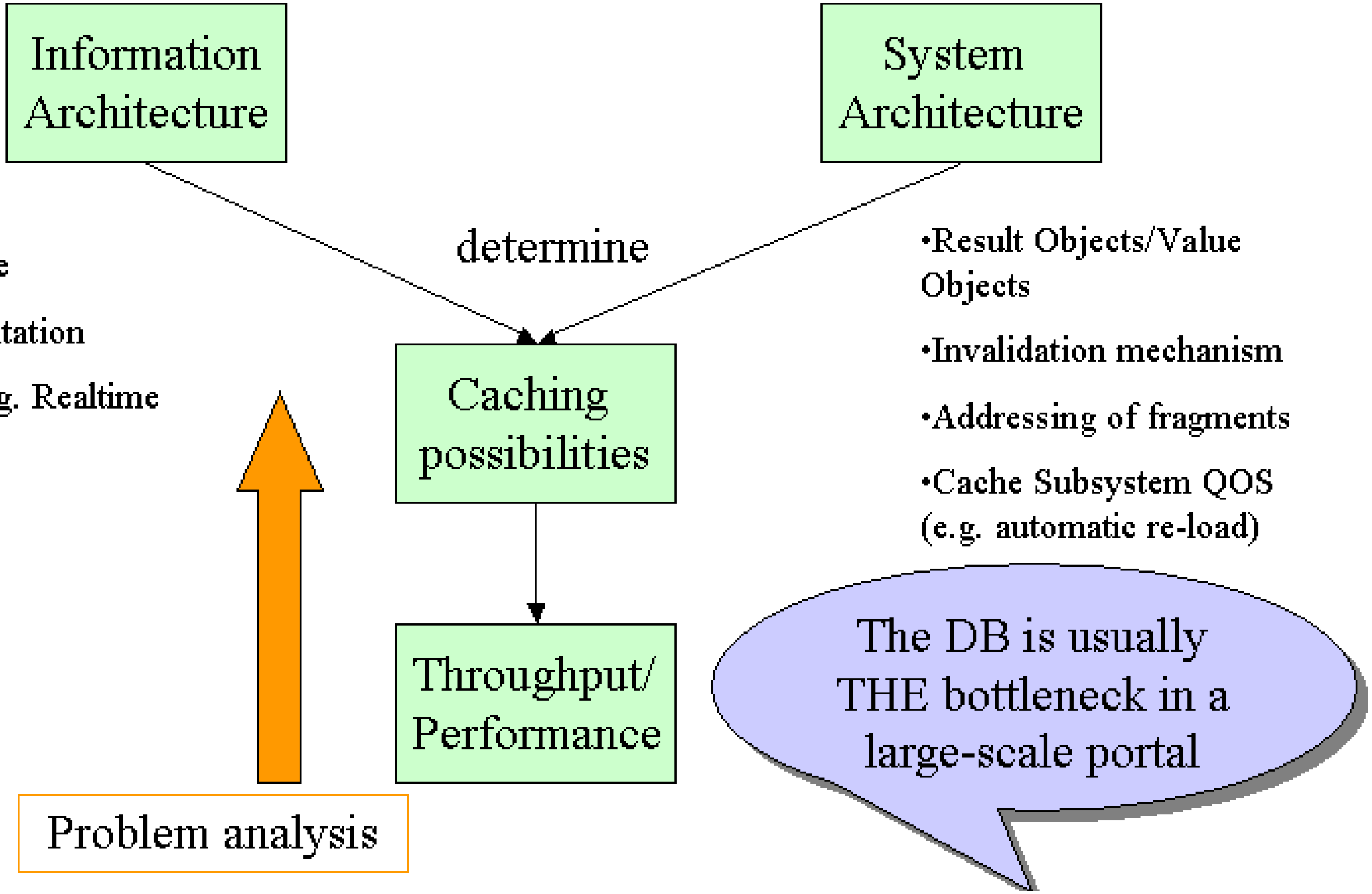
	Source	Protocol	Port	Avg. Resp.	Worst Resp.	Down-times	Load-bal.	Security	Contact/SLA
News	hostX	http/xml	3000	100ms	6 sec.	17.00-17.20	client	plain	Mrs.X/News-SLA
Research	hostY	RMI	80	50ms	500ms.	0.00-1.00	server	SSL	Mr.Y/res-SLA
Quotes	hostZ	Corba/IDL	8080	40ms	25 sec.	Ev.Monday 1 hour	Client	plain	Mr.Z/quotes-SLA
Personal	hostW	JDBC	7000	30ms	70ms	2 times Per week	server	Oracle JDBC dr.	Mrs.W/pers-SLA

Getting this information requires tracking backend services and writing test programs. The results determine what can be combined on a personalized homepage.

Performance, Caching and Architecture

- No Information Architecture existed: Information not qualified with respect to aging and QOS.
- Caching possibilities not used (http) or underestimated (20 secs. Are static!)
- No compression or web accelerators used.
- Architecture not fit to support caching (where and what analysis missing)
- Large scale portal needs fragment architecture
- Tactical mistakes: no automatic service time control, no automatic DB connection hold control, internal threading introduced too early…

Caching: Why, What, Where and how much?



Information Architecture

System Architecture

determine

Caching possibilities

Throughput/ Performance

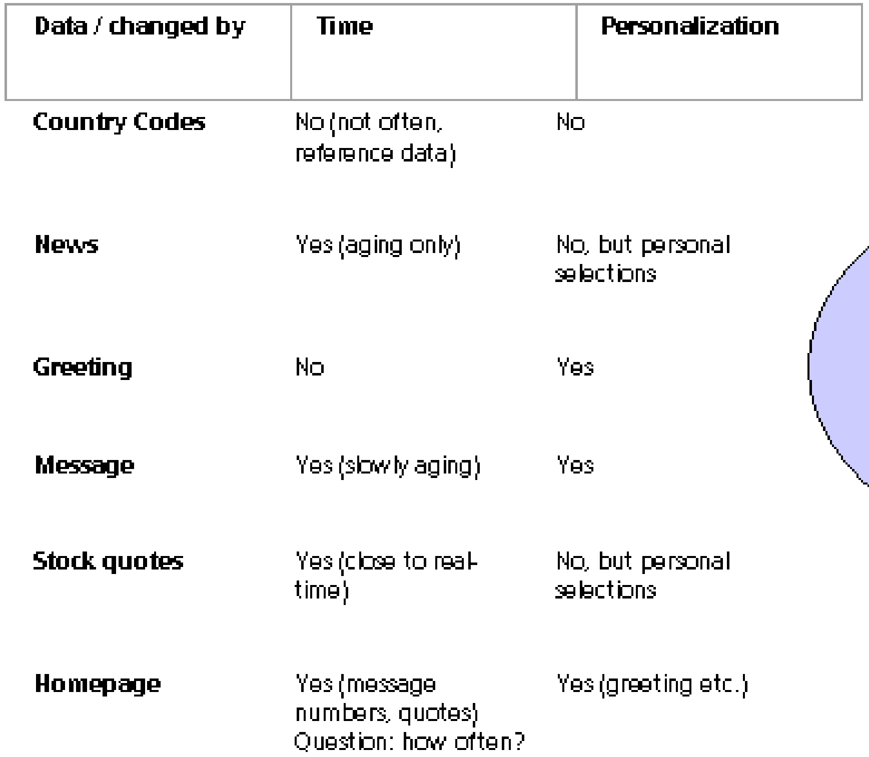
Problem analysis

- Lifecycle
- Fragmentation
- QOS (e.g. Realtime quotes)

- Result Objects/Value Objects
- Invalidation mechanism
- Addressing of fragments
- Cache Subsystem QOS (e.g. automatic re-load)

The DB is usually THE bottleneck in a large-scale portal

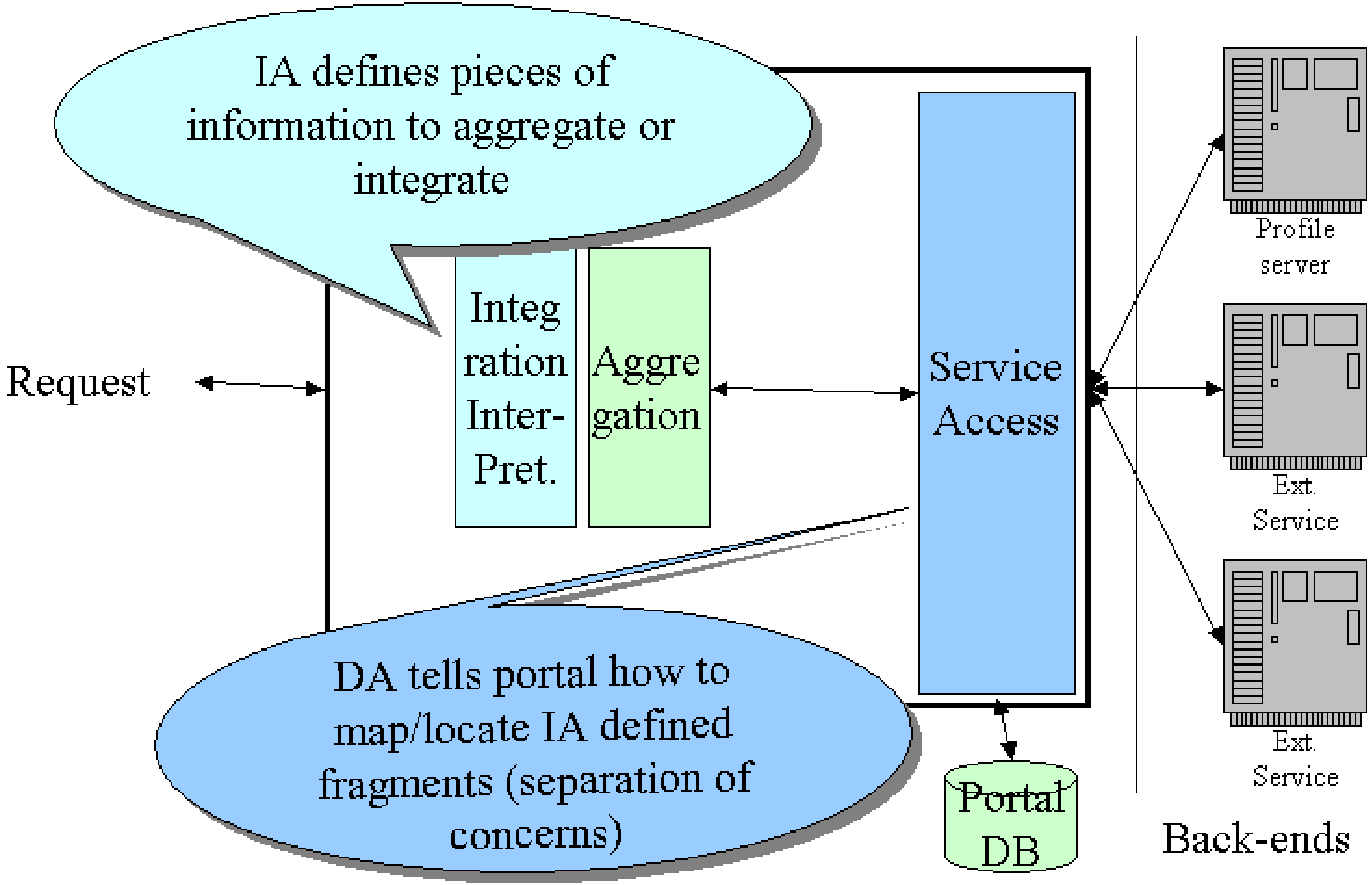
Information Architecture – Lifecycle Aspects



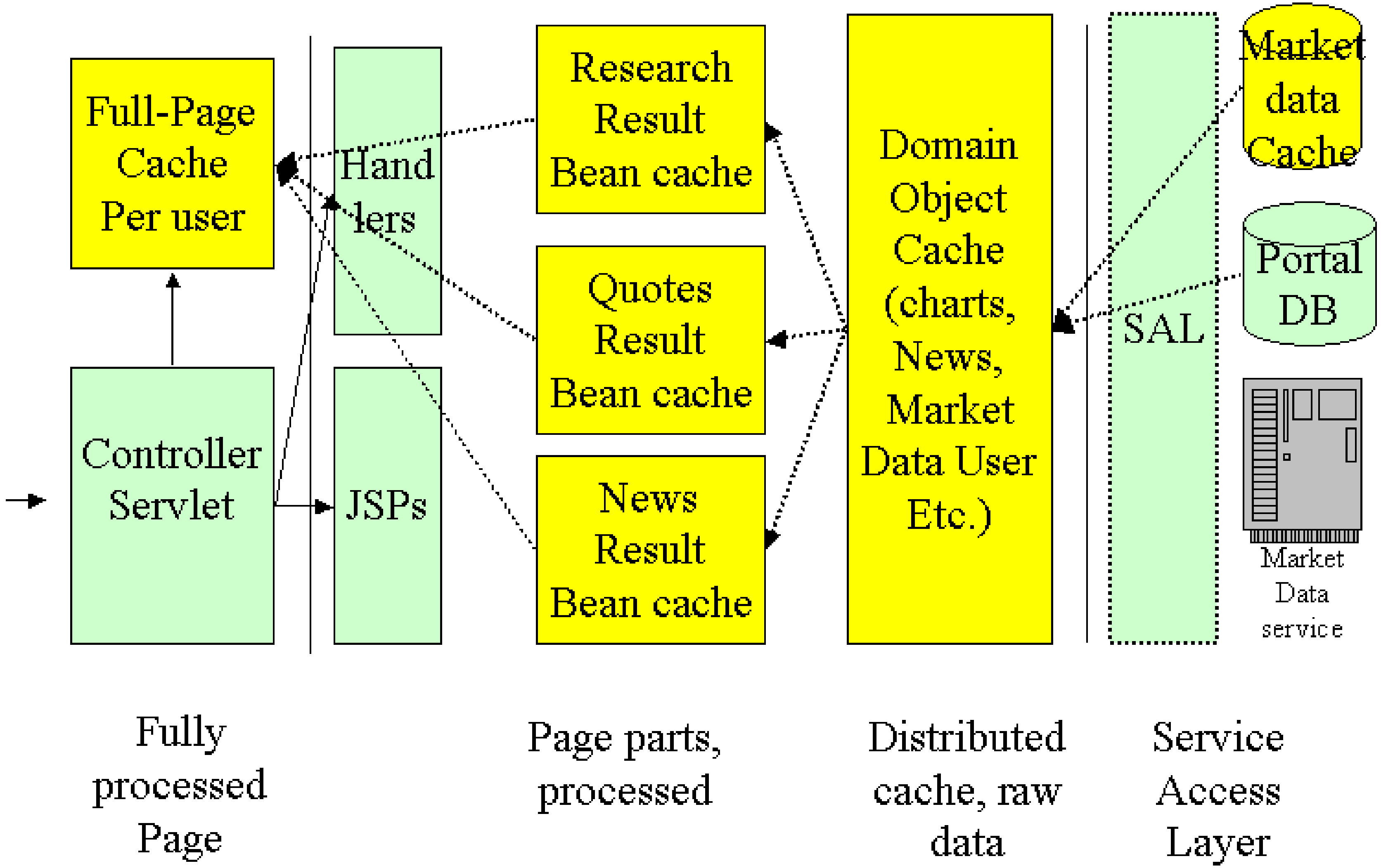
Data / changed by	Time	Personalization
Country Codes	No (not often, reference data)	No
News	Yes (aging only)	No, but personal selections
Greeting	No	Yes
Message	Yes (slowly aging)	Yes
Stock quotes	Yes (close to real-time)	No, but personal selections
Homepage	Yes (message numbers, quotes) Question: how often?	Yes (greeting etc.)

For every bit of information you must know how long it is valid and what invalidates it

How Information- and Distribution Architecture drive the Portal


IA defines pieces of information to aggregate or integrate
Request
Integ ration Inter- Pret.
Aggre gation
Service Access
Profile server
Ext. Service
Ext. Service
DA tells portal how to map/locate IA defined fragments (separation of concerns)
Portal DB
Back-ends

Cache fragments, locations and dependencies (without client and proxy side caches)



Common: customize, filter, contact etc.

User did not customize this service: use standard

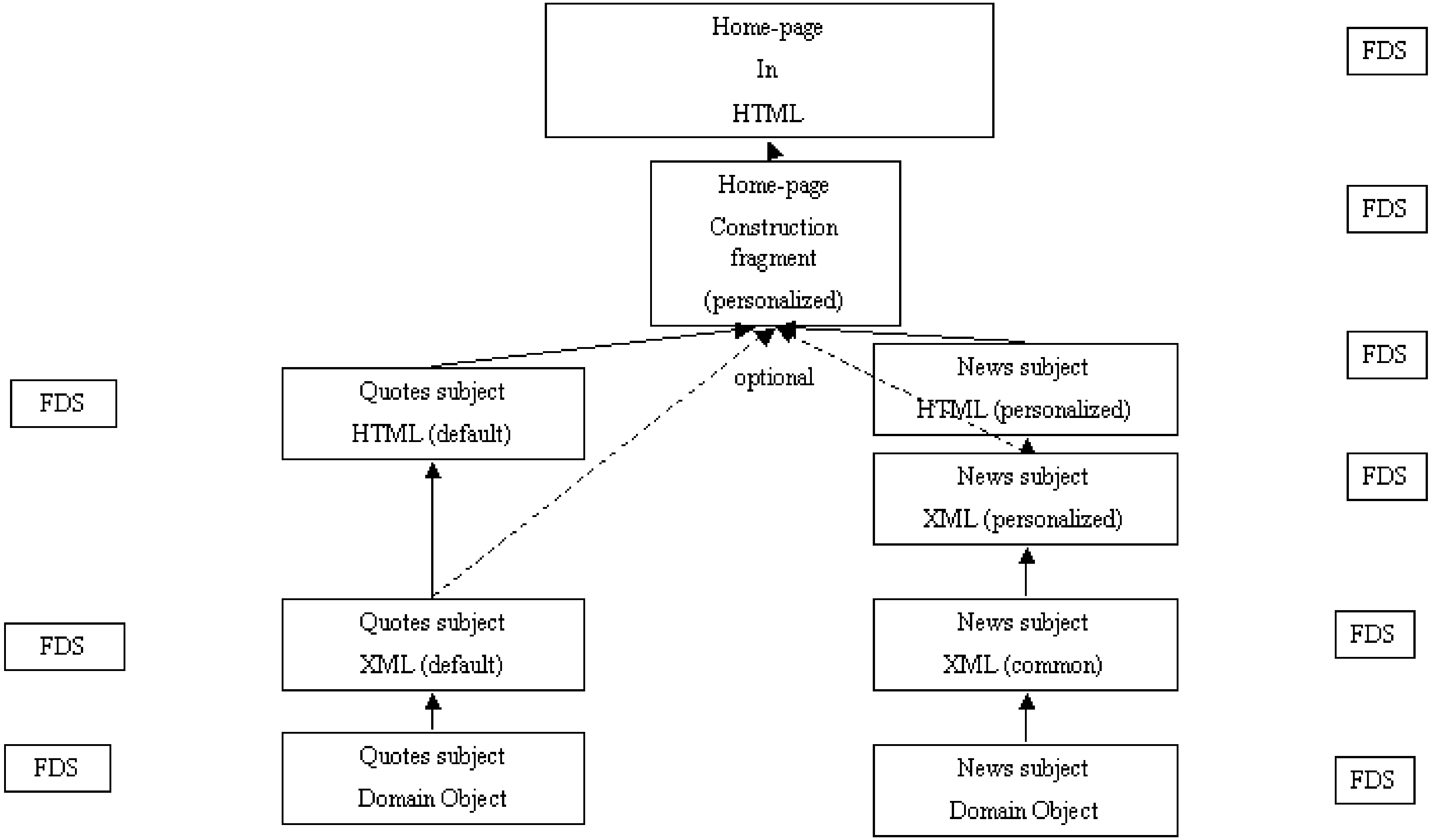
User did customize this service. Keep personalized copy

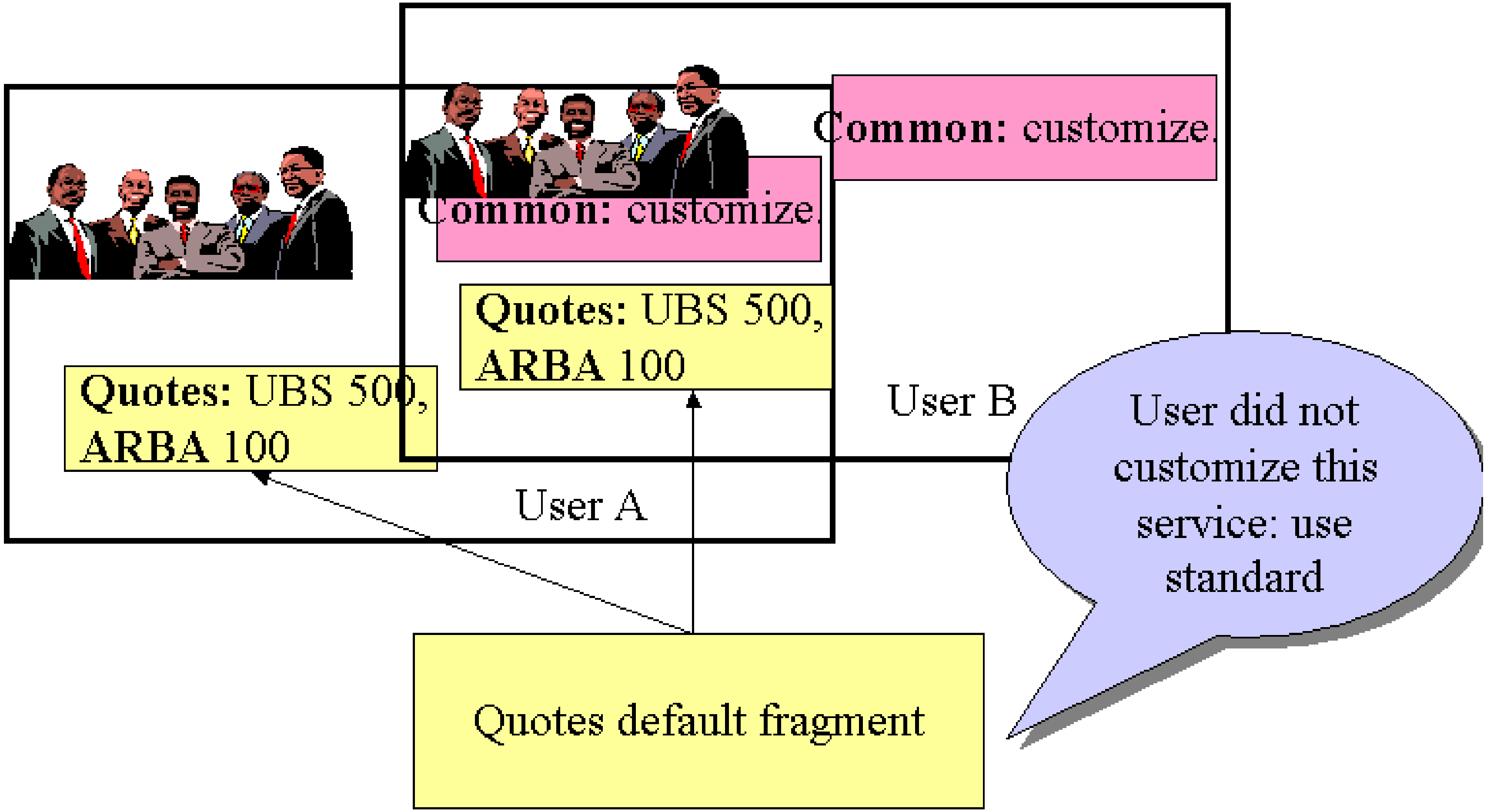
Quotes: UBS 500, **ARBA** 100

News: IBM invests in company **X**, **X** now listed on NASDAQ

The Information Architecture for services/portlets defines what parts are global or personalized, where they come from and how long they are valid

Personalized and standard content mixed together for one homepage





Common: customize.

Common: customize.

Quotes: UBS 500, ARBA 100

Quotes: UBS 500, ARBA 100

User B

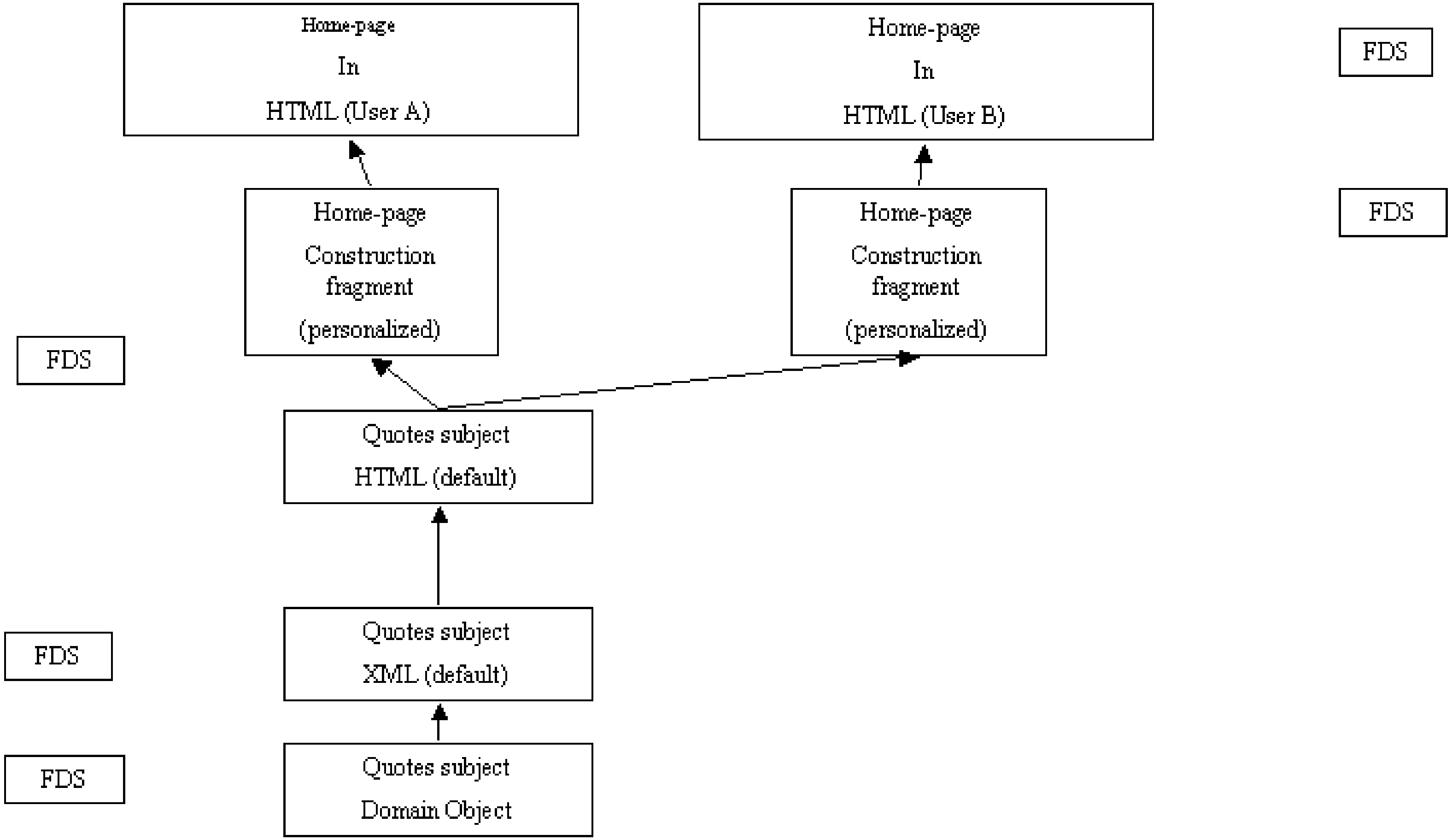
User A

User did not customize this service: use standard

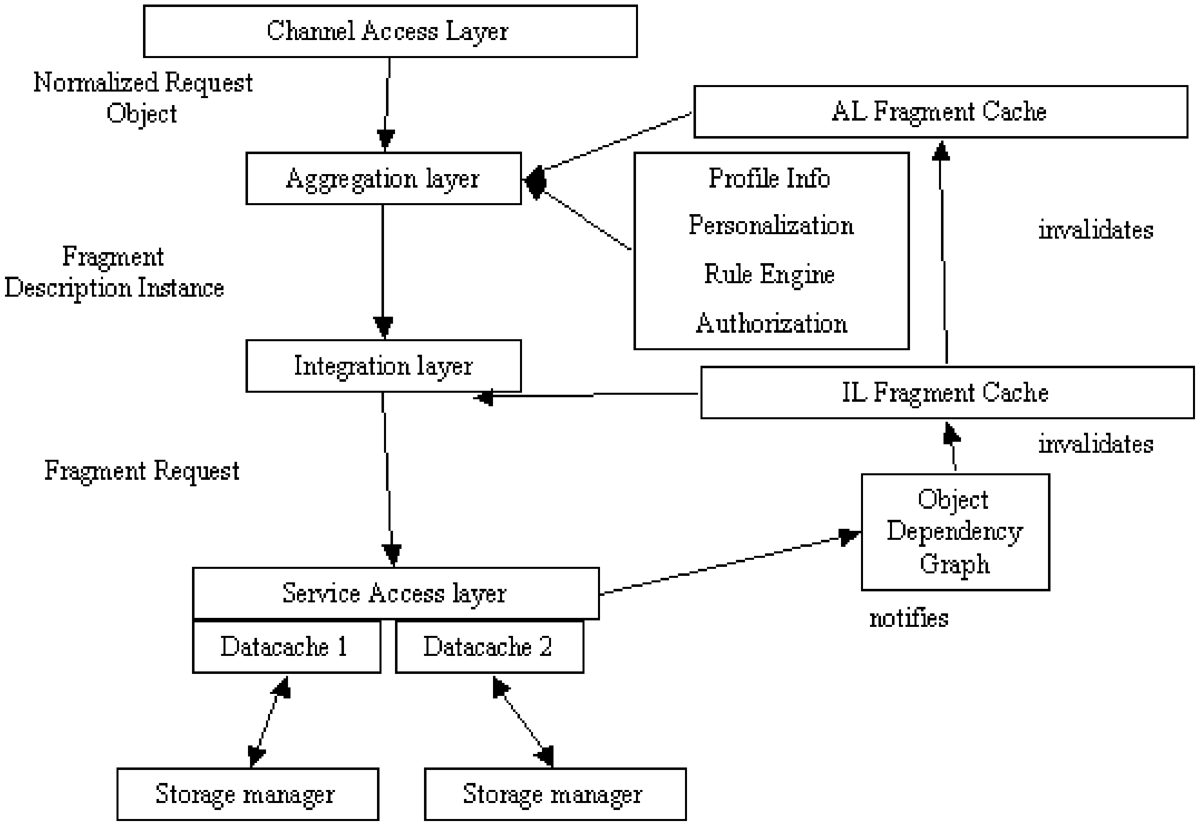
Quotes default fragment

80% of users do NOT customize many services. Using the standard (cached) quotes fragment saves HUNDREDS of backend requests/min. and makes the AEP possible!

Standard content fragments shared across homepages

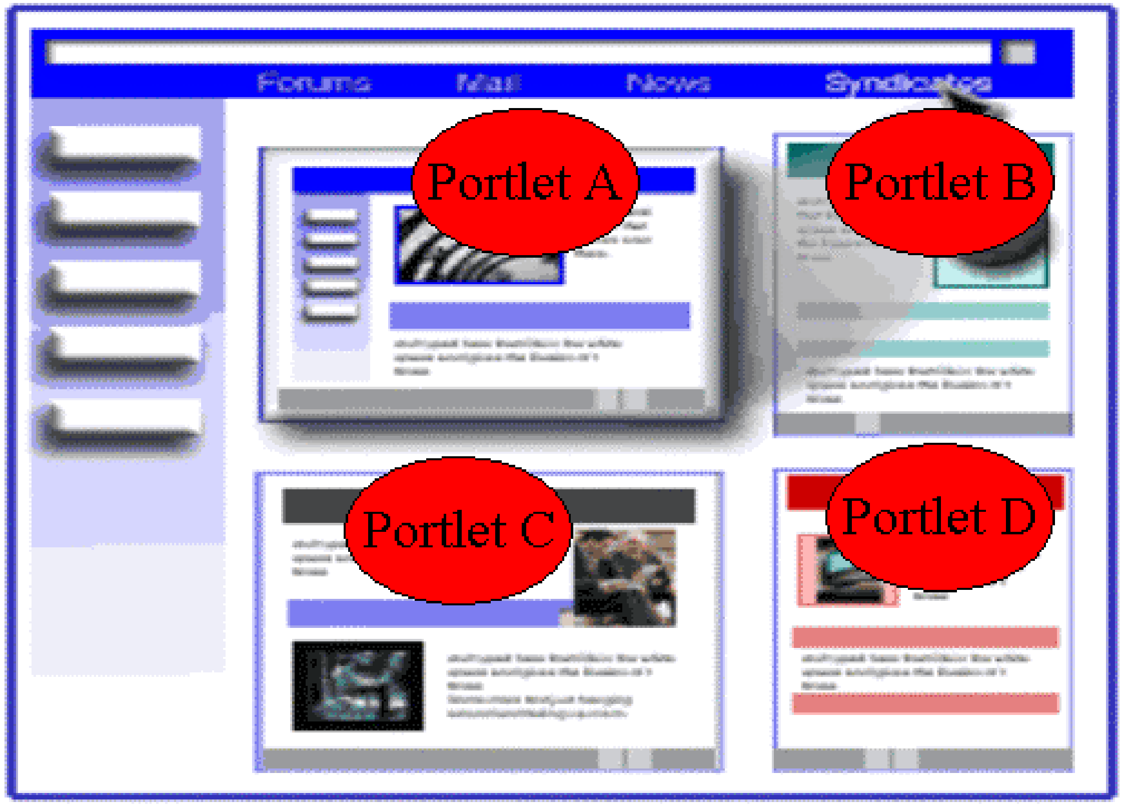

Home-page In HTML (User A)
Home-page In HTML (User B)
FDS
FDS
Home-page Construction fragment (personalized)
Home-page Construction fragment (personalized)
FDS
Quotes subject HTML (default)
FDS
Quotes subject XML (default)
FDS
Quotes subject Domain Object

Fragment Based Information Architecture



Goal: minimize backend access through fragment assembly (extension of IBM Watson research)

Jetspeed, an alternative to a fragment architecture

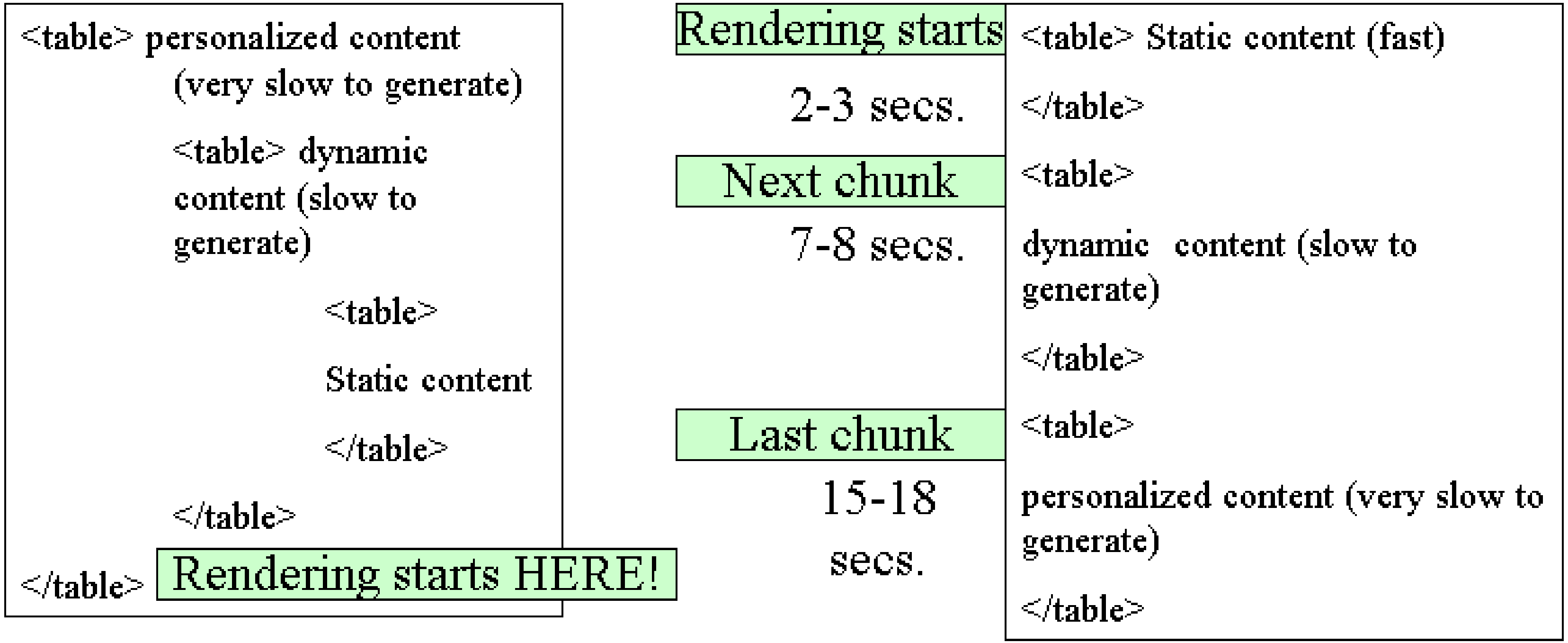


Will it scale on an AEP level?

GUI Design Problems

- Incremental load process (Homepage not wrapped in a big table)
- Static, dynamic and dynamic and personalized fragments ordered for sequential delivery on the homepage
- Limit the information shown on a homepage in accordance with your distribution architecture

Tables, Information Order and performance



```
<table> personalized content
        (very slow to generate)
        <table> dynamic
        content (slow to
        generate)
                <table>
                Static content
                </table>
        </table>
</table>
```

Rendering starts HERE!

Rendering starts

2-3 secs.

Next chunk

7-8 secs.

Last chunk

15-18 secs.

```
<table> Static content (fast)
</table>
<table>
dynamic  content (slow to
generate)
</table>
<table>
personalized content (very slow to
generate)
</table>
```

It makes a bad (slow) user experience to show nothing for 20 seconds and then the complete page. It is much better to show something quick, the next piece after 7-8 sec. and the rest when it’s done. Of course, this requires a properly structured homepage.

Implementation Problems

- Too many objects built and collected
- Too many exceptions thrown
- XML performance: no parser pooling, wrong parser selected
- No object pool
- Database connection hold times too large

Tools and Technologies

- Apache Web Server
- Web Application Server
- Visual Age Java IDE
- Oracle Database
- CVS
- Twiki collaboration tool
- Photoshop/Dreamweaver
- TogetherJ

- Object Oriented Development
- Design Patterns
- Java Idioms
- Web (http, html)
- SQL, XML, XSL, XPATH, JSP
- TCP/IP, SSL,
- EJB, J2EE, JMS, JNDI,
- RMI, CORBA

Design Patterns and Idioms: Double Checked Locking

```
// Single threaded version
 class Foo {
   private Helper helper = null;
   public Helper getHelper() {
     if (helper == null)
         helper = new Helper();
     return helper;
     }
   // other functions and members...
}
```

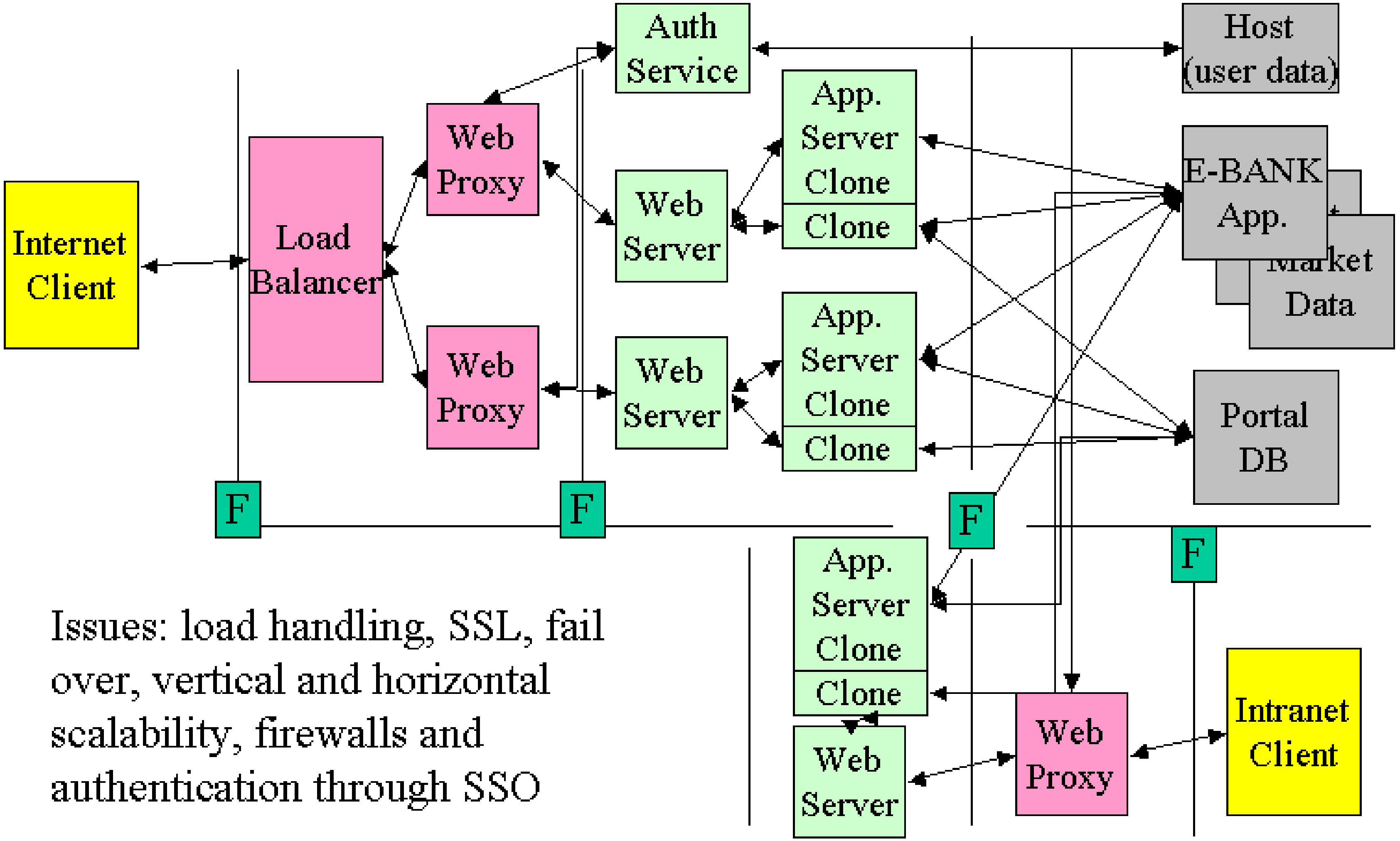
```
// Broken multithreaded version
 // "Double-Checked Locking" idiom
 class Foo {
   private Helper helper = null;
   public Helper getHelper() {
     if (helper == null)
       synchronized(this) {
         if (helper == null)
           helper = new Helper();
       }
     return helper;
     }
   // other functions and members...
   }
```

```
Symantec JIT compiled code:
0206106A   mov        eax,0F97E78h
0206106F   call       01F6B210
; allocate space for
; Singleton, return result in eax
02061074   mov        dword ptr
[ebp],eax
; EBP is &singletons[i].reference
; store the unconstructed object here.
02061077   mov        ecx,dword ptr
[eax]
 ; dereference the handle to
; get the raw pointer
02061079   mov        dword ptr
[ecx],100h
```

Infrastructure Problems

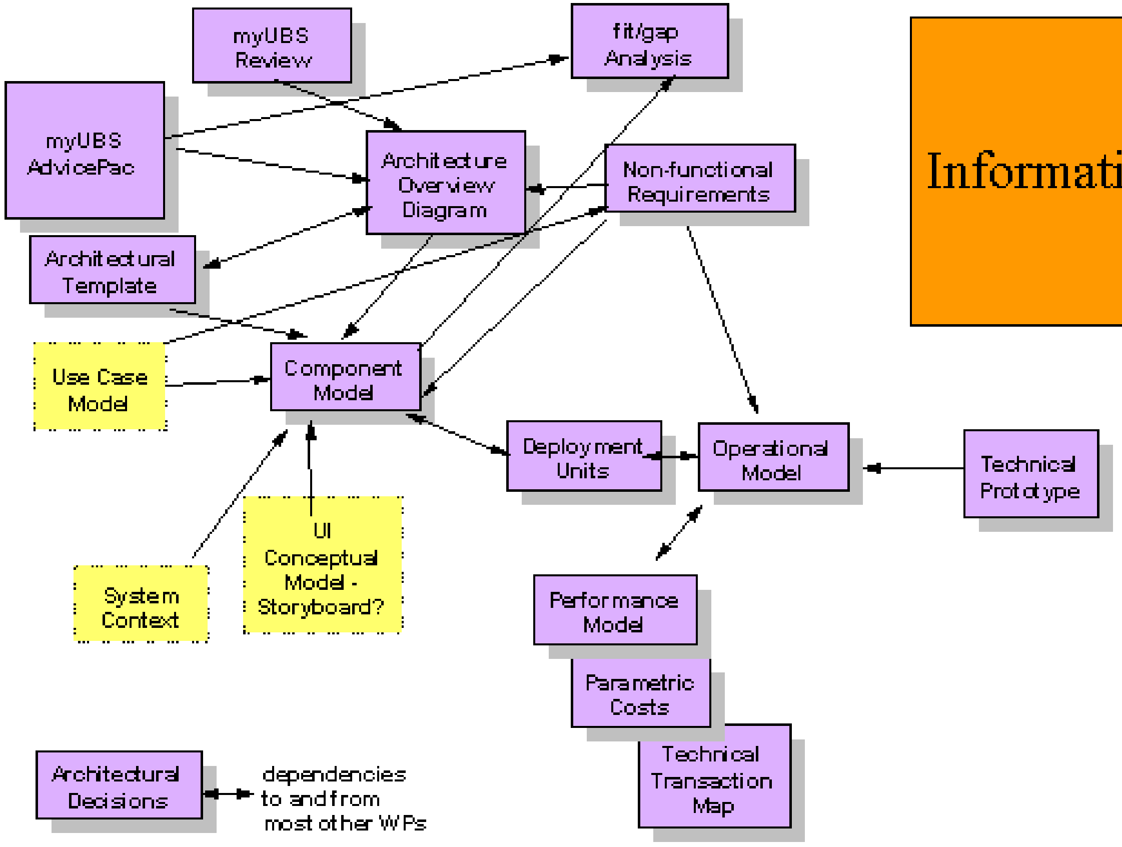
- JVM version did not support multiple CPU’s of target platform
- No SSL acceleration used, wrong CPU’s
- No end-to-end load testing possible
- No distributed cache for Application Server clones: Too much memory used, Database load not reduced (Websphere problem)

Physical Portal Architecture: Web Cluster



Issues: load handling, SSL, fail over, vertical and horizontal scalability, firewalls and authentication through SSO

Architecture Domain for myUBS



myUBS Review

fit/gap Analysis

myUBS AdvicePac

Architecture Overview Diagram

Non-functional Requirements

Architectural Template

Use Case Model

Component Model

Deployment Units

Operational Model

Technical Prototype

UI Conceptual Model - Storyboard?

System Context

Performance Model

Parametric Costs

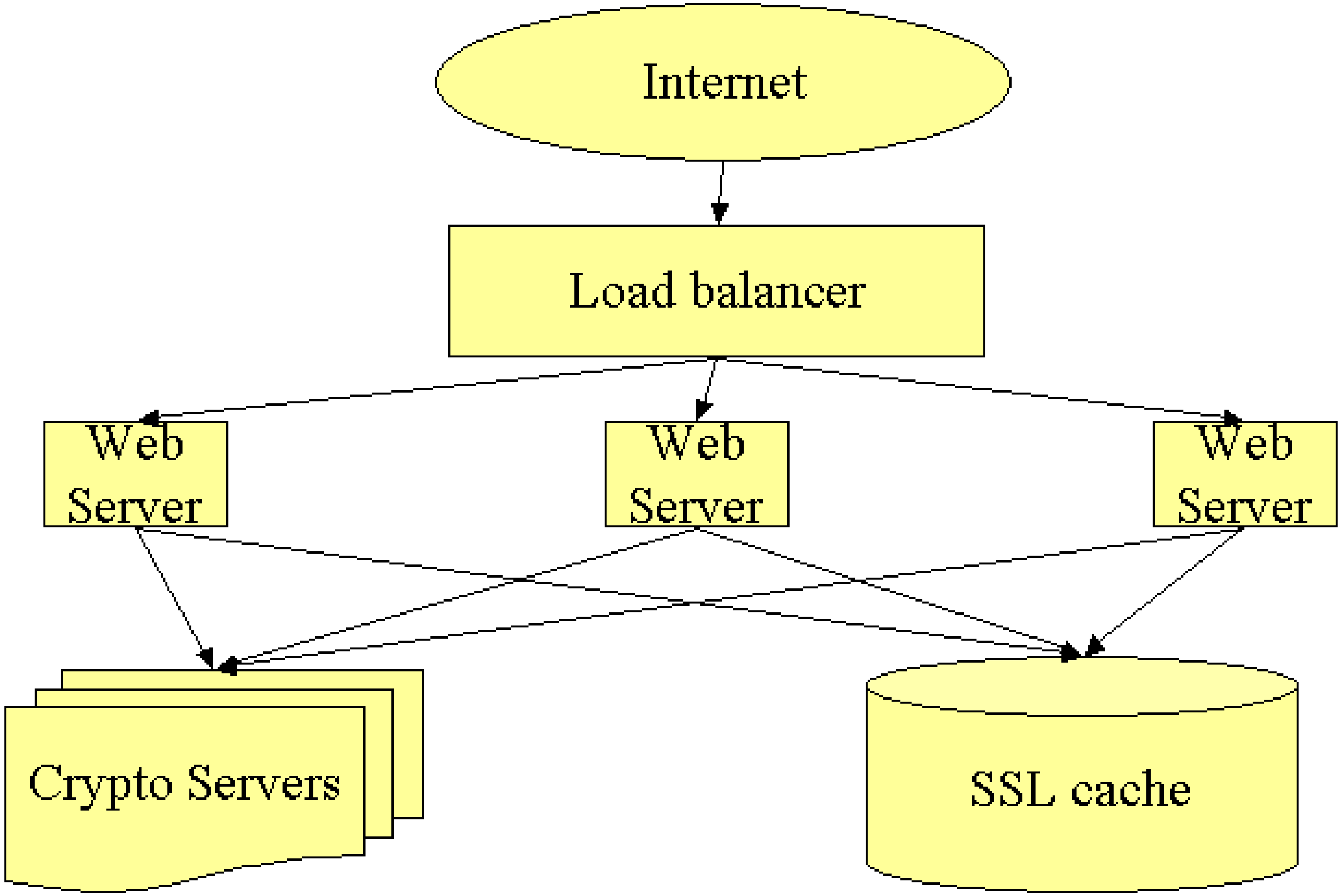
Technical Transaction Map

Architectural Decisions

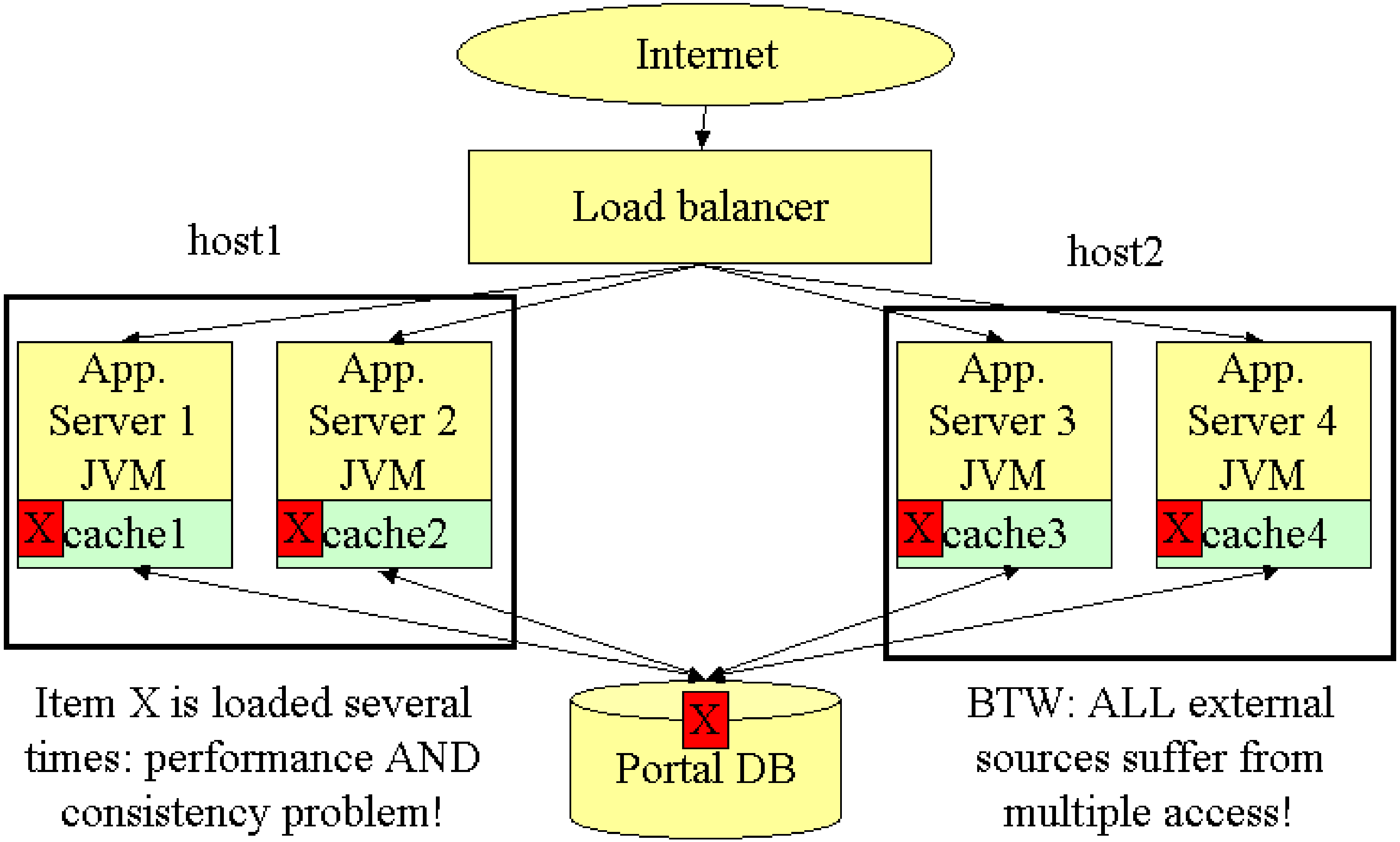
dependencies to and from most other WPs

Information Model???

Load-balanced SSL (apache case study)



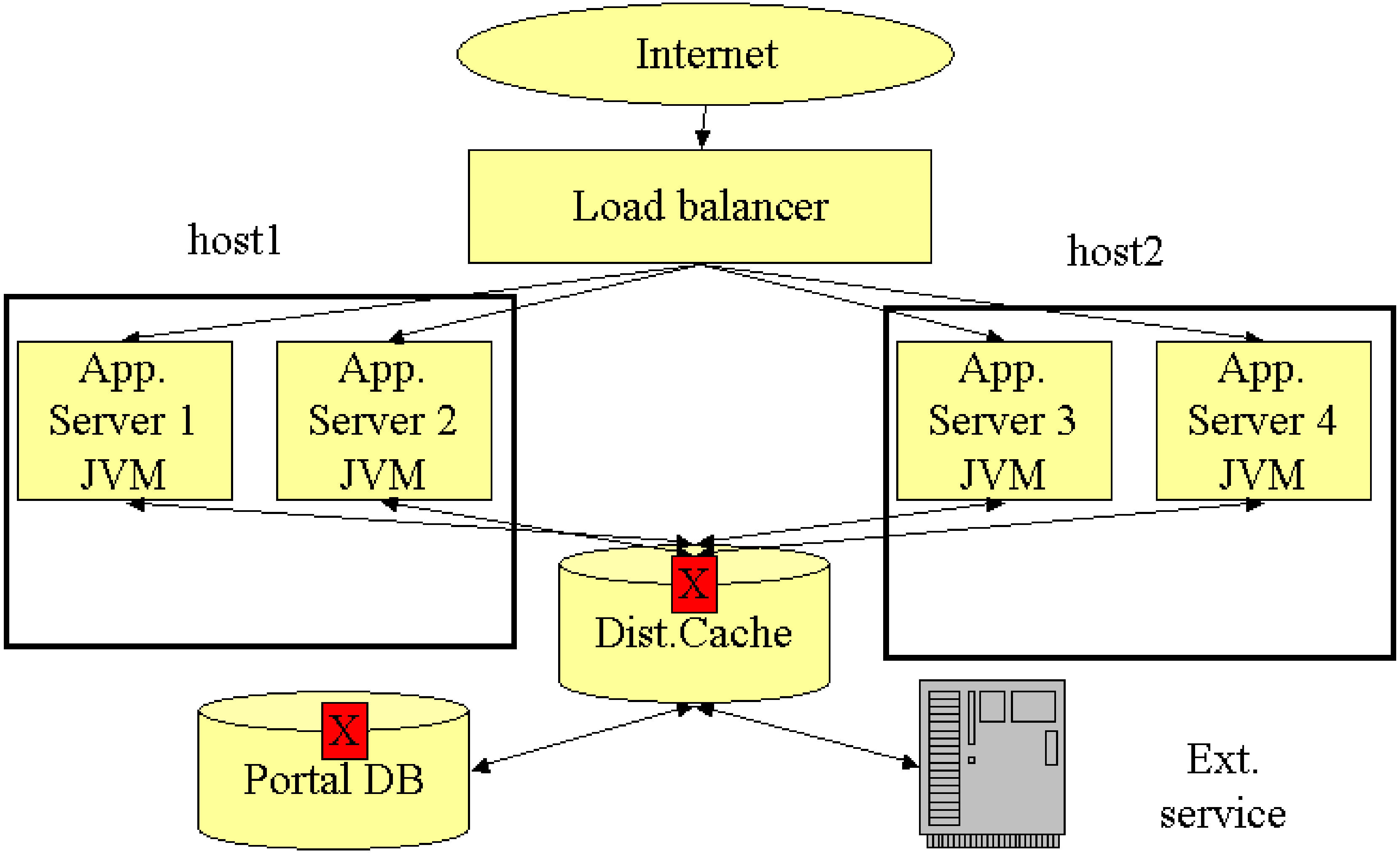
Pull Model Without Distributed Cache



Item X is loaded several times: performance AND consistency problem!

BTW: ALL external sources suffer from multiple access!

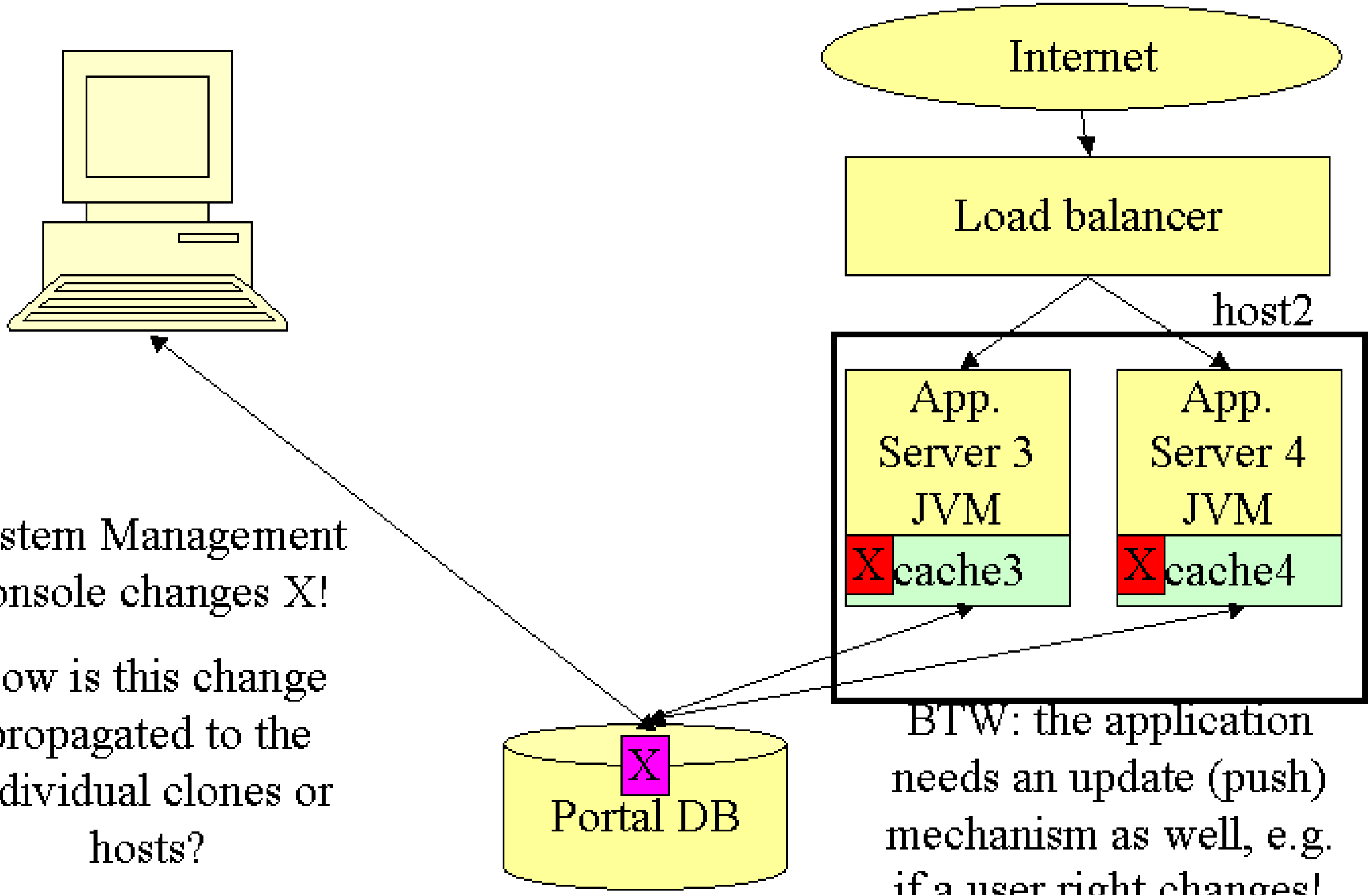
Pull Model With Distributed Cache



Maintenance Problems

- Upgrades of live system
- Extensions to live system
- Interface changes of suppliers/backends
- New system management requirements to manage application server clones

Pull Model: Update Problem

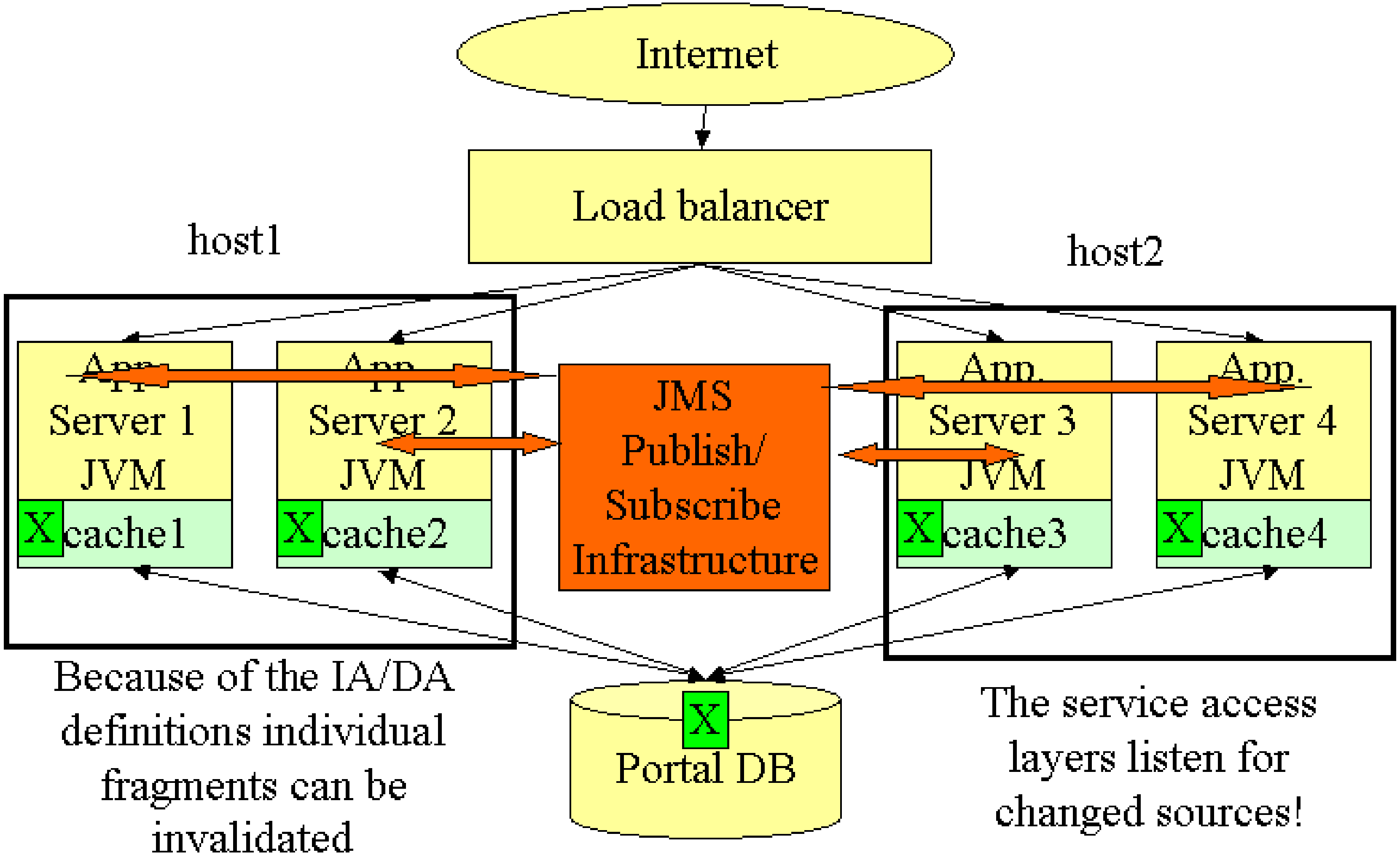


System Management console changes X!

How is this change propagated to the individual clones or hosts?

BTW: the application needs an update (push) mechanism as well, e.g. if a user right changes!

Push Model With Update Notifications



Because of the IA/DA definitions individual fragments can be invalidated

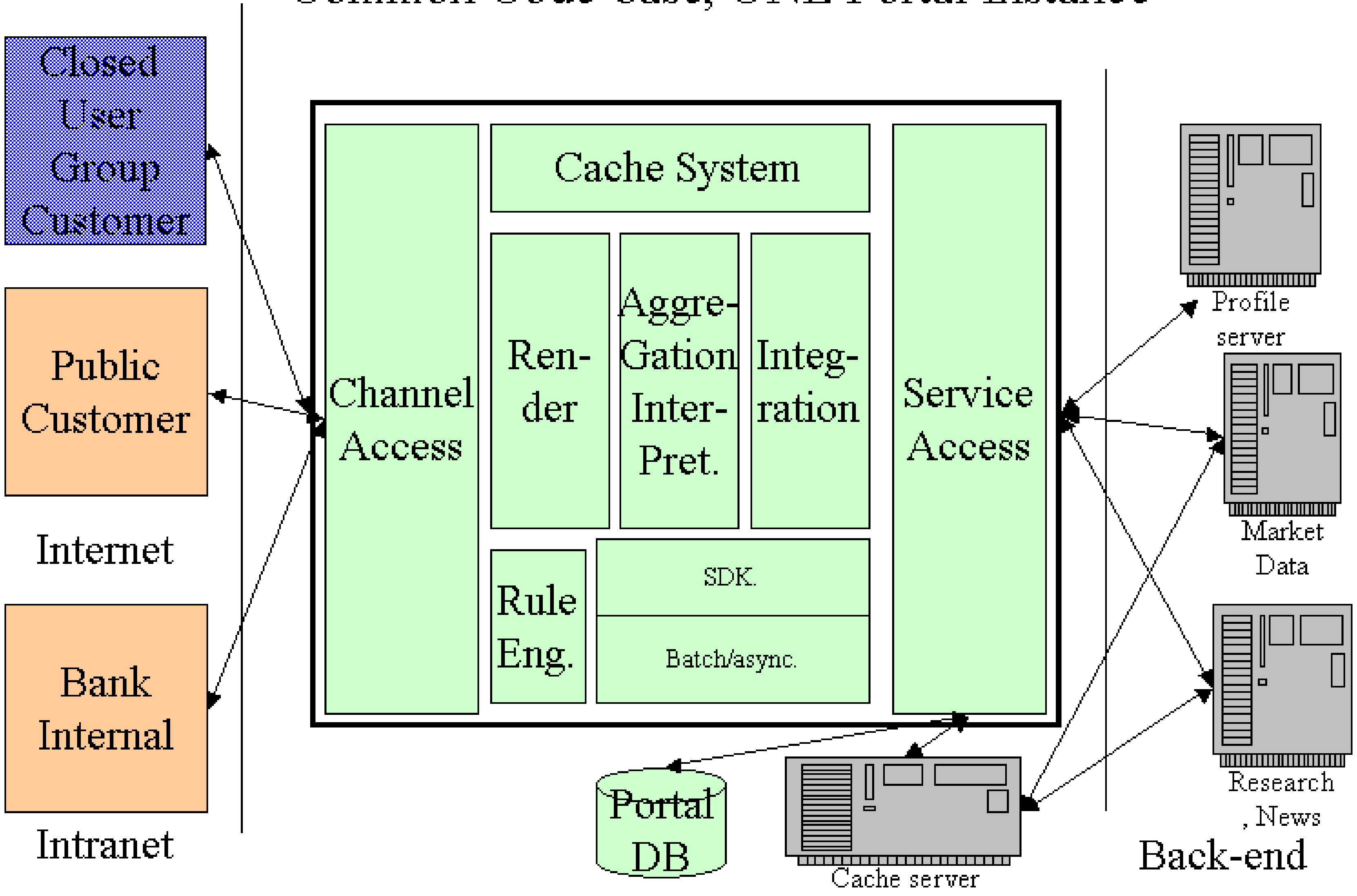
The service access layers listen for changed sources!

AEP: can there only be ONE?

- Is it clever to have only one physical **instance** of an AEP? (update and extension problem, QOS for special customers)
- What is the price of having only one **code-base**? (missed time to market, missed optimization, missed functionality, missed opportunities)

Scalability issues will force you to use different architectures, implementations and infrastructure. Timeframes will differ considerably

Common Code-base, ONE Portal Instance



One Portal Code-Base Only

- Must contain implementations for ALL functional requirements across ALL scalability needs: expensive and hard to develop
- Needs Service development kit
- Driven by the “re-use” evangelists

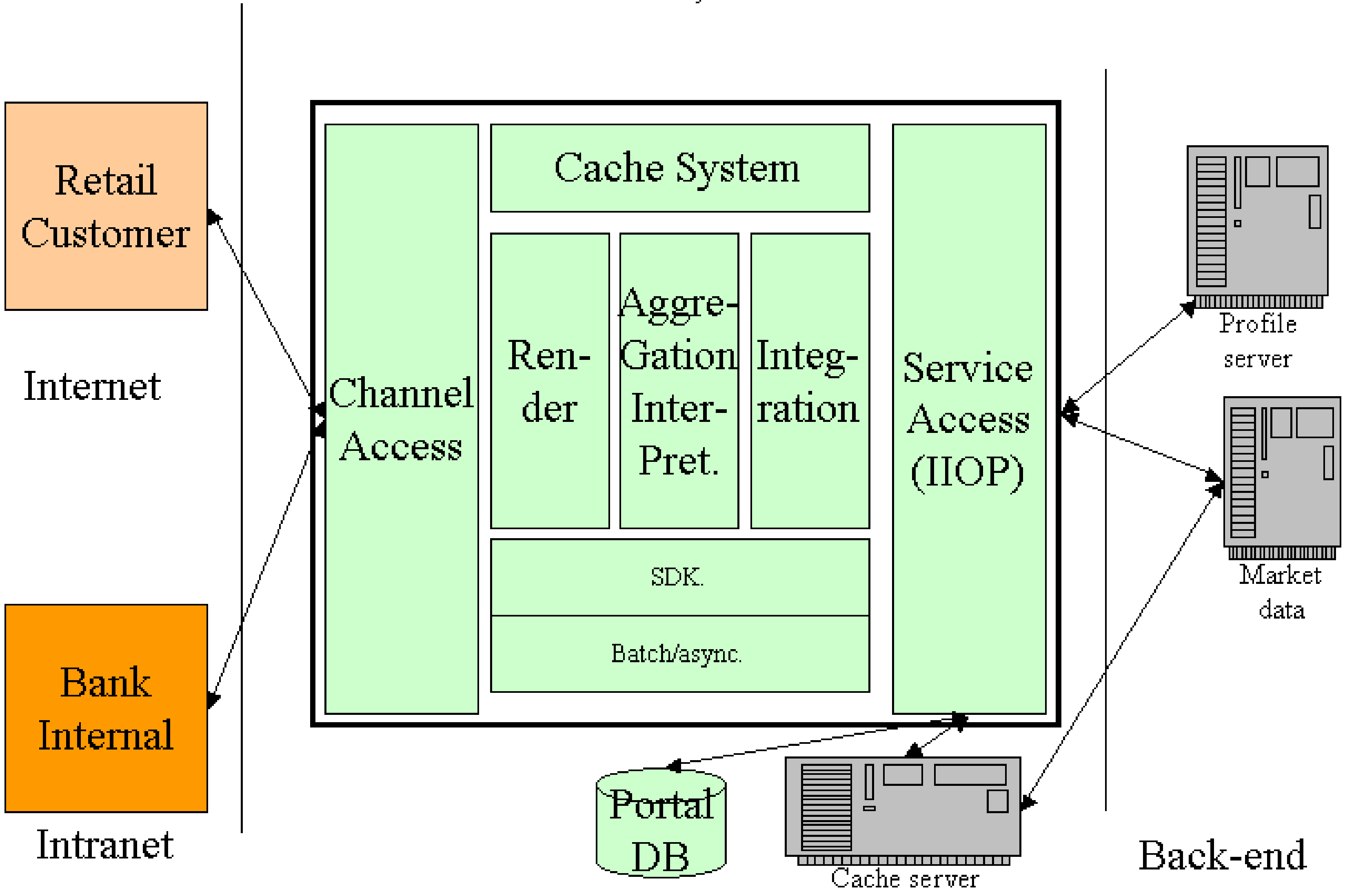
My guess: It won’t work!

One Portal Instance only

- Hard to guarantee Quality of Service for special (paying, high-net-worth) customers
- Upgrades are hard and dangerous!!
- Upgrades to individual components are tied to general release plan!

My guess: It won’t work!

Common Code-base, Public Portal Instance



Retail Customer

Internet

Bank Internal

Intranet

Channel Access

Cache System

Ren-der

Aggre-Gation Inter-Pret.

Integ-ration

SDK.

Batch/async.

Service Access (IIOP)

Profile server

Market data

Portal DB

Cache server

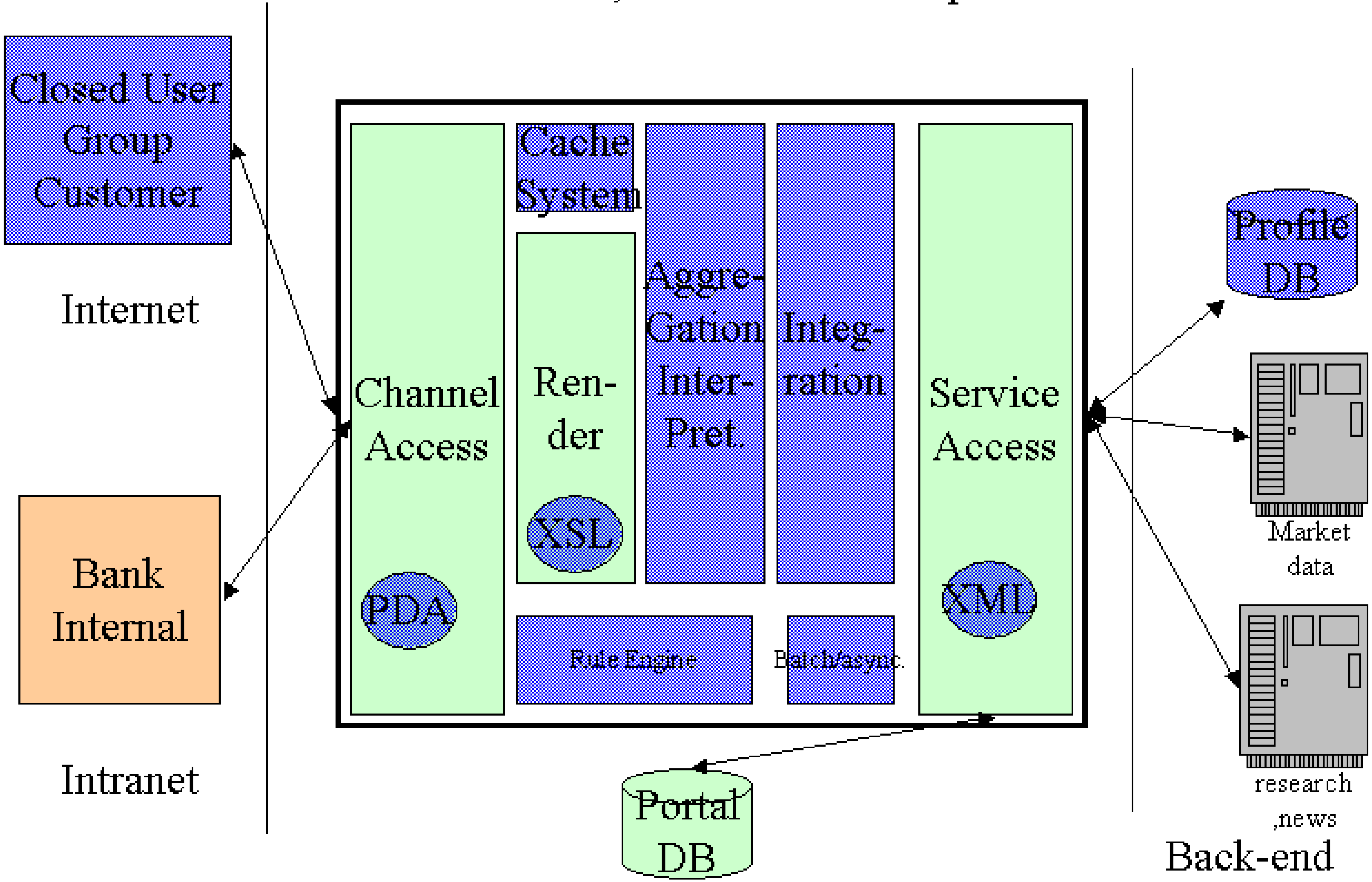
Back-end

Public Portal

- Implementation must work on a much larger scale
- Implementation requires different architecture (fragments etc.)
- No rule engine (performance). Static template based rendering
- High-speed profile server necessary
- High-speed cache server necessary

The common code base would have to follow the public portal requirements first - because of the scalability problems.

Different Code-base, Closed User Group Portal Instance



Closed User Group Portal

- Can live with a simpler architecture because of fewer scalability problems
- Does not need Service Development Kit. Needs less caching and batch processing.
- Rule engine possible (fewer user). Advanced XSL based rendering, better integration and aggregation
- No high-speed profile server necessary
- No high-speed cache server necessary
- Much faster time to market.
- No need to change back-ends because of fewer requests

Federated Portals?

- Given the problems of centralized AEPs – are federated portals an alternative? Both technically and politically?
- Could the rush to “Webservices” help?
- Is external service access realistic? During request time?
- How is SSI handled? How would one share personalization information?

Management Problems

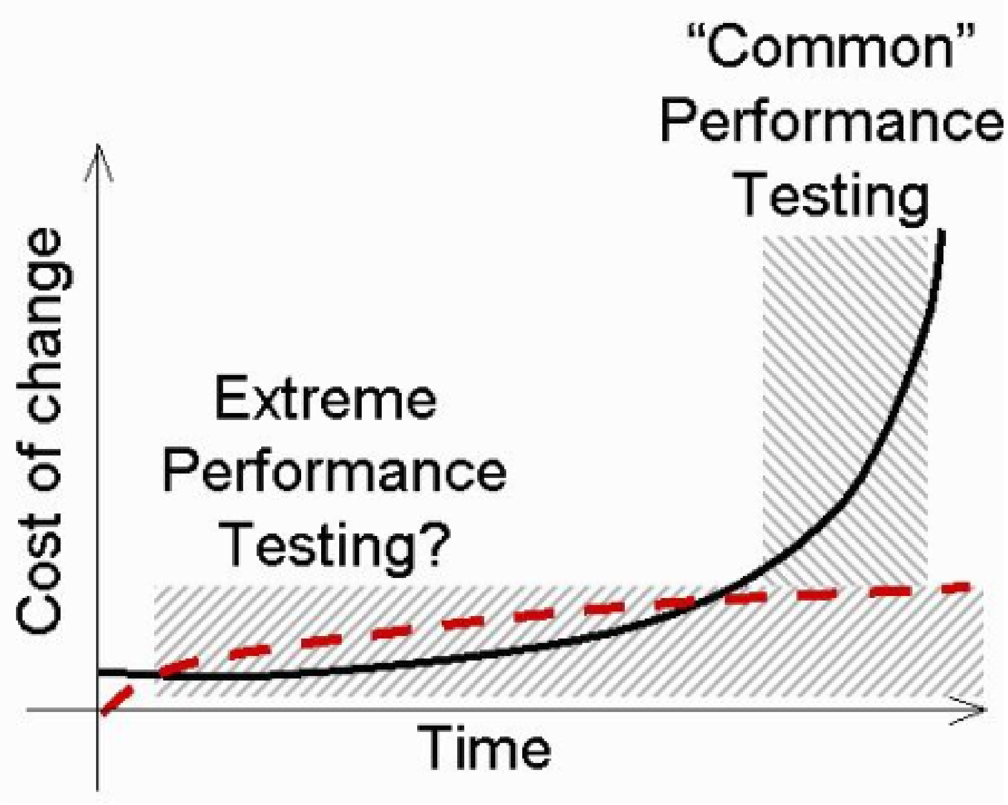
- AEP require incremental but end-to-end improvement processes to reach performance and throughput – hard to sell to management and business, looks like “probing”. Especially if combined with an XP development approach.

- Separating portal (services) and infrastructure (SSI, authorization etc.) is hard: Is database replication a feature of an AEP or of the infrastructure? Who pays for it?

- Load-tests are NO LONGER acceptance tests! They are permanently required and will cause software changes.

- Many intranet applications are currently implementing portal features (offering content from different sources, storing personalization information etc.). Centralized AEP approaches are not very popular here.

Portal Load Tests are “Extreme”



Source: Ted Osborne from Empirix

Lessons learned

- Create Information Architecture first
- Do not introduce new technology without previous scalability and stability tests (e.g. rule-engine)
- Start with a closed user group.
- Do permanent load tests to improve the architecture
- Change physical setup to fit your application architecture
- Track legacy system performance
- Use architecture re-factoring approach
- Learn about the technology specific problems of your approach (e.g. Java performance)

There is NO SINGLE cause of performance or stability problems. An Advanced Enterprise Portal needs to be optimized from end-to-end, from Browser caching to backend system speed and reliability

Resources I

Caching:

Engineering Highly Accessed Web Sites for Performance, J.Challenger, A.Iyengar IBM Watson Research Center

Design Alternatives for Scalable Web Server Accelerators (j.Song, et. Alii) IBM T.J.Watson Research Center. Uses cache arrays with CARP for caching.

Pooling:

www.jboss.org, Minerva Object Pool

Graham Glass, When less is more: a compact toolkit for parsing and manipulating XML http://www-106.ibm.com/developerworks/xml/library/x-elexml/index.html?open&l=136,t=gr,p=electricXML

Resources II

Performance:

Java Performance Tuning (Java Series (O'Reilly)) -- by Jack Shirazi;

Architecture:

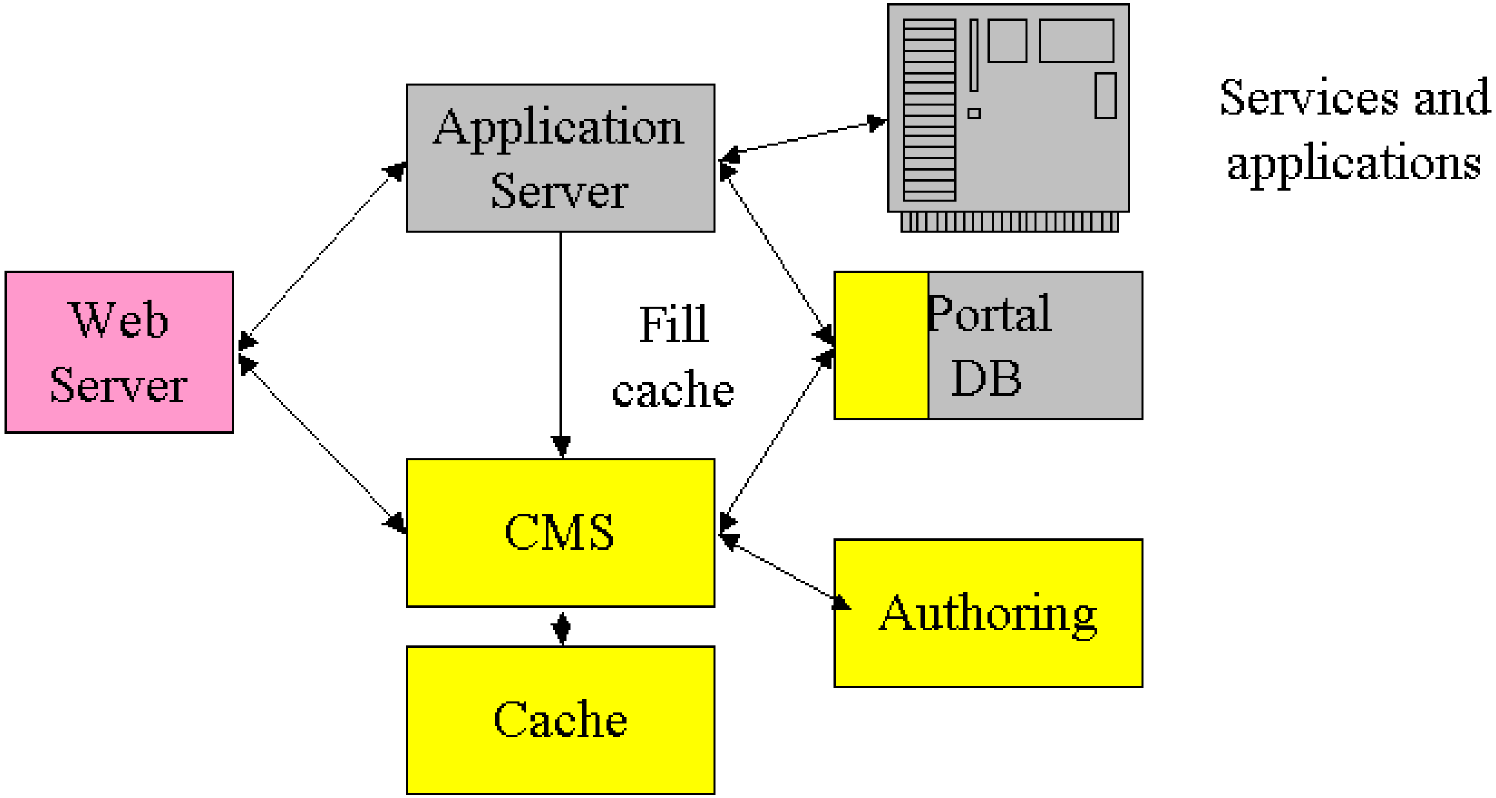
Building and Managing Dynamic Database Driven Web Sites. A talk from a Seybold seminar. Most important: to realize that using the typical JSP/J2EE push model (like myUBS does) the business users wont have a chance to EDIT the dynamic sites the way they are used to e.g. with their static intranet sites). Without an information-centric "pull-model" dynamic content always implies "programmed" content.

http/Servlets/Compression

www.servlets.com: server site caching example from Jason Hunter. Servlet interceptor for dynamic but non-personalized page PER servlet.

Fineground Condenser Product Brief. Like packeteers.com a product that does browser detection and compression. http://www.fineground.com

Portal Architecture Option: Content Management System (CMS) and Application Server



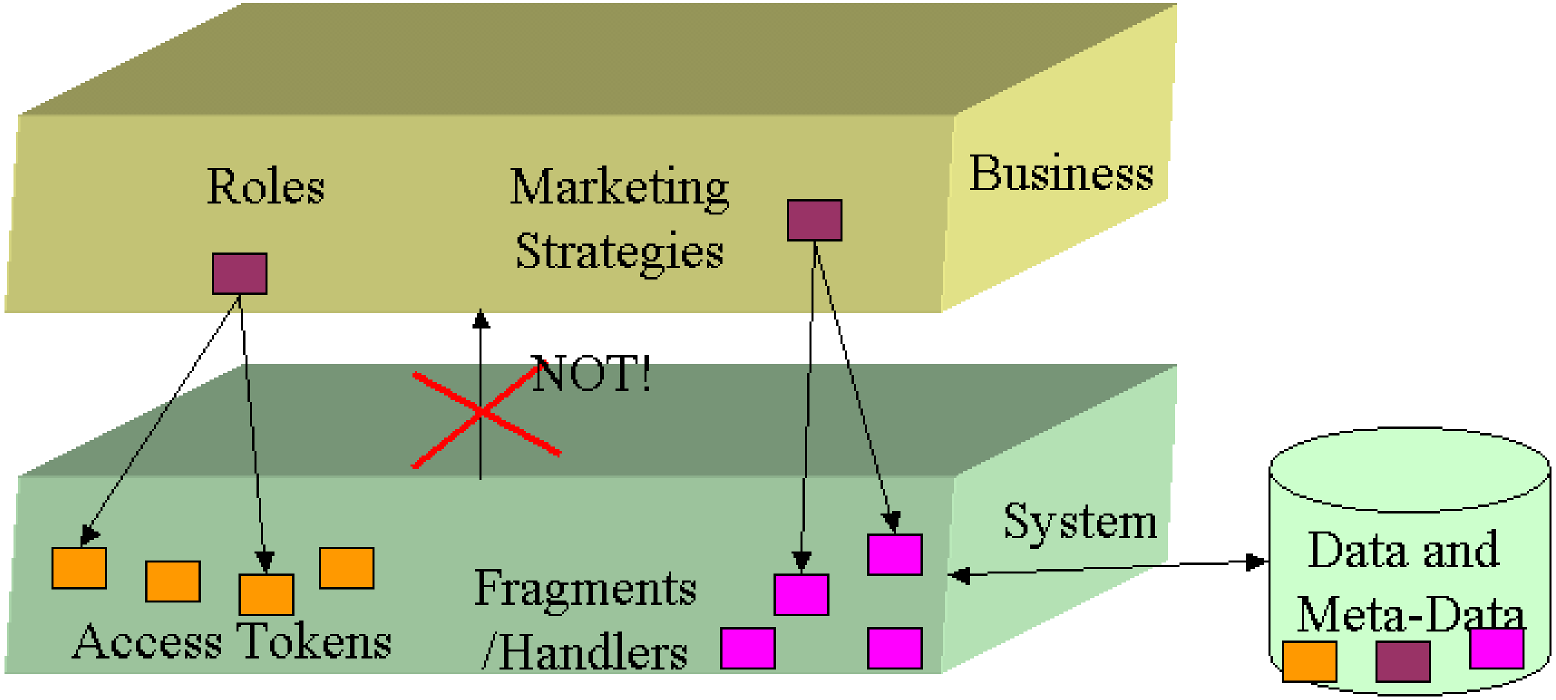
Who assembles complex pages? Who invalidates cache? Can App.Server and CMS share authorization concepts?

Domain Analysis

- Business Logic vs. System Behavior
- Information Architecture vs. Distribution Architecture

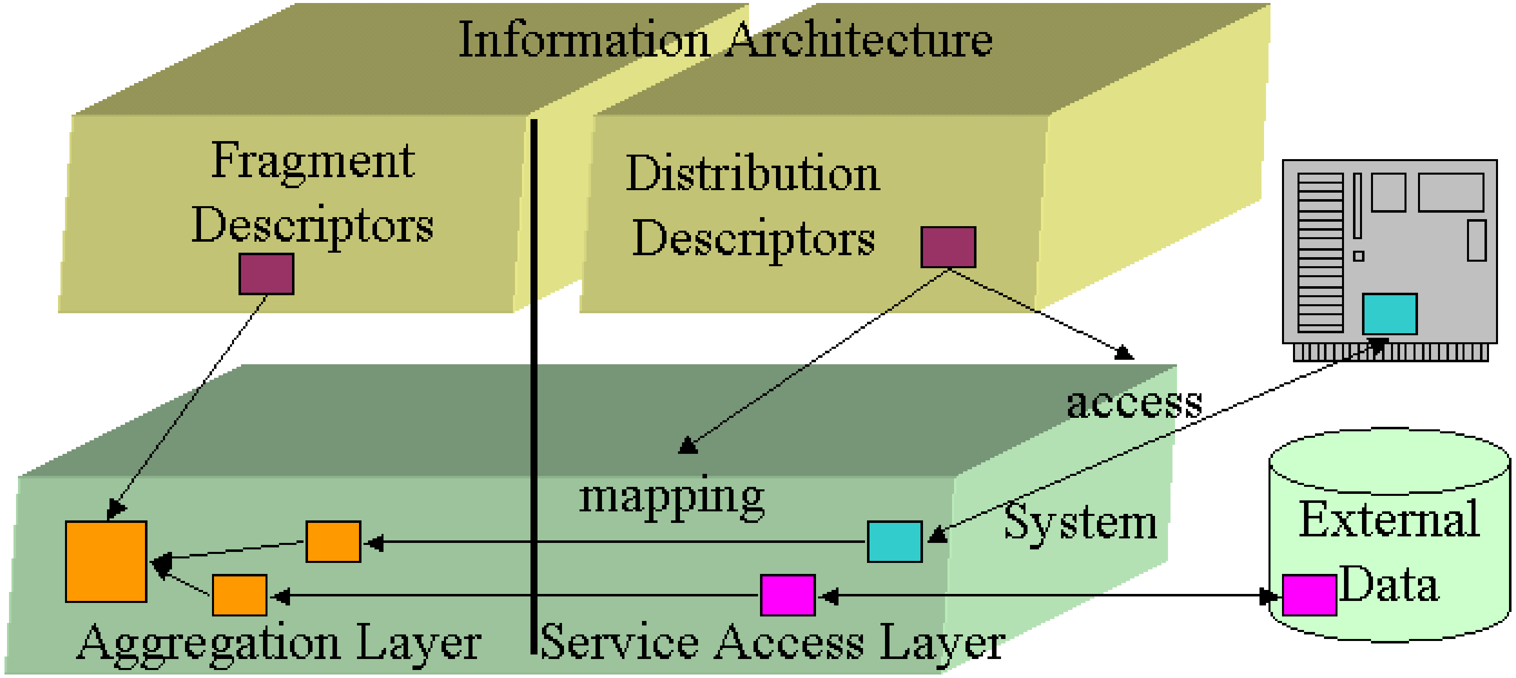
It is a Domain Analysis job to define the hot spots. In this case: the changing business rules (marketing concepts) as well as service changes in front-end and back-end

Business Conceptual Model vs. System Model



The portal realizes both conceptual levels. The system level does NOT use business conceptual terms and needs not change if the business concepts change!

Information Structure, Aggregation and Access



By separating logical (what) and physical (where, how) qualities, information can be easily re-structured, extended or physically moved