



Industry Vision



1



Sang Shin

sang.shin@sun.com
www.plurb.com
Java™ Technology Evangelist
Sun Microsystems, Inc.


2



Courses I am teaching

- ? XML (2001)
- ? Distributed programming using Jini and JavaSpaces technology (2002)
- ? Web services programming using XML and Java technology (2002)
- ? J2EE programming (2003, currently going on right now, **you can take it online free**)
- ? Advanced J2EE programming (summer, 2003)
- ? Java and Web services security (in the future)
- ? All course materials are available online at www.plurb.com


3




Agenda

- ? Evolution of Computing
- ? Software as a Service Model
- ? Evolution of Distributed Computing
- ? Evolution of Enterprise Application Development Frameworks
- ? Why Java and J2EE?

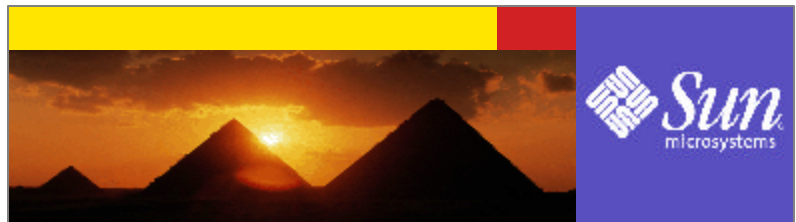
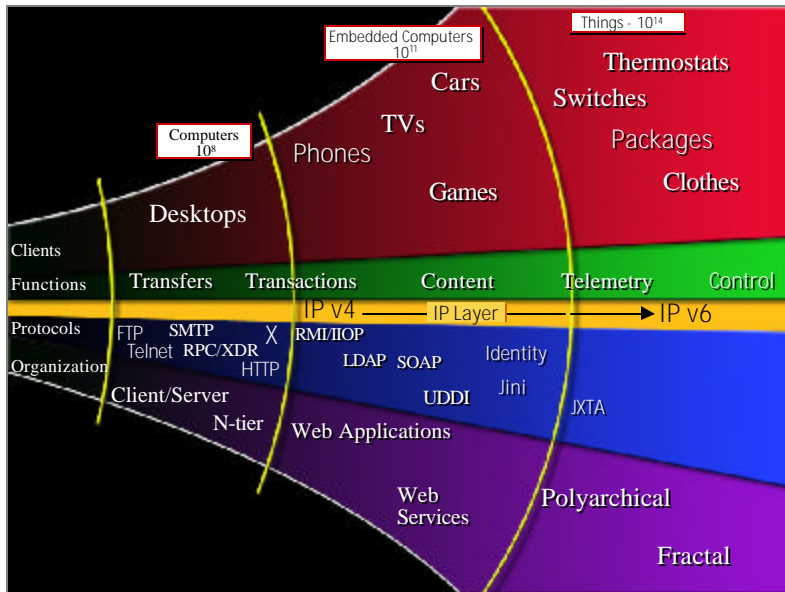
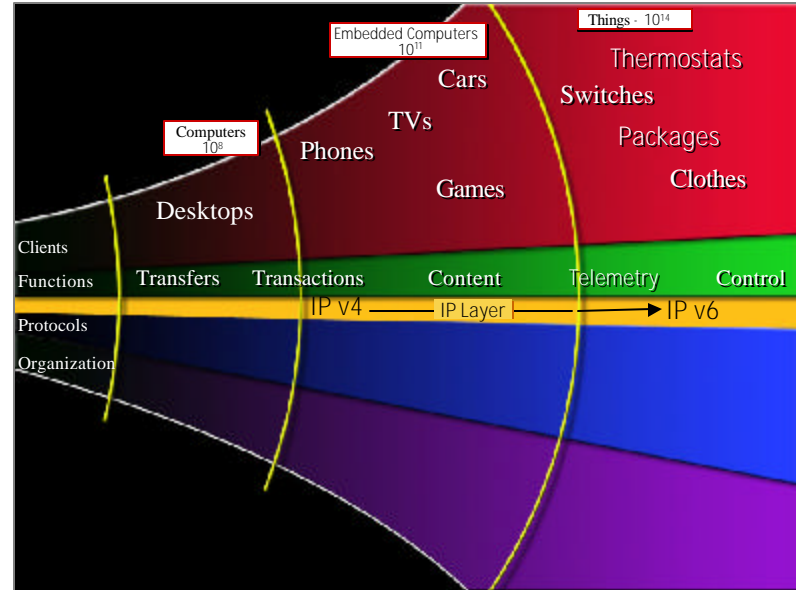
4




Evolution of Computing



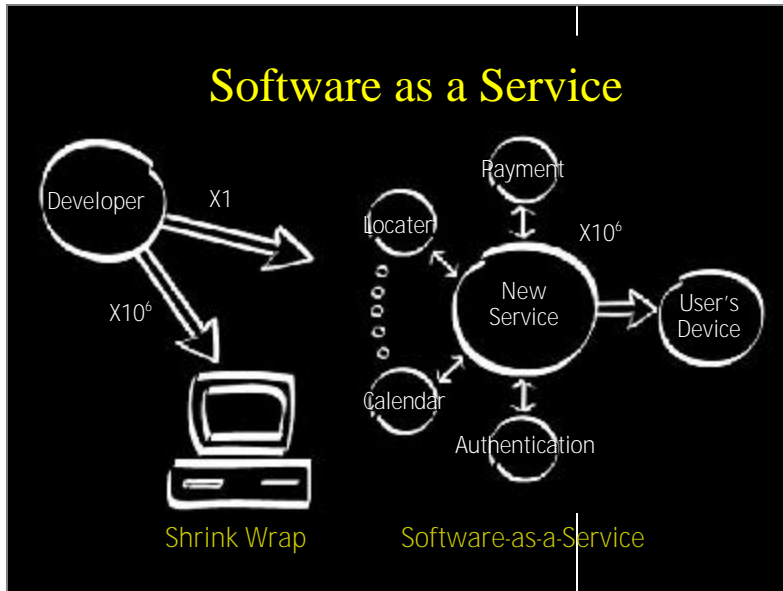
5



Software as a Service



8



Shrink Wrap Model of Today

- ? End user
 - Have to pay for functionalities you don't need
 - Have to install and configure in order to use software
 - No possibility to plug-in the feature you need
 - You have to do your own version control
- ? Developer
 - No chance to distribute your software to the mass
- ? Deployer
 - Constant headache of software upgrade and maintenance
 - Close to 70% of TCO is maintenance of computers and software

10

Service-driven Network

- ? End user
 - Download only pieces you need during runtime
 - Always get up-to-date software pieces
 - You pay only the things you use
- ? Developer
 - Level-playing field for everybody
- ? Deployer
 - No software upgrade and maintenance headache
 - TOI is way down

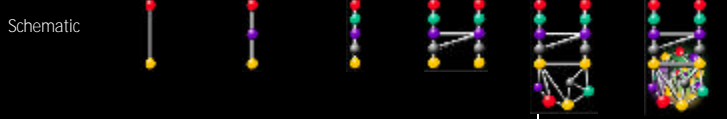
11

Evolution of Distributed Computing

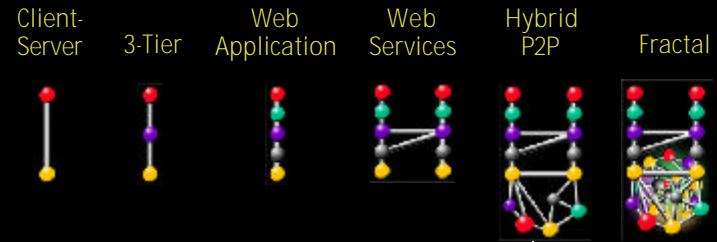
12

Platform Evolution

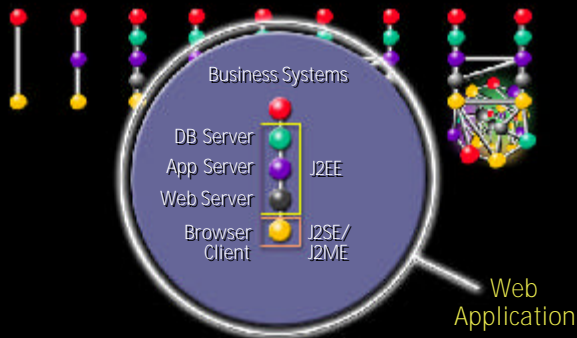
Catch Phrase	The Network Is the Computer	Objects	Legacy to the Web	The Computer Is the Network	Network of Embedded Things	Network of Things
Scale	100s	1,000s	1,000,000s	10,000,000s	100,000,000s	100,000,000s
When/Peak	1984/1987	1990/1993	1996/1999	2001/2003	1998/2004	2004/2007
Leaf Protocol(s)	X	X	+HTTP (+JVM)	+XML Portal	+RM	Unknown
Directory(s)	NS, NS+	+CDS	+LDAP(*)	+UDDI	+Jini	+?
Session	RPC, XDR	+CORBA	+CORBA, RM	+SOAP, XML	+RM/Jini	+?



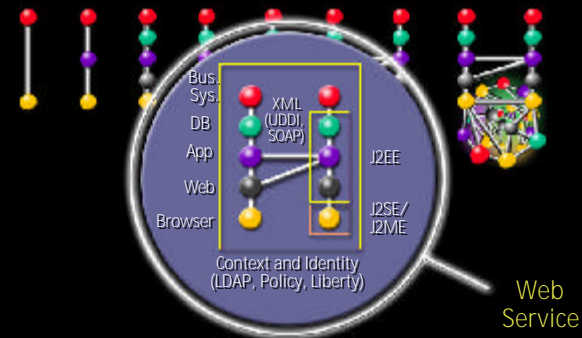
Communication Patterns

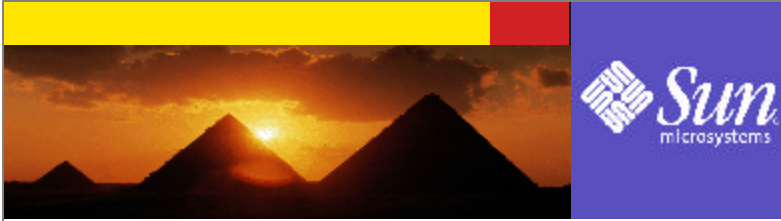


Communication Patterns: Java™ 2




Communication Patterns: Sun ONE





Evolution of Enterprise Application Frameworks




17



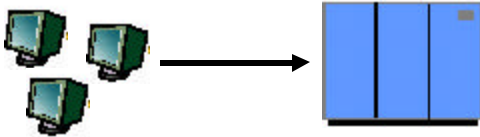
Evolution of Enterprise Application Framework

- ? Single tier
- ? Two tier
- ? Three tier
 - RPC based
 - Remote object based
- ? Three tier (HTML browser and Web server)
- ? Proprietary app server
- ? Standard app server

18




Single Tier (Mainframe-based)



- ? Centralized model (as opposed distributed model)
- ? Dumb terminals are directly connected to mainframe
- ? Presentation, business logic, and data access are intertwined in one monolithic mainframe application

19

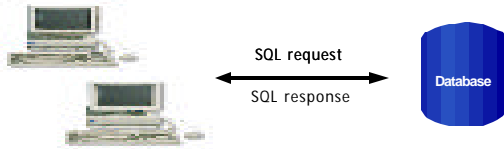


Single-Tier: Pros & Cons

- ? Pros:
 - No client side management is required
 - Data consistency is easy to achieve
- ? Cons:
 - Functionality (presentation, data model, business logic) intertwined, difficult for updates and maintenance and code reuse

20

Two-Tier



- ? **Fat clients talking to backend database**
 - SQL queries sent, raw data returned
- ? Presentation, Business logic and Data Model processing logic in client application

21

Two-Tier

- ? Pro:
 - DB product independence (compared to single-tier model)
- ? Cons:
 - Presentation, data model, business logic are intertwined (at client side), difficult for updates and maintenance
 - Data Model is "tightly coupled" to every client: If DB Schema changes, **all clients break**
 - Updates have to be deployed to all clients making System maintenance nightmare
 - DB connection for every client, thus difficult to scale
 - Raw data transferred to client for processing causes high network traffic

22

Three-Tier (RPC based)



- ? Thinner client: business & data model separated from presentation
 - Business logic and data access logic reside in middle tier server while client handles presentation
- ? **Middle tier server is now required to handle system services**
 - Concurrency control, threading, transaction, security, persistence, multiplexing, performance, etc.

23

Three-tier (RPC based): Pros & Cons

- ? Pro:
 - Business logic can change more flexibly than 2-tier model
 - ? Most business logic reside in the middle-tier server
- ? Cons:
 - Complexity is introduced in the middle-tier server
 - Client and middle-tier server is more tightly-coupled (than the three-tier object based model)
 - Code is not really reusable (compared to object model based)

24

Three-Tier (Remote Object based)



- ? Business logic and data model captured in objects
 - Business logic and data model are now described in "abstraction" (interface language)
- ? Object models used: CORBA, RMI, DCOM
 - Interface language in CORBA is IDL
 - Interface language in RMI is Java interface

25

Three-tier (Remote Object based): Pros & Cons

- ? Pro:
 - More loosely coupled than RPC model
 - Code could be more reusable
- ? Cons:
 - Complexity in the middle-tier still need to be addressed

26

Three-Tier (Web Server)



- Browser handles presentation logic
- Browser talks Web server via HTTP protocol
- Business logic and data model are handled by "dynamic contents generation" technologies (CGI, Servlet/JSP, ASP)

27

Three-tier (Web Server based): Pros & Cons

- ? Pro:
 - Ubiquitous client types
 - Zero client management
 - Support various client devices
 - ? J2ME-enabled cell-phones
- ? Cons:
 - Complexity in the middle-tier still need to be addressed

28



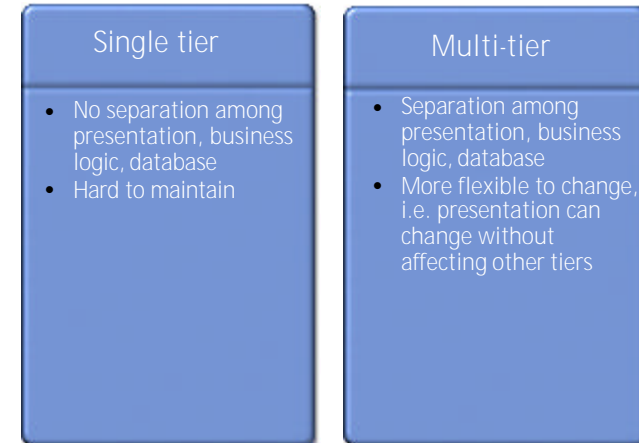
Trends

- ? Moving from single-tier or two-tier to **multi-tier** architecture
- ? Moving from monolithic model to **object-based** application model
- ? Moving from application-based client to HTML-based client

29



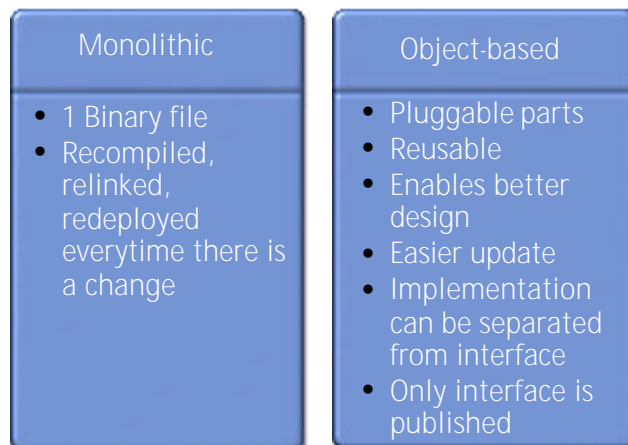
Single-tier vs. Multi-tier



30



Monolithic vs. Object-based



31



Outstanding Issues & Solution

- ? Complexity at the middle tier server still remains
- ? Duplicate system services still need to be provided for the majority of enterprise applications
 - Concurrency control, Transactions
 - Load-balancing, Security
 - Resource management, Connection pooling
- ? How to solve this problem?
 - **Commonly shared container** that handles the above system services
 - Proprietary versus Open-standard based

32



Proprietary Solution

- ? Use "component and container" model in which container provides system services in a well-defined but with proprietary manner
- ? Problem of proprietary solution: Vendor lock-in
- ? Example: Tuxedo, .NET

33



Open and Standard Solution

- ? Use "component and container" model in which container provides system services in a **well-defined and as industry standard**
- ? J2EE is that standard that also provides portability of code because it is based on Java technology and standard-based Java programming APIs

34



Why J2EE?



35



Platform Value to Developers

- ? Can use any J2EE implementation for development and deployment
 - Use J2EE RI or Sun ONE Platform Edition which are free for development/deployment and then use high-end commercial J2EE products for actual deployment
- ? Vast amount of J2EE **community resources**
 - Many J2EE related books, articles, tutorials, quality code you can use, best practice guidelines, design patterns etc.
- ? Can use off-the-shelf **3rd-party** business components

36



Platform Value to Vendors

- ? Vendors work together on specifications and then **compete in implementations**
 - In the areas of Scalability, Performance, Reliability, Availability, Management and development tools, and so on
- ? **Freedom to innovate** while maintaining the portability of applications
- ? Do not have create/maintain their own proprietary APIs

37



Platform Value to Business Customers

- ? **Application portability**
- ? Many implementation choices are possible based on various requirements
 - Price (free to high-end), scalability (single CPU to clustered model), reliability, performance, tools, and more
 - Best of breed of applications and platforms
- ? Large developer pool

38



Programming Language & Java Platform

39



Sun's Approach on Programming Language

- ? Java platform provides significant value to our customers and developers
 - It does not become the **most popular programming language** of our time by accident
- ? It's a **platform**, not just a language
 - Standard APIs available for almost anything
- ? It's the community, Stupid!
 - Most open source projects are done in Java

40

The Java™ Platform Scales!

The diagram illustrates the scaling of the Java platform across four device categories, each supported by a specific edition:

- Java Technology Enabled Devices:** Supported by the **Micro Edition**.
- Java Technology Enabled Desktop:** Supported by the **Standard Edition**.
- Workgroup Server:** Supported by the **Standard Edition**.
- High-End Server:** Supported by the **Enterprise Edition**.

The editions are represented as a 3D bar chart with three segments: Micro Edition (red), Standard Edition (blue), and Enterprise Edition (green).

Java™ Programming Language

Language use by North American developers
(Evans, Spring 2002)

J.S. Enterprises programming in the Java language
(Gartner, December 2001)

42

Java™ and Web Services

Language developers expect to use when building Web services
(Evans, Spring 2002)

Developers choice for most effective platform for building Web services
(ComputerWorld, December 2001)

43

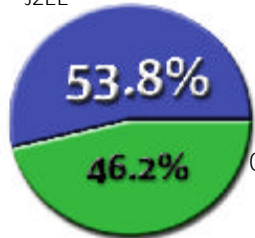
Java™ Technology for Server

Company satisfaction with their Platform of choice
J2EE

Company satisfaction with their Platform of choice
.NET

44


Java™ Technology for Server





Category	Percentage
J2EE	53.8%
Other	46.2%

Developers currently using or planning to use J2EE technology (Evans, Spring 2002)


J2EE app server market grew 39% in 2001 (Giga, March 2002)



45



Thank You!



46