JavaOne℠

Sun's 2000 Worldwide Java Developer Conference

eXtreme Development with the Java™ 2 Platform, Enterprise Edition (J2EE™)

Lessons from a B2B Start-up

Paul Hodgetts, Denise Phillips
Escrow.com
Oscar Chico
Sun Microsystems

Session Overview

- These are our experiences as a B2B dot-com start up
  - What we did
  - What worked
  - What didn't work
- We'll cover a few significant topics
- There will be Q&A time at the end
  - (Hint: We like to talk about XP!)

Overview of Topics

- Escrow.com Overview
- Challenges Faced
- Projects and Products
- System Architecture
- Architecture & Design
- Java Technologies
- J2EE Technologies
- XML/XSLT Technologies
- Tools
- Development Process
- Human Factors

Escrow.com Overview

- Provider of Transaction Settlement Solutions
  - Coordinates payment/settlement, shipping, inspection services
- Founded October 1999 as a spin off a major escrow & title transfer company
- Successive stream of products & services
  - Escrow
  - Open account, payment guarantees
  - Will call, staged payments, deposits
  - Global trade services

Service Model

- Implemented as a “BSP” (Business Service Provider)
  - Hosted applications plus back-office services
  - Integrated a back end system providing individual services
  - Integrated as co-branded components of partner site
  - Available a stand-alone site

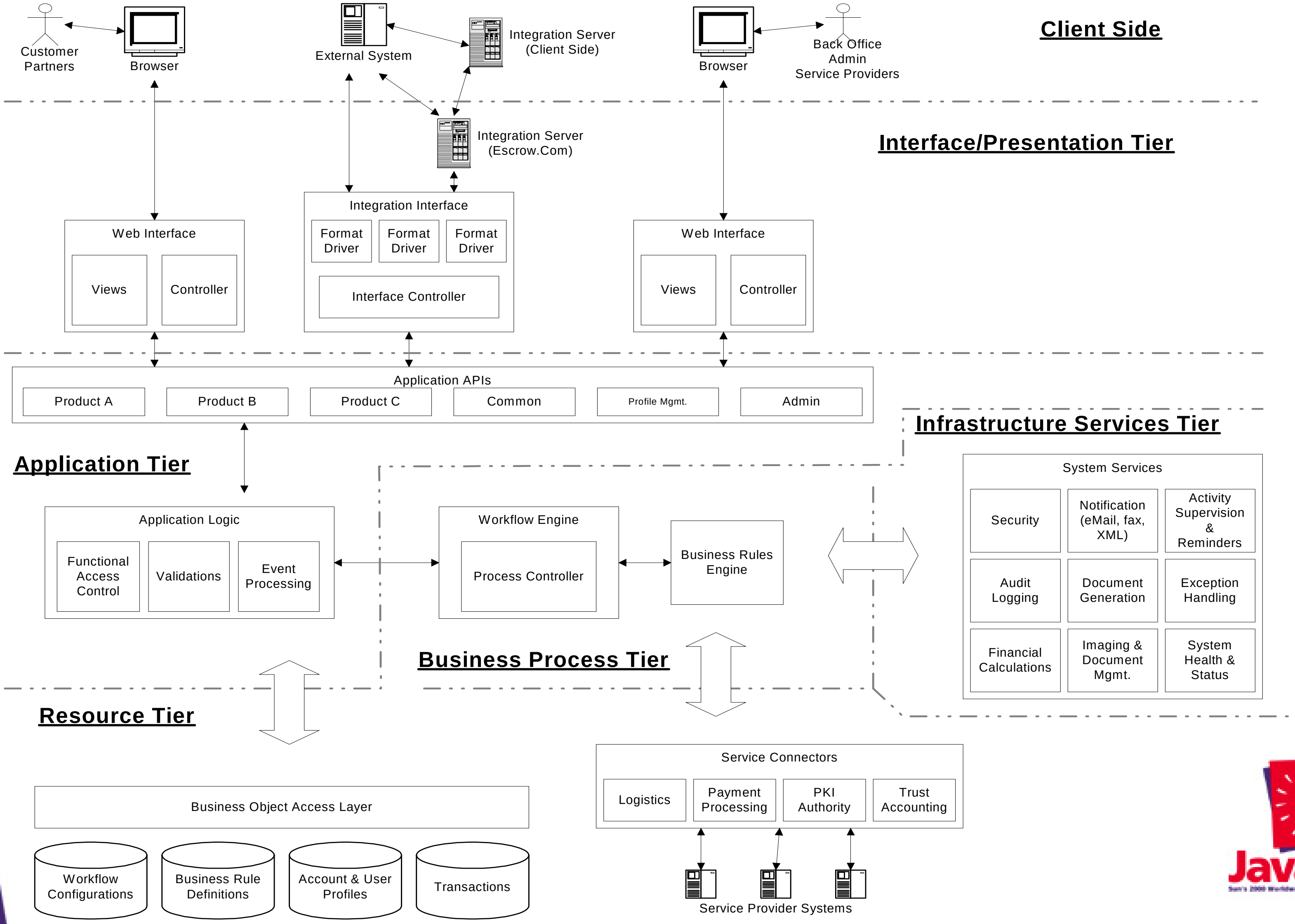
Challenges Faced

- Chaotically changing requirements
  - Immature, rapidly changing B2B marketplace
- Critical time-to-market
  - Broad competition in emerging market
  - Market share needed to execute business plan
- Quality and service availability
  - Financial services require strict controls
  - Critical component in client's business process
- Scalability and flexibility
  - Market projected to grow rapidly
  - Unpredictable client business requirements

Projects and Products

- “Version 1” - Initial “gold rush” system
  - Two-tiered architecture
  - ASP/VB Script, MS SQL Server
- “Version 2” - The “ultimate” platform
  - Multi-tiered architecture
  - ASP front end, Java/EJB middle tier, Oracle
  - XML for external interfaces and “glue”
- “Version 3” - Business value focus
  - Multi-tiered architecture
  - Full J2EE technology base, Oracle

System Architecture



Client Side

Customer Partners

Browser

External System

Integration Server (Client Side)

Browser

Back Office Admin Service Providers

Interface/Presentation Tier

Integration Server (Escrow.Com)

Web Interface

Views

Controller

Integration Interface

Format Driver

Format Driver

Format Driver

Interface Controller

Web Interface

Views

Controller

Application APIs

Product A

Product B

Product C

Common

Profile Mgmt.

Admin

Infrastructure Services Tier

Application Tier

Application Logic

Functional Access Control

Validations

Event Processing

Workflow Engine

Process Controller

Business Rules Engine

System Services

Security

Notification (eMail, fax, XML)

Activity Supervision & Reminders

Audit Logging

Document Generation

Exception Handling

Financial Calculations

Imaging & Document Mgmt.

System Health & Status

Business Process Tier

Resource Tier

Service Connectors

Logistics

Payment Processing

PKI Authority

Trust Accounting

Business Object Access Layer

Workflow Configurations

Business Rule Definitions

Account & User Profiles

Transactions

Service Provider Systems

JavaOne

Architecture and Design

- Broad J2EE blueprints provide a framework for incremental designs
- J2EE architecture inherently supports scalability, reliability, availability
- Important to properly distribute business logic to facilitate maintenance
- Overall architecture has good subsystem boundaries, very flexible
- Design patterns very effective
  - Notification services
  - Model-View-Controller
  - Factories, other GOF patterns

Architecture and Design
(the dark side)

- Problems arose mostly from failure to communicate, utilize and adhere to the architecture
- Bad distribution of business logic hurt maintainability and extensibility
- Short-changing design time created implementation problems
- Failure to manage dependencies resulted in brittle code
- Entropy needs to be attacked early to limit redesign and/or rewrite costs
- No substitute for design expertise

Java Technologies

- Java APIs offer a wide range of functionality
- Enables rapid development – less code
- Well known and documented
- Good support for object-oriented designs
  - Interfaces enable decoupled components
  - Reflection enables flexible factories
- Homogenous environment
- Easy to learn (for old C++ hacks)
- Popular and cool

Java Technologies (the dark side)

- Language missing some features (generics, enumerations, overloaded operators)
- Lots to learn
- Technology still developing

J2EE Technologies

- EJBs provide convenient transactional support
- Session beans work great for distributed components
- JSP/Servlets enable a homogeneous environment front-to-back
- Challenges mapping our access control needs to security model (JAAS?)
- New and cool

J2EE Technologies (the dark side)

- Entity beans
  - EJB 1.x provides weak O-R mapping & finder support
  - 1:1 beans brittle, subject to database changes
  - Added a lot of extra code with minimal pay back
  - Lifecycle model can result in performance hits
  - Debugging can be difficult

J2EE Technologies (the dark side)

- HTML/JSP Java conflicts (tag libraries?)
- JSP errors found later in deployment
- Issues with cached JSP pages
- JDBC temptations to write lots of in-line code

XML/XSLT Technologies

- Used for:
  - External integration interfaces
  - Dynamic document generation
  - Glue between ASP and Java
  - XML Testing tool
- Well known format for data exchange
  - Many standards emerging (ebXML, cXML, BizTalk, RosettaNet)
- Text-based, easy to learn and manipulate
- Good integration with Java, tools available
- Self-describing and validating (DTDs, schemas)
- XSLT provides powerful way to transform XML to a variety of formats with minimal coding
- New and cool

XML/XSLT Technologies
(the dark side)

- Emerging technology, competing standards, adoption is slow
- Standards are evolving, incompatibilities
- Hierarchical data model requires mapping to object model (e.g., Breeze, JAXB)
- Parsing and element access can be tedious and error prone
- XSLT is complex and cryptic
- Hard to find XML/XSLT gurus

Tools

- BEA WebLogic Server
  - Solid server with good standards support
- Java IDEs (Jbuilder, Visual Café)
  - IDEs primary value to have a consistent development environment
  - Code generation not widely used to maintain control of code
- JIndent
  - Enforces code formatting standards
- Jikes
  - Much faster compiles, but some deviations from javac

Tools

- Apache Ant (future CruiseControl)
  - Initial long build cycles lead to this tool
  - Allows flexible builds and tasks
  - Integrates testing with build (e.g., JUnit tasks)
  - Ability to check out latest source code automatically
  - Produces consistent build packages for deployment
- JUnit, HttpUnit (future Cactus)
  - Vital for thorough unit testing and test first designs
  - HTTPUnit enables testing of web conversations
- StarTeam Version Control
  - Continuous Integration Possible with XP process
  - Allow Integration in a distributed way

Development Process

- Defaulted to typical dot-com development
- Initial process was mostly ad hoc
- Attempts made to introduce elements of more formal methods (e.g, RUP)
  - Value of process not appreciated by management or developers
  - Process overhead considered too high

But that didn’t work…

- **Several weeks per iteration for bug fixes**
- **Internal “customers” did not know product**
- **Specialists in disciplines and features**
- **Integration difficulties, manual and inconsistent testing**
- **Decreasing ability to respond to changing requirements**
- **Time between releases grew**
- **Development effort became a “black hole”**

We Needed a New Process

- Needed customer focus to deliver most important features quickly
- Had to work in a changing business environment
- Searched for a lightweight, but rigorous process
- Process visibility needed across the organization

We Took On XP (***not*** Windows XP)

- Up Front Training
  - Initial JavaCon 2000 sessions
  - XP Immersion for seed team
  - On-site training and mentoring
  - Self-directed research by developers
- Big bang cut-over
  - Tried piecemeal adoption but not effective
- XP practices strictly followed
  - Sink or swim approach

Growing Pains

- **Resistance to change**
- **Environmental issues – common war room and shared cubes**
- **Legacy code issues with testing and refactoring**
- **Large team size created communication and overhead difficulties**
- **Personality conflicts**
- **YAGNI vs. BDUF**
- **Test first, micro-incremental development is hard**

Everyday Practices

- All production code written in pairs
  - Even when interviewing
- Test first design
  - Test – run – code – run – refactor – run – repeat
  - JUnit & HttpUnit
- Continuous Integration
- Developers
  - Make their own estimate and set velocity on yesterday’s weather
  - Pick their own tasks
  - Pick their pair partner
- Work on related tasks to complete stories
- Iteration ends on end date; customer determines if task / story continues in next iteration

Tangible XP Benefits

- Able to sustain a consistent, rapid development pace
- Organization always knows the complete current project status
- Always have a correctly working, “shippable” product at the end of every day
- Reduced QA cycles through intensive testing
  - Pre-XP, 2 month development produced 3 week test and fix cycle with 200 reported bugs
  - Post-XP, 2 month development produced 11 reported bugs

Tangible XP Benefits

- System is built to address immediate business requirements
- Short, incremental iterations deliver small working systems faster
- Clean, simple, documented code base enables quick understanding and easier changes
- Collective development provides easier cross training, mentoring, reduces truck factor, continuous product reviews
- Process enables small teams to be hyper-productive

XP Lessons

- XP pushes business requirement orientation throughout organization
- Accountability for business and technical decisions placed on appropriate parties
- Tough adjustment for some developers used to more traditional development
  - Developers give up private workspaces
  - Need to become comfortable with exposing mistakes
- Emergent design is a viable alternative to big design up front
  - Less elegant initial solutions evolve, providing collective understanding of designs
  - Overall blueprint (metaphor) guides evolution

Human Factors

- Job market in 1999/2000 very tight
- Qualified developers scarce
  - Most have limited dot-com experience
  - Object-oriented, J2EE expertise and experience limited
  - Intelligence, motivation, curiosity vital
- Augmented in house talent with targeted consulting expertise
  - Sun Java Center – Architecture, Java, J2EE, Process
  - Object Mentor – XP Process
  - Ultimate Knowledge – Java, QA
  - eBuilt – Java
  - iRise – WebLogic
  - Aquent – UI Design, Web Development

Human Factors

- Team culture and dynamics important
  - Group values (integrity, continuous improvement, intensity)
  - Personalities and interactions
- Team building needs to be carefully considered
  - Senior staff essential for core designs and mentoring
- Environment should foster encouragement and intelligent risk taking

If We Had to Do It Over Again

- Provide more up front training
  - Object-oriented design principles
  - J2EE architecture & technologies
- Dedicated time for continuous improvement
  - Allocate specific time for project reviews
  - Overcome inertia on tools, environment issues

If We Had to Do It Over Again

- Strong Focus on people & process
  - Agile, rapid development process (e.g., XP)
  - Careful team building, courage to restructure
- Don’t overbuild initial systems
  - Focus on simple set of initial customer requirements
  - Build into J2EE framework, but start simple

JavaOne℠

Sun's 2000 Worldwide Java Developer Conference™

Q&A

More Info

- Javaone@extremejava.com