J2EE: A Standard In Jeopardy

Richard Monson-Haefel
Senior Analyst

rmonson@burtongroup.com
www.burtongroup.com

Thursday – October 21, 2004
Friday – October 22, 2004

J2EE: A Standard in Jeopardy

Thesis

- The primary value proposition of J2EE is that it defines a common, standard programming model for enterprise development
- Five technical and market factors threaten the future of J2EE as a standard.
  - It's the fact that there are so many threats to J2EE that is concerning
- Java 2, Standard Edition is *not* threatened
- Businesses should choose a leading J2EE vendors if J2EE is central to IT.
  - Otherwise, consider open source J2EE, alternative frameworks and Microsoft .NET
- Vendors should not let J2EE die.
  - Preserving the J2EE standard is critically important to competing with alternative frameworks and Microsoft .NET.

J2EE: A Standard in Jeopardy

Agenda

- The J2EE common programming model
- Threats to J2EE
- Recommendations
- Conclusion

Agenda

- *The J2EE common programming model*
- Threats to J2EE
- Recommendations
- Conclusion

The J2EE Common Programming Model

Before J2EE

- Microsoft COM/MTS/COM+
  - Serious vendor lock-in
  - Platform limited to Windows
- Java-CORBA binding and proprietary models
  - CORBA was less-than portable or interoperable
  - Vendors offered proprietary models w/CORBA services
  - Businesses were locked in to proprietary models
- Vendors implemented a mix Java enterprise APIs
  - JDBC, Servlet API, Enterprise JavaBeans developed independently
  - No standard offering
  - No version consistency
  - No common programming model

The J2EE Common Programming Model

J2EE is introduced

- J2EE 1.0 released in December 1999
- Sun, IBM, BEA, and others created J2EE
- J2EE is an Uber-specification
- J2EE defines a standard, common programming model for enterprise computing
- J2EE was quickly adopted by vendors

The J2EE Common Programming Model

Benefits of J2EE common programming model

- Businesses have less risk of vendor lock in
- Businesses have application portability across products
- Developer's J2EE skills valuable across projects, vendors, employers
- Businesses have an expanding the pool of skilled developers
- Leveled playing field for smaller J2EE vendors. Brand becomes less important
- Provided large vendors a solid marketing strategy (standardization) for combating Microsoft COM/MTS/COM+
- Expanded the third-party market to many vendors

Agenda

- The J2EE common programming model
- *Threats to J2EE*
- Recommendations
- Conclusion

Threats to J2EE

The five threats

- Commodization
- Disruptive Technologies
- J2EE 5.0
- Microsoft .NET
- Model Driven Development

Threats to J2EE

The threat of commodization

- Commoditization characterizes a market that is:
  - Undifferentiated
  - Saturated with cheap or free offerings
  - High margins not possible
- The J2EE Standard, common programming model inhibits differentiation
- Open source J2EE projects (JBoss, Apache, ObjectWeb) are saturating the market with free offerings
- Commercial vendors find it increasing difficult to compete with free open source options, so that normally high margins are threatened

Threats to J2EE

The threat of commodization (cont.)

- JBoss, Apache Geronimo, ObjectWeb will be fully J2EE 1.4 compliant
- Performance of open source on J2EE level is consistent with commercial products
- Open source supported by big and small companies
  - RedHat
  - Novell
  - JBoss Group

Threats to J2EE

The threat of commodization (cont.)

- The specification is threatened by open source, not the leading J2EE vendors
- Smaller vendors will probably suffer if J2EE fails to thrive as a standard
- The Java 2, Standard Edition platform is not threatened.\
  - Java programming language
  - Java virtual machine
  - J2SDK APIs

Threats to J2EE

The threat of commodization (cont.)

- Vendors focus moving away from J2EE
  - Vendors are focusing resources higher up the stack (e.g. SOA, ESB, etc.)
  - J2EE compliance is being minimized in favor of holistic platform marketing. Notice the very late adoption of J2EE 1.4 by BEA
- Vendors may abandon or slowdown innovation of J2EE
  - Less licensing money for Sun means less resources for spec
  - A specification mitigated by lack of advancement

Threats to J2EE

The threat of disruptive technologies

- A disruptive technology is:
  - Less features and lower performance: Narrowly defined
  - "Good Enough": Do the job that most businesses need without overreaching
  - J2EE has overshot the low-end of the market
  - Eventually disrupters service high-end and displaces incumbents
- The J2EE standard and vendors are classic incumbents in a disruptive market
  - J2EE specification is far too complex - overshooting low-end of market
  - J2EE products are expensive
- Disruption has not happened, but the potential is very good
  - A single disrupter is unlikely – several disrupters are more likely

Threats to J2EE

The threat of disruptive technologies (cont.)

- Alternative Java open source frameworks
  - are focused, simpler to use, and free
  - Hibernate
  - Spring
  - Struts
  - Tomcat
  - Others
- LAMP
  - Linux, Apache, MySQL, PHP/Python/Perl (LAMP)
  - Less feature rich, but improving. “Good enough” and installed base is growing

Threats to J2EE

The threat of J2EE 5.0

- Current direction looks good for simplification
  - Plain old Java objects (POJO): No required interfaces
  - Dependency injection (DI): No required context or JNDI
  - Annotations (JSR-175 & JSR-250): No XML deployment descriptors
- Current direction also has huge potential for backfire
  - Use of POJO and DI lessens dependence on standard programming model and opens the door to any framework – hard to create IP around POJO/DI technologies
  - Over emphasis on annotations simply shifts complexity, it doesn't reduce it. Vendor specific annotations should be prohibited
  - Yet-another-persistence model: The Hibernate-influenced model. Its nice, but do developers want to learn a new persistence. Backlash is possible

Threats to J2EE

The threat of J2EE 5.0 (cont.)

- Catch 22
- Radically simplify J2EE or loose to alternative open source frameworks
- Radical simplification will mitigate J2EE's primary value proposition, a common, standard programming model

Threats to J2EE

The threat of Microsoft .NET

- .NET is “just as good”
  - From an enterprise architecture perspective, J2EE and .NET are comparable
  - Microsoft is no longer perceived as offering an inferior platform for enterprise computing, which gives it a boost when competing for business and developer mindshare
- .NET and VisualStudio is simpler
  - Right or wrong, the general perception is that .NET and VisualStudio provide a simpler, more productive environment for developing applications

Threats to J2EE

The threat of Microsoft .NET (cont.)

- Microsoft is poised to take advantage of a weakened J2EE standard
- Microsoft can leverage near-monopoly positioning (ubiquitous Windows desktops and servers) and deep pockets (60 billion war chest)
- .NET threatens, but it will not consume the entire market

Threats to J2EE

The threat of Model Driven Development

- Model Driven Development (MDD)
  - Uses graphical tools
  - Reduces direct code development
  - Reduces dependency on APIs
- MDD is both good and bad for Businesses
  - Could make developers more productive
  - Increases dependence on vendor tools

Threats to J2EE

The threat of Model Driven Development (cont.)

- Leading vendors beginning to emphasize graphical RAD tools over code-centric development tools
  - BEA's WebLogic Workshop is more RAD than code-centric
  - IBM's Rational Application Developer is increasing its RAD and modeling emphasis
  - Oracle Developer Suite includes a lot of RAD
- MDD emphasizes modeling and auto-code generation
  - Greater abstraction; more separation between developers and APIs
  - Less need for API, shifts desired skill sets to tool-specific and modeling
  - As development becomes more "divorced" from J2EE APIs, the J2EE standard becomes less significant

Agenda

- The J2EE common programming model
- Threats to J2EE
- *Recommendations*
- Conclusion

Recommendations

Recommendations for businesses

- Know how important J2EE is to your IT infrastructure
- If J2EE is not critical to IT
  - Consider open source J2EE projects
  - Consider potential disruptive technologies
    - Spring, Hibernate, LAMP, etc.
  - Consider Microsoft .NET
- If J2EE is critical to IT
  - Use commercial market leaders: IBM, Oracle, and BEA
  - Market leaders will provide best transition to non-J2EE world
  - Market leaders increasing emphasis on RAD/MDD helps insulate development teams through abstraction

Recommendations for J2EE vendors

- Don't allow the J2EE standard to die or whither away
  - J2EE provides vendors with their most significant competitive edge over disruptive technologies and Microsoft .NET
- Remain united and supportive of J2EE and the continued evolution of the standard
- Focus on the life-cycle and fixed set of annotations for standardization
- Protect the intellectual property rights associated with standardized annotations and runtime-behavior in the absence of APIs

Recommendations for J2EE vendors

- Ensure that J2EE 5.0 is a truly simpler platform and more productive
- Simplify programming model even if it opens the doors to alternative Java framework
  - Needed to compete with .NET
- Avoid shifting complexity: Employ annotations frugally
  - Don't allow vendor specific annotations
- Consider a J2EE-light or J2EE a la carte.
  - Still requires support and implementation of specific APIs, but businesses can buy only what they need (e.g. J2EE w/o EJB container or JMS provider)[1]

1. This recommendation was added recently. It is not in the published report.

Recommendations

Recommendations for J2EE vendors (cont.)

- Commodization of the J2EE stack is inevitable
- Stop sinking resources into J2EE stack
  - Invest higher up the stack
- Invest in the open source J2EE projects
- Support the open source project that best aligned with your organization with personnel, facilities, or cash
  - JBoss (LGPL license)
  - Apache Geronimo (BSD-style license)
  - ObjectWeb (mix of LGPL and BSD-style licenses)

Agenda

- The J2EE common programming model
- Threats to J2EE
- Recommendations
- *Conclusion*

J2EE: A Standard in Jeopardy

Conclusion

- The J2EE standard faces many threats
  - Commodization from open source J2EE projects
  - Alternative frameworks as disruptive technologies
  - The definition of the next specification, J2EE 5.0
  - Microsoft .NET
  - Model Driven Development
- The threats are not coordinated, but they are compounding
  - Result of natural market forces and technical advances, not orchestrated
  - Individual threats are addressable, but the combined pressures of all of them requires a change in vendor strategy and re-invigorated vendor unity

J2EE: A Standard in Jeopardy

Conclusion (cont.)

- Businesses should reevaluate the importance of J2EE in their infrastructure
  - Consider alternatives (LAMP, .NET, alternative Java frameworks)
  - Consider aligning with a major J2EE vendor
- Vendors should preserve J2EE as a standard
  - Continue to rally around standard
  - Focus on simplification of J2EE 5.0

References

- Burton Group's Application Platform Strategies
  - The Advent of the Network Platform: Web Services Move into the IT Fabric
  - Open Source Software: Risks and Rewards
  - Microsoft's .NET Platform: Bringing Web Services into the Core of Windows
  - Model-Driven Development: Rethinking the Development Process
- Other sources
  - The Innovator's Dilemma and other books on innovation theory by Clayton M. Christensen