Kodo JDO

An ROI Analysis Framework

SolarMetric

January 2003

Persistence Frameworks – JDO Simplifies the “Buy versus Build” Decision

Introduction

Organizations that leverage object oriented languages like Java often face the object-relational impedance problems inherent of working in an elegant object-oriented manner while having to store persistent information in the rows and columns structure of a relational database. To solve this issue, developers in such organizations face a decision between internally developing the persistence framework or licensing persistence middleware software from a third-party vendor. This “buy vs. build” decision typically incorporates factors that are easy to quantify like total cost of ownership (TCO), initial start-up effort, opportunity costs, and protection from vendor lock-in, as well as factors that are impossible to quantify like strategic direction. This decision becomes even more challenging because either answer can achieve the desired goal of persisting information into a relational database.

In the past, the available “buy” options included proprietary object relational mapping software and solutions that employed entity beans. Proprietary object-relational mapping solutions often came with problems of vendor lock-in and poor support whereas entity beans were complex to code and created performance bottlenecks. Both solutions were expensive and risky to implement making the “buy vs. build” decision difficult.

The Java Data Objects (JDO) standard dramatically changes the dynamics of the “buy vs. build” decision for persistence by providing a clear and simple winner to this problem. Traditional compelling reasons to buy transparent persistent middleware from a JDO vendor include:

- Faster time-to-market.
- Focus limited resources on core competencies, not infrastructure coding, especially in current environments where availability of corporate resources are limited.
- Ongoing cost savings in maintenance (traditionally, companies underestimate maintenance of internally developed solutions).
- Higher reliability through rigorous commercial testing necessary to sell into thousands of organizations.
- Greater level of integration with third-party technologies.
- Faster innovation with new features required by not just one client but many clients.
- Access to third party consultants and professional training.

Mature vendors supporting the JDO standard, like SolarMetric, offer additional benefits:

- Portability across data stores, not just the ones currently being used by the stakeholders of the “build” project.

- No vendor lock-in, common among proprietary solutions
- Simple API designed for ease-of-use and power with the JDO developer in mind, and not for the engineer of the implementation. This is in contrast to proprietary solutions and internal projects, which are often designed to get the persistence framework out the door rather than to be as simple and intuitive as possible for the end user.

Unless you are gaining a strategic competitive advantage by building your own solution (which is difficult to achieve in low level technology like data access middleware) or you can not fulfill your business requirements with existing tools like Kodo JDO (which is unlikely due to the extensibility of Kodo JDO), experts indicate that you should not endeavor to build your own persistence product.

This white paper explores the key financial reasons impacting the buy vs. build decision. We examine both building from scratch and building upon existing open source solutions. By focusing on shareholder value through the calculation of Return on Investment (ROI), this white paper attempts to deliver a quantitative framework to assist with the buy vs. build decision. By examining total cost of ownership (TCO) and risk, the white paper comes to the conclusion that most organizations cannot justify the cost of maintaining an internally developed persistence solution, let alone actually developing such a solution when mature JDO alternatives are available. Kodo JDO offers a lower TCO, greater functional capabilities, reduced risk and therefore higher ROI.

Return on Investment (ROI) – The Ultimate Financial Measurement

There are a number of ways to perform an ROI analysis, from Net Present Value calculations to payback. In this white paper, we only examine the factors that go into calculating an ROI. Regardless of how your organization prefers to handle its ROI analysis, the factors used to determine ROI of a buy vs. build persistence decision are the same:

- Risk
- Application Functionality
- Total Cost of Ownership (TCO)

Risk

An over-arching factor in determining ROI is Risk. Internal persistence projects are fraught with risks of both a business and technical nature that can result in cost overruns, project delays, and unanticipated development and maintenance, all negatively affecting ROI.

Inflexible Architecture and Limited Portability

Although the promise of internally developed persistence solutions is a solution that is specifically designed and developed for the particular technology problem, this is rarely the case. Often the design of internally developed solutions only takes into account the current problems that are faced by the organization. Even if the persistence project is designed appropriately, aggressive deadlines cause the development team to typically focus on solving current problems first.

For example, a persistence solution developed for Oracle 8i may need large amounts of maintenance work when the organization upgrades to Oracle 9i, and even more work when the organization moves to Sybase as its primary database. Imagine the integration challenges after acquiring a company that doesn't use Oracle but rather IBM DB2 and SQL Server 2000. An internally developed solution locks the organization into the design and development constraints that are usually associated with current business needs, not a holistic view of current and future requirements.

To meet the needs of a wide customer base, Kodo JDO must support as wide a range of data stores as possible (See the Supported Databases and Application Servers Section). In addition, by providing source code for its database dictionaries, Kodo JDO can easily be extended to access the most esoteric relational databases. Kodo JDO can even use legacy data stores in conjunction with relational databases.

Mature JDO vendors like SolarMetric offer solutions that are highly extensible, making it easy to extend the persistence framework.

Poor Alignment with Future Needs

Although seemingly well suited to meet current needs, internally developed applications often fail to address future needs. Performance and scalability are often compromised because of uncertain future needs. Additionally, internally developed solutions often make significant constraints on standard object-oriented concepts such as polymorphism or encapsulation, placing undue restrictions on the design and architecture of object-oriented projects.

Open source projects are typically no more aligned to the business needs than a packaged solution.

Based on the Java Data Objects (JDO) standard, Kodo JDO benefits from a specification team represented by a diverse cross section of industry. This team makes enhanced design decisions compared to individual companies or development groups.

On top of the 3 years (and continuing) of effort into the specification, JDO solutions are designed and specified by professional product managers that gather technical and business requirements from a broad customer base. Organizations that develop solutions internally derive no benefit from this requirements gathering process or the past experiences and practices of other companies. Although internally designed applications are intended to be well-aligned to the organization's needs, such applications often fall short and do not account for future needs.

To remain competitive, SolarMetric must remain on the cutting edge of evolving technologies and business needs, and can do so in part through its large customer base and user community. As a consequence, SolarMetric is in a far better position to anticipate and, consequently, prepare for future market needs. SolarMetric's Kodo JDO is designed to scale and perform for the largest enterprise customers and offer standards-based, market-proven architectures and data models.

Unchallenged by Competitive and Commercial Demands

Internally developed solutions rarely have to face competition from external solutions. Safely nestled within an organization, political pressures and personal biases affect fact-based decisions. Often, organizations have to live with technology limitations for extended periods of time that would not be acceptable in the commercial marketplace. The result is often an internal vendor lock-in that can be more costly than external vendor lock-in.

Competition and commercial demands drive product improvements, performance enhancements and innovation. The market determines the winners and losers based on technical functionality. If a JDO vendor does not provide superior support, powerful functionality, and innovation, their product can be easily replaced by another JDO compliant vendor.

Unlike open source solutions, SolarMetric's Kodo JDO has to meet commercial documentation and support requirements. As a result, developers using Kodo JDO to handle their organization's persistence requirements aren't digging through source code but rather leveraging Kodo JDO's documentation and highly responsive support system to quickly get answers.

Risk of Developer Turnover

In general, internally developed solutions do not have the documentation that is common among packaged solutions. As a result, there is tremendous risk that an internally developed solution will become obsolete should the inevitable turnover occur among the developers of the solution. In such situations, proper documentation initially or ongoing maintenance are both extremely expensive.

Even internally developed solutions that are based on an open source product become "one offs" if their changes are not incorporated into the core build of the open source project. In time this internally developed software turns into a proprietary, unmaintained application. This can be avoided by ensuring that the open source project remains aligned with an organization's business needs, but doing so requires committing significant resources to the open source project.

As a vendor whose purpose is to maintain Kodo JDO, SolarMetric offers professional training courses, prompt technical support, and commercial grade documentation to ensure that transition among developers is easy. In addition, consulting organizations have familiarity with the JDO specification and the Kodo JDO implementation. Five books are scheduled for publication from 2002 through early 2003 on JDO. The resources of an accepted standard and an organization committed to its product greatly reduce the risks of turnover among the development staff.

Application Functionality

Another factor affecting ROI is application functionality. In the buy versus build decision, Kodo JDO must meet the functionality requirements of the developer. Kodo JDO benchmarks favorably against all competing technology on a functionality level.

Support for the Final Version of the Java Data Objects Standard

Kodo JDO is a robust implementation of the Java Data Objects specification that not only meets the requirements of the specification's Technology Compatibility Kit, but also supports almost all of the optional features in the JDO specification.

SolarMetric also helps shape the direction of future versions of the specification by participating on Sun Microsystems' Java Community Process JDO expert team. In addition, SolarMetric helps market the standard as a charter member of JDOcentral and at a grassroots level by presenting at Java User Groups around the world.

Over the first eighteen months that Kodo JDO has been available, it has been downloaded over 18,000 times and is used by customers throughout the world and in all industries. SolarMetric makes it a point to continue to innovate ensuring Kodo JDO is the best performing data access tool available.

Full Life Cycle of Object Persistence

With Kodo JDO, SolarMetric offers the full life cycle of object relational mapping tools.

Kodo JDO's reverse engineering schema tool creates persistent class definitions, metadata, and mapping extensions from an existing schema. This provides value when in the common situation of working with existing database structures.

In addition, Kodo JDO offers its Schema Tool for taking a class and creating the corresponding database schema. Thus, with Kodo JDO, it is possible to round-trip schema modifications to and from the data store. Finally, Kodo JDO permits schema evolution over the life cycle of a project, meaning that you can add or remove fields in your business objects, and automatically propagate the necessary changes to your database without doing any export or import of data already in your database.

Performance Pack

In Kodo JDO, SolarMetric has focused on developing a high performance JDO implementation. The achievement of this goal can be seen in several ways.

Kodo JDO offers a high-performance cache that dramatically enhances performance over standalone JDO by 20 – 40 times. This cache can be used in a standalone configuration or across a distributed cluster of machines. The cache results in the greatest performance gains in applications that frequently access the same data – a common pattern in many server-based applications

When used with a JDBC 2.0 driver, Kodo JDO uses batched statements when possible. This can lead to quite significant performance enhancements when performing many repetitive operations, such as creating or deleting many new objects. A future version of Kodo JDO will include the capability to dynamically re-order SQL to maximize the amount of batching possible.

Extensive internal performance testing projects help us keep a close eye on the scalability of Kodo JDO, both in multi-threaded environments and with long-lived systems. Scalability is a critical issue when deploying an enterprise solution.

Developer Friendly

Kodo JDO is considered to be the most developer friendly JDO implementation available. This manifests itself in a number of ways but most predominantly:

- Kodo JDO is highly extensible. A number of Kodo JDO's subsystems are configurable and pluggable via publicly available and documented APIs. Examples include our performance cache component, database sequence generation, SQL generation control to support particular dialects of SQL, and subclassing of our PersistenceManager implementation. Other components can be easily customized upon request. Examples include support for non-relational databases, particular sets of stored procedures, and transparent recognition of custom implementations of the Collections interfaces.
- SolarMetric open-sourced its standards-based byte code enhancement tool (serp on Source Forge). This allows organizations evaluating Kodo JDO to see exactly how the byte code is being manipulated.
- Kodo JDO also provides source code for selected utilities, allowing users to quickly debug problems they might have and extend Kodo JDO more easily.
- SolarMetric offers pre-sales technical support for developers evaluating Kodo JDO. SolarMetric also requires that every member of its technical staff (from engineering management to summer interns) participate in some technical support activities on a weekly basis. For these reasons, SolarMetric has some of the best technical support in the software industry.
- SolarMetric believes very strongly in not creating *another* GUI tool that developers have to learn and deal with but rather make sure that Kodo works well

with the tools that developers currently use. Tight Apache Ant integration makes it possible to support any IDE that supports Ant. In addition, a seamless high-level integration has been completed with JBuilder with additional high-level integrations planned for other popular IDEs.

- JDOQL extensions are supported. This means that it is possible to execute custom SQL that may not have an analogy in JDOQL. Additionally, we provide default extensions to do things like searching for a substring anywhere inside a string (JDOQL only supports String.startsWith() and String.endsWith()) and case-insensitive querying.
- SolarMetric publishes its bug tracking database. Unlike traditional software companies, SolarMetric recognizes that bugs will crop up. As a result, SolarMetric feels it is important to provide developers with a resource to identify bugs, manage the bugs identifies, and help guide product rollouts by voting on enhancements.

Supported Databases and Application Servers

Kodo JDO supports a wide variety of databases (both commercially available and open source databases):

- Oracle
- IBM DB2
- Microsoft SQL Server 2000
- MySQL
- PostgreSQL
- Sybase Adaptive Server Enterprise
- Cloudscape
- Hypersonic Database Engine
- InstantDB
- Pointbase

SolarMetric provides source code for its DB dictionaries, thereby making it very easy for our customers to support any special databases that they may use. For example, customers are using Kodo JDO with Oracle Lite, Microsoft Access, and SAPDB.

In addition Kodo JDO supports the major application servers:

- BEA WebLogic Server
- IBM WebSphere Application Server
- JBoss

Kodo JDO can be integrated with any application server that supports the Java Connector Architecture.

Total Cost of Ownership (TCO)

When looking at TCO, one must look at the initial project development costs and ongoing maintenance costs as well as opportunity costs of skilled developers focusing on what is likely not a core aspect of the organization's business.

Initial Project Development

Kodo provides a 20-40 % time savings in the total coding effort of an enterprise application. Additionally, Kodo reduces the size of the code base, improving code quality. Planning, developing, and implementing an internally developed proprietary system is a limited, more expensive option. Again, the smaller group or individual does not benefit from the resources of the standards body specification team or our user community.

Ongoing Maintenance and Future Planning

Often, internally developed products focus on immediate problems and do not take into consideration the future needs of the organization. Six months or 3 years from now when upgrades or enhancements to the internally developed product are necessary, the software often needs to be completely redesigned, and the original development staff will likely no longer be available. The initial effort to implement the old system is often repeated.

Again, aside from specification improvements made by the JDO standards body, a maintained product like Kodo takes into consideration the feedback of tens of thousands of users in our community. The feedback is invaluable when planning new enhancements such as performance capabilities, improved customizability, and documentation and error message improvements.

Opportunity Cost

Typically, organizations leverage their most skilled and experienced developers to internally create a persistence framework. The complexity of writing such a framework and the high importance of getting it right requires that the developers be skilled to ensure that future projects based on the persistence mechanism work as prescribed. Unless extremely well documented, the ongoing maintenance will typically be done by the same skilled developers who wrote the persistence framework.

The cost of having the high-powered developers focusing on low-level persistence costs the organization in lost opportunities to develop true business applications. Does the benefit justify the cost especially when mature JDO implementations are available today? Unlikely.

Advanced Features

Internally developed persistence solutions often stagnate once they become ‘good enough’. That is, once a solution can more-or-less persist data into a database, the developers involved in the project often get moved on to other projects. SolarMetric, on the other hand, has a full-time research and development staff working on designing and implementing innovative new features.

Recent research projects that have made an appearance in production versions of Kodo JDO include our distributed caching functionality, the capability to automatically generate Java files and JDO metadata from an existing schema, flexible inheritance mapping capabilities, and SQL statement batching.

As a result of our performance-focused research efforts, systems using Kodo JDO often operate considerably faster than comparable systems using hand-coded SQL. We have accomplished this by developing algorithms that optimize common database hot spots to transparently accelerate data access and manipulation.

It is important to consider these advanced features when calculating the final TCO. Sticking with an existing product that is in maintenance mode will make it impossible to realize the benefits that an innovative solution can offer, both today and in the future.

Cost

The cost of simply maintaining an internally developed persistence framework is, very conservatively, 2 days per month or approximately 5 weeks per year for a persistence team. An average salary for an inexperienced developer fully loaded is approximately $100,000 per year. Simply maintaining an internally developed persistence framework costs approximately (for a team of 5 developers) $50,000 per year. Customers using Kodo JDO as their persistence framework indicate that maintenance is driven to almost zero even when porting from one backend database to another. Kodo licensing costs are a fractional portion of the long-term internal resource requirement for maintenance.

Conclusion

ROI analysis can play a critical role in helping to see the benefits of SolarMetric’s Kodo JDO, especially when compared to internally developing from scratch or from an open source starting point. This is the case whether using EJBs Bean Managed Persistence / Container Managed Persistence, Session Beans, JSPs, Java Objects or other JDBC / SQL options.

SolarMetric’s Kodo JDO provides valuable functionality while reducing the risks of internal development, at a price point that is far less than an internally developed solution. Battle-tested by supporting the JDO standard and proven commercially by tens of thousands of members of our user and customer community, Kodo JDO is a robust and mature implementation of the JDO standard and a valuable product for persisting data to data stores. Kodo is based on over 50 man-years of development effort and challenged everyday by competitors and customers alike.

The persistence buy vs. build decision has been transformed by the Java Data Objects specification. Organizations who are currently maintaining internally developed persistence frameworks should do an apples to apples comparison of functionality, performance and costs. Organizations who openly consider Kodo JDO will be hard pressed to justify continuing to maintain their own persistence solutions. Organizations considering internal development efforts clearly need to evaluate Kodo JDO.

Given the risks of internally developed persistence frameworks, the high costs of maintenance and internal development, and reduced robustness and functionality available when building internally, Kodo JDO provides a viable and high-ROI alternative. In recent customer ROI studies, Kodo JDO provided first year returns of 300% - 600% in enterprise application development situations.

To get started, you can download a complete and free evaluation of Kodo JDO at http://www.solarmetric.com/Software/Evaluate/.

Interested in learning more about Kodo JDO? Contact SolarMetric's Sales Department at 202-595-2064 x2 or sales@solarmetric.com.