

Interim Platform Recommendations for New Software Services Development in IST-AS

IST - AS Architecture Group (ASAG)

v 1.0.0

I. Introduction, Context, and Scope

Scope: Recommendations made in this document apply to new user-facing web application/service development in IST-AS. This document is also intended as a signal to (and call for feedback from) the campus developer community regarding the direction IST-AS intends to take in the area of web application/service provision.

The paramount consideration is achievement of key, interdependent IT objectives -- Quality, Speed, and Flexibility (QSF) -- as articulated by CIO Shel Waggener for IST and the UC Berkeley campus. These objectives will guide all architectural decisions that impact our organization's application service offerings.

This is an interim recommendation, intended for long-term adoption after (approximately) six months of early-adopter use and feedback. These recommendations are intended to dovetail with formal tool, methodology, and deployment platform recommendations that are currently in-progress. *Developer toolkit recommendations will follow in short order. Prototype implementations on the recommended platforms will be developed once critical (but not complete or formal) buy-in on the content of this interim document is achieved.* In the long-term, tools, prototypes, components, documentation, training, code review, and similar offerings that may be applied to new web application development, as well to replatforming or refactoring of existing projects, will be made available to both IST and the UC Berkeley Campus.

Open, vigorous deliberation, animated by careful yet spirited challenges to our own assumptions, has been the modus operandi of the AS Architecture Group in developing this interim recommendation. Deliberation and consequent evolution continues as drafts are presented for feedback to colleagues outside IST-ASAG. Feedback may be offered directly by e-mailing ist-as-architecture@lists.berkeley.edu.

II. Broad Considerations

The fact of ubiquitous interdependence underlies the analysis undertaken to produce these recommendations: IT developers require support in the form of deployment and database services; development platform and tool provision; training and developer communities; and a well-managed set of skills to keep current and honed. Decisions that place burdensome demands on any of these critical aspects of software development threaten an organization's ability to maintain excellent levels of QSF.

Fundamental concerns in making these recommendations therefore involve impediments to QSF that occur in IT environments that are more heterogeneous than necessary, including, in general, UC Berkeley's, and in particular the IST-AS Web Application unit. These impediments include:

1. the small number of frameworks in which a developer can maintain currency & expertise, and, consequently:
 - constrained staff resources in each of N platform/framework silos, tightening as N increases
 - barriers to deployment of staff to "hotspots" as business requirements demand
2. complexity in infrastructure support (deployment platforms, development platforms, toolkits) -- cf. #1
3. licensing fees for overlapping coverage of technology and business needs

Reduction of the number of software development platforms and frameworks in use in IST-AS, in IST, and on the UC Berkeley campus is therefore seen as a desirable goal, and facilitating achievement of this goal is a driving consideration in this analysis. Achievement of this goal would result in:

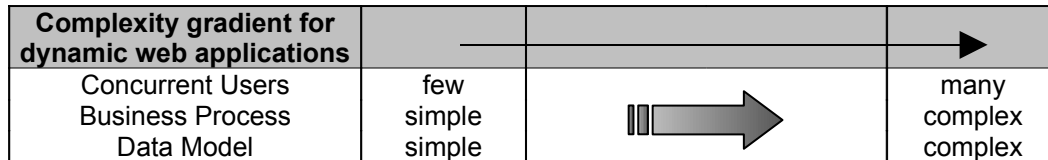
- A smaller number of more fully-staffed pools of developers, where
- the set of pools provides full or nearly-full coverage of IT needs; and where
- developers will be able to achieve a higher level of expertise in a narrower, market-aligned, broadly-applicable set of skills.

A consequence of this consideration was to privilege more general-purpose platforms and frameworks, as these would enable fewer, deeper skill-sets to be applied to the full UC Berkeley problem-space.

Implicit in these concerns is that development on any platform must occur in the context of selected, standards-based, proven frameworks in order to deliver Quality, Speed, and Flexibility in IT applications/services. Proliferation of home-grown solutions to problems already addressed by commonly-adopted frameworks would forfeit the organizational benefits that are intended to follow from implementation of this document's recommendations.

III. Software Application/Service Complexity Matrix

A simplified matrix describing levels of complexity in dynamic web applications typical to the UC Berkeley campus was used to frame this analysis. The levels of complexity range from small, departmentally scoped applications; to mission-critical applications with many users, complex business rules, and a complex data model. The many variants between these poles are represented in the diagram by a gradient arrow.



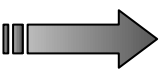
IV. Evaluation Criteria

Twelve evaluation criteria were developed by which to measure the suitability of candidate platforms.

Evaluation Criteria	Description
Vendor Lock-in	Avoid solutions offered by a single vendor. Vendors' long-range plans won't always align with UC Berkeley's; licensing costs can be/become burdensome; more frequent upgrades may be required, which are disruptive and costly; vendor-specific solutions and tools don't always play nicely with tools from other vendors or open source.
Development & Deployment Choices	More options are better; most IST developers use Windows, but faculty & student colleagues may prefer Mac/*nix. Option of deploying to Windows or multiple flavors of Unix removes the application (framework) as a deployment constraint. Toolset, desktop, and platform support considerations may nevertheless influence convergence toward a subset of available options.
Key Integration Points	Oracle ERP [Peoplesoft] (Fusion); Sakai & other Higher Ed open-source solutions; PPS
Enterprise Platform Integration	For scripting languages, the ability to execute in the Java Virtual Machine, for example, offers a higher-performance execution environment.
Standard & Available Libraries	Platforms should have solutions for standard tasks (SMTP, LDAP, XML, etc.), either as part of the standard distribution or available at little or no cost.
Existing Applications	Any applications implemented in languages/frameworks not on the recommended list will still need support as well as a migration/sunsetting plan.
Existing skills	If current staff are experienced and expert on a platform, no training is required.
Availability of skills	How easy is it to recruit for skills required to develop in a given language/framework?
Learning curve	Some IST and campus developers will likely need training in any recommended language/framework. Ease of learning and time to mastery are considerations.
Number/Preeminence of Frameworks	For a particular language, more (web) frameworks means more choice, but may also imply a more fragmented community.
Testing Support	Support for, and ideally tight integration with, test-driven development (TDD), automated functional testing, and automated load testing is critical.
Tools Integration	Integration between platform and IDE, source code repository, bug-tracking tools, etc. is critical.

V. Platforms Considered

The platforms listed in the following chart were considered for the two "poles" of complexity described in Section III, as well as for applications in the middle-range. These candidates were culled, on the basis of the Evaluation Criteria detailed in Section IV, from an informal survey of development platforms in IST-AS and on the campus, and strongly represent prevalent languages and frameworks in current use by web application developers at UC Berkeley and beyond.

Platforms Considered			
<i>Concurrent Users</i> <i>Business Process</i> <i>Data Model</i>	<i>few</i> <i>simple</i> <i>simple</i>		<i>many</i> <i>complex</i> <i>complex</i>
Candidate Platforms	Cold Fusion Perl PHP Python Ruby	Cold Fusion Java Platform .Net Perl PHP Python Ruby	Java Platform .Net

VI. Complex Applications/Services

The Java platform and .NET were both judged to be well-qualified candidates for implementation of complex, enterprise applications and services at UC Berkeley. They were seen as broadly similar in that:

- they are both compiled, high-performance languages
- the learning curves to achieve expertise are similar in each case
- each has a rich enterprise ecosystem (well-developed APIs and frameworks)
- each has rich, well-integrated source-control, unit-testing, and IDE choice(s)
- each is tightly integrated with target deployment servers
- each has a record of proven successful enterprise deployments

The Java platform was judged a better fit to UC Berkeley's environment, despite the advantages .NET currently enjoys in the maturity of its visual-development environment and its simpler deployment configuration, because its selection:

- avoids vendor lock-in
- enables a broader range of development and deployment environments
- is better aligned with most UC Berkeley key integration points (Oracle [Peoplesoft] Fusion, Community Source Higher Ed, PPS)
- is better aligned with key internet2.edu efforts, such as CAS, Shibboleth, and JA-SIG
- can be substantively simplified by standardization on selected frameworks that are well-supported by advanced IDE tools
- enables integration with Java-conversant campus resources in the student and faculty populations (EECS, SIMS), and among UCOP & UC campus IT staff (who create and/or support, e.g., CASA, DARS, Cost Sharing)

Of key importance is the assessment that it would be a significant, and possibly crippling strain on IST resources to support multiple platforms for complex enterprise application development (see Section II, above).

This document does not address selection of frameworks from among the very many available for web application development on the Java platform. Selection of recommended frameworks for Java-based web applications will be forthcoming, as will prototypes developed on those frameworks.

It is expected that Service Oriented Architecture (SOA) will enable heterogeneous software development groups to freely consume and provide services. Platforms that conform to UC Berkeley's security standards, and in which it is possible to implement standards-conformant message exchange -- including .NET -- will thus be able to interoperate over standards-based protocols (e.g., Web Services).

VII. Simple Applications/Services

Of the five languages and their associated frameworks considered at the "simple" end of the continuum, Cold Fusion was judged to be an outlier because its use tends to result in page-based (non-MVC) applications. While MVC frameworks are plentiful in PHP-space, its historical focus and dominant developer-focus is, similarly, page-based. Neither Cold Fusion nor PHP are as suitable for general purpose, object-oriented scripting.

Perl, Python, and Ruby were judged to be well-qualified candidates for implementation of simple and midrange applications and services at UC Berkeley. They were seen as broadly similar in that:

- all three are general-purpose, object-oriented scripting languages
- all three are dynamically typed
- all have active developer communities
- acceptable MVC web frameworks exist for all three
- all three are open source (no vendor lock-in)
- each has a rich pool of implementations / libraries
- all three run on multiple platforms
- tools-integration can leverage the Eclipse platform, widely used in Java platform development via plug-ins available for all three


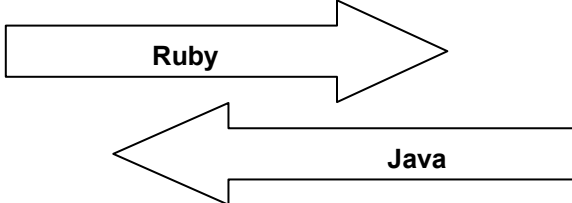
Ruby (and the Rails framework) was judged the best fit to UC Berkeley's environment for web application/service development. Though Perl and Python have multiple frameworks from which to choose, and Python offers more mature JVM integration than JRuby, the following reasons tilt the balance in Ruby's favor:

- its web framework (Rails) effectively shepherds developers toward separation of concerns (MVC)
- robust, automated test coverage is built into the Rails framework
- a highly active Rails developer community is continuously delivering value-adding implementations, allowing rapid provision of new features & widgets that the campus community is looking for
- the learning curve is less intimidating to most developers than those of Perl or Python
- its implementation is more purely and cleanly object-oriented
- it is more expressive (and therefore more readable)

It is expected that Service Oriented Architecture (SOA) will enable heterogeneous software development groups to freely consume and provide services, as noted with regard to implementation platforms for complex applications and services, above. Platforms that conform to UC Berkeley's security standards, and in which it is possible to implement standards-conformant message exchange -- including Perl and Python -- will thus be able to interoperate over standards-based protocols (e.g., Web Services).

VIII. Recommendations

Interim recommendations for new web application/service development in IST are:

Recommendations			▶
<i>Concurrent Users</i> <i>Business Process</i> <i>Data Model</i>	<i>few</i> <i>simple</i> <i>simple</i>		<i>many</i> <i>complex</i> <i>complex</i>
Recommended Platforms			

These selections are compelling in large part because each platform may be credibly applied to problems more complex (in the case of Ruby) or simpler (in the case of Java) than the far end of the spectrum that each single selection addresses.

Selection of these two platforms -- Java and Ruby -- facilitates achievement of Quality, Speed, and Flexibility in application / service provision:

- The full (or nearly-full) problem space that must be addressed by web application & service providers on the campus are addressable with only two application platforms, from which it follows that:
 - A campus IT staffperson can be deployed to address a broad range of business needs even if s/he is expert in "only" one of these two platforms.
 - It is possible to imagine that a significant proportion of campus IT staff could become conversant, or even expert, in both these platforms, providing even greater options to IT managers.
- These choices are well-suited to economies inherent in supporting narrower ranges of IT development infrastructure (deployment platforms, development platforms, toolkits).

This recommendation has been put forward by the IST-AS Architecture Group, <http://ist.berkeley.edu/as/ag> .
The current draft of this document is maintained at: <http://ist.berkeley.edu/as/ag/technology/webapp-platform-rec.html> .

Please offer comment, questions, and other feedback by sending e-mail to ist-as-architecture@lists.berkeley.edu .