arXiv Technical Evaluation Rubric

Part of planning for the next generation technical architecture for arXiv was evaluating technologies that could be adopted or adapted for arXiv. At this point, we looked at technologies that were developed with similar motivation to what arXiv-NG needs. The evaluation included open source frameworks, services that could fulfill part of arXiv's needs, and commercial solutions. We met with technical leads of each candidate technology and looked at code and documentation when it was in an open repository. The arXiv-NG development team created a "rubric" (see below) that covered the many aspects of what a technology would require to meet the needs of arXiv-NG going forward. In recognition that any one technology would not be able to solve all of arXiv's needs, we set a **priority** for each item on the rubric. At the end of the evaluation period, we narrowed our focus to two top candidates (i.e., open source approaches that were modular and evolvable), at which time we did testing and proof of concepts to essentially "kick the tires" for essentials for arXiv.

arXiv-NG Technical Evaluation Rubric v2

RATING KEY:

no	weak	neutral	good	excels
◉	◓	○	◒	●

PART 1: ENVIRONMENTAL EVALUATION		Candidate 1 [name]	Candidate 2 [name]
1. Fitness for Purpose - arXivNG	**Priority**	**Rating**	**Rating**
a. Avoids architectural lock-in (one platform; integrated web app)			
b. Maximizes potential for strategic evolutionary strategy			
c. Minimizes need to re-architect or develop code for arXiv			
d. Easily integrated with existing arXiv workflows and services			
e. Designed with assumption of distributed system			
f. Well-defined APIs for arXiv essential components			
g. APIs for arXiv submission, moderation, admin			
h. API for arXiv export features			
2. Application Architecture	**Priority**	**Rating**	**Rating**
a.Extensible and modular framework			
b. Containerization and nodes in cloud			
c. Amenable to horizontal scaling			
d. Amenable to vertical scaling			
3. Code Quality (for existing open source)	**Priority**	**Rating**	**Rating**
a. Well written code			
b. Well-defined APIs			
c. Open source license			
d. Preferred language (Python/Ruby/Java; JavaScript framework)			
e. Well supported (regular bug fixes; upgrades; documentation)			
f. Low barrier to contribution			
g. Best practices in coding: tests, version control, documentation, code guidelines, automated build, automated tests			
h. Dependency management			
4. Technical Infrastructure	**Priority**	**Rating**	**Rating**
a. Easy to setup a developer environment			
b. Deploy and run services in cloud (e.g., VM images + application components; Docker)			

c. Elastic provisioning			
d. Flexible logging			
e. System monitoring (cron; periodic monitoring scripts)			
f. Log monitoring and system alerts			
g. Security monitoring			
h. Service-level agreements			
i. Uniform code management			
j. Admin access to servers and VMs			

PART 2: FUNCTIONAL EVALUATION		**Candidate 1**	**Candidate 2**
1. Data model	**Priority**	**Rating**	**Rating**
a. Generic (core data model)			
b. Flexible and Extensible (core model can be extended)			
c. Model for key entity types (e.g., publication entities; scholarly objects; research objects)			
d. Supports relationships (e.g. part/whole; reference links; semantic			
e. Supports custom metadata			
f. Supports multiple identifiers (local and global)			
g. Supports aggregation of entities (e.g., via part/whole relationships)			
h. Amenable to serialization in multiple standard formats			
2. Metadata Store	**Priority**	**Rating**	**Rating**
a. Flexible database schema			
b. Ability to store metadata as objects (e.g., json)			
3. Object Store	**Priority**	**Rating**	**Rating**
a. API is well-defined and documented			
b. Flexible content type (text, binary, data, video, image, PDF, other)			
c. CRUD operations on objects upon objects in persistent storage			
d. Versioning of objects			
e. Backup of objects			
f. Redundancy/replication of objects			
g. Pluggable storage backend (e.g., file system; cloud storage; storage system)			
h. Audit trail of changes			
i. Create checksum of objects			
j. Monitor object integrity (e.g., via checksums)			
4. Workflows	**Priority**	**Rating**	**Rating**
a. Specification of workflows (outside of code)			
b. Modern, reactive user interfaces for agents of workflows			
• Submission			
• Moderation			
c. Execution and tracking of workflows			

• Submission			
• Moderation			
d. Integration with external services (e.g., call outs)			
• Classifier, overlap, TeX,			
• Reference link extraction, metadata extraction			
• iThenticate			
e. User communication and notification			
• email			
• Messaging			
• Event audit trails (also visible via user web pages)			
f. Logging and history (e.g., of submitters and submissions)			
g. RT integration or similar functionality			
5. Publishing	**Priority**	**Rating**	**Rating**
a. Persistent URLs for arXiv papers			
b. Pre-print DOIs			
c. Changes at specific times: freeze time/ publish time			
d. Move metadata and content to published storage			
e. Triggers for announcements			
• email announcements to submitters			
• email announcements of articles			
6. Discovery and Access	**Priority**	**Rating**	**Rating**
a. Indexing easily configured			
b. Robust search engine (e.g., Elastic; Solr)			
c. Faceted browse			
7. Services	**Priority**	**Rating**	**Rating**
a. TeX system - Ability to run as external service for processing of submission, moderation, publishing of articles.			
b. Overlap detection			
c. Classification			
d. Reference extraction			
8. User Interfaces	**Priority**	**Rating**	**Rating**
a. All Web UIs use modern javascript framework			
b. Out-of-box moderation UI			
c. Custom-built moderation workflow UI			
d. Out-of-box submission UI			
e. Custom-built submission workflow UI			

f. Out-of-box browsing UI and preprint page			
g. Custom-built browsing UI and preprint page			
h. Out-of-box administrative UI			
i. Custom-built administrative UI			
j. Faceted search UI			
9. User account management	**Priority**	**Rating**	**Rating**
a. Integrating existing user account data and history			
b. Flagging user actions			
c. Log in as a user for dev purposes			
d. ORCID integration			
e. Open standards			
f. OAuth2			
g. User action auditing			
10. Email	**Priority**	**Rating**	**Rating**
a. Sending nightly email			
b. Totally Decoupled			
c. Large scale (~10K daily)			
d. Ability to send one-off emails			
e. emails to specific users			
f. Public relation emails			
g. emails to sets of users (such as those affected by an incident)			
h. Emails to users via user account system			
i. User subscriptions to nightly email categories via email.			
11. I18N and accessibility	**Priority**	**Rating**	**Rating**
a. Unicode character handling			
b. Should at least not cause problems in this area			
12. CMS or Wiki (for public, admin and policy documentation)	**Priority**	**Rating**	**Rating**
a. Public relation features like front page news posting			
b. RSS feed not tied to records/articles			
c. Help and FAQs pages			
d. Admin policies			