Comment on: Draft Guidelines for Advancing UA Adoption

Drafter: Mark W. Datysgeld

Context

Governance Primer offers this comment from direct experience in the UA field. For 10 years, we have contributed to practical UA work by means of deliverables, evaluations, methodologies, and training activities developed in the context of the Universal Acceptance Steering Group ecosystem, including technical assessment and implementation-oriented work. We therefore approach this consultation not as outside observers, but as practitioners familiar with the persistent gap between awareness of UA and the routine production of UA-compliant systems.

Comment

Governance Primer appreciates the opportunity to comment on the "Draft Guidelines for Advancing UA Adoption". The draft is particularly useful where it identifies concrete technical surfaces of persistent UA failure, including validators, local-language email support, linkification, URL handling, authentication flows, email systems, and higher-level application dependencies. It is also constructive in recommending UA-compliant starter templates, issue logs, checklists, roadmaps, and CI/CD-compatible testing practices.

Our main concern is that the draft underestimates the single most scalable implementation channel now available, which was not available to prior large-scale UA efforts: AI-assisted software development, especially large language model (LLM)-driven code generation. The guidelines discuss outreach to developer communities, learning platforms, conferences, open-source ecosystems, and large technology organizations, but they do not appear to treat LLMs, coding assistants, or model-driven code generation as a distinct strategic vector for UA adoption.

In 2026, this omission is impossible to justify. The software development workflow is changing rapidly, and regardless of differing views on the long-term shape of the current AI market, code generation and code completion through LLMs have already become embedded in ordinary development practice around the world. Where generated code is wrong, errors propagate at scale. Where generated code is correct, compliance also propagates at scale.

We believe the draft should be revised to make AI-assisted code generation a first-order implementation priority. This is not a secondary matter. If leading AI companies, coding-assistant vendors, and model providers are engaged correctly, UA knowledge can be embedded directly where new software is actually produced. Model behavior can be improved through better training inputs, benchmark sets, retrieval sources, reference implementations, and better conformance-oriented examples.

In the UA context, this includes correct handling of internationalization principles, UTF-8, IDNA2008, multifaceted EAI scenarios, Unicode-aware validation, storage and display practices, multilingual email handling, and the many edge cases that have repeatedly surfaced through years of UA implementation work. The result would not be limited to awareness. It would produce more UA-compliant code by default.

This is, in our assessment, the strongest available bet because it has unusually high returns. Traditional outreach has value, but it depends on persuading individual actors to learn, remember, prioritize, and correctly implement technical requirements in future work. Past outreach-heavy approaches have required substantial effort while producing very uneven results.

By contrast, improving model defaults changes the baseline output of the development process itself. A corrected reference pattern can reappear across thousands of projects. A corrected validator, template, test suite, login flow, or email-handling pattern can circulate widely and continuously with no need to re-convince each developer from first principles. This does not eliminate the need for training, standards work, or ecosystem coordination, but it materially changes the economics of adoption in UA's favor.

Accordingly, we recommend that ICANN org explicitly add AI-assisted software development to the guidelines, especially in the sections dealing with open-source systems, software development professionals, and capacity-development materials. At minimum, the guidelines should call for engagement with leading AI and developer-tooling companies. The recommendation to create UA-compliant starter templates and CI/CD-integrated checks is already present in the draft; our suggestion is that these outputs should also be designed for direct uptake by LLM training, evaluation, retrieval, and toolchain integration.

In practical terms, ICANN should not treat this as a future possibility. It should treat it as an immediate deployment opportunity. If the goal is greater real-world acceptance of domain names and email addresses in all valid scripts and languages, then one of the most effective actions available is to ensure that the systems now helping write the Internet's software stop reproducing old UA mistakes and start generating compliant patterns by default.

Governance Primer therefore recommends that the draft be revised to recognize LLM-based coding systems as a strategic UA adoption channel and to prioritize direct engagement with the organizations shaping modern code generation. This should include benchmark suites, reference corpora, negative test cases, and implementation patterns suitable for model training, retrieval, and evaluation. In our view, this would materially strengthen the guidelines and better align them with how software is actually being built today.