Programming Language Reviews – Evaluation Criteria

Sara Dickinson

UASG @ ICANN 58
March 2017, Copenhagen

What is the problem?

- Observation: UA issues found in applications
  - 3 letter TLDs, ASCII only email address
- UA in software: Many applications, many languages, many requirements
- Q1: How well do languages natively support UA?
- Q2: Where is improvement needed?
  - Languages or applications

High level goals

- Evaluate popular programming languages
- Results => deliver guidance to stakeholders
  - Library developers (Compliance issues)
  - App Developers (Best libraries, example code)
  - Managers (Architectural decisions)
  - ‘Influencers’ (Input on overall status of UA)

How to proceed?

1. **Develop evaluation criteria** - based on multiple considerations

2. **Produce a report** -

    1. *Rating system* => recommendations

    2. *Remedial action* (fix the broken/missing stuff)

        - Either through engagement or possibly patches

Scope - what is a ‘programming language’?

- Flexible definition - a ‘component’
  - Package, framework, libraries
- “Web sphere” languages - **but** need to consider back end languages too! (see next slide)
- Initially focused on Open Source

Some candidate languages

- **Javascript npm module.** Supports IDNA2003 and IDNA2008. Bundled with Node.js.
- **PHP IDN functions.** Part of the PHP standard library, supporting IDNA2003 and IDNA2008.
- **Python encodings.idna** / **idna** modules
- **Go idna package.** Part of the Go standard library supporting IDNA2008.
- **GNU Libidn.** / **GNU Libidn2.** (Written in C with bindings for Perl and Ruby)

Evaluation Criteria

- Scoring system (weighted)
    - **Basics** - license, portability, cost?
    - **Support** - bugs, maintained, documentation
    - **Implementation notes** - issues, pitfalls, etc.
    - ***Test suite*** *- directly asses functionality*

Test suite:
Basic functionality

- As we all know:
    - Accept, Validate, Store, Process, Display
- 3 ‘identifiers’:
    - Domain names, Email, URLs

Specifics of evaluation

- **Low level** vs **High level** functions (for all 3 identifiers)
- **Low level**: basic transformation services (defined in relevant RFCs)
    - e.g. ToASCII, ToUnicode, equivalence,…
- **High level**: Application level functions
    - Decompose identifiers to elements
    - Syntactic and semantic checks (incl. common operational restrictions?)

Current status

- Evaluation document under review:

Evaluation Criteria

- Feedback welcome!