Evaluation

Unix Philosophy

From **Beginning Linux Programming, 4th Ed,** Matthew & Stones, Wrox/Wiley Pub.

☐ Simplicity

Many of the most useful UNIX utilities are very simple, and, as a result, small and easy to understand. ***KISS*** is a good technique to learn. Larger, more complex systems are guaranteed to contain larger, more complex bugs, and debugging is to be avoided.

☐ Focus

Often it is better to make a program perform one task well, rather than to throw in every feature, along with the kitchen sink. A program with "feature bloat" can be difficult to use and difficult to maintain. Ones with a single purpose are easier to improve, as better algorithms or interfaces are developed. Small utilities can often be combined to perform demanding tasks when needed, rather than trying to anticipate needs in larger programs.

☐ Reusable Components

Make the core of you application available as a library. Well-documented libraries with simple but flexible programming interfaces can help others to develop variations, or apply the techniques to new application areas.

☐ Filters

Many UNIX applications can be used as filters. They transform their input and produce output. Quite complex applications can be developed by combining UNIX programs in novel ways.

☐ Open File Formats

The more successful and popular UNIX programs use configuration files and data files that are plain ASCII text or XML. These enable users to use standard tools to change, search for configuration items, and develop new tools to perform new functions on data files.

☐ Flexibility

We can't anticipate exactly how ingeniously users will use programs. Programs should be flexible and avoid arbitrary limits on field sizes or numbers of records. Programs should be made network-aware and able to run across a network as well as locally. We should never assume we know everything that the user might want to do.

KISS

Keep **I**t **S**imple and **S**mall

Usability

WHY

☐ **Need ~** ☐ Requirement ☐ Use Case

☐ **Clear Description**

HELP

☐ Community

☐ Commercial Support

☐ Documentation ☐ FAQs ☐ README

◘ man pages ◘ info pages ◘ other

RELEASE MODEL

◘ Funded ◘ Unfunded

Cost

☐ Time ☐ Money

Release Schedule

◘ Point ◘ Rolling

HOW TO USE

☐ Installation

☐ Configuration

☐ Demonstration

INTERFACE

☐ Command Line Interface (CLI)

☐ Graphical User Interface (GUI)

Desktop Environment (DTE)

☐ Gnome ☐ KDE ☐ MATE ☐ Other

☐ **EULA** ☐ **TLDR**

◘ **GPL2** ◘ **GPL3** ◘ **Other License**

TOOLS

◘ Metrics ◘ Performance

Functionality

Key Performance Indicators

(Expectation of Features)

☐ **Usability**

☐ **Capability**

☐ **Performance**

☐ **Manageability**

☐ **Scalability**

☐ **Security**

Evaluation Short List

☐ Comparable Alternatives

(Compare top alternatives)

Plus/Minus

Compare Significant Features

Evaluation

Cross Platform Portability
☐ bsd ☐ linux ☐ mac ☐ ms win ☐ UNIX

CROSS DISTRIBUTION AVAILABILITY
Repositories~ ☐ Bin ☐ Source

Packages **~ ☐ DEB ☐ RPM ☐ TAR**
☐ PPA ☐ Romeo ☐ Other
☐ Flatpack ☐ Snap

Compatibility

Three measures of compatibly:

☐ Portability
– software can run on different systems.

☐ Scalability
– software to run on computers of differing levels of performance.

☐ Interoprability
– computer systems can communicate with each other.

Dependability

Security

DESIGN ISSUES

☐ BUGS
Vulnerabilities
☐ Zero Day ☐ Responsible Disclosure
☐ Severity Level

☐ Mitigation
◘ Risks ◘ Benefits ◘ Effectiveness

SOURCE CODE
◘ Closed ◘ Open
☐ Audited

MAINTENANCE
◘ New ◘ Mature
◘ Active ◘ Abandoned
☐ Roadmap
Rate of repair
Team size

☐ Trust No One (TNO)

Confidentiality
Integrity
Accessibility

Authentication
Encryption

Privacy

Does product have cost, or are you the product?

Anonymity

What could possibly go wrong?

What We Don't Know!

? Consequences ?
? Unknown Consequences ?

? Responsibility ?
? Transparency ?
? Ethics ?

Suggest considering Bloom's Taxonomy.