Published DRAFT

OPUSVL

Document overview

- Scoring overview
- Implementation
- Selection criteria

DITO Project

OpenChain Governance - code selection criteria

Measures for inclusion of Open Source code

Date: 01/09/2020
By: Stuart Mackintosh, OpusVL development team
Version: v0.1

hello@opusvl.com
01788 298 450
opusvl.com

Drury House
Drury Lane
Rugby CV21 3DE

Opus Vision Limited t/a OpusVL
Company No. 03905104
VAT No. GB 747 8294 82

Open source **business management software.**
Customised to **fit the way you work.**

Contents

1 Summary

With the aim of developing a practical and sustainable governance policy with which imported Open Source can be measured against, this document presents the output of the OpusVL developer workshop held 1st September 2020.

The policy must be usable by developers who rely on Open Source components and provide a simple and

The briefing for the workshop was as follows:

- Develop policy
  - based on rating / strategy
  - to apply to the selection of Open Source code and components
  - when working on a project
  - that is subject to OpenChain compliance
  - and practically useful

1.1 Scoring overview

Scoring is based on a set of selection criteria.

Each selection criteria has an importance factor (low,medium,high); weighted accordingly and a rating factor.

Overall scoring is assigned with a higher score being preferred for each criteria.

The software developer is responsible for validating any software against requirements; this selection criteria does not reflect suitability for purpose or functionality.

2 Implementation notes

2.1 Dynamic loading and CDN

Although dynamic loading of code is not itself related to structure, it must be considered with a view to managing compliance.

In order to maintain compliance, code must not be dynamically loaded from uncontrolled 3rd parties or CDN, but only from the controlled software origin as this will invalidate OpenChain compliance.

2.2 Code elements

An element of code is described as the unit of code to be imported. This may be a code library, function or application.

As an element, it will be subject to one licence, have one maintainer and one community.

Code composites should be broken down to elements before measuring against the selection criteria.

2.3 Calculating the score

Each code element is measured against the selection criteria and score is calculated based on the importance and rating against the selection criteria.

The Importance factor is stated as Low, Medium, High and scored from 1 to 3 respectively.

Rating factor is measured from 1 to 5.

The section score for each criteria is calculated as: importance x rating

Therefore the lowest score for any criteria is 1 and the highest is 15.

The scores for each selection criteria are added together, giving a total score for the code element.

Scores are relative to each other; higher scoring code elements are preferred over lower scoring elements.

Acceptable use of a code element for a particular purpose is defined through the code selection filter

process of which the score feeds in to along with other parameters.

3 Selection criteria

When measuring code elements against the criteria, the Importance factor should be used as guidance as to the level of diligence given to each criteria.

3.1 Component criticality to solution

Importance: high

- Depends on the importance of the component required
- If critical to the solution, only high rating
- Does the operation rely on this component, or is it periphery
- If one choice of many, lower rating acceptable

low rating Component is critical to the solution that it is part of
high rating Component has little impact on on the overall solution

3.2 Security

Importance: high

- High rated CVE
  - Typical time to address CVE
  - Time since report
- Security test record
  - External penetration test
  - Functional testing
  - Human testing
  - Software scanning
- Compatible with contemporary security practice
  - https
  - tls 1.2+
  - Data protection policies
- Published security policies
  - Responsible disclosure
  - Security report instructions

low rating unresolved long-standing CVE (+60 days) or known issues

high rating CVE's promptly addressed, availability of security testing logs

3.3 Community

Importance: high

- Values
- Availability
- Accessibility
- Breadth / diversity
- Governance
- Stability
- Release cycle

low rating Incompatible values, unavailable, no clear release strategy
high rating Diverse community across sectors and use cases, strong published governance, stable core team

3.4 Maintainer

Importance: medium

- Activity
- Availability
- Individual or organisation

low rating Single unavailable maintainer
high rating Maintainer active and engaged with community, succession plan

3.5 Commercial sponsor

Importance: medium

- Reputation
- Motivations
- Level of community involvement

low rating Commercial sponsor is known to change licence terms, has reputation for agression to open source community
high rating Commercial sponsor is transparent with motivations and has reputation fore good behaviour in software, or no commercial sponsor

3.6 Code maturity

Importance: medium

- Quantity of code commits

- History of closed issues
- Time since release

low rating Few public commits, issues not closed, no track record
high rating Proven code, many commits, issues addressed, long stable history

3.7 Code quality

Importance: medium

- Unit tests
- Coding standards used ie linting
- Commented
- Optimised / normalised
- Structure of code / libraries / project
- Compatibility with other components

low rating No test coverage, no structure or standards
high rating good test coverage, normalised functions, works well with other components

3.8 Foundation sponsor

Importance: low

- low rating: no foundation
- high rating: Linux Foundation/Apache Foundation/Eclipse Foundation

low rating No foundation
high rating Active and engaged foundation

3.9 Activity

Importance: low

- Frequency of code commits
- Community discussion
- Releases
- Open/closed issues

low rating Inactive code and community
high rating Active code and community

3.10 Reputation

Importance: low

- Known enforcement notice

- Known to make breaking changes
- Documentation standards

low rating Known enforcement or disputes
high rating Clean record

4 Appendix 1: Default OpenChain example

The OpenChain policy template includes an example structure as follows:

- Commercial Sponsor
  - Low score: none
  - High score: Intel/IBM/Microsoft/Red Hat
- Foundation sponsor
  - low score: none
  - high score: Linux Foundation/Apache Foundation/Eclipse Foundation
- Code maturity
  - low score: New
  - high score: long-established
- Stability
  - low score: project has forked multiple times
  - high score: project has never forked
- Activity
  - low score: no commits for 5 years
  - high score: daily substantive commits
- Reputation
  - low score: multiple enforcement actions
  - high score: no enforcement action
- Quality
  - low score: Buggy
  - high score: Stable
- Security
  - low score: Known multiple security issues
  - high score: No security issues exist
- Community involvement
  - low score: We are not and do not want to be involved in this community
  - high score: We are or want to be heavily involved in the community

OPUSVL

Since 1999, OpusVL has helped organisations get the business management software they were looking for (but just couldn't seem to find). We combine our 'off-the-shelf' products with the craftsmanship it takes to tailor your software to your needs.

With over 20 years' experience, we're a multi-disciplined team of developers, people-people and project management experts. But, above all else, we're proud to be a team that our clients trust to get the software – and results – that they need.

opusvl.com
01788 298 450
hello@opusvl.com

Drury House
Drury Lane
Rugby CV21 3DE

Cyber Essentials

Crown Commercial Service Supplier

odoo Community Member

UKRI Innovate UK