Questionnaire on Opens Source Software (OSS) Adoption

Xiaozhou Li*[a], Sergio Moreschini*[a], Zheying Zhang[a], Davide Taibi[a]

[a] *Tampere University, Tampere (Finland)*
∗ *the two authors equally contributed to the paper*

Keywords: Open Source, Software Selection, Open Source Adoption

---

1. **Personal Information**
   - 1.1. Name
   - 1.2. Age
   - 1.3. Role
   - 1.4. Time in the company (years)
   - 1.5. Unit/Department
2. **General Questions regarding OSS Adoption**
   - 2.1. How experienced are you with OSS?
   - 2.2. Are you following any specific model for selecting OSS or are you adapting any model to your needs? If yes, which changes are you applying to the standard model?
   - 2.3. What are the key factors that you consider when you adopt OSS? How important is each factor from 0 to 5?, where 0 means not important at all, and 5 means extremely important?
   - 2.4. What factors do you think personally important? How important is each factor from 0 to 5, where 0 means not important at all, and 5 means extremely important?

     F1: Community & Support

     F2: Documentation

     F3: Economic

Email addresses: xiaozhou.li@tuni.fi (Xiaozhou Li*), sergio.moreschini@tuni.fi (Sergio Moreschini*), zheying.zhang@tuni.fi (Zheying Zhang), davide.taibi@tuni.fi (Davide Taibi)

January 7, 2021

F4: License

F5: Operational SW Characteristics

F6: Maturity

F7: Quality

F8: Risk

F9: Trustworthiness

F*: Other factors

2.5. *Note for the interviewer:* For each factor evaluated with importance higher or equal than 3, ask the interviewee to rank the sub-factors and to specify which measure they adopted to evaluate them, and from which source they commonly obtain the measures (e.g. GitHub web interface, GitHub APIs, SonarCloud, manual measurement, ...)

Factor	Measure
Community Support and Adoption	
Popularity	
	# Watch
	# Stars
	# Fork
	# Downloads
Community reputation	
	Member of a foundation (e.g. Apache, Linux)
	Complete administration mechanism
	Proactive checking on OSS quality
	Transparent OSS quality
	Fast response to issues,
Community creativity	
Community size	
	# Contributors
	# Subscribers
	Community age
	# Involved developers per company
	# Independent developers
Support and service	
Community support	
	Activeness
	Responsiveness
Communication	
	Availability of questions/answers
	Availability of forum
	# Mailing lists
	Traffic on the mailing list
	Responsiveness of postings
	Friendliness
	Relation between stakeholders
	Quality of postings
Involvement	
	Coordination
	Clear project management
	# mailing list members
Sustainability	
	Existence of maintainer
	Maturity Implications of Mailing List Support
Paid support	
	Availability of official support
	Availability of 3rd party support
	External service provider support
	Quality of professional support
	Availability of training
	Availability of end user training
Failure support or maintenance	
Product Team	
	Developer quality
	Productivity

Factor	Measure
Economic	
Cost	
	License
	Support
	Training/Learning
	Staffing
	Promotion
	Ownership (TCO)
	Return of investment (ROI)
Clear project management	
Profitability	
Resources and investment	
Documentation	
Documentation completeness	
Availability of documentation/books/online docs	
	Responsiveness to issue
	Acccuracy
	Updated Documentation
Availability of development process documentation	
	Coverness
	Readability
Reference documentation	
	Available Get start document
	Reference Completeness
Tutorial/Guidelines documentation	
	How to start from 0
	Coverage
	Accuracy
Usage documentation	
	Instructions and Examples
	Completeness
	Readability
Best practices	
	Classic Best example and hwo to optimize
	Guidance
Community's experience	
Availability of architectural documentation	
	Accuracy
Distribution media	
Software requirements	
	Completeness
Hardware requirements	
Dissemination	
Road map	
Test case documentation	
Quality report	
	Coverage
	Accuracy
Coding comments	
	Readability

Factor	Measure
License	
	License type Law conformance
Operational SW Characteristics	
Trialability	
	Available for independent verification and compile Provide demo for quick evaluation
Independence from other SW	
	Adopted SW architecture Run independently Supports independent libraries Fewer dependences
Development language	
	Mainstream dev Lang Language know in the company Programming language uniformity
Multiplatform support	
	Portability
Standard compliance	
Maturity	
	# forks Stability Release version stability # bugs Release frequency # releases # releases per period Age (#Years) # commits # bug reports Development versions System growth New feature integration

Factor	Measure
Quality	
Resilience	
Reliability	
	Component reliability
	Architecture reliability
	System reliability
	# faults (open, closed, ...)/Fault density
	Average fault age
	Mean time between software failure (MTBF)
Performances	
	Scalability
Security	
	# security vulnerabilities
	# security vulnerabilities on NVD portal
	Information for security
Modularity	
Usability	
	Understandability
	Operability
	Learnability
Portability	
	Adaptability
	Installability
Flexibility/Exploitability	
Code Quality	
	Code complexity (class, methods, ..)
	Change proneness
	Fault proneness
	Test coverage
	Code size
	Technical difficulty
	Other (Please specify)
Longevity	
	Age of product
	Version number
Coding conventions	
	Method calling conventions
	Class naming conventions
	Loop/Switch/If conventions
	Public/private conventions
	Overriding converntions
Coding practices	
	Usage of Linters
	Definition of customs/rules for Linters
	Issue/Commit traceability
Maintainability	
Analyzability	
Testability	

Factor	Measure
Quality	
Changeability	
	Avail. and type of Defect management system
	Avail. and type of Version management system
	Opened defects
Source code changes/modifiability	
Resolved defects	
Fixed defects	
Enhancements	
Resolving time for non-fixed defects	
Resolving time for fixed defects	
Resolved defects life cycle	
Fixed defects life cycle	
Detect initiated source code changes	
Defect proneness	
Defect Management	
Version Management	
Update/Upgrade/Add-ons/Plugin	
Visibility	
Development process	
As-is utility/Quality in use/External quality	
	Effectiveness
	Efficiency
Satisfaction	
	Functionality/Non-functionality
	Usefulness
	Trust
	Pleasure
	Comfort
Freedom from risks	
	Economic risk migration
	Health & Safety risk migration
	Environmental risk migration
Context coverage	
	Context completeness
	Flexibility
Architecture	
	Dependence & constraints

Factor	Measure
Risk (Perceived risks)	
Perceived lack of confidentiality	
Perceived lack of integrity	
Perceived high availability	
	Test according to context
	Analysis and pre-examination
	Comply with business requirements
Perceived high structural assurance	
Business Risks	
Strategic risks	
Operational risks	
	Influence of operation specified
Financial risks	
Hazard risks	
	consequences specified
	Moment metrics
	Code security
	Virus scanning
Trustworthiness	
Component	
	Component development & certificate processes
	Component reputation & user community
	Functionality
Architecture	
	Dependence & constraints
System	
	Component installation
	Delivery
Functional requirements satisfaction	
	% of functional requirements implemented
	Compliant with description
	Low extra useless features
	# companies adopting it
	Performance and Reliability
Customer satisfaction/requirements	
	Matching customer satisfaction/Features
	Function analysis and verification evaluation
Interoperability	
Facilities for modifying and customizing	
OSS provider reputation	
Interface language localization	
Existence of benchmark/test	
	Fast responsiveness to Issues
	Fast responsiveness to malicious affairs
	Transparency
	Fame of community
	Support for mainstrame languages
	Test cases availability
	Basic functions covering
Collaboration with other product	
	Commercial vendor integrations
	Community-created integrations
	The percentage of existing needed integrations