

FLOSS Quality and Maturity Models

Project: EUX2010SEC

Lars Strand

Norwegian Computing
Center

Bergen

29. Oct 2008

Free and Open Source Software

- ▶ SourceForge.net ~180 000 software projects
- ▶ GNU Savannah ~2900 software project
- ▶ Several other FLOSS projects hosted elsewhere
- ▶ Gartner prediction
 - 2011 – 27% market share of the infrastructure market
 - 2012 – 80% of all commercial software will include elements of FLOSS
- ▶ Problem:
 - How can we decide that one FLOSS project is better than another? Can we measure its quality?

Quality

- ▶ Not a clear concept
- ▶ Can be a feature, or as one or more attributes to an object
- ▶ Attributes can be perceived and interpreted differently
- ▶ Determined by testers preference, knowledge and experience
- ▶ Thus making quality an abstract concept and highly subjective

Quality definition

- ▶ ISO9000
 - “Degree to which a set of inherent characteristics fulfills requirements”
- ▶ IEEE
 - “The degree to which a system, component or process meets specific requirements”
 - “The degree to which a system, component or process meets customer or user need, or expectations” (customer satisfaction)

ISO/IEC 9126

- ▶ International standard for evaluation of software quality
- ▶ Model not related to open source specifically
- ▶ Six quality characteristics defined
 - Functionality
 - Reliability
 - Usability
 - Efficiency
 - Maintainability
 - Portability
- ▶ Each with a set of subcharacteristics which in total gives metric for each characteristic

- ▶ Product quality
 - measuring by examine code quality
 - underlying design of program
 - usually very specific tests

- ▶ Methods/development model
 - How does the FLOSS development model differ?
 - organization, leadership, development tools, sustainable communities
 - A lot of research in the past few years

Current efforts

- ▶ CapGeminis Open Source Maturity Model (OSMM)
 - 27 grouped indicators computes an overall score
 - Service offered by CapGemini
 - Not updated since 2003
- ▶ Navicas Open Source Maturity Model (OSMM)
 - Six different “product elements” are measured and given a score based on customer requirements
 - Not updated since 2005

- ▶ Method for Qualification and Selection of Open Source Software (QSOS)
 - Licensed under GNU FDL
 - Two sections
 - generic – common for all
 - specific – focus on functionality (databases, wikis,...)
 - Good online documentation
 - Tools available – but starting to show its age (2006)
 - A range of metrics, but no overall score
 - Results can be uploaded to the QSOS website
 - Several assessments available on the website

QSOS report

OpenLDAP 2.4.11

Information

Sheet created on 2008-10-15

Language: en

Application: OpenLDAP

Release: 2.4.11

License: Open Content License

Url: <http://www.openldap.org/>

Desc:

Authors of this sheet: [David Pillant](#)

You can access to the sheet change log on [the CVS](#).

Generic section

Generic criteria from QSOS version 1.6

▣ Intrinsic durability

▣ Maturity

▣ Age

less than 3 months

if between 3 months and 3 years

after 3 years

The OpenLDAP project was started in 1998

Score : 2/2

▣ Stability

Unstable software with numerous releases or patches generating side effects

Stabilized production release existing but old. Difficulties to stabilize development releases

Stabilized software. Releases provide bug fixes corrections but mainly new functionalities

roadmap: <http://www.openldap.org/software/roadmap.html>

Score : 2/2

▣ History, known problems

Software knows several problems which can be prohibitive

No know major problem or crisis

History of good management of crisis situations

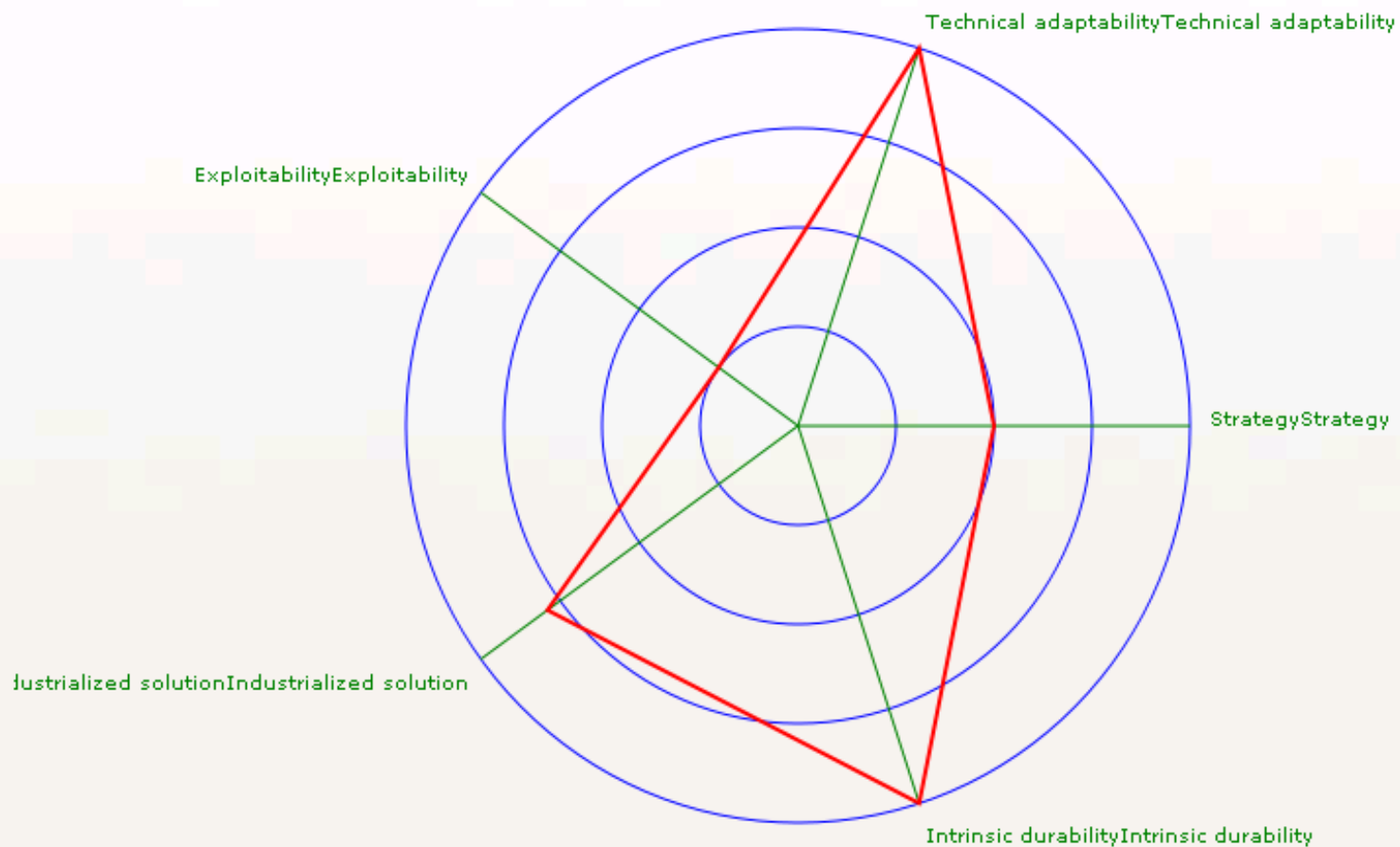
<http://www.openldap.org/its/index.cgi>

Score : 2/2

▣ Fork probability, source of Forking

QSOS chart

Asterisk 1.4.20 > Generic section



OpenBRR

- ▶ Open Business Readiness Rating (OpenBRR)
 - Last updated 2006 – ceased development?
 - Build on (both) OSMM models
 - A hierarchy with 12 categories with a total of 29 metrics
 - Each category can be weighted according to requirements
 - Each metric is typed into a OpenBRR spreadsheet
 - The metrics are specific and easy to quantify

OpenBRR example

Evaluated Technology And Functional Orientation

Component Name: Asterisk v1.4.20.1

Component Type: PBX (VoIP)

Usage Setting: Business Critical

Rank	Category Title	Weight	Unweighted Rating
1	Functionality	25,00%	3

Rank	Category Title	Weight	Unweighted Rating
2	Operational Software Characteristics	15,00%	4,35

2,1 Usability

Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
End user UI experience	Advanced software with many features demands manual for proper use.	5,00%	3	0,15

Test Description

This measures how well the UI is perceived by an end-user. (Intuitive interface/ navigation/control scheme)

Test Score Specification

Test Score Specification	Score
Simple & Intuitive, information is well organized, no manual required	5
Takes little time to learn, information somewhat organized, some use of the manual	3
Complex, too much information, no obvious organization, cannot use without a manual	1

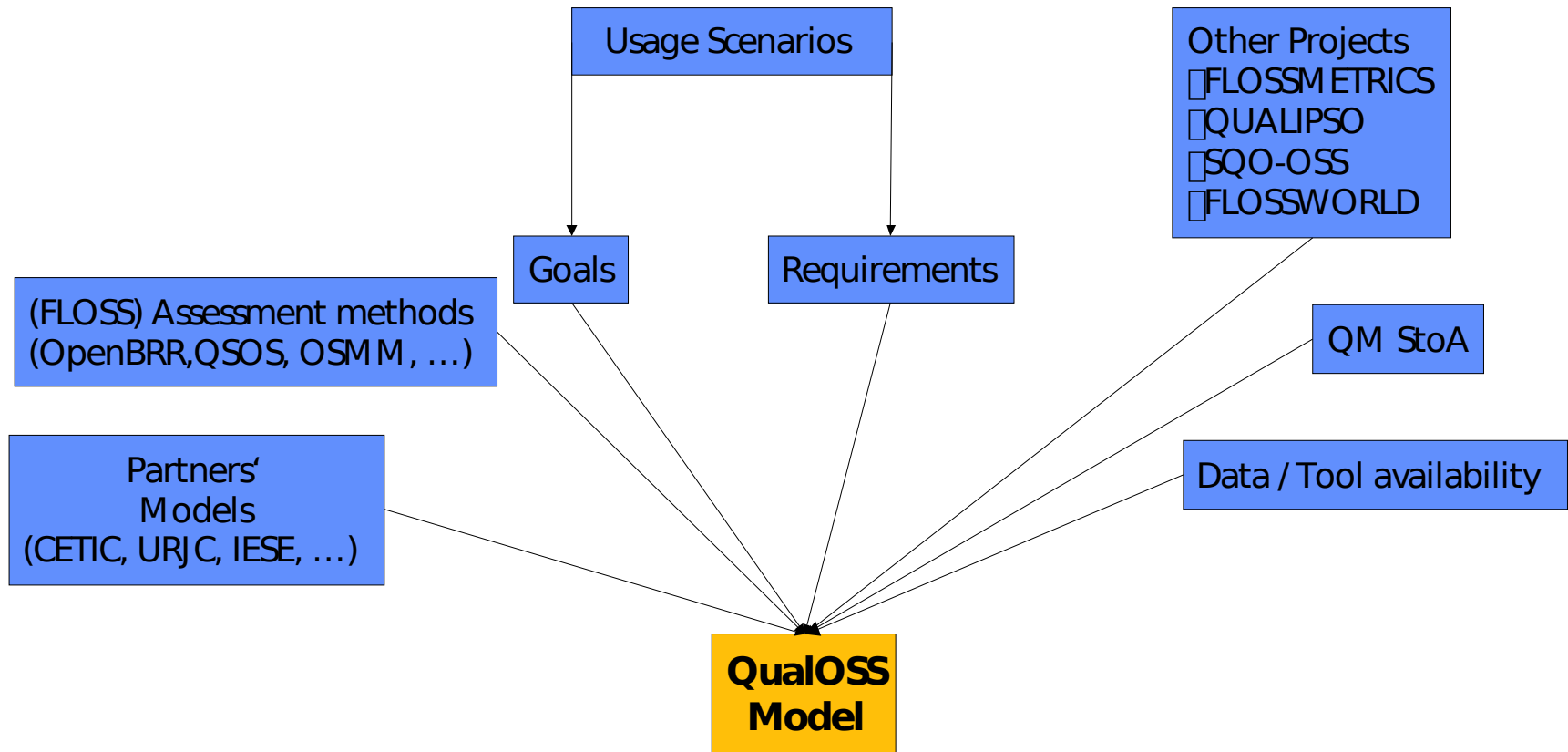
OpenBRR scores

Categories	Metrics	Description	Scoring				
			5 - Excellent	4 - Very good	3 - Acceptable	2 - Poor	1 - Unacceptable
	Number of point/patch releases in past 12 months	Typically, official point/patch releases are fixes for P1 bugs like deadlock, memory, and security vulnerabilities.	3 - 4		1 - 2, or 5 - 6		0 or > 6
	Number of open bugs for the last 6 months	This measures the quality of product usage.	< 50	50 - 100	100 - 500	500 - 1000	> 1000
	Number of bugs fixed in last 6 months (compared to # of bugs opened)	This measures how quickly bugs are fixed.	> 75%	60% - 75%	45% - 60%	25% - 45%	< 25%
	Number of P1/critical bugs opened	This measures the seriousness of quality issues found.	0	1 - 5	5 - 10	10 - 20	> 20
	Average bug age for P1 in last 6 months	This measures the responsiveness to fixing critical issues.	< 1 week	1 - 2 weeks	2 - 3 weeks	3 - 4 weeks	> 4 weeks
Security							
	Number of security vulnerabilities in the last 6 months that are moderately to extremely critical	This measures the quality related to security vulnerabilities. How susceptible the is software to security vulnerabilities.	0	1 - 2	3 - 4	5 - 6	> 6
	Number of security vulnerabilities still open (unpatched)	This measures whether the project is capable of resolving all security issues.	0	1	2	3 - 5	> 5
	Is there a dedicated information (web page, wiki, etc) for security?	This measures how aware of and seriously the project takes security issues.	Yes, well maintained		Yes		No

QualOSS

- ▶ QUALity in Open Source Software (QualOSS)
 - Ongoing research project
 - Builds on OSMM, QSOS and OpenBRR
 - Goal: Build a high level methodology to benchmark the quality of FLOSS
 - The metric will be collected by (mostly) automated tools from sources such as:
 - Project releases
 - Version control systems (CVS, SVN, etc.)
 - Bug tracking systems (Bugzilla, etc)
 - Mailing lists archives
 - Some advanced metrics might be collected manually

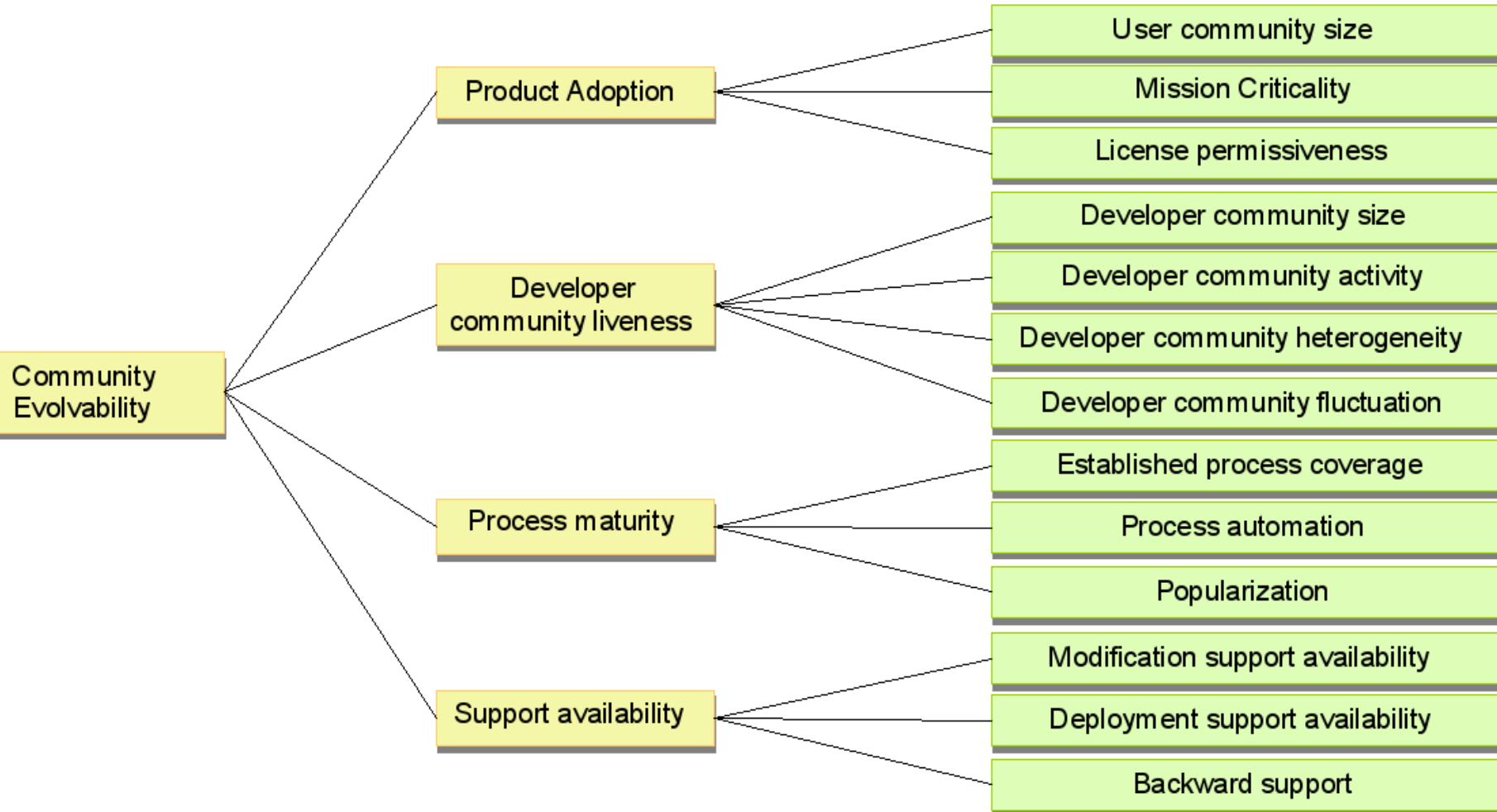
QualOSS input sources



QualOSS model

- ▶ Evolvability – *“the general ability of a FLOSS project to deliver over time”*
 - Product evolvability – ability of a product to be corrected, adapted and extended over time
 - Community evolvability – that a community remains able to maintain the product over time
- ▶ Robustness
 - Product robustness – in presence of invalid inputs or stressful environments
 - Community robustness – the ability to guarantee the delivery of robust products.

QualOSS excerpt



QuaLOSS: Community Evolvability...

Evolvability: Community Quality Model

Metric		Issue Tracking	Security DB	Ver. Control Rep	Publication db	Packaged Dist List	Website	mailing list	Value	Comment regarding measures for example, what values were used for each field in the query	Comment regarding other info available (or lack of info) or any other remarks	Indicators
Product adoption => User Community Size	<i>NumOfDevelopers</i>			X					42	select count(distinct(commiter)) from committers;		Green
	<i>NumOfPostersMailingLists</i>							X	9570			Green
Product adoption => Mission Criticality	<i>No metrics defined</i>											
Product adoption => License permissiveness	<i>LicenseUsedSourceCode</i>			X			X		GPL	If we do not have a tool to analyze and obtain license from a project, there are some cases where the license is not possible to obtain in a fast way. For instance, in Maemo or Plone project there are several directories and each one has several files, so it is not possible to analyze manually.	Perhaps this is not a basic metric	Yellow
	<i>LicenseUsedDocumentation</i>			X			X		¿?	If we do not have a tool to analyze and obtain license from a project, there are some cases where the license is not possible to obtain in a fast way. For instance, in Maemo or Plone project there are several directories and each one has several files, so it is not possible to analyze manually.	Perhaps this is not a basic metric	

QuaLOSS: File Complexity

FileCyclomaticComplexityAverage

Threshold Avg. Compl.	0,2
Num Files over Threshold	57
Total Number of Files	477
HotFilesPercentage	12%

File Name	MethodCom	Lines	Ratio
agi\eagi-sphinx-test.c	38	231	0,16
agi\eagi-test.c	26	165	0,16
apps\app_adsiprog.c	337	1581	0,21
apps\app_alarmreceiver.c	127	813	0,16
apps\app_amd.c	66	418	0,16
apps\app_authenticate.c	46	212	0,22
apps\app_cdr.c	5	63	0,08
apps\app_chanisavail.c	17	157	0,11
apps\app_channelredirect.c	7	93	0,08
apps\app_chanspy.c	134	730	0,18
c	25	168	0,15
apps\app_db.c	18	139	0,13
apps\app_dial.c	440	2047	0,21

FLOSS development model

- ▶ Successful FLOSS project have
 - 1) Modularity – “can the work be broken into pieces?”
 - 2) Granularity - “can each piece be small enough?”
 - 3) Low cost integration
- ▶ These three characteristics is the foundation of the new production model called “*Commons Based Peer Production*” (CBPP) -- Y. Benkler
- ▶ The maturity models should reflect the findings in the CBPP model

Questions?

Thank you