Can software be healthy?

Jesus M. Gonzalez-Barahona

Universidad Rey Juan Carlos
@jgbarah http://jgbarah.github.io/presentations

SoHeal 2019
Montreal (Canada), May 28th 2019

Health,
what is health?
Can anyone be healthy
at all?

Speaker: What do we want?
Crowd: Patience!
Speaker: When do we want it?
Crowd: Right now!!!

Adapted from a well kwnown joke
by Eugenio (Spanish humorist).

The theory

Software should behave
according to requirements,
be cheap to mantain,
be easy to use,
have good performance,
...

"We want software of good quality"

The practice

In most cases...

- Functionality: shallow verificaton
- Requirements: from inexistent to incomplete
- Maintainability: very expensive
- Usability: many facets
- Performance: only a relative target

"Good enough", depending on the stakeholder

Improving quality

The quest for quality

"Traditional" approach in software engineering:

- Product quality
  (ISO 9126, CISQ)
- Process quality
  (ISO 9001, CMM)

Follow the rules, increase quality

CISQ (code) quality model

- reliability
- efficiency
- security
- maintainability

https://www.it-cisq.org

CISQ (code) quality model

SOFTWARE QUALITY CHARACTERISTIC	CODING PRACTICES UNIT LEVEL	ARCHITECTURAL PRACTICES SYSTEM LEVEL
RELIABILITY	• Protecting state in multi-threaded environments • Safe use of inheritance and polymorphism • Resource bounds management, Complex code • Managing allocated resources, Timeouts	• Multi-layer design compliance • Software manages data integrity and consistency • Exception handling through transactions • Class architecture compliance
PERFORMANCE EFFICIENCY	• Compliance with Object-Oriented best practices • Compliance with SQL best practices • Expensive computations in loops • Static connections versus connection pools • Compliance with garbage collection best practices	• Appropriate interactions with expensive or remote resources • Data access performance and data management • Memory, network and disk space management • Centralized handling of client requests • Use of middle tier

Measuring quality

There are other motivations

What if the focus is "knowing" instead of "improving"

- comparison
- tracking
- self-awareness

There are other subjects

What if the people are also important?

- the builders
- the evaluators

The builders

Specially important in FOSS:

- diverse people working together
- different motivations, agendas...
- the sense of community

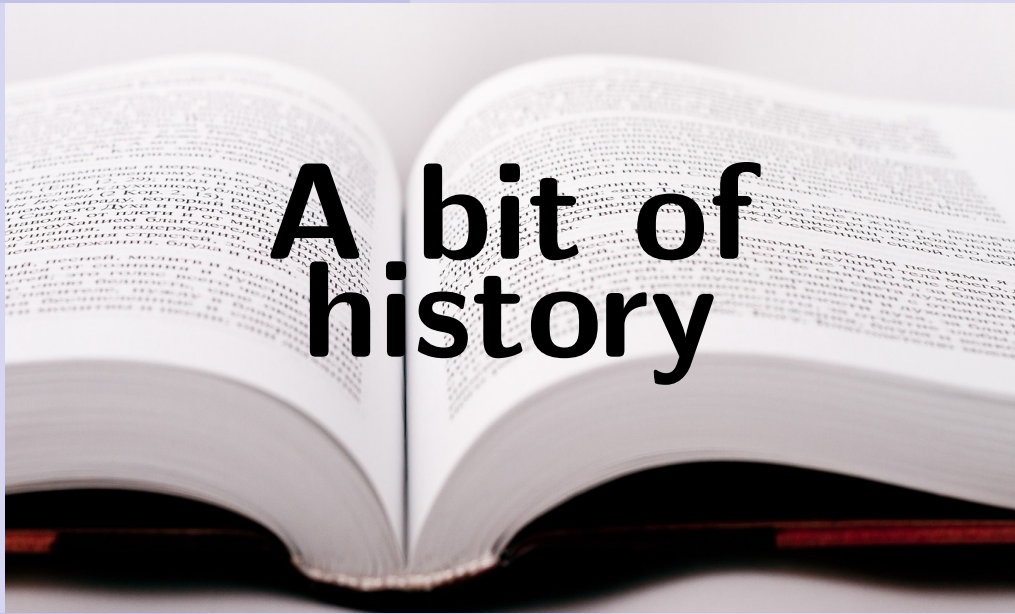
And we still have the context...

Software is not used in a vacuum:

- legalese
- support
- economy
- ecosystem
- ...

The evaluators

Different goals / interests
mean
different definitions of "good"



A bit of history

OpenBRR

Business Readiness Rating™

A Framework for Evaluating Open Source Software

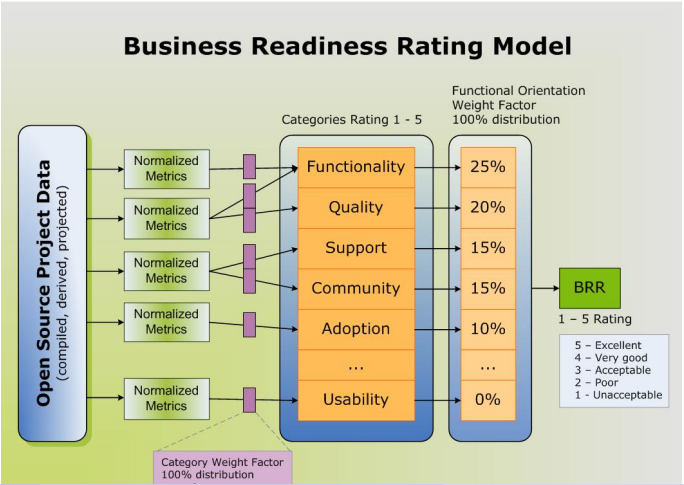
STEERING COMMITTEE

Larry Augustin, Open Source Strategist
Michael Goulde, Forrester Research
Peter Kronowitt, Intel
Murugan Pal, SpikeSource
Josh Berkus, PostgreSQL
Marc Hedlund, O'Reilly CodeZoo
George Pace, Prudential Financial
Anthony Wasserman, Carnegie Mellon West (Chair)

FOUNDING SPONSORS

Carnegie Mellon West spike SOURCE intel. O'REILLY code Zoo

OpenBRR

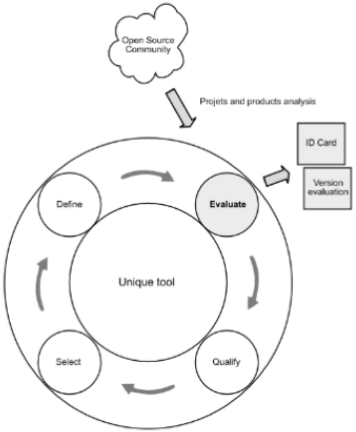


Qualoss

Atos Origin

QSOS

QSOS



- ID card and version evaluations;
- Scoring of criteria on three major axis:-
  - Functional coverage;
  - Risks from customer perspective;
  - Risks from Atos Origin perspective;
- Weighted metrics for product scoring;

QSOS

Intrisic robustness
- Maturity
- Adoption
- Development Roadmap
- Activity
- Development independence

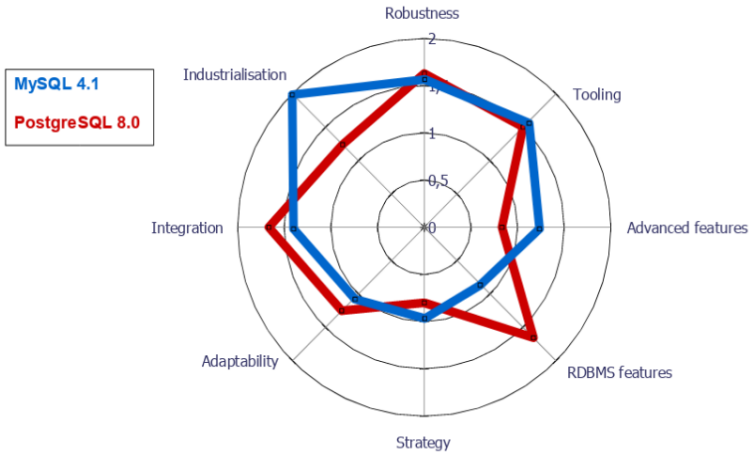
Industrialised solution
- Services
- Documentation
- Quality Assurance
- Exploitability

Integration
- Adherence to standards
- Interface with other products

Technical adaptability
- Modularity
- By products

Strategy
- Licence
- Copyright owners
- Modification of source code
- Roadmap
- Sponsor
- Strategical independance

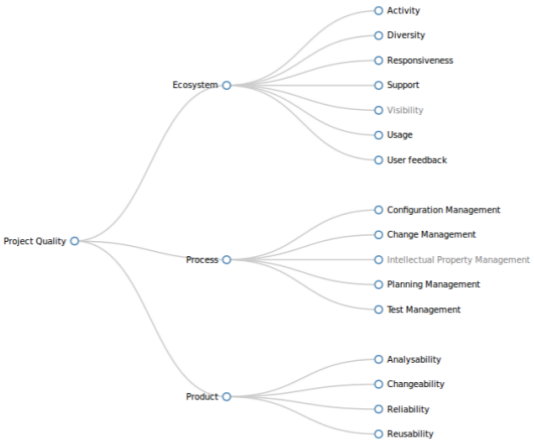
QSOS



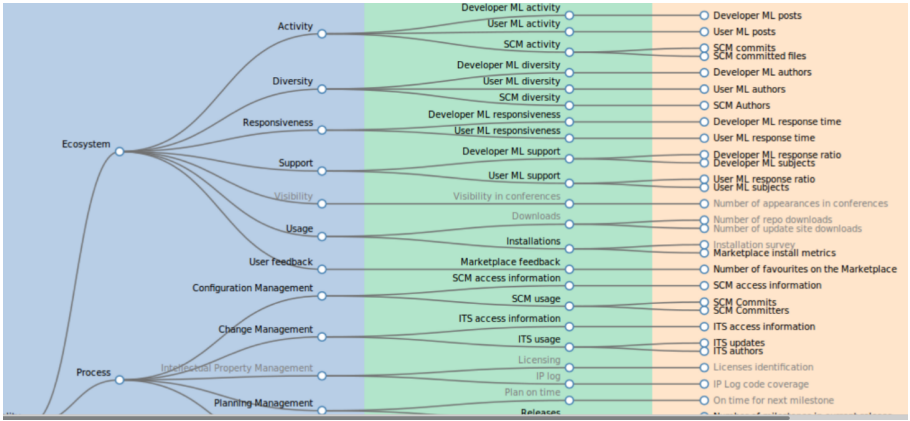
OMM

QualiPSo OpenSource Maturity Model

Polarsys Quality Model



Polarsys Quality Model



"A set of **characteristics**
of a software project
making it capable of producing
software of good quality,
according to certain **criteria**,
in a **sustainable** *way"*

What is software health?

- Criteria to define quality
- Characteristics that allow for that quality
- Timeframe for sustainability

Measuring software health

- Quantify quality criteria
- Find indicators that summarize criteria
- Find values for them that characterize health
- Track their evolution

Example

- Criteria for quality: minimize unfixed errors
- Indicator: unfixed bug reports
- Healthy value: X unfixed bug reports per KLoC
- Alarm when number below X

The causes for health

The really interesting matter
is to know the causes
for variation in indicators
Example: unfixed bug reports are minimized by good code review

The future

On the shoulders of giants

The importance of dependencies

Evolution

Making decisions for tomorrow

Integrating metrics with development

Metrics as a part of continuous integration

Working with stakeholders

- Builders
- Integrators
- Users

Health for different actors
for different purposes

CHAOSS

Understanding dynamics

How do specific actions impact
on the health model
for a software development system?

Towards a new research framework

Define health conditions

Find out how to measure indicators of health

Study deviations from healthy conditions

Learn how to help to go back to healthy

Include all of this in the development process

Simple example

Health condition: no regressions

Indicators: tests failing

Deviations: old errors appear

Mitigation: automatic testing

Continuous integration system

Finding evidence

Evidence that the inditicator shows deviation from healthy condition

Evidence that mitigation::

- condition go back to healthy
- indicator go back to normal

Can we do this in non-trivial cases?

This document (including source) is available from https://jgbarah.github.io/presentaciones