* Discusses how quality evaluation is usually based on hierarchal models that measure various aspects of quality
* They take steps to present a hierarchical quality model that evaluates that evaluates source code and community processes (missing from normal modes)
* OSS is becoming more relevant so proper methodology should be developed to adapt to this
* Mainly metric oriented, and describe metrics as directly measurable attributes of software
* Contrasts itself with other models
  + Criticizes other models like OSMM for even though they are simple to implement, they don’t consider things like source code
  + Criticizes OpenBRR method for even though it includes community, it considers the reference application a weak idea
  + QSOS allows for objective results among users, but lacks flexibility
* SQO-OSS uses continuous monitoring, is focussed on automation (no heavy user interference), focuses on source code, and considers OSS community
* Two phases: definition of the evaluation model, phase two: definition of the aggregation method (data collection)
* Breaks down source code analysis into the hierarchy: analyzability, changeability, stability, testability, maturity, effectiveness, security
* Break community down into: mailing list, documentation, developer base
* Uses weighting from poor to excellent to assess quality, and grades it up the hierarchy (main categories must match)
* Dangerous assumption in stating that all metrics are of equal importance, however simplified
* They themselves mention the limitation of some metrics
* Not mentioning metrics because mentioned in a previous article
* Lack of user interference can be a negative
* Doesn’t consider usability as it requires human intervention
* Makes the point that models themselves have problems
* Makes a good point on why we can’t realistically test functionality and the importance of source code
* Their community metrics seem to be unreliable in my mind, and could be simplified down
* Software too is limited as it’s still in testing phase and will need to verify thresholds
* Similar to traditional software, the quality assessment of closed source software must also be analyzed
* Takes into account product quality as well as process maturity and sustainability of the underlying community
* Mentions the idea that OSS doesn’t necessarily need to be concerned with licensing costs, and allows for greater independence between software vendors; however picking one is risky without assessing and could create a negative impact for an organization
* It’s a comprehensive process involving both robustness and evolvability (very different from each other)
* It criticizes the fact that while many other models generally cover relevant data, they have a very rudimentary perspective and a lack of coverage
* Model is composed of quality characteristics, metrics, and indicators
* Makes the point that metrics correspond to concrete aspects we can measure
* Quality characteristics are organized into two levels of characteristics and subcharacteristics (uses a hierarchy model)
  + Product related: maintainability, reliability, transferability, operability, performance, functional suitability, security, compatibility
  + Community related: maintenance capacity, sustainability, process maturity
  + Summarize each quick
* Makes the point that development communities are the main difference between closed and OSS
* Current assessment processes include: change submissions and review, peer review of changes, propose significant enhancements, report and handle issues with the product, test the program produced by the project, plan releases, release new versions of the product backport corrections in the current release to previous stable releases12

00D

* Common theme of OSS models is the inclusion of community
* Makes a valid point that source code is always available and can be readily analyzed, therefore it should
* Limitation on evolvability, as some of the information for the metrics may not be available readily (example: access contributors email or web addresses
* Must of the community argument is based not on fact, but presumptions and generalizations
* Doesn’t get specific with metrics or processes
* Note: makes point that successful OSS project aren’t rushed or disorganized, and are based on strong communities
* Note: this is another model based in part on the ISO9126 guidelines
* At the time of writing this article the model hadn’t been fully tested (mainly theoretical still)