Open Source Software Development (OSSD)

Open source software (OSS) is software that affords users the independence to practice, learn, and revise the software centered on indigenous requirements and inclinations. This free will is dynamic for advancing the evolution and improvement of Computer Sciences. (Kotwani et al., 2011). OSS has been quite instrumental and facilitated the prospect to realizing Lesser charges; Dependability, performance and confidence; Construct durable dimensions; Open beliefs; Inspire inventions; Substitute to prohibited plagiarism; Prospect of localization; Erudition from source code. Which serve as huge benefits to the software development community and interested concerns alike.

The development process in OSS is admitted to a lifetime progression that combines a supervisory context with task-prone project segments. The prototype encompasses six stages such as characters and duties, classifying work to be completed, allocating and executing improvement work, pre-release analysis, assessments, and handling releases. (Kaur et al., 2011) admitted that, numerous rudimentary dissimilarities exist concerning OSSD and traditional approaches. The System Development Life Cycle (SDLC) of traditional approaches encloses common stages addicted to entire project undertakings that can be structured into planning, analysis, design, implementation and maintenance. Consequently, open source life cycle for open source software development (OSSD) model establishes a number of mutual characteristics such as comparable development and peer assessment, speedy response to user and developer influences, extremely endowed developers, equivalent fixing, user participation, and prompt release intervals. This suggests that an open source project can comprise completely the fundamentals of a traditional SDLC. Evidence from the characteristic OSS projects such as BSD, BIND and SendMail averred that open source
projects make use of typical software engineering processes of analysis, design, implementation and maintenance.

However certain quality issue emerged in the form of dependability of OSSD, though aware that the traditional development approaches are susceptible to failures as admitted by (Lee et al., 2009) such as:

1. Insufficient thoughtful of the scope and difficulty the development projects combined with uncompromising, impractical time bounds and underprivileged budget approximations;
2. Dearth of user participation;
3. Deficits in experienced workforces and
4. Project budgets upsurge by the worth of warranty charges for software and tools essential for application development as well as supplementary expenses for conversation controls.

The foregoing elicited colossal concern about the immunity of OSSD from the consequences arising as a result. (Rugina et al., 2011) affirmed that, in traditional development practices, carrying out dependability assessment alongside with additional analyses at architectural level permits both constructing architectural compromises and forecasting the consequences of architectural verdicts on the dependability of an application. This provokes the knowledge which electrifies emotional interest that draws the leaning from (Mustafiz et al., 2008) that, dependability is that characteristics of a computer system in which confidence can admissibly be engaged on the provision it supplies. Such encompasses sustaining numerous necessities in terms of: availability, reliability, safety, maintainability, confidentiality, and integrity. The dependability constraint differs with the objective application, in as much as a restraint can be indispensable for one situation and not so for others.
OSSD as a consequence of the foregoing is disposed to certain dependability issues such as those which position it for comparative advantage over traditional development approaches as follows:

1. Concerted, corresponding development concerning source code distribution and recycle;
2. Concerted methodology to problem deciphering over continuous response and peer appraisal;
3. Enormous pool of universally circulated, extremely endowed, interested experts;
4. Exceptionally swift release periods;
5. Improved user participation as they are regarded as co-developers;
6. Quality software and
7. Right to use and modify obtainable source code. (Kaur et al., 2011)

However, it is susceptible to the influences of the following shortcoming alike:

1. The user boundaries of OSSD products remain precisely not instinctive;
2. OSSD had no particular source of information and service desk consequently there exist no conclusive responses to difficulties raised or encountered by users and,
3. OSSD is principled on swift releases and characteristically has numerous repetitions compared to profit-making software. This generates an organization difficulty such that fresh release is necessarily required to be effected to achieve or reap complete profits consequently. (Kaur et al., 2011)

In conclusion these listed issues coupled together unleash dependability concerns for the OSSD to deliver software products which are measure of it attributes.
References:


Bibliographies:


