Practices of the KDE ecosystem

Social Practices. For the social view, we grouped practices related to working together in the community.

. S1: Create confidence in a member based on his work history. Based on this, provide different levels of responsibilities.

. S2: Require that at least two persons nominate a member to be promoted up the levels of responsibilities.

. S3: Conduct annual meetings among translators by projects.

. S4: Promote the networking in the work Market.

. S5: Promote the social relationships to members in other countries.

. S6: Create opportunity to practice another human language such as English.

. S7: Conduct a general annual meeting to discuss face-to-face several questions of the community.

. S8: Perform elections to assign responsibility levels for new members.

. S9: Teach experienced members how to use tools to be more productive. For example: use of shortcut keys, scripts, and plug-ins.

. S10: Active members can be given trips to events sponsored by the ecosystem.

. S11: Use tools to support communication in the group such as: mailing lists, forums, IRCs, wikis, blogs to communication.

. S12: Promote Happy Hours and/or dinners to facilitate the social integration during events.

Business Practices. For the business view, we grouped practices related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.

. B1: Provide flexibility in the translation schedule and negotiate deadlines.

. B2: Reach an agreement in the community to face questions that are not registered in the tutorials or guidelines.

. B3: Make decisions based on discussions in the mailing list.

. B4: Provides lectures to teach how to translate and how to become a contributor.

. B5: Provide a nonprofit corporation to manage legal and financial issues.

. B6: Divide the activities into working groups responsible for several areas such as: marketing, infrastructure, design, community in order to keep the group healthy.

. B7: Make technical decisions independently inside each project.

. B8: Attract companies that invest money to support the ecosystem.

. B9: Provide flexibility in the time to work with better code quality.

. B10: Use lessons learned by past mistakes to modify and improve the practice.

. B11: Define a schedule of releases that will affect the work of the entire community.

. B12: Provide a feature plan that will be used for the implementation.

. B13: Provide an accountability report every three months to community.

. B14: Define and support a special group to manage all business issues.

. B15: Define a hierarchical structure for members with different levels of responsibilities.

. B16: Assign a lead maintainer for each project for making technical decisions.

Technical Practices. These practices address issues related to product development (core and applications), technologies used, code rules, among others.

. T1: Each person chooses the work they desire to perform among available tasks in the ecosystem. For example: file to translate, code development, test of applications, and so on.

. T2: Review all code before accepting into the release.

. T3: Do not review translations of artifacts.

. T4: Provide guidelines for how to translate artifacts.

. T5: Provide guidelines for how to create the environment to work with translation and code development.

. T6: Control the translation work separating artifacts into two groups to translate: stable - artifacts have experienced none or few changes and unstable - artifacts are continually being changed and checked constantly.

. T7: Provide different access levels to the KDE repository to check-in artifacts for release.

. T8: Provide specific tools to optimize the translation work. For example: the tool,

called Localize, searches text to translate and save the specific format. It also suggests words to use in translation.

. T9: Provide different kinds of translation environments to work: online and offline.

. T10: Provide a freeze period to stabilize the translation of a version before the launch of a version.

. T11: Provide tools for code optimization, perform static analysis of code, code review, and test automation.

. T12: Provide checklists of tasks to conclude the work. For example: before adding the code to the repository, the developer needs to document, test, integrate with other applications, code must be reviewed by another person and translation must be checked.

. T13: Provide continuous integration tools that check code every day and report errors.

. T14: Require that code must follow a standard defined previously.

. T15: Provide a manifest which the project must follow to be considered part of the ecosystem.

. T16: Require that all infrastructure must be under the control of the ecosystem and be based on the technologies created by the ecosystem.

. T17: Provide a design style to be followed for all applications.

. T18: Conducting incremental transitions rather than versions and releases all at once.

. T19: Use scripts to do an initial code review and catch errors.

. T20: Execute the code with several different versions of operational system and compilers, keeping compatibility and avoiding errors.

. T21: Use scripts to evaluate improper dependencies.

. T22: Each member announces the feature that he will implement to avoid duplicated work.

. T23: Develop code to be extensible. The use of plug-ins and compilation is separated between the core and applications.

. T24: Number versions with 3 numbers indicating what was changed in the version.

. T25: Keep backward compatibility for a long time (around 6 years).

. T26: Provide guidelines informing about actions that are allowed and not allowed to keep backward compatibility.

. T27: Communicate changes in the API to the community in advance.

. T28: Generate code documentation automatically.

. T29: Provide documentation about operation of projects such as wikis, user guidelines, and meetings event log.

. T30: Use discussion mailing lists divided by different topics such as test, development, etc.

. T31: Use a group of members to perform tests in beta version.

. T32: Use a bug triage task force to remove repeated or non-existent bugs.

. T33: Conduct and control refactoring in project as necessary.

. T34: Use technology to reuse GUI components to facilitate the development work.

. T35: Developers handle manually change management.

. T36: Developers and End Users create requirements for core and applications.

. T37: Provide blogs for members where they can post news about new functionalities and other topics of the project.

. T38: Develop two versions in parallel, a old and a new version until that new version is stable and ready to replace the old version

. T39: Different types of contributors help to test code that needs special hardware, operational system or network connection. This happens because normally a developer does not have all devices to test code.

. T40: Mark the code as deprecated in features that will be removed from the software.