Architectural Practices of the KDE Ecosystem

Identity	Architectural Knowledge
AK1	Architect leaders provide some tutorials and/or video courses for the technology available
AK2	Build maps of modules and runtime elements during face-to-face meetings
AK3	Create personal blogs and/or wikis to inform about the development and architectural issues
AK4	During code review, provide feedback information about architecture, good practices to code, doing refactoring and show the best way to solve the problems
AK5	Document APIs daily
AK6	Provide mentoring programs to training new developers
AK7	Keep different architects with different levels of specialty to spread the knowledge in the community
AK8	Provide code recommendations, defining a standard in the community
AK9	Provide a development manual for newcomers to know how to start contributing
AK10	Keep a register of meetings available to community to know all decisions of the meeting

Identity	Choice of Technology
CT1	The organization board defines some technologies that should be used by the whole community as tools for testing, communication, coding review, bugging manager, and navigation
CT2	The project leaders define specific technology that impact into the work of the project community
CT3	The developer chooses tools that impact only in your local work like IDEs and simulators

CT4	The organization board recommends a tool to create UML diagrams of the architecture to be used by KDE projects

Identity	Quality Management
QM1	Define minimal quality criteria requirements to add an application to the ecosystem (documentation, automatized tests, dependence restrictions)
QM2	Projects of the applications define quality criteria in accordance with the organization board recommendations
QM3	Define a team to test performance and behavior of the application
QM4	Use some tools to compute some quality metrics

Identity	External Management
EM1	Keep the backward compatibility for a medium or long time to allow the community update their software
EM2	Provide different levels of security access for the parts of the ecosystems in accordance with the degree of commitment and tasks in the ecosystem
EM3	Use automatized tests to gather problems with the code
EM4	Use tools to publish known cyber security vulnerabilities. For example the Common Vulnerabilities and Exposures (CVE®)
EM5	Keep a team to manager the security problems registered
EM6	Maintainers checks and removes malicious code
EM7	Preference to use the English language written to avoid misunderstanding
EM8	Do online meetings in a time zone adequate for most of the community
EM9	Provide several online meetings to discuss architectural problems by IRC, email
EM10	Create partnership with third-party to solve problems of the core and their interfaces

Identity	Resources Management
RM1	Allocation of responsibilities in accordance with the interest of the developer/architect and discussing with the other members
RM2	The organization board provide hardware and software resources to be used by the community
RM3	Define a financial board to manage the financial resources
RM4	Provide financial resources to support meetings face-to-face

Identity	Design-Making
DM1	The architect/maintainer taken design decisions to guide the developers considering the communication mechanisms (between systems, between your system and external entities, between elements of your system)
DM2	Application architects should follow decisions of the core architects and add their decisions in agreement with their recommendations
DM3	Define recommendations to reuse all resources available and tested by community
DM4	Provide meetings (sprints) face-to-face to accelerate the development of critical issues
DM5	Provide autonomy to work on the architecture in accordance with the degree of commitment and experience of work in the community
DM6	Create a demand register to the community know what are the next steps

Identity	Change Management
CM1	Divide the parts of the software in layers defining restrictions for managing dependencies among the layers

CM2	Discuss with the community about critical changes into architecture that will impact in the applications
CM3	Publish widely the architectural changes for the community
CM4	Only architects leaders take decision about remove obsolete code, respecting the backward compatibility
CM5	Provide specifics face-to-face meetings with teams of different applications to solve development problems with their interdependent modules
CM6	Build the architecture based in plug-ins to facilitate the coupling of applications