Status of Frameworks 5 & Plasma 2

KDE's Next Generation user interfaces will run on top of Qt5, on Linux, they will run atop Wayland or Xorg as display server. The user interfaces move away from widget-based X11 rendering to OpenGL. Monolithic libraries are being split up, interdependencies removed and portability and dependencies cut by stronger modularization.

For users, this means higher quality graphics, more organic user interfaces and availability of applications on a wider range of devices.
Developers will find an extensive archive of high-quality, libraries and solutions on top of Qt. Complex problems and a high-level of integration between apps and the workspace allow easy creation of portable, high-quality applications.

The projects to achieve this goal are KDE Frameworks 5 and Plasma 2.

Status Frameworks 5

Development of KDE's Frameworks5, which focuses on modularization of APIs currently contained in kdelibs and kde-runtime, loosening its internal structure and making it possible to only use specific parts by splitting it into individual libraries and solutions.

The entire work to be done for Frameworks 5.0 is split into 7 topics[1]. Three of these "Epics" are done:

- Initial communication and documentation (Kevin Ottens),
- Merging of code into Qt 5.0 (David Faure)
- Reduction of duplication with Qt by removing classes and using their Qt alternatives (Stephen Kelly)

Four Epics are currently work in progress, three of them are monstrous:

- Build system[2] (Alex Neundorf, Stephen Kelly)
  - CMake (upstreaming some stuff, modularization, porting)
  - Modularization of CMake KDE settings (work in progress)
  - Modularization of macros
  - Review and inventarize Find* CMake modules
- kdelibs cleanups[3] (David Faure)
  This is a large Epic, containing many bite-sized tasks. Roughly 50% of them are done, 37 tasks remain open and 7 are being worked on, an extensive list is on the wiki.
- Qt 5.1 merging (David Faure)
  This is the list of things that we haven't been able to merge upstream into Qt 5.0, so we hope we can upstream as much as possible into Qt 5.1. This can potentially cause timing problems, if we can't get all the necessary things we need into Qt 5.1. 9 tasks are work in progress by David Faure, Thiago Maciera, Richard Moore and others. 52 tasks are on the todo list, most of them currently unclaimed.
- Splitting kdelibs (blocked)[4] (Kevin Ottens)
  Another large Epic, in bigger chunks, meaning going through all libraries one by one,

1 http://community.kde.org/Frameworks/Epics
2 http://community.kde.org/Frameworks/Epics/CMake
3 http://community.kde.org/Frameworks/Epics/kdelibs_cleanups
4 http://community.kde.org/Frameworks/Epics/Splitting_kdelibs

porting their build system to the changes in Frameworks5, cut out certain library dependencies and changing the translation system. 13 tasks are done, 12 work in progress and 8 on the todo list, not all of them assigned.
An extensive list of libraries and their status can be found on the wiki.

Frameworks 5 currently compiles on top of Qt 5.0 and basic system services run (kdeinit5), although not all of its dependencies have been ported to Qt 5. Work on Frameworks5 is ongoing, so it is currently quite a moving target, and will remain so for a while.

Plasma and KWin

An architecture based on Qt5 and Wayland makes it possible to use a more modern graphics stack, which means moving from X11-based rendering to OpenGL graphics rendering. QtQuick2 (which is the QtQuick shipped with Qt5) makes it possible to offer a very nice and extensible development API, while using the full power of the graphics hardware to produce excellent visual possibilites. Plasma offers development APIs that make it easy to create well-integrated applications as well as workspaces that are flexible, extensible and fully featured on top of QtQuick, and in the future QtQuick2.

As KDE moves forward towards Frameworks5, Plasma is taking the opportunity of the source and binary compatibility break of Qt5 to do necessary updates to its architecture. The goal is to have a leaner Plasma Development API and depdendency chain and achieve a better user- and developer experience by moving the UI fully to Plasma Quick, which is QtQuick plus a number of integration components for theming, compositor interaction, internationalization, data access and sharing, configuration, hardware, etc..

This constitutes a major refactoring of the Plasma libraries and components. First, their UI needs to be done in QML. This effort of porting workspace components to QML is already well underway.
Second, the Plasma library and runtime components need to be ported from the QGraphicsView-based canvas to QML. This means cutting out dependencies on classes such as QGraphicsItem and QGraphicsWidget to their equivalent in QML. In the case of painting and layouting code, it means porting this code to QML.

Detailed Status Plasma Framework

The first bigger set of tasks is done:

- API changes have been made in libplasma2 (done)
- De-QGraphicsViewification of libplasma2 (done)

The open tasks are:

- Remaining libplasma2 design tasks[5]
  - Out-of-process dataengines
  - Dataengines as models
  - More fine-grained data retention in dataengines
  - Improved remote widgets
  - Change libplasma2 directory structure to reflect Frameworks5 policies[6]

5 http://community.kde.org/Plasma/libplasma2
6 http://community.kde.org/Frameworks/Policies#Framework_directory_structure

- Create a scenegraph-based shell, make it load a QtQuick2Corona, Containments and Applets
- Port QML Scriptengine to QtQuick2
- Port scriptengine from QScriptEngine to QDeclarativeEngine
- Remove dependency on graphics backend (QGraphicsView, or scenegraph) (Marco Martin)
- Port imports to QtQuick2
  - Plasma Core (containing Theme, FrameSvg, DataSource, etc.)
  - Plasma Components (containing a basic QtQuick widget set)
  - QtExtras (containing components missing in Qt, such as MouseEventListener)
  - PlasmaExtras (containing additional UI widgets for better integration, such as animations, text layout helpers, Share-like-connect integration, etc.)
- Making scriptengines (such as the Python scriptengine) only export QObject-deriven classes to the QML runtime

Plans for KWin Plasma Compositor

Plasma Compositor refers, in a *Wayland world,* to the compositor used for Plasma workspaces, which is essentially Kwin in disguise as Wayland compositor.
In KWin, we benefit from an ongoing effort to modularize and clean it up architecturally. For most of its UI, KWin already supports QML (Window decorations, tabswitcher, etc.). Some mechanisms which currenty work through XAtoms will need to be ported, the API impact of that will likely be quite limited for application developers.
The strategy for KWin is to port KWin to Qt 5, then make it possible to run KWin outside of an X server on top of KMS, using the graphics hardware more directly. The next step is to use KWin as compositor for Wayland display servers. The dependency of X11 can be removed once it is not needed anymore to provide compatibility with X11 applications, or can possibly be made optional.
Milestones for KWin (Martin Graesslin):

1. Kwin on Qt5 (work in progress, planned for 4.11)
2. on top of KMS (planned for 4.11)
3. Kwin as Wayland compositor (planned for 4.11 or 4.12)
4. no X11 dependency (planned for the distant future)

Plasma Workspaces

The porting strategy of the workspaces is to port plasmoids and containments to QML, in order to make them ready to run on top of the new infrastructure. In the case of C++ and Python, Ruby, JavaScript and "Web API" applets, it means rewriting them in QML. For portability and maintainability reasons, pure QML widgets are preferred. For some complex use cases, that can not be easily done in QML, we ship a combined C++ QML applet. For Plasma2, the only way to do the UI is using QML. QGraphicsWidget based Uis will not be supported anymore.[7]

Once we have a working libplasma2 and a useful set of QML Plasmoids, we can think of running an entire workspace in QML and on top of QtQuick2, either on top of X11, or with Kwin's plans in mind, on Wayland.

7 http://vizzzion.org/blog/2012/09/randa-meetings-2012-the-future-of-qgraphicsview-in-plasma/

Porting status of important widgets to QML / Plasma Quick needed for the workspace.[8]

- Taskbar (close to first review, target: 4.11) (Eike Hein)
- Folderview (work in progress) (Ignat Semenov)
- Desktop containment (second revision close to review, target: 4.11) (Sebastian Kügler)
- Calendar (work in progress, target: 4.11) (Sebastian Kügler)
- Kickoff (about to be merged into master, target: 4.11)
- KRunner (work in progress) Aaron Seigo, Aleix Pol
- Done: System tray, pager, notifications, device notifier, battery, lock/logout, weather, Wallpaper (Aleix Pol), Containment support (Sebastian Kügler)
- Open:
    - Digital Clock
    - Icon
    - Picture Frame
    - others from kdeplasma-addons
    - and more (see wiki)

8 http://community.kde.org/Plasma/QMLPorting