ABSTRACT

MeeGo is a Linux-based open source mobile operating system project. Primarily targeted at mobile devices and information appliances in the consumer electronics market, MeeGo is designed to act as an operating system for hardware platforms such as netbooks, entry-level desktops, nettops, tablet computers, mobile computing and communications devices, in-vehicle infotainment devices, SmartTV / ConnectedTV, IPTV-boxes, smart phones, and other embedded systems. MeeGo is hosted by the Linux Foundation. This open source project brings together the Moblin project, headed up by Intel, and Maemo, by Nokia, into a single open source activity. MeeGo integrates the experience and skills of two significant development ecosystems, versed in communications and computing technologies. The MeeGo project believes these two pillars form the technical foundations for next generation platforms and usages in the mobile and device platforms space.

TABLE OF CONTENTS

ABSTRACT 01

LIST OF FIGURES 02

Chapter 1 INTRODUCTION 04

1.1 MeeGo and Moblin 04

1.2 Intel's MeeGo strategy 05

Chapter 2 MEEGO SOFTWARE ARCHITECTURE 06

4.1 MeeGo Architecture Layer View 06

4.2 MeeGo Architecture Domain View 08

4.3 MeeGo Architecture API View 14

Chapter 3 BENEFITS OF MEEGO PLATFORM 15

CONCLUSION 18

REFERENCES 19

LIST OF FIGURES

Fig No.	Figure Description	Page No.
2.1	MeeGo Architecture Layer View	06
2.2	MeeGo Architecture Domain View	08
2.3	MeeGo Architecture API View	14

1. INTRODUCTION

MeeGo is a Linux-based OS built for the next-generation of computing devices. Different from other mobile OSes, MeeGo is a truly open source platform that includes the core OS, UI libraries and tools, references user experiences for multiples devices and applications, a standard set of APIs across all target device types, and the flexibility to support proprietary add-ons. MeeGo supports a magnitude of mobile client devices and provides choice and flexibility to create and deliver a uniquely differentiated service offering. MeeGo currently targets platforms such as netbooks/entry-level desktops, handheld computing and communications devices, in-vehicle infotainment devices, connected TVs, and media phones. All of these platforms have common user requirements in communications, application, and internet services in a portable or small form factor. The MeeGo project was announced on February 15, 2010 and since its announcement theproject has had multiple releases and progressed significantly.

MeeGo includes:

- Performance optimizations and features which enable rich computational and graphically oriented applications and connected services development
- No-compromise internet standards support delivering the best web experiences
- Easy to use, flexible and powerful UI/app development environment based on Qt
- Open source project organization managed by the Linux Foundation
- State of the Art Linux stack optimized for the size and capabilities of small footprint platforms and mobile devices, but delivering broad linux software application compatibility

1.1 MAEMO AND MOBLIN

The Maemo project, initially created by Nokia, provided a Linux-based software stack that runs on mobile devices. The Maemo platform is built in large parts of open source components and its SDK provides an open development environment for applications on top of the Maemo platform.
A series of Nokia Internet Tablets with touch screen have been built with the Maemo platform. The latest Maemo device was the Nokia N900, powered by Maemo 5, that introduced a completely redesigned finger-touch UI, cellular phone feature, and live multicasting on the Maemo dashboard.

The Moblin project, short for Mobile Linux, is Intel's open source initiative created to develop software for smartphones, netbooks, mobile internet devices (MIDs), IVI systems, and other mobile devices. It is an optimized Linux-based platform for small computing devices. It runs on Intel Atom, an inexpensive chip with low power requirements. A unique characteristic to devices running Moblin is that they can boot up quickly and can be online within a few seconds.

1.2 INTEL'S MEEGO STRATEGY

Intel's overall software strategy for the Intel Atom processors is known as "architecture of choice." This means that Intel works to ensure that all possible software solutions run best on Intel processors. Intel executes this strategy by implementing Intel Atom processors on the best process technology in the world, by designing CPU features specifically with underlying software requirements in mind (silicon/software co-design), and by working with software suppliers to optimize their products to run best on Intel Atom processors. As Intel developed the Atom processor family, it became clear that the fragmentation of Linux was hindering OEMs and service providers in their efforts to offer solutions that span the many consumer electronics segments they now want to serve. The MeeGo OS was created under the auspices of the Linux Foundation to address the fragmentation that dilutes the efforts of application developers. MeeGo provides a common set of APIs across device segments, with compliance tests and segment-specific profiles to ensure that an application written for MeeGo works on multiple implementations within and between device segments.

2. MEEGO SOFTWARE ARCHITECTURE

The MeeGo platform has been carefully created to provide the components necessary for the best device user experience. The MeeGo platform architecture can be viewed in three different ways:

- **Layer view** shows the separation of different layers and user experience (UX) verticals
- **Domain view** shows the grouping of subsystems into architecture domains, based on similarities in technology and functionality
- **API view** shows the grouping of MeeGo API into functional areas

2.1 MeeGo Architecture Layer View

The Layer view consists of three layers: User Experience, Application API, Core OS layers

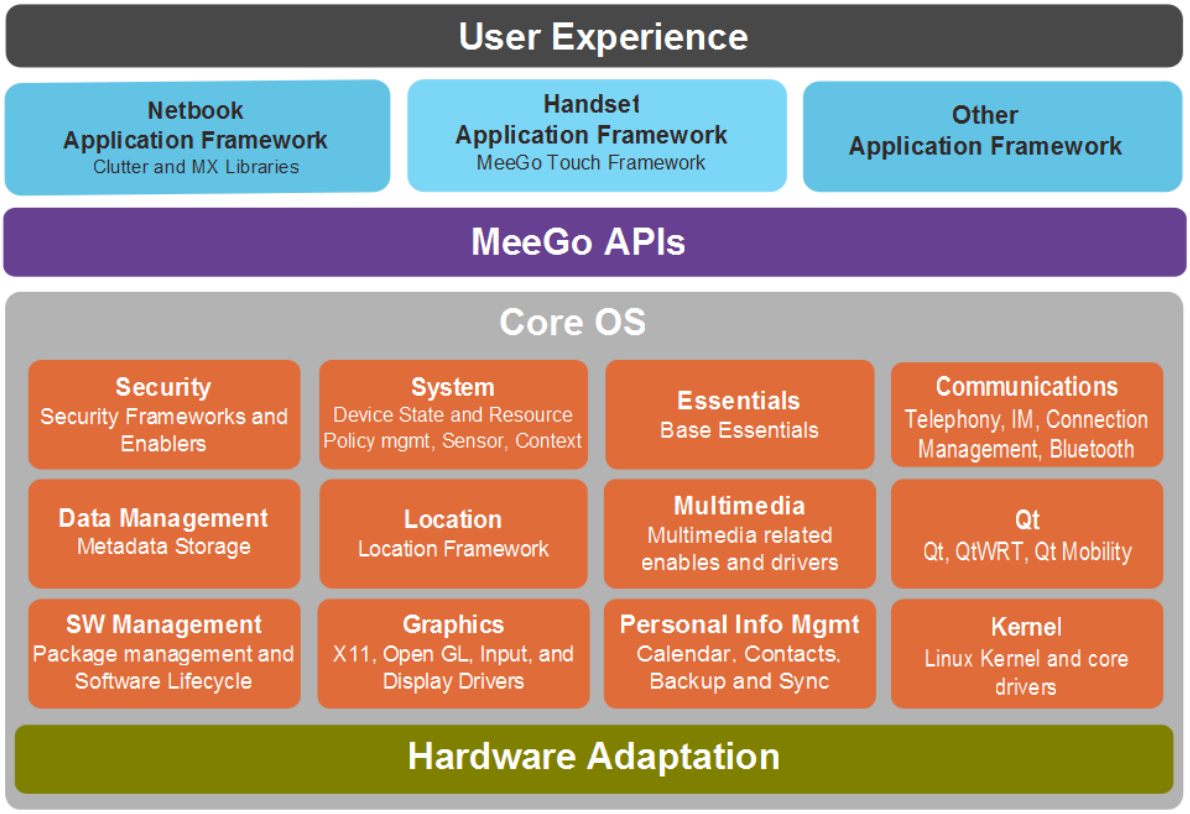

Fig 2.1

The user experience layer contains the UX verticals. It provides reference user experiences for multiple platform segments. MeeGo 1.1 contains reference user experiences for handhelds and netbooks. The User experience layer provides the Application Framework for each device profile. Netbook UX uses Clutter and MX Libraries. The Handset UX uses MeeGo Touch Framework with haptics, gestures, and input methods.

Application API layer contains the MeeGo API. It provides the interface for application development. The released versions consists of Qt 4.7, Qt mobility 1.0, Open GL ES 1.1, and Open GL ES 2.0.

Core OS layer contains all the middleware/OS service domains and the hardware adaptation services. It includes the Linux kernel and all the middleware needed to define hardware and usage model independent API for building both native applications and web run time applications. The Hardware Adaptation API is for adapting MeeGo to support various hardware architectures.

MeeGo Core OS architecture is grouped into domains based on functionality in that area:

- **Security** - Security framework and enablers
- **Data Management** - Meta-data storage
- **Software Management** - Package Management and software lifecycle
- **System** - Device State and Resource Policy Managmeent, Sensor, Context
- **Location** - Location Framework
- **Graphics** - X11, OpenGL, input and Display drivers
- **Essentials** - System essential libraries
- **Multimedia** - Multimedia related enablers and drivers
- **Personal Information Management** - Calendar, Contacts, Backup, and Sync
- **Communication** - VOIP, IM, Presence, Cellular Telephony, and IP Connectivity
- **Qt** - Qt, QtWRT, Qt Mobility
- **Kernel** - Linux Kernel and core drivers

Hardware Adaptation Software

There are multiple software components that a hardware vendor must provide for MeeGo to run successfully on their platform architecture, including platform kernel drivers, core architecture additions, kernel configuration, X software additions and configuration, modem support, and hardware specific media components. These specific software components are called the hardware adaptation software .

The MeeGo Core OS defines interfaces for platform dependent hardware. It's the responsibility of a chipset's hardware adaptation software to implement these interfaces. The hardware adaptation software is divided into to the following *adaptation subsystems*: security, sensor, device mode, haptics and vibra, audio, camera, imaging and video, location, cellular, connectivity, input, display and graphics.

2.2 MEEGO ARCHITECTURE DOMAIN VIEW

The Domain view expands each domain and details the subsystems required to provide that functionality.

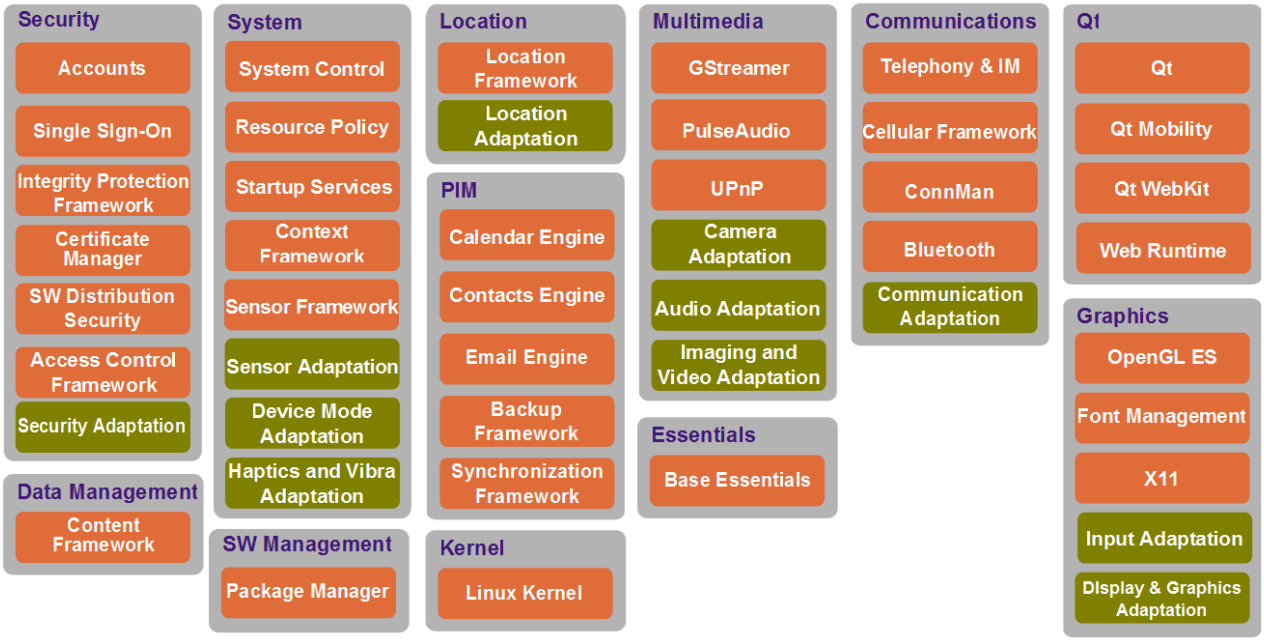

Fig. 2.2

Security

Security domain is responsible of security deployment across the system. It provides enablers for platform security and user identity.

- **Accounts** - Provides a storage solution for user accounts. Applications which need to store and access user settings for the service they provide over a user account will use the Accounts API. Instant messaging, e-mail, calendar, and sharing are examples of such applications.
- **Single Sign-On** - Responsible for providing secure storage for credentials and framework for authentication plugins to different services
- **Integrity Protection Framework** - Integrity protection of executables, configuration, and data files.
- **Certfificate Manager** - Services for storing and validation of security certificates for various purposes (such as email, wifi, and browsing).
- **Software Distribution Security** - Security aspects of software distribution including new application installations and updates.

- **Access Control Framework** - Access control enforcement and access control policy for the device
- **Security Adaptation** - Platform specific abstraction of security and crypto services.

Data Management

Data Management domain provides services for extracting and managing file meta-data (for example to support extracting and searching metadata for media files). The Data Management domain includes the following subsystem:

- **Content Framework** - Tracker provides indexing, meta-data extraction, and search capabilities for a variety of data types, including media files, and documents.

Software Management

Software Management domain is responsible for package manager and its backend functionality.

- **Package Manager** – Package Kit uses distribution package management tools to make installing and updating software on devices easier. It is a system activated daemon meaning that it is only run when the user is using the tools, and quits when it is no longer used.

System

System domain is responsible for device state/mode handling, time management, policy control, startup services, and sensor abstraction.

- **System Control** - Device state and time management
- **Resource Policy** - Plugin based framework for audio, video, and system policy management.
- **Startup Services** - Components related to system startup.
- **Context Framework** - High level API to numerous context properties of the device.
- **Sensor Framework** - Provides an interface to hardware sensors through logical sensors.
- **Sensor Adaptation** - Sensor specific plugins for sensor framework

- **Device Mode Adaptation** - Hardware abstraction layer for device mode related information (such as watchdogs, temperature sensors)
- **Haptics and Vibra Adaptation** - Hardware abstraction layer for vibra and haptics devices

Location

Location domain provides location services.

- **Location Framework** - GeoClue provides location data combined from number of sources, such as GPS, GSM cell, or wifi network.
- **Location Adaptation** - Hardware abstraction layer for location source devices such as GPS

Kernel

Kernel domain contains Linux kernel and device drivers.

- **Linux Kernel** - Linux kernel 2.6.35 or newer.

Personal Information Management

Personal Information Management domain enables managing user data on the device, including managing calendar, contacts, tasks, and retrieving data about the device context (such as device position, cable status). The domain includes the following subsystems:

- **Calendar Engine** - Calendar engine provides an interface for accessing calendar data.
- **Contacts Engine** - Contacts engine provides an interface for accessing contact data.
- **Email Engine** - Email engine provides an interface for accessing emails.
- **Syncronization Framework** - Synchronizing calendar, email, and contacts data between different devices via various transport layers like USB and Bluetooth

Multimedia

Multimedia domain provides audio and video playback, streaming, and imaging functionality to the system. In general, the domain takes care of the actual audio and video data handling

(retrieval, demuxing, decoding and encoding, seeking, etc.). The domain includes the following subsystems:

- **Imaging and Video Adaptation** - Platform specific codecs and containers for GStreamer
- **Camera Adaptation** - Platform specific codecs and containers for GStreamer. Adaptation interface is CameraBin.
- **UPnP** - Universal Plug and Play provides a UPnP stack, the UPnP profile for audio and video.
- **Gstreamer** - GStreamer, through its plugins, provides playback, streaming, and imaging functionality to the system.
- **Audio Adaptation** - Platform specific modules for PulseAudio
- **Pulse Audio** - The audio subsystem handles audio inputs, post and pre processing, and outputs in a system. The purpose is to provide a proxy between audio applications and audio hardware.

Essentials

Essentials domain provides all system essential packages.

- **Base Essentials** - Fundamental system tools and libraries

Communications

Communications domain provides Cellular and IP Telephony, Instant Messaging, Presence, Bluetooth, and Internet Connectivity services.

- **IP Telephony, Instant Messaging and Presence** - Telepathy is a modular communications framework that enables real-time communication via pluggable protocol backends.
- **Cellular Framework** - oFono provides cellular telephony stack and services in MeeGo. Plugin based architecture supports multiple platforms and modems.
- **ConnMan** - Connection Manager provides services for managing internet connections.
- **Bluetooth** - The Bluetooth subsystem consists of the Linux Bluetooth stack BlueZ, as well as related extensions.

- **Communication Adaptation** - Platform specific modules for WiFi and Bluetooth devices and oFono plugins for different platforms and modems.

Qt

Qt domain contains cross platform toolkits such as Qt, Qt Mobility, Qt WebKit, and Qt WebRuntime.

- **Qt** - Qt application and UI toolkit.
- **Qt Mobility** - Qt Mobility APIs for MeeGo.
- **Qt Webkit** - MeeGo provides Qt Webkit as a layout engine. It renders web content (HTML, XML, XHTML, SVG, CSS, JavaScript, etc.) for on-screen display within applications.
- **Web Runtime** - Provides an execution environment for Web Widgets and extends the standard JavaScript environment with device-specific APIs providing access to other subsystems.

Graphics

Graphics domain enables the core 2D and 3D graphics capabilities for the platform, including support for rendering internationalized text and taking advantage of underlying hardware platform acceleration for graphics. The Graphics domain includes the following subsystems:

- **Font Management** - Service to locate fonts within the system and select them according to requirements specified by applications
- **Input Adaptation** - Input adaptation abstracts the hardware behind drivers and exposes an input event interface for user space. Hardware buttons, qwerty keyboard, and touch screen are provided as input devices. Typical HW buttons are: power button, camera, volume up, and volume down.
- **X11** - Implementation of the X11 Window system with architecture specific drivers, patches and configuration.
- **OpenGL ES** - Provides Khronos interfaces and implementation of OpenGL, OpenGLES and EGL. Includes also platform specific implementation of GL/ES driver and libraries.
- **Display and Graphics Adaptation** - Framebuffer and display panel related platform specific abstraction.

2.3 MEEGO ARCHITECTURE API VIEW

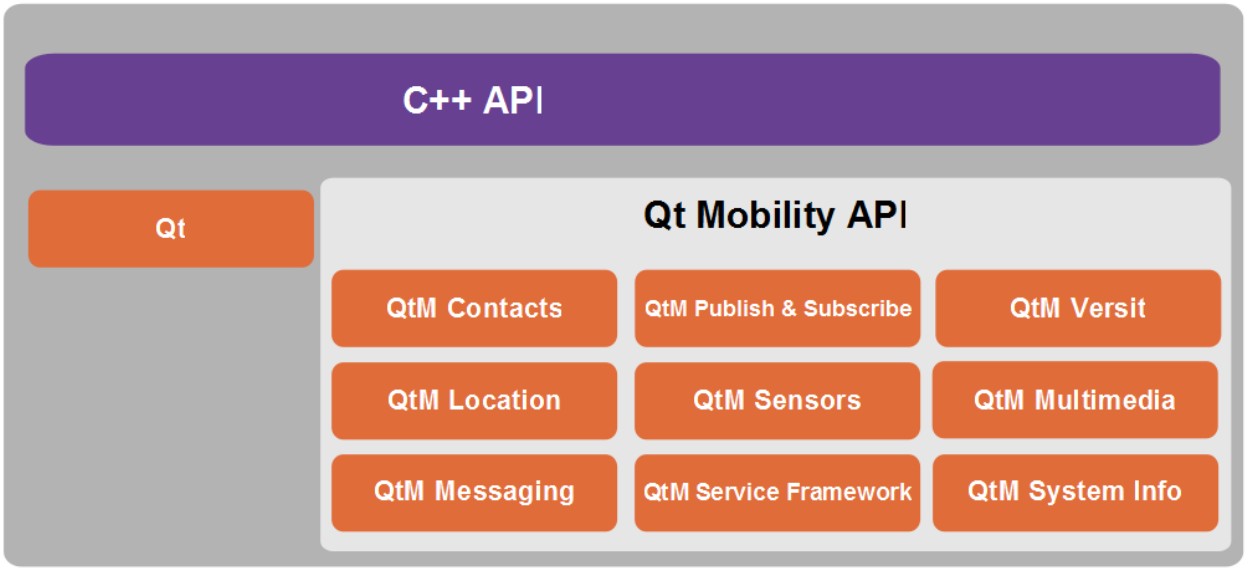

Fig 2.3

MeeGo API is based on Qt and Qt Mobility.

Qt: Qt provides application developers with the functionality to build applications with state-of-the-art graphical user interfaces. Qt is fully object-oriented, easily extensible, and allows true component programming.

Qt mobility: This delivers a set of APIs to Qt, with features that are well known from the mobile device world. However, these APIs allow the developer to, with ease, use features from one framework and apply them to phones, netbooks, and non-mobile personal computers.

3. BENEFITS OF MEEGO PLATFORM

The MeeGo open source project is unique in that it offers benefits to everyone in the ecosystem starting from the developer all the way up to the operator and the industry as a whole. Meego allows participants to get involved and contribute to an industry-wide evolution towards richer devices, to rapidly address opportunities and to focus on differentiation in their target markets.

Benefits to Open Source Developers

The MeeGo project is a true open source project hosted by the Linux Foundation and governed by best practices of open source development. From meego.com, as an open source developer, you have access to tools, mailing lists, discussion forum, accessibility to technical meetings, and multiple options to make your voice heard over technical and non-technical MeeGo related topics. Furthermore, all source code contributions needed for MeeGo will be submitted to the upstream open source projects from which MeeGo will be built.

Benefits to Application Developers

As an application developer, MeeGo significantly expands the market opportunities for you being the only open source software platform that supports deployments across many computing device types. MeeGo offers Qt and Web runtime for application development, cross platform environments, so application developers can write their applications once and deploy easily on many types of MeeGo devices or even on other platforms supporting the same development environment.

Furthermore, MeeGo offers a complete set of tools for developers to create easily and rapidly a variety of innovative applications. The major advantage from this approach is having a single set of APIs across client devices. In addition, in this context multiple devices is much broader than just multiple handset for instance; MeeGo device types include media phones, handhelds, IVI systems, connected-TVs and netbooks. In addition, MeeGo application developers will the opportunity to make their applications available from multiple application stores such as the Nokia's Ovi Store and the Intel's AppUp Center. In addition, there is the opportunity of hosting the applications on other app stores for specific carriers carrying MeeGo devices as part of their device offering. These MeeGo capabilities, cross-device and cross-platform development, are major differentiator and offer huge benefits to the developers.

Benefits to Device Manufacturers

MeeGo helps accelerate time to market using an off-the-shelf, open source and optimized software stack targeted for the specific hardware architecture the device manufacturer is supporting. From a device manufacturer perspective, MeeGo lowers complexities involved in targeting multiple device segments by allowing the use of the same software platform for different client devices. In addition, as an open source project, MeeGo enables device manufacturers to participate in the evolution of the software platform and build their own assets for it through the open development model.

Benefits to Operators

For operators, MeeGo enables differentiation through user interface customization. Although many devices can be running the same base software platform, they can all have different user experiences. Furthermore, it provides a single platform for multitude of devices, minimizing the efforts needed by the operators in training their teams and allows their subscribers to be familiar with the experience common to many device types.

Benefits to the Linux Platform

For operators, MeeGo enables differentiation through user interface customization. Although many devices can be running the same base software platform, they can all have different user experiences. Furthermore, it provides a single platform for multitude of devices, minimizing the efforts needed by the operators in training their teams and allows their subscribers to be familiar with the experience common to many device types.

In addition, MeeGo is helpful for Linux as a platform as it combines mobile development resources that were recently split in the Maemo and Moblin projects into one well-supported, well-designed project that addresses cross-platform, cross-device and cross-architecture development. One major benefit from the MeeGo project is that all other Linux mobile and desktop efforts that use the components as MeeGo will benefit from the increased engineering efforts on those components. This is the power of the open source development model.

CONCLUSION

MeeGo is an open source project developed under the auspices of the Linux Foundation. MeeGo is intended to run on a variety of hardware platforms including handhelds, in-car devices, netbooks and televisions. All platforms share the MeeGo core, with different "User Experience" ("UX") layers for each type of device. Since it was announced in February 2010, the world has been able to both watch and participate as the project builds up and delivers the core software stack, in addition to three reference user experiences for handsets, IVI systems, and netbooks, with more to come as MeeGo also targets connected TVs and tablets. MeeGo is challenging Android as the Smartphone Linux Standard.

REFERENCES

- www.meego.com
- www.moblin.org
- www.maemo.org