Advanced messaging with mBuddy - app showcase

Messaging Buddy (mBuddy) is integrated solution for managed text messaging (sms) that supports message and recipient groups, complex schedules and sending jobs. This article explains the main elements of mBuddy design, and the differences between Symbian and MeeGo implementation.

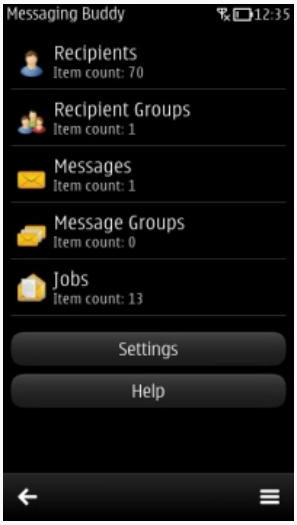
Introduction

mBuddy is integrated solution for managed text messaging (sms) that supports message and recipient groups, complex schedules and sending jobs.
The system is comprised of two major components:

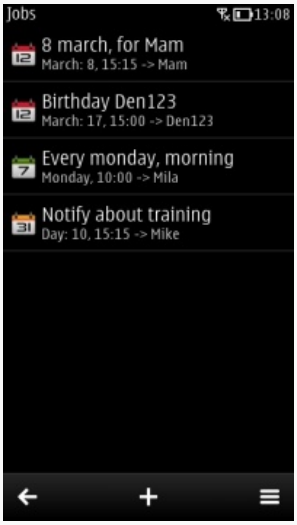
- **GUI** - front-end for convenient sending jobs configuration. UI is based on Qt Quick Components 1.1
- **Server** - core application engine that performs all the messaging. The server is a background process, based on QtMobility.

Main application as a front-end does not need to be kept open all the time and could be closed at any point, while engine part keeps running in the background independently. Originally developed for MeeGo OS it has got several new features when ported to Symbian.

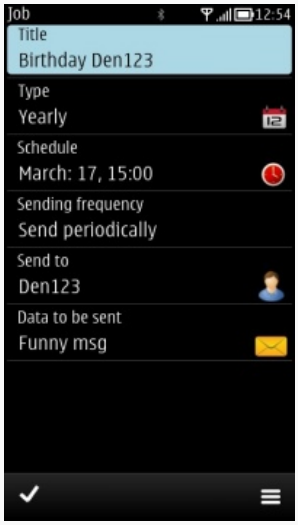
Screenshots:



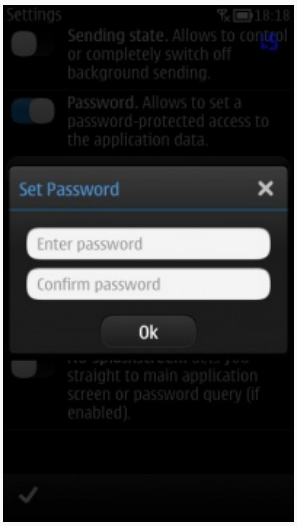
Main screen



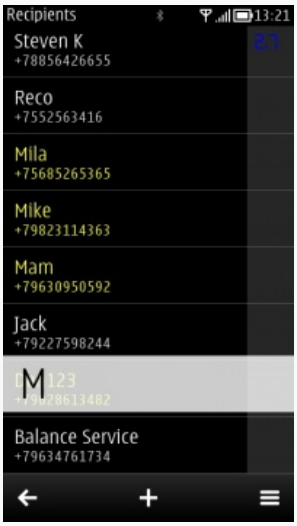
Job list



Job editor



Password change dialog



List scrolling

UI design transformations in Symbian port

(C) Copyright Nokia 2013. All rights reserved.

Main application screen contains links to the following objects:

- recipients
- recipient groups
- messages
- message groups
- jobs

Instead of grid in MeeGo version Symbian app utilizes list (most commonly used GUI element in Symbian), that allows to use screen space more efficiently, providing additional buttons for quick access to app settings and help on the main screen.

MeeGo version uses own QML **TopBar** component to display information about current object. That components shows object title, icon and sorting order change button (only in lists). To meet Nokia Belle look&feel requirements Symbian version makes no use of `TopBar` at all. Object title's moved to `StatusBar`, while additional menu item added to change sorting order.

MeeGo version also uses **TopBar** for "Ok" and "Cancel" buttons in modal editors. Symbian style does not allow such a design in UI. Now it uses a tick mark in the bottom left corner of `ToolBar` to save changes and return to previous screen. This change comes with constraint though: no way to discard changes made to the object.

List of objects allows user to find out which list item is used in mailing (one or more jobs linked to this this item). Each element in list data models has special attribute "inUse" and that attribute is used in list delegates:

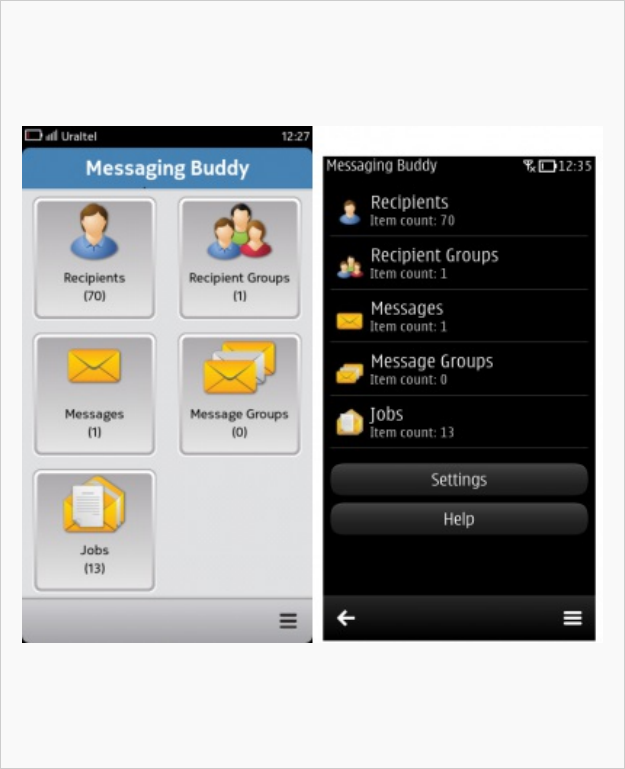
- MeeGo version displays special icon on the left of the list item whenever "inUse" property is true
- Belle app highlights such list items with yellow color

Multiple selection is also implemented slightly differently. In Belle application selection mark moved to the left side, leaving the other side to the list scroller.

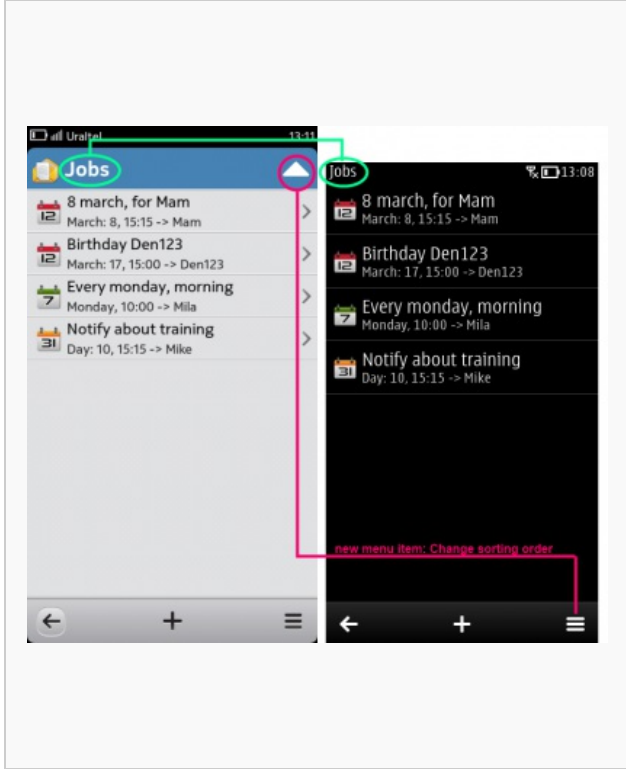
The look&feel of **modal dialogs** used in MeeGo and Nokia Belle varies considerably. Symbian version uses less modal dialog boxes overall and every dialog box has "OK" button only, while "Cancel" button remains unclaimed (dialog Close button is used instead). The source code of several dialog components:

- QML Change Password Dialog
- QML Day of Week & Time Dialog

UI design transformations:

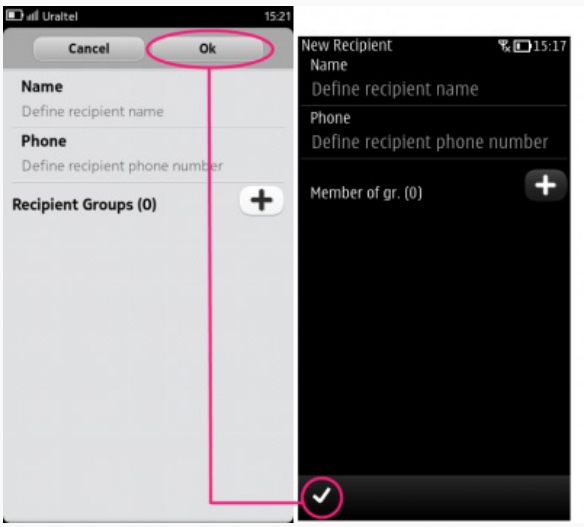


Main screen

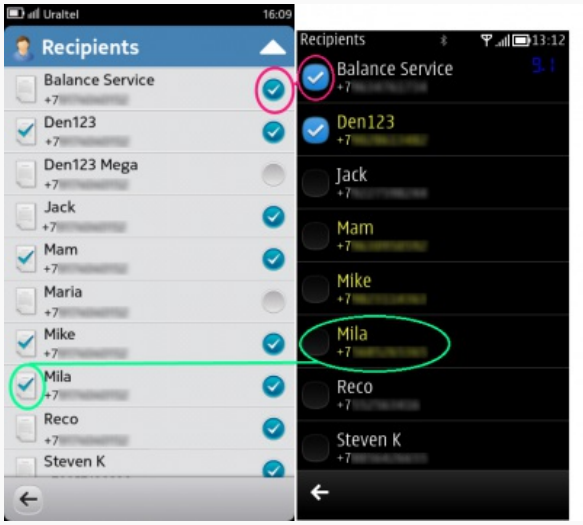


TopBar

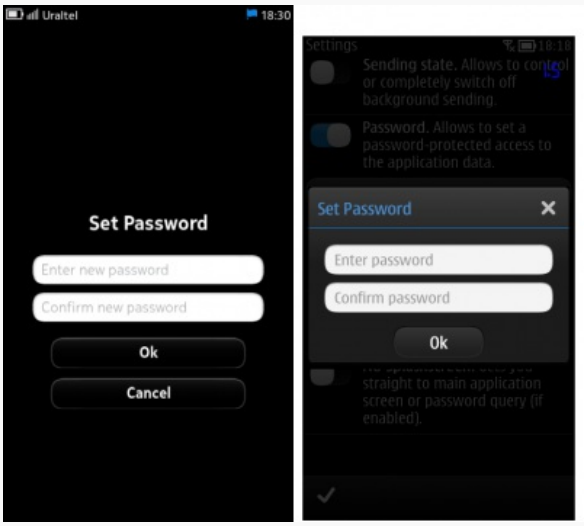
(C) Copyright Nokia 2013. All rights reserved.



TopBar in Editor



Multiselection list



Password change dialog

Complex components and advanced UI behaviour

mBuddy uses own custom QML component **MyScrollBar** for quick navigation in lists. The wide semitransparent scrollbar (MeeGo style) demonstrates the ability to drag the scroller. This element is only visible while scrolling the list, it contains section scroller which allows to fast find necessary items. Any change of data model (adding, deleting, renaming items or change sorting) requires a new initialization of MyScrollBar.

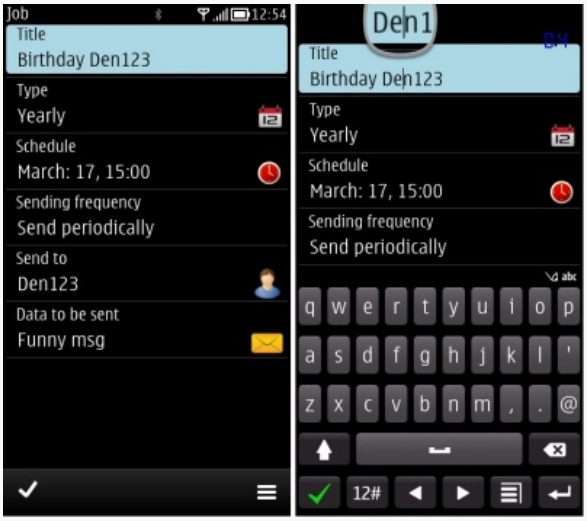
Developer must take into account the availability of free space on top of the field for comfortable input with the virtual keyboard. That space is needed to show magnifying glass when user moves the cursor within the text. To meet this requirement mBuddy performs following actions when virtual keyboard appears on the screen:

- hides `StatusBar` from the screen
- moves input field slightly down

These actions are performed with short animation that improves perceived user experience. When virtual keyboard leaves the screen, `StatusBar` comes

(C) Copyright Nokia 2013. All rights reserved.

back to its place and the top margin disappears.



Challenges

Text input fields might get covered by virtual keyboard when it appears on screen, so developer must consider this possibility designing UI for Symbian app. To effectively address this issue, developer should use `Flickable` component (to change the size of visible area) and make a good use of `inputContext.visible` property that shows whether virtual keyboard is visible or not. It is worth mentioning that MeeGo does not have such a problem at all.

Standard QML Symbian Components (such as `TextField`, `TextArea`) do not allow to change appearance of descendants. Corresponding MeeGo components are more flexible: you can override `TextFieldStyle` (`TextAreaStyle`) property in descendant and completely change component appearance. To achieve similar results on Symbian would take a lot more efforts.

Links

- mBuddy for MeeGo
- mBuddy for Symbian

Note: This is an entry in the Symbian Qt Quick Components Competition 2012Q1

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.