Mobile Widget Technology as a Solution for Smart User Interaction

Miroslav Behan and Ondrej Krejcar

University of Hradec Kralove, FIM, Department of Information Technologies,
Rokitanskeho 62,Hradec Kralove, 500 03, Czech Republic
Miroslav.Behan@uhk.cz, Ondrej.Krejcar@remoteworld.net

Abstract. Widget technology will increase its potential in time due to visualization comprehensiveness, fast content reachability and easy event driven possibilities. The most positive factor of spreading widget technology world widely is usability for tasks on daily bases. Widget users are not bordered pointless middle steps to acquired correct information. The widget technology will shape modern view of applicant use due to utilization software platforms leads by simultaneously of mobile devices increasing influence. We would see future in user friendly environment where are interactions with surroundings devices based on simple, smart and customizable widgets, gadgets or plugins.

Keywords: Widget, Mobile, Device, User, Interaction, GUI.

1 Introduction

The access to content is nowadays driven mainly by web or mobile application bases and therefore the changing of human behavior in cyberspace shaped by mobile devices is recognized as an increasing influence of application design where intuitivism and user experience are on the first place. The mobile applications, which conceptually are enhanced with easy, fast and understandable content bundled into simple application package consists of graphics, code and in some cases external resources, is due to necessity of small and handy mobile device resolutions perfect environment which impacts development for higher user experience. Also that pushes development design level further to designing applications with real importance according to content delivery and applicability. Other motivation which speeds up application evolution process is covered in user's feedback possibility and is called market place environment where applications are rated by users according to satisfaction of goals and quality of user experiences. And also identification of mobile applications which are expressed by visualized icon and simple words benefits over current main domain name of identification for web application. The informational stream or content could be directly accessed by single user touch thought widgets technology and there is no need to remember or type correct domain name and load whole web content repeatedly to get daily bases kind of information but just simple to have a look on home screen where all required content is already loaded with actual information.

C.T. Angelis, D. Fotiadis, and A.T. Tzallas (Eds.): AMBI-SYS 2013, LNICST 118, pp. 113–122, 2013.

2 Problem Definition of Widget Technology

The fast, intuitive and easy to use access for basic devices features by widget technology is an issue over the specific platforms. The concept of widget technology comes historically from desktop computers where users could extract fragments of functionality or information of desktop to defined area. The usability of such kind of content is positive and could be also negative in specific cases. The inconvenient behavior could be caused by incorrect design or fragmented aim of informational stream in terms of remote based information provided by 3rd part provider. We describe in this chapter history of technology, platforms overview and architectural aspects. Typical widget became common on desktop computers where the story of 3rd side reusable small applications starts. We could call them also gadgets, portlets, modules or plugins and we could divide them by background used technologies or by type of client. At first types are simple categorized into desktop, web, mobile or TV widgets. For further more granulation let's focus on technological point of view which are used and are typical for widgets. In case of web background we recognized types as HTML + JavaScript, Adobe Flash, SilverLight or Java applet. These widgets could be embedded in website as well as ported into desktop widgets by particular container as Webkit, Adobe AIR, SilverLight Desktop or Java Virtual Machine. This kind of solution is preferred in case of high needs of variability and portability but could not equally supplement native desktop client application due to specific system functions restrictions. Therefore the desktop widgets are better solutions in case of more system or hardware functionality requirements for instance as hardware

Table 1. Mobile Platform Comparison [1]

Platform	Android	iOS	Windows Phone 7
Developer	Google	Apple	Microsoft
Copy/Paste	Yes	Yes	-
Multitasking	Yes	Yes	-
Flash Support	Yes	-	-
SilverLight Support	-	-	-
HTML5 Support	Yes	Yes	-
Unified Inbox	Yes	Yes	-
Exchange Support	Yes	Yes	Yes
Threaded Email	Yes	Yes	-
Visual Voicemail	Yes	Yes	-
Video Calling	Yes (3rd side)	Yes	-
Universal Search	Yes	Yes	-
Internet Tethering	Yes	Yes	-
Removable Storage	Yes	-	-
Folders	Yes	Yes	Hubs
Widgets Technology	Yes	-	Tiles on Home Screen

monitoring or operational system calls which are not implemented in containers for web widgets because of multiple platforms consistency. The desktop widgets are developed mostly as clones of C objective language such as C# (Windows), C++ (Unix) or Object-C (Mac OS) and also with popular Java (any JVM embedded). Others alternatives are developed in Perl (Unix) or Visual Basic (Windows) dependable on desired platform of use. Mobile widgets are nowadays fast growing area which is fully supported on Android. Others mobile platforms partial or even not support widget technology at all (see fig.1).

The next chapters we will focus on descriptions problematic cross platforms and at last we mentions future heading of TV widgets.

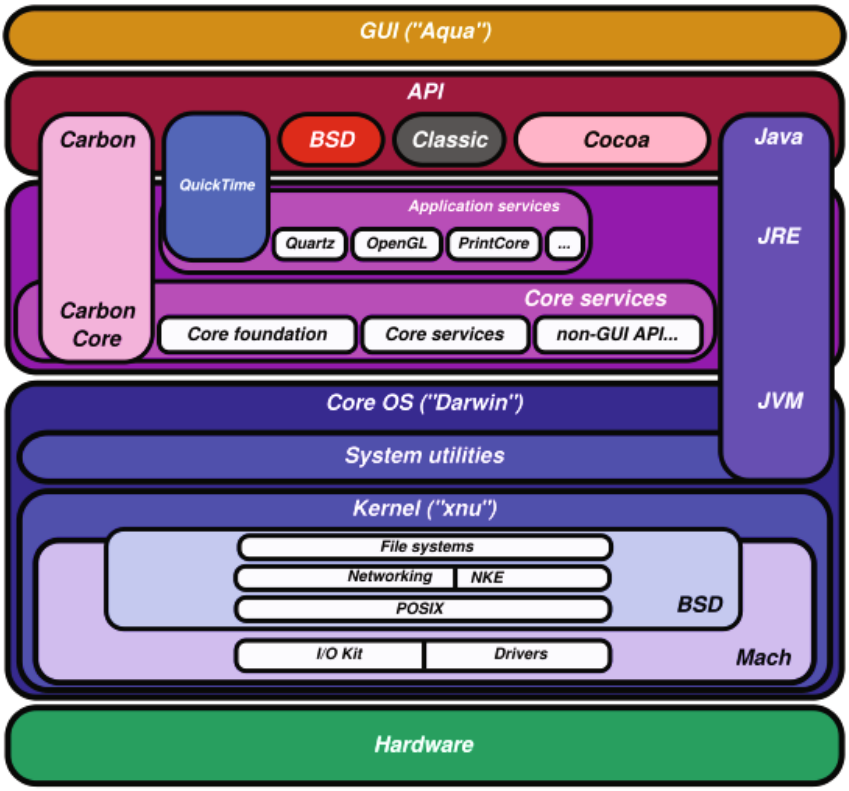

Fig. 1. Apple OS X Architecture [2]

2.1 Apple Platform

The solution based on Apple platform provides for developers fundamental and well prepared design support with framework named COCOA, which is basically using Object-C as programmatic language. There are others extensions from point of developers view where Java or others scripting languages could be used and also when we talked about apple platform we have to announce there are two operation

systems the iOS for mobile devices and OS X (see fig.1) for desktops. The differences are in architectural aspects and supportive environmental tools. Undependable on system bases the most convenient way for developing application or widget applications are in usage common system calls as application interfaces, application services and core services (see figure 1). The advantage of apple platform is basically comprehensive, publishable and distributive widget channel over internet by widgets download center but only for desktop applications. Mobile devices widgets are not supported and therefore the usability of iOS is by this inconvenience decreased.

2.2 Windows Platform

The desktop widget applications are known for users Windows XP, Windows Vista and Windows 7 as desktop gadgets used through sidebar as docking content container. The comprehensive user's environment and easy installation process represent high usability of 3rd side content providing. However Windows Phone is only provide partial solution for widgets by simple tiles which trigger widget application.

2.3 Android Platform

The mobile platform which enables full widget technology and suits to designed solution which is described in next chapter is Android platform. The architecture of Java based platform fully provides multithreading environment where widgets [6] are processed by application widget provider which for defined scheduled time rerun widget application process for limited amount of time where over time consummation is not allowed and the process is killed. The widget would be designed with

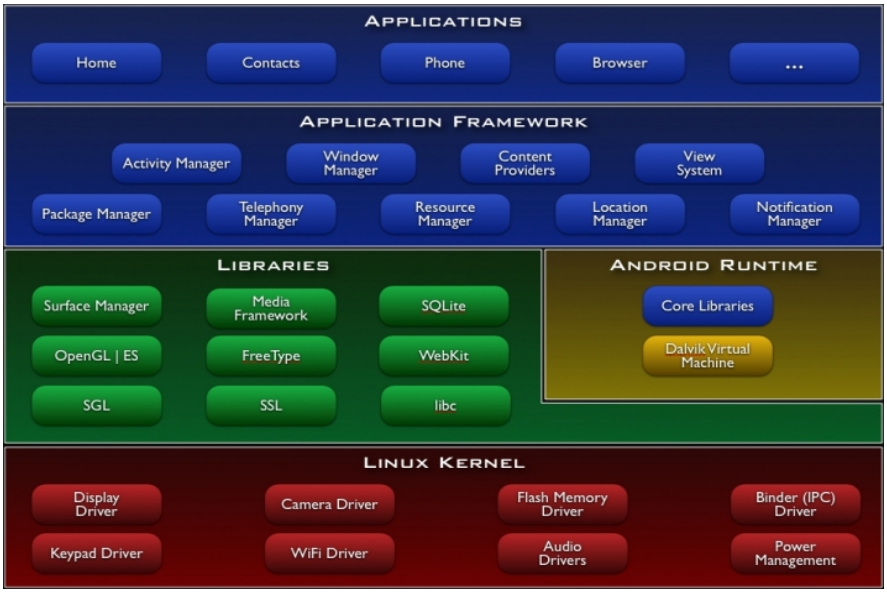

Fig. 2. Android Architecture [3]

responsible battery management and processing time. In case of usage more complicated tasks as internet resource or web services gathering there are allowed services which are running on background and the widget process itself could be only as initializer of background service. The visualization of widgets is based on RemoteViews classes which defines views for particular content and are representation also of user interface in terms of events triggering. For better overview following figure represents Android architecture (see figure 2).

2.4 TV Platform

Last band new area of widgets usage is television platforms where directions we evolution we recognized more in change of TV rather than TV embedded solution. For instance some of vendors supports embedded widget engine based on XML+JS for instance Konfabulator but with low reflection in mass use and there for others vendors as Samsung who directly comes with whole embedded platform Android in TV called Smart TV would change establishment and reality of understanding what television really is.

3 Solution Design

This chapter overviewed application development solution which is based on widget technology and provides core mobile devices functionality. In term of usability we decided to develop such widget application where user could easily control mobile devices features with simple one click functionality. The concept and implementation is based on Android and programed in Java with well-known application architecture model view control (MVC) concept. As visualization of views is used customized component which change its status dependable on mobile device features status change or by user's required actions (see figure 3). Model component represents feature's functionality and provides access to system call or platform API which are allowed. Control component of concept receiving and sending broadcast event and control behavior of view and model component. View in terms of MVC concept we called as a

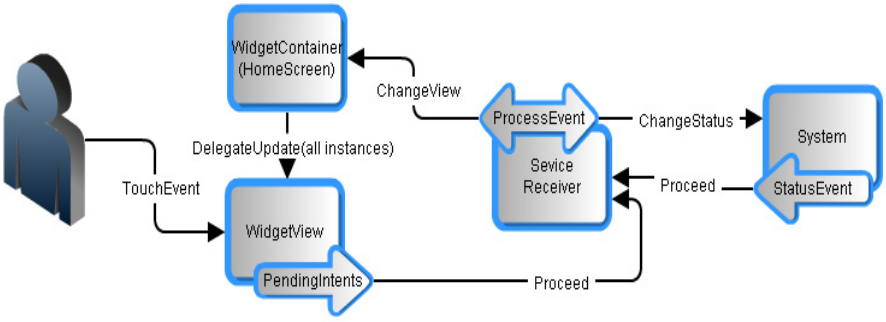

Fig. 3. Mobile Widget Architecture - Events

layout which consist of widgets or other layouts and views based on XML configuration file. In eclipse android plug-in there is a standard palette of views, layouts and widgets. Also developers could define their own views where it is overwritten by *onDraw()* method of *View* class. All raster graphical resources have to be assigned in *res* folder of project specified by type of resolution for concrete device screen.

The essential for application architecture as widgets is message driven model which responses on customized service calls which leads to widget view update. All services are in consideration of one service pattern which interacts with system. The views are managed by widget container which also handles user events called *PendingIntent.* The user events are posted to correct receiver and invoke inner process as an application owner. Process called as Service or Receiver is responsible for view updates and interaction with system. For better overview we outlined following class diagram (see figure 4) and describe application in more detail.

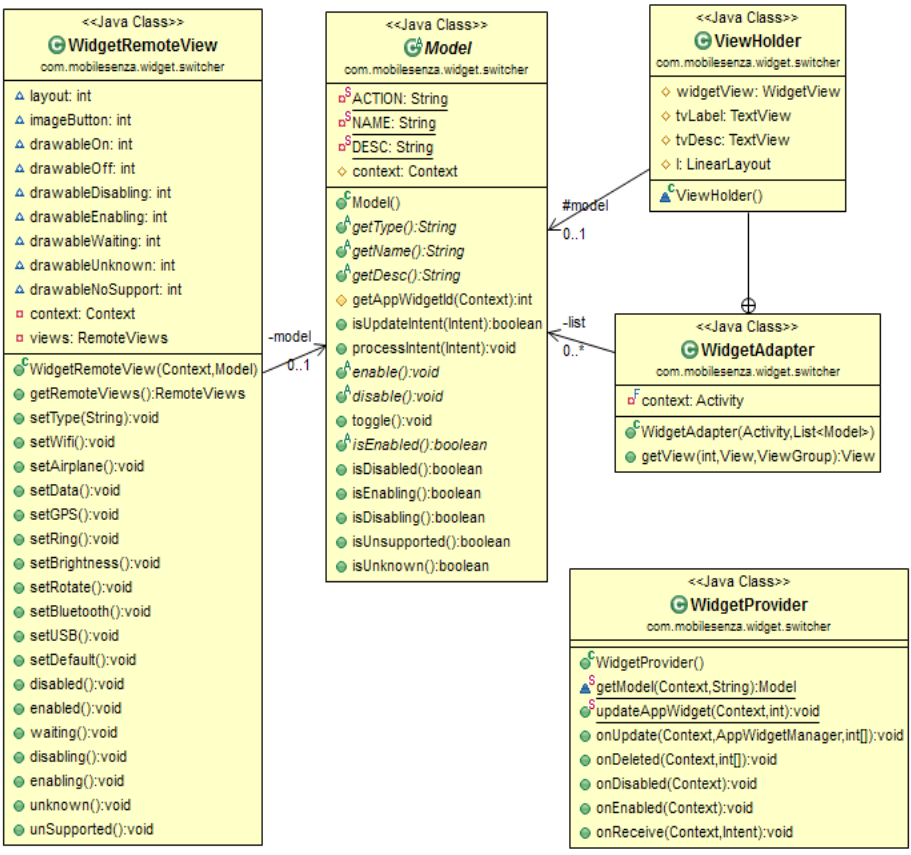

Fig. 4. Mobile Widget Architecture – Class Diagram

The *WidgetRemoteView* class is responsible for correct visualization and user event handling of all widgets. Basically is a container for customized view in easy and comprehensive way of approach to *RemoteViews* class which in terms of Android architecture is fundamental for widget concept. All widget instances are maintained by widget manager and therefore Views of widgets are not directly accessible therefore the only consistent way of changing widget views or associated user actions is by delegating *RemoteViews* instances to widget manager. For communication with widget manager is responsible *WidgetProvider* class which generating correspondent *RemoteViews* by method *getRemoteViews()* and post them by *updateAppWidget()*. The user's actions are represented by *PendingIntent* class as an event which could trigger 3rd side *Service* or *Receiver* due to consistency of developed authorization mechanism. The access is encoded by definition of Intent and we could consider them as authorization tokens with message based delivery which triggers particular task. The Model abstract class is pattern for all device features model classes which could be handled by known states as a enable, disable, toggle methods are connected to system calls and changing the statuses of desired feature accordingly to user defined service permissions. Finally *WidgetAdapter* class represents list container for all implemented modeled features of device to overview statuses on one place in real-time [9], [13] and also as choice optionality for widget selection before placement on home screen [5].

4 Testing Use Case Application – Widget Switcher

The *Widget Switch* is mobile application enables, disables or change status of specific mobile device features. The application is available on Google Play market for any device with Android [4]. We implement following (see table 1) mobile device features based on toggle buttons which are displaying status and are for touch screen events responsible to correspondent system change. Core application is running in background service which is driven by widgets. The widgets are places over widget native device selector on any place on home screens. All instances of the same type of toggle widget are changing upon system status change at the same time.

Widget switch is free for download application where we consider many possible extensions as other device features to be widget like controlled as a system monitoring tools (processor, networks, and memory), external service informational content providing, environment context status resolver, cost/effective communication connectivity, financial data overviewing, etc. We outline just subset possibilities of huge range of widget technology usage.

Table 2. Implemented Android Features Overview

Widget	States	Description
Wifi-WLAN	Disabled Enabling Enabled Disabling Unknown	The functionality is based upon system calls or receives of WLAN radio status. The connectivity to concrete wireless network is beyond application based on configuration and network priority policy [7], [8]
Air Mode	On Off	Mode of in airplane device behavior where could be specified also omitted or included networks explicitly.
Bluetooth	Disabled Enabling Enabled Disabling Unknown	The Bluetooth radio is behave on android platform close to wireless radio where widget application only enables or disables Bluetooth connectivity.
Brightness	Auto Manual	Settings auto is corresponding with saving battery policy; manual values are in 0-100% interval.
GPS	Enable Disable	From 2.3 Android version is only navigate over link to native settings.
Ring	On Silent Vibrate	Status of incoming calls where on is defined ring tone, vibrate only or silent for mute all.

Fig. 5. Widget Switch Application

5 Conclusions

The widget based technology empowers usability of corresponding platform whether is desktop or mobile device. Widgets would be ideal extensions for common applications where summarized data or fast user reaction is required. The advantages of widget technology will be revealed with emerging mobile device and television widget platforms market where 3rd side content providers are required by advantages of mass creative content human power. Presented solution is also usable in numerous cases of image or video processing solutions [10-19], where it can significantly speed up usage by intuitive design.

Acknowledgement. This work was supported partially by (1) “Smart Solutions in Ubiquitous Computing Network Environments”, Grant Agency of Excellence, University of Hradec Kralove, Faculty of Informatics and Management under the project GAE/2012/2213 and (2) "SmartHomePoint Solutions for Ubiquitous Computing Environments", University of Hradec Kralove under the project SP/2013.

References

1. PCWorld mobile platform comparisation, http://www.pcworld.com/article/208491/mobile_os_smackdown_windows_phone_7_vs_ios_vs_android.html (retrieved October 22, 2010)
2. Apple Developer Site, iOS, https://developer.apple.com/library/ios/#documentation/ (retrieved May 24, 2012)
3. Wikipedia, Android OS, http://en.wikipedia.org/wiki/Android_(operating_system) (retrieved May 23, 2012)
4. Widget Switch Application, Google Play Marketplace, https://play.google.com/store/apps/details?id=com.mobilesenza.widget.switcher
5. Komatineni, S., MacLean, D., Hashimi, S.Y.: Home Screen Widgets, Pro Android 3, pp. 711–743 (2011), doi:10.1007/978-1-4302-3223-0_22
6. Murphy, M.L.: Beginning Android 2, Employing Fancy Widgets and Containers, pp. 95–116 (2010), doi:10.1007/978-1-4302-2630-7_9
7. Machaj, J., Brida, P.: Impact of Radio Fingerprints Processing on Localization Accuracy of Fingerprinting Algorithms. Elektronika ir Elektrotechnika 18(7), 129–132 (2012), doi:10.5755/j01.eee.123.7.2391
8. Brida, P., Machaj, J., Gaborik, F., Majer, N.: Performance analysis of positioning in wireless sensor networks. Przeglad Elektrotechniczny 87(5), 257–260 (2011)
9. Herrero, A., Navarro, M., Corchado, E., Julian, V.: RT-MOVICAB-IDS: Addressing real-time intrusion detection. Future Generation Computer Systems 29(1), 250–261 (2013) ISSN 0167-739X, doi:10.1016/j.future.2010.12.017
10. Cheng, W.C., Liou, J.W., Liou, C.Y.: Construct Adaptive Template Array for Magnetic Resonance Images. In: IEEE International Joint Conference on Neural Networks, Brisbane, Australia, June 10-15 (2012), doi:10.1109/IJCNN.2012.6252560

11. Penhaker, M., Darebnikova, M., Cerny, M.: Sensor Network for Measurement and Analysis on Medical Devices Quality Control. In: Yonazi, J.J., Sedoyeka, E., Ariwa, E., El-Qawasmeh, E. (eds.) ICeND 2011. CCIS, vol. 171, pp. 182–196. Springer, Heidelberg (2011)
12. Machacek, Z., Slaby, R., Hercik, R., Koziorek, J.: Advanced System for Consumption Meters with Recognition of Video Camera Signal. Elektronika ir Elektrotechnika 18(10), 57–60 (2012), doi:10.5755/j01.eee.18.10.3062
13. Machacek, Z., Srovnal Jr., V.: Communication Network Model for Industrial Control. In: Proceedings of the 9th RoEduNet IEEE International Conference, Sibiu, Romania, June 24-26, pp. 293–298 (2010)
14. Juszczyszyn, K., Nguyen, N.T., Kolaczek, G., Grzech, A., Pieczynska, A., Katarzyniak, R.: Agent-based approach for distributed intrusion detection system design. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3993, pp. 224–231. Springer, Heidelberg (2006)
15. Vybiral, D., Augustynek, M., Penhaker, M.: Devices for Position Detection. Journal of Vibroengineering 13(3), 531–535 (2011)
16. Sojka, M., Pisa, P., Faggioli, D., Cucinotta, T., Checconi, F., Hanzalek, Z., Lipari, G.: Modular software architecture for flexible reservation mechanisms on heterogeneous resources. Journal of Systems Architecture 57(4), 366–382 (2011), doi:10.1016/j.sysarc.2011.02.005
17. Kasik, V., Penhaker, M., Novák, V., Bridzik, R., Krawiec, J.: User interactive biomedical data web services application. In: Yonazi, J.J., Sedoyeka, E., Ariwa, E., El-Qawasmeh, E. (eds.) ICeND 2011. CCIS, vol. 171, pp. 223–237. Springer, Heidelberg (2011)
18. Behan, M., Krejcar, O.: The Concept of the Remote Devices Content Management. Journal of Computer Networks and Communications 2012, Article ID 194284, 7 p. (2012), doi:10.1155/2012/194284
19. Vybiral, D., Augustynek, M., Penhaker, M.: Devices for position detection. Journal of Vibroengineering 13(3), 531–535 (2011)