WIRELESS WORLD RESEARCH FORUM

OUTLOOK

Visions and research directions for the Wireless World

December **2013, No 10**

UIs – past, present and future

WIRELESS WORLD RESEARCH FORUM

WWRF WGB Services, Devices and Service architectures

White Paper
UIs – past, present and future

Editors:

Dr. Kari Heikkinen, Lappeenranta University of Technology, Finland
Prof. Jari Porras, Chair WGB, WWRF

Version 1.0

Project website address: www.wwrf.ch

Abstract— *A User Interface (UI) is something that we as users or as developers see first. It affects greatly on our performance how well or how fast we do things with the application or service behind the UI. Furthermore, it plays a major role in the overall user experience of the product the UI is part of. This white paper focuses on the evolution of the UI from past and current state into the possible future and recognizes the challenges of the future UIs. Mobile device aspects are also covered as a separate part of UI evolution.*

Index Terms—User Interface, UI development

1. INTRODUCTION

Smith and Mosier [9] have defined User-System-Interface back at 80's. It shares the similarities what we think UI is supposed to do. Back at that time, the focus was on systems, to which users gave commands via UI. Nowadays, we talk more about applications that can sometimes have different kind of systems in the background e.g. for data. However, the fundamentals have not changed. The user interacts with either system or application through an UI.

The definition of an UI emphasizes an important aspect, namely that the UI can also be understood as a two directional interface that allows both inputs and outputs. In the early days the inputs and outputs were clearly separated but the evolution of technologies will bring these closer to each other and finally tie them together into indistinguishable user interaction 'space'.

This white paper focuses on the evolution of the UI from past and current state into the possible future. Section II will introduce how UIs have developed, and section III focuses on UI development in mobile environments. Section VI will shortly focus on UI development environments. Section IV tries to look into future and find out the major challenges. Section VI will conclude this white paper

2. EVOLUTIONARY VIEW ON UI

UI can be understood in from many point of views; some see it as a 'screen' only, and some see it as everything that effects on us interacting with it including e.g. peripherals and other hardware. In this paper the view is both software and developer oriented. Thus, a good analogy would be: as the APIs give for the applications an access to the underlying resources, similarly UIs give that for the interaction between the computer and the user.

Based on Nielsen [22], UIs can be classified in generations, which align to the development of the hardware; according to Nielsen, the first generation was named as pre-historical generation (before 1945), in which the users had a direct access to the hardware, and no software existed. The next ten years is being described as batch programming (1945-55), and the ten years after that as command line interfaces (CLIs, 1955-1965). The next two generations, are given names traditional and modern generation, respectively. The traditional (1965-1980) has menus and form fill-in and the modern has windows, icons and menus (this is also called WIMP - Windows, Icons, Menus and a Pointing device interface, 1980-1995). The current era is named as future generation. In this white paper we focus on user interfaces starting from the command line generation.

It can be said that first modern generation UIs were command line interfaces (CLI) and the computers were mainframes. There was a limited need for interactions. We can divide CLIs into three distinctive types; non-interactive, interactive line-based and text-based. Non-interactive CLIs do not require any user input after the command has been

run (simple Unix command e.g. ls is a good example). Interactive, line-based CLIs do require a further command or typing after the command/next line (a line-based editor being a prime example). Text-based CLIs are actually somewhere between the CLI and GUI (see example of CLI and GUI in Figure 1). As an interaction style, CLIs can be seen instructive. The first generation UI used mostly keyboard (input) and screen (output) for interactions and feedback.

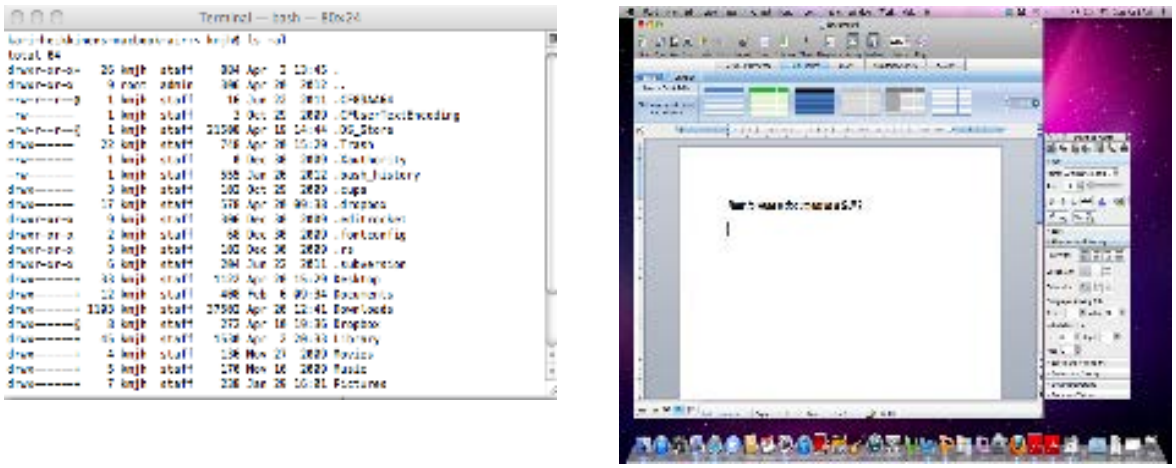

Figure 1: A screenshot of CLI and GUI

All the major interaction styles, conversational, manipulative and explorative, became more apparent by the appearance of graphical user interfaces (GUI), one that can be considered as second generation of UI (traditional or modern generation in Nielsen's taxonomy). Conversational interaction design emulates the means of communication of humans. It promotes discussions and relies on dialogs. What comes to the required technologies, voice recognition is an important technology. Manipulative interaction design is always direct. It gives the user / developer an opportunity to manipulate the items on the screen, whether they are icons, forms, menus or something else that can be used to make the UI behave dynamically. Touch screens have provided new ways to manipulate the items on the screens. Exploration inside the UI is the last interaction style. It is basically needed in applications, where the user has to explore the world behind the UI. Games and 3D user interfaces utilize this interaction design style.

GUIs have relied on exploratory design, and the user is encouraged to look for items on the screen, whether they are icons, pop-ups, dialogs, menus or etc. GUIs introduced mouse and track ball as ways to interact with objects. As the GUIs developed, there became the need for their (faster and more efficient) development. This was the era when first UI development environments emerged. With the development of GUIs, more sophisticated and more direct interaction devices became available, such as e.g. stylus and the use of touch screen.

Then NUIs arrived as third generation of UI (future generation in Nielsen's taxonomy). Before that, an interaction designer only had to identify a correct interaction style that fit into the application scope and supported the tasks carried out with the application. Obviously it had to support the cognition of the user. With NUIs it became more important to understand the cognitive process of a human to communicate in a natural manner. It required new kind of understanding of means of communication without interaction devices such as a mouse or keyboard. These so called non-standard interaction "devices" include for example eye tracking, sensors and cameras, wearable equipment such as Head-Up-Displays (HUDs) or data gloves, haptic feedback by analyzing the capability of a human skin and its characteristics to feel temperature, pain, pressure etc., and tangible items.

Often NUI's means of communication/interaction is based on gestures that are done either by fingers in touch screens or gestures that are based on human body movements that are collected by cameras (e.g. Kinect concept) and analyzed by machine vision software or collected by sensors (e.g. Wii concept) and analyzed by the processing device. The gestures on the touch screen have already become very common and the learning curve on gesture is very low e.g. for a child in comparison for a keyboard. The development of NUIs has been very quick as game platforms such as Nintendo Wii and Playstation Move or Xbox Kinect had brought it to the every day life. Such a NUI examples are shown in Figure 2. In scientific side, for example Islam et al. [19] introduce an approach to utilize Xbox Kinect platform; they have used the SDK for the platform to create an emulator for being able to emulate the Kinect sensor without the real sensor by capturing the sensor data and storing it for later purposes.

Figure 2: Left, Kinect using full body gestures (http://www.gameshop.com.pl/pl/p/Kinect-Sports-Xbox-360/519) and Right, Leap motion for hand gestures (http://www.techspot.com/products/remotes/leap-motion-controller.93087/)

Another UI generation or genre is SUI (Spatial User Interfaces). It can combine all the previous types of UIs and some could cluster it under NUI, but it is here separated due to its interaction concept (see Figure 3 as an example, Surface Table). The basic idea is roughly that any UI in a physical space should be aware or itself and its surrounding and can communicate with the objects (real or virtual) in its proximity. Currently, it also benefits of two recent developments on UIs. Firstly, on top of list and already in our daily life are 3D user interfaces. With 3D we can see the UI from different perspectives and we can manipulate it even richer in comparison to the standard GUI. There exist already own set of interaction design rules for 3D user interfaces. Another relevant technology for SUIs is augmented reality (AR). Bringing together both virtual and real world brings many benefits. Actually, with that eyes may become our main "interaction device" as we guide the UI with our gaze and the movement of an eye. Obviously, with the 3D and AR we are at a new frontier from user point of view. It is still unknown, how these new evolving technologies affect to us. Young children are prohibited of long time use of 3D devices. How much cognitive overload these new technologies will bring?

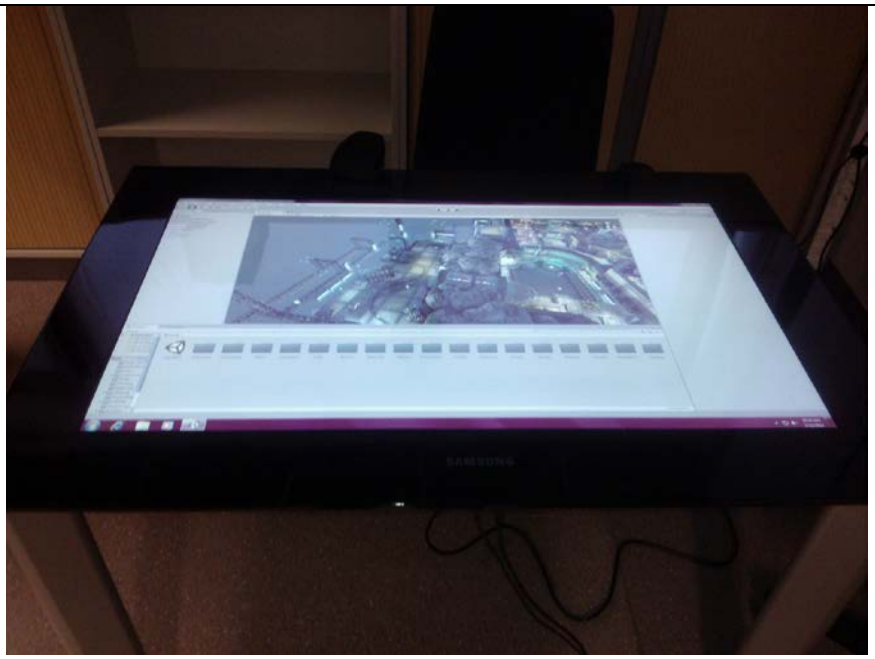

Figure 3: A screenshot of SUI, surface with machine vision software

Gellersen et al [6] introduce a new term, spontaneous interaction, which is a natural part of both NUIs and SUIs. They introduce a spatial model, named RELATE, to make such an interaction possible based on proximity and relative positioning of the targeting devices / equipment. Goth [8] has a nice insight in his article discussing the recent development of NUIs. He discusses how low-cost motion and depth-sensing, such as Microsoft Kinect or PS Move or Wii inertial sensors may bring bigger change to our capability to interact with our applications or environments. It also lists some non-commercial solutions for NUIs, such as algorithms and open-source drivers and middleware, e.g. OpenCV and OpenNI. Furthermore, the paper contains a criteria list for next-generation standard UI; it should be affordable, UI should be intuitive to use and software development should not have a learning curve, meaning that the tools are easy to use and accommodate.

As such, NUIs need to be differentiated from other future development, i.e. virtual reality and augmented reality. Even though, virtual reality utilizes gestures or touch or speech for interaction, it also requires the immersion. This is missing in ‘standard’ NUIs. Augmented reality is another issue, as its immersion is the opposite. The objects of the virtual world are brought into real world. Nevertheless, NUIs do not need another reality in order to be NUI. These approaches share the main elements, namely capability to interact naturally and getting sensory feedback.

Table 1 compares the different UI types, i.e. CLI, GUI and NUI. SUI is omitted as it uses all the other types in creating spontaneous interaction. Furthermore, the scientific literature has provided other acronyms such as TUI, tangible user interface and so on. TUI, according to Shaer and Hornecker [20] can be considered as a new kind of interface that combines the real world with the digital world. In essence, the word tangible is the key here. Based on that the physical objects in our daily life are being used as tangible representation of some digital data, such e.g. one can carry e-mail in a real world pen. The essence of touch is important for TUIs. Shaer and Hornecker [20] describe the birth of TUIs , its current status and where it might be going. According to them, the origin shares some similarities with the early concepts on Augmented Reality and Ubiquitous Computing. However, it differentiated from the rest due to the reason that physical world and its physical objects are more familiar to the users, so that slower pace in shifting to the virtual world would be user-friendlier rather than

immersion into virtual reality. In [20], there is presented some important examples of TUIs, e.g. Graspable UI, Tangible Bits, Slot and Marble Answering Machines. It also has some cross-connection to e.g. augmented reality through the concept of tangible augmented reality, in which the input of the user is tangible but the output is augmented (in display). Examples listed include tangible tabletops, ambient displays and embodied user interfaces. Another piece of cross-connection presented by [20] is the term tangible computing, which in essence is saying that the combination of tangible user interfaces, augmented reality, ubiquitous computing, context-aware devices and smartness in the environment could be put into that umbrella. In the perspective of this paper, these proposals will be part of the future challenges chapter.

Table 1. Comparison of CLI, GUI and NUI [1, 2, modified]

Feature or Characteristic	Command Line Interface (CLI)	Graphical User Interface (GUI)	Natural User Interface (NUI)
Style of Interaction	The user interface is static. It does not contain e.g. icons, forms or menus and did not provide dialogs for interaction. No real interactivity. Yet, it can be considered as instructive interaction style. Keyboards are used as input devices and displays as output devices.	The user interface is response-based. It relies on icons, forms, menus and dialogs to able to perform. Allows using all styles of interaction, although one main style should be used. Adds mouse, trackball and even touchpad for input control.	The user interface evokes our memory, immediately reflects to our natural way of doing things. Allows using all styles of interaction, and support of parallel is more easily understood. Adds devices for gesture recognition for input purposes.
Means of communication / interaction	There is no interaction and no real communication between the UI and the user / developer.	The means of communication or interaction is indirect as the user / developer requires interaction devices, e.g. mouse and keyboard.	The means of communication / interaction is more of cognitive process as it is unmediated. The interaction takes place by haptics or by locomotion as gestures.
The flow of operation	Is directed to the single purpose. The user interface is guided by giving shortened commands. It needs language design for commands.	Can be considered as UI to be explored. Making it graphical gives a reason to look for icons, menus, forms or dialogs.	Natural UIs are highly contextual. The correct interaction needs an understanding of the current underlying context of the situation.
User perception	User has to remember commands and has to recall a lot from the memory. Is not suitable for a novice user but very good for an expert user.	User recognizes (when learned) the icons, menus, forms or dialogs and does not have to remember a lot. It is very suitable for a novice user.	The UI is intuitive. User can do things based on instinct. By following natural means of human communication the use of the UI becomes very easy.
Where it is especially good?	When there is need for a speed. When there are only expert users. When the user needs to have a perception of being in control. CLIs do not need a lot of resources (e.g. drivers to be loaded). Terminals are good examples of CLIs that can be used for remote access.	When there is a clear need to explore. When you want to build an UI based on dialogs, menus or icons. When there is no time for the learning curve. When you want to a comprehensive list of manipulation on the UI (removing, dropping, selecting, etc.).	When there is no possibility to add traditional interaction devices. When the use of the software needs to be intuitive. When there is need to use haptic feedback or motion. When there is need for multimodality. When there is a special user group (e.g. deaf).
Current use	Some experts still favor command line interfaces. Actually, modern CLI is on every mobile phone and is also widely used in web 2.0 applications. When you text a message, you emulate the use of CLI (even though not a command). When you send a tweet or write in a Facebook, the message is often textual.	The mainstream UI. Is well known and well learned. Is used in most applications and especially in desktop applications, which is still the majority application class. Some mobile devices can also be clustered under the GUI.	Modern Smartphone and tablets are all touch-based (need haptic feedback). Surface computing can support multi-touch and can use machine vision to recognize gestures. All applications that require use beyond the GUI benefit.

3. FROM DESKTOPS TO MOBILES

One of the major changes of the last decade is the transition from desktop environment to the mobile environments. The development of the user interfaces in traditional environment is partly affected by the mobile devices through the direct use of mobiles as a part of the user interface and partly by enabling context awareness aspects. It should be however remembered that at the same time the user interfaces of the mobile devices evolve as well. One primary design rule for mobile UIs has been to understand how the user is expected to provide input and how it affects to the usage ergonomics. Table 2 compares these design rules with the general mobile UI heuristic rule set.

Table 2. Comparison of Mobile devices, with UI design heuristics

Design rule	Standard phone	Pen or stylus supported device, tablet	Full Keyboard device	Wearable device
How the user types the input [23, modified]	One hand. From transportable to pocket size phones the keys were: numeric (also letters for messaging), soft keys, navigation keys (mobile phone history has seen several different navigations) and call management keys. The touch smart phones changed the way of typing, by adding e.g. gestures for easier navigation.	Two hands. One hand holds the device and the other hand uses the pen / stylus. Alternatively touch based interaction in modern tablets. These devices can support hand writing recognition.	Two hands. The user can type with both hands like with using standard keyboard (although keys are smaller). The current trend is to have a virtual keyboard; it is in smart phones and in tablets.	Theoretically two hands. It is dependent on where the device is embedded. Possibly limited set of functions included to the wearable interface.
UI characteristics and main five mobile UI heuristics	Character or pixel-based. Can be touch-screen. Keyboard in screen. #1: Unity-based look and feel of making input, navigation etc. has become important design factor #2: Error prevention and management is more important in mobile UIs, as the mobile use often makes them easier due to e.g. contextual settings #3: Easy undo should be possible. Power button is easy to use and also if screen has a separate button for	Character or pixel-based. Can be touch-screen. Keyboard in screen. #1: Needs equally to support unity in making input, navigation etc. #2: Needs support for error prevention and management #3: Needs equal support for easy undo. Need also take into account the style of input through pen/stylus. #4: Needs more support for memorability, as the additional input possibilities are alternatives and might need different	Character or pixel-based. Can be touch-screen. Can have keyboard in the screen or separate keyboard. #1: Needs equally to support unity in making input, navigation etc. #2:Needs equally support for error prevention and management. #3: Needs equal support for easy undo. #4: Needs equal support for memorability. #5: Needs more	Character or pixel-based. Can be touch-screen. Does not have keyboard. #1: The use is likely different, and it might be cumbersome to make unity-based input, navigation etc. #2: Needs partial / different support, as the use is different #3: Needs to support easy undo too. Is likely through hard keys or voice recognition. #4: Should be designed so that

	it. #4: Memorability is more important than in desktops. Should use more voice and haptic feedback. #5: Design for-the-go. The use is literally mobile; thus design it for into use in different contextual settings. Utilize sensors.	feedback. #5: Needs more support or could utilize pen/stylus interaction in different contextual setting in which the user might benefit from it	support as the user could use two hands, and some applications or contextual settings might benefit from two-hand use.	memorability would not come as a problem. #5: Needs partially different support as the screen is smaller than in counterparts, and yet the same data need to be relayed.
UI generation	CLI, GUI and NUI supported. Early phones had only CLI (with layered menus) and keys. GUI (and icons) evolved with smart phones. With touch-based smart phones came the gestures, sensors and haptic feedback.	CLI, GUI and NUI supported. Early phones had only CLI (with menus) and keys. With smart phones came GUIs (and icons). With touch-based smart phones came the gestures, sensors and haptic feedback.	CLI, GUI and NUI supported. Firstly came CLIs (and menus) and GUIs (and icons). With touch-based smart phones came the gestures, sensors and haptic feedback.	Mostly have been CLI (with menus and text icons), as the screen is smaller. New devices like Samsung Gear extend the use close to GUI type of usage with touch screen.

4. UI DEVELOPMENT

UI development can be simplified by clustering it into development technologies and support tools such as development environments. The first ‘cluster’, namely development technologies have evolved through the past 30 years into different approaches and is either trying to focus on some UI related concerns or on offering complete UI development solution. The second ‘cluster’ was development environments, i.e. an integrated development environment (IDE). In short, we are dealing with an application that is created for developers and programmers for making software more efficiently. The link to the software development, in general, in very strong and sometimes it is difficult to differentiate the development environments from software engineering.

Mijailovic and Milicev [7] introduce a survey that gives us some insight on both the evolution and state-of-the-art of UI development technologies. It mainly focuses on categorizing the development technologies (e.g. form-oriented frameworks such as Microsoft Access) and their concerns related to the UIs. These concerns are the layout of the UI, how to bind the data for the UI, how the behaviour (of UI elements) is addressed, support for data formatting, what kind of style/theme support can be taken into account and how to validate the input (how ever it is given). In addition, it also lists models- and domain-specific approaches. As a result, declarative UI development technology (widely applied in web development, e.g Eclipse XWT, XAML Android etc.) is the current trend with a shift to standard technologies and platforms.

History of the IDEs has begun with the Darthmouth Basic programming language (http://dtss.dartmouth.edu/timeline.php). It was basically an interactive command line and with it the Basic programs could be run from the terminals. Maestro product was first official IDE (http://www.computerwoche.de/a/interaktives-programmieren-als-systems-schlager,1205421) and was developed by laboratory in Munich. From there the driving force was the need for visually code the UIs, and thus modern IDEs have

been developed. During the past 50 years the UIs have developed from mainframes' CLIs to desktop-derived GUIs or more intuitive NUIs. This has been accomplished partly by different integrated development environments (IDEs). Obviously, there exist a lot of platforms and a lot of different programming languages, which often require own kind of IDE.

Kats [3] provides a good insight of the history of IDE; as the first UI was CLI and focused on commands given by the expert user, there was no urgent need for IDEs even though the same functionalities existed, i.e. producing the code, compiling the code, debugging and executing the final result. Kats [3] further continues the discussion by explaining the features of the modern desktop IDEs; the UI has become much more richer and the above mentioned functionalities are present and some provide support for software development (e.g. version control) and often focus on doing the programming work with a one particular programming language. An example of a modern desktop IDE is Eclipse, which is an extensible IDE. Rubel [4] has presented an article, which focuses on explaining "the heart of Eclipse". This Eclipse platform parts are illustrated in Figure 4. On the bottom there is the platform runtime, which helps in binding the basic initialization and loading with set of plug-ins. Inside the platform the UI workbench is responsible for handling all what the user sees. It is linked to both SWT, which is a graphics library for Java GUI and Jface, that is a GUI abstraction layer on top of SWT.

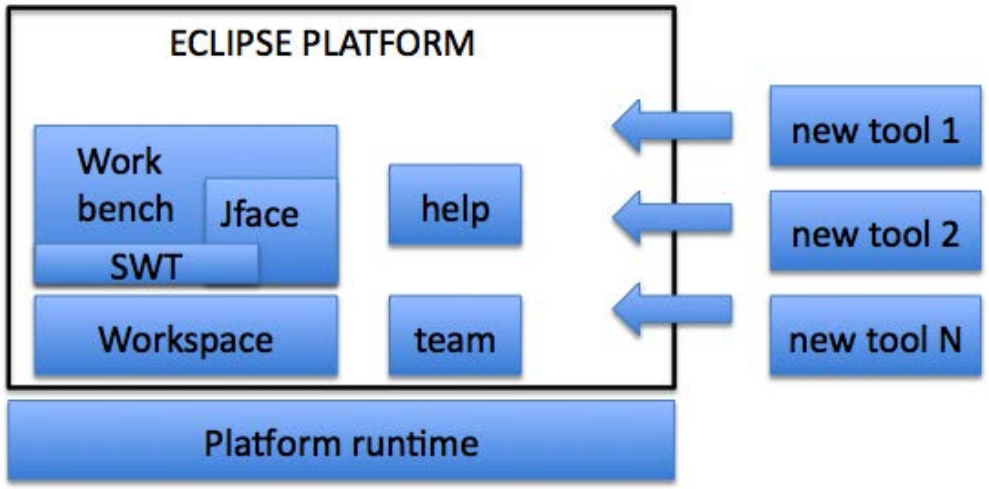

Figure 4: Eclipse platform and its components

Rubel [4] explains each component in more detail; basically the workbench is the fundamental from the development point of view as it provides for the developer different views, editors and actions. According to Rubel [4] it saves work from the developer and he/she does not have to make same work while changing e.g. operating system. Thus, promoting both modularity and re-use of components.

A very interesting concept has been presented by Bruch et al. in [5]. They emphasize the idea of knowledge sharing. That immediately puts some additional questions into our mind? What if our development environments could be collaborative? Can we create code in real-time together while standing / sitting around the shared workspace? Could the team be in more agile, while e.g. four persons could be making the same application around the same e.g. SUI, and each one would be having own view and can create the same code collectively with real time debugging and execution at the background?

Another two examples of extensible IDEs are Android development environment for mobile development and Visual Studio for Microsoft ecosystem (see figure 5, next page). The upper hand figure below belongs to Android development environment and the lower hand is a screenshot of Windows 8 development environment. Both development environments have the outlook of a modern IDE, having set of actions available, having certain views for the developer and contains an emulator for end-use view.

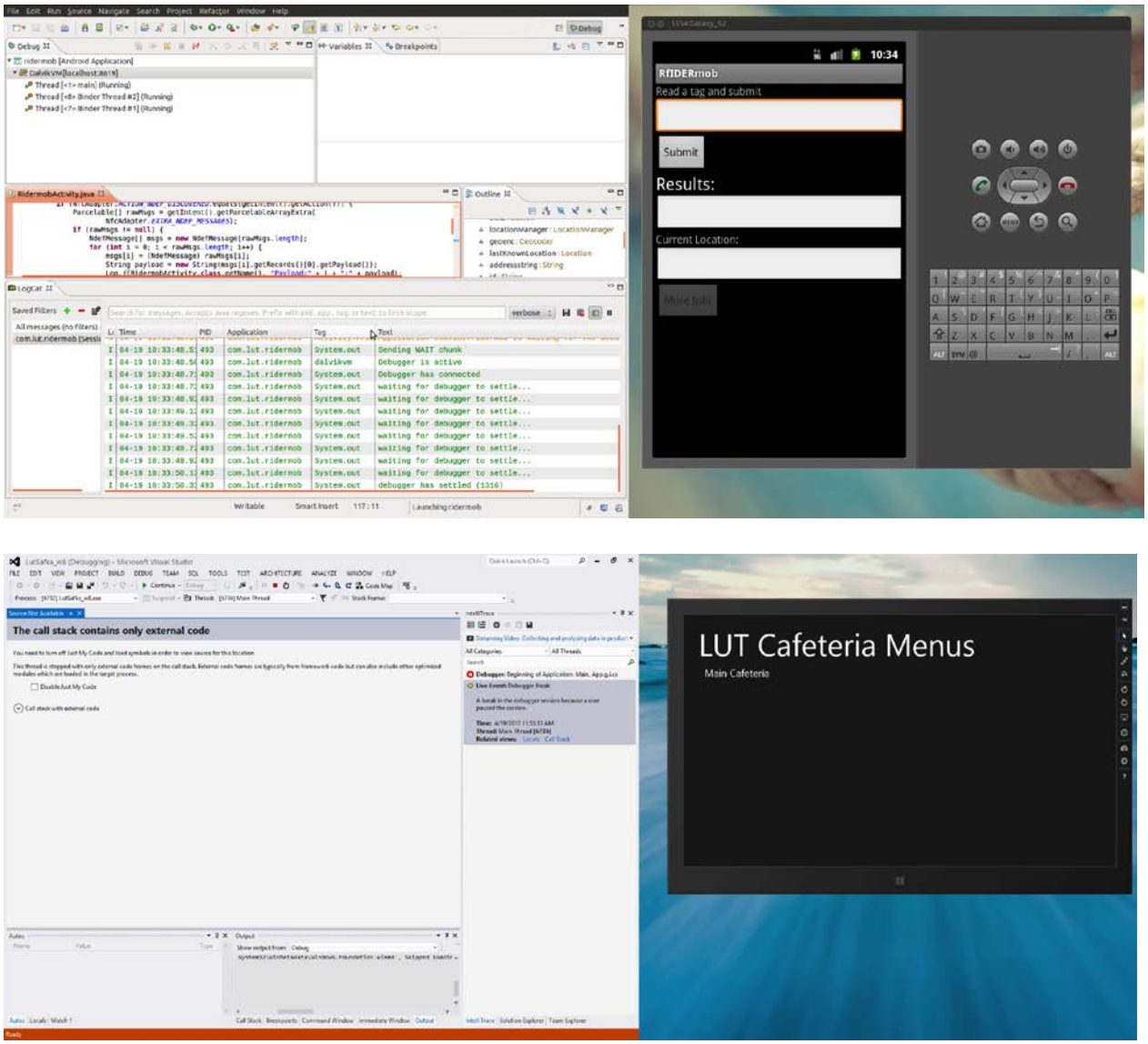

Figure 5: Two examples of extensible IDE (Android and Visual Studio Environments)

5. FUTURE VIEW ON UIS

The UI has been very device-, equipment-centric. At the beginning we had centralized mainframes to which we could access by terminals. Then came the personal computers, which still were terminals, and the UI was focused on the terminal and its capabilities. Still, most of the UIs are built into a given device, whether it is a mobile phone, tablet, laptop or any other device. Future UI is not necessarily device centric, it could rather be environment-derived (e.g. such an example is illustrated by Thebault et al. [16], in which services are embedded into the environment, in other terms it is being ambient): future UI can be any ‘screen’, whether it is augmented reality supported eye-

glass (Figure 6), 'smart' air (at nano level) or any other projection into the proximity of the user. Various input option will also evolve e.g. Brain-to-Computer Interface as presented in Figure 7. It seems that the input and output parts of the user interaction space tend to become more and more separated. It will be possible to combine various input and output options towards a personalized user interface.

Figure 6. Google Glasses for interfacing environment (http://www.google.com/glass/)

Figure 7. NeuroSky MindWave BCI equipment (http://www.neurosky.com/)

We can identify at least three major trends that will have an effect on the future UI evolution. Firstly, we have pervasive computing. Secondly, we have matured augmented reality. Thirdly and finally, we have nano and material technology that can have disruptive effect on UI development.

Pervasive computing (or ubiquitous) is a paradigm that we are stepping in at the moment. There are already established products that make computing for our everyday life available anytime and anywhere. Clear elements of pervasive computing are connectivity (via different wireless/wired technologies), algorithmic intelligence (smartness) and natural (or even spatial) user interaction (by gestures, motion detection or voice). The examples are so many, that we focus on examples that try to merge with the other two, namely augmented reality and nanotechnology. Sandor and Klinker [10] presents user interface architecture to handle in future even more complex challenges of interactions within UIs. It combines different UI approaches from e.g. multimodal, wearable, tangible and augmented solutions for a single UI. Thus, it could be used in future pervasive environments. Another integration example shows also the benefit of combining elements; Rodrigues et al. [14] present a result of tangible user interface with an augmented reality. Furthermore, third example by Alam et al. [17] shows the integration of haptic interface with 3D immersion into ubiquitous environment. Finally, the fourth example by Grubert and Schmalstieg [18] show how the use of lens with augmented reality is handled in a public place. Its interesting contribution is towards finding information how users might possible use augmented reality in a public space from usability and user interface design perspective.

Augmented reality (AR) as such, it is not a new idea, rather the tools and its applicability has reached maturity level. At the literature, it has been studied already over 50 years. Prime examples in the literature show the potential of the augmented reality for the future UI. Debernardis et al. [11] illustrate the extra benefit of AR instead of computer graphics, i.e. the information on the UI being both real-time and co-located with the real environment. Johnson and Sun [12] show the power of AR in how it can be addressed into spatial collaboration. A very good example of learning case is presented in [14], how your own body can be used in e.g. learning your anatomy. Eck and Sandor [13], on the other hand present an approach, in which visuo-haptic is presented together with AR. In such approach, the users are able both to touch and see the virtual objects at the same time. Markov-Vetter et al. [15] illustrates the depth of studies on AR; it focuses on how to input data in a complex setting of task with gravital powers (flight-related) in presence.

Development of nanotechnology and new advanced materials will affect our societies equally disruptive manner as pervasive computing and augmented reality does. It has also effect on the development of UIs as it opens another new set of possibilities for what can be considered as a screen. Even though the most research focuses on energy efficiency of the devices and very thin nano scale displays, in the future, UI design might be more in the demand.

Innovation for the future UI evolution can be searched for example from Marcus [21]. He discusses the relation of the Sci-Fi movies and HCI, and how these movies at the same time show the creativity of possible future design.

6. CONCLUSION

As the UIs have developed through the time, the fundamentals have remained; the user has a certain need / capability to give an input in order to have something accomplished. The UI acts as sort of broker for both an input and output (provided by the screen or feedback of some form). In the beginning, the input/output was mostly clearly separated but in the future the border of what is input or output is more blurred. However, if the technology is becoming more and more smart, the more the user has opportunities for intuitive interaction with the technology in a way the user feels comfortable, secure and in control. This is in essence the calm computing, Mark Weiser predicted.

REFERENCES

[1] B.Shneiderman, C.Plaisant, M.Cohen and S.Jacobs, Designing the User Interface: Strategies for Effective Human Computer Interaction, 5th. Edition. Published by Prentice Hall, 2009 ;ISBN: 978-0321537355

[2] Y.Rogers, H.Sharp and J.Preece, Interaction Design – Beyond Human-Computer-Interaction, 3rd Edition. Published by Wiley, 2011, ISBN: 978-0470665763

[3] L.Kats, R. Vogelij, K. Trygeve Kalleberg and E.Visser, Software development envinronments on the web: a research agenda, Proceedings of ACM symposium on new ideas, paradigms, and reflections on programming on software at Onward! 12 conference, 2012, ISBN: 978-1-4503-1562-3.

[4] D.Rubel, The heart of Eclipse, Queue, Volume 4 Issue 9, 2006, ISSN: 1542-7730.

[5] M.Bruch, E.Bodden, M.Monperrus and M.Mezini, IDE 2.0: collective intelligence in software development, Proceedings of the FSE/SDP workshop of future of software engineering research at FoSER conference, 2010, ISBN: 978-1-4503-0427-6.

[6] H.Gellersen, C.Fischer, D. Guinard, R.Gostner, G.Kortuem, C.Kray, E.Rukzio and S.Streng, Supporting device discovery and spontaneous interaction with spatial references, Personal and Ubiquitous Computing, Volume 13, Issue 4, Springer-Verlag, 2009, ISBN: 00779-008-0206-3

[7] Z.Mijailovic and D.Milicev, A retrospective on user interface development technology, IEEE Software, 2013, ISSN: 0740-7459.

[8] G.Goth, Brave NUI world, Communications of the ACM, Volume 54, Issue 12, December 2011, ISSN: 0001-0782.

[9] S.L.Smith and J.N.Mosier, Guidelines for designing user interface software, 1986, Available online at http://hcibib.org/sam/

[10] C.Sandor and G. Klinker A rapid prototyping software infrastructure for user interfaces in ubiquitous augmented reality, Personal Ubiquitous Computing, 2005, ISSN: 1617-4909.

[11] S. Debernardis, M. Fiorentino, M. Gattullo, G. Monno, and A. E. Uva , Text readability in Head-Worn Displays: color and style optimization in video vs. optical see-through devices, IEEE Transactions on Visualization and Computer Graphics, 2013, ISSN: 1077-2626.

[12] A.S. Johnson and Y.Sun, Exploration of Spatial Augmented Reality on Person, Virtual Reality, 2013, ISBN: 1434-9957.

[13] U.Eck and C.Sandor, HARP: A Framework for Visuo-Haptic Augmented Reality, Virtual Reality, 2013, ISBN: 1434-9957.

[14] F.Rodrigues, F.Sato, L.Botega and A.Oliveira, Augmented Reality and Tangible User Interfaces Integration for Enhancing the User Experience, Proceedings of the 11th ACM SIGGRAPH conference on Virtual-Reality Continuum and its Applications in Industry, 2012, ISBN: 978-1-4503-1825.

[15] D.Markov-Vetter, E.Moll and O.Staadt, Evaluation of 3D selection tasks in parabolic flight conditions: pointing task in augmented reality user interfaces, Proceedings of the 11th

ACM SIGGRAPH conference on Virtual-Reality Continuum and its Applications in Industry, 2012, ISBN: 978-1-4503-1825.

[16] P.Thebault, D.Decotter, M.Boussard and M.Lu, Embodying Services into Physical Places: Toward the Design of a Mobile Environment Browser, ACM Transactions on Interactive Intelligent Systems, Volume 3, No.2, 2013, ISSN: 2160-6455.

[17] K.Alam, A.Mahfujur and A. El Saddik, Mobile Haptic E-Book System to Support 3D Immersive Reading in Ubiquitous Environments, ACM Transactions on Multimedia Computing, Communications and Applications, Volume 9, No. 4, 2013, ISSN: 1551-6857.

[18] J.Grubert and D.Schamlstieg, Playing it real again: A repeated evaluation of magic lens and static peephole interfaces in public space, Proceedings of ACM MobileCHI 2013, ISBN: 978-1-4503-1965-2.

[19] M.Islam, S.Rahaman, R.Hasan, R.Noel and A.Salekin, A Novel Approach for Constructing Emulator for Microsoft Kinect XBOX 360 Sensor in the .NET Platform, Proceedings of Intelligent Systems Modeling and Simulation, 2013, ISBN: 978-1-4244-9809-3.

[20] O.Shaer and E.Hornecker, Tangible User Interfaces: Past, Present and Future Directions, Foundations and Trends in HCI, Volume 3, Issue 1-2, 2010, ISSN: 1551-3955.

[21] A.Markus, The History of the Future: Sci-Fi movies and the HCI, Interactions Magazine, Volume XX.4, July-August 2013, ISSN: 1072-5220.

[22] J.Nielsen, Usability Engineering, Academic Press, ISBN: 0-12-518405-0, 1993.

[23] H.Kiljander, Evolution and Usability of Mobile Phone Interaction Styles, Dissertation thesis, 2004, ISBN: 951-22-7319-5.

Imprint

Wireless World Research Forum
c/o Format A AG
Pfingstweidstrasse 102b
CH-8005 Zürich

Secretariat:
Vinod Kumar
Alcatel-Lucent France
Centre de Villarceaux
Route de Villejuste
91 620, NOZAY
France
e-Mail: vinod.kumar@alcatel-lucent.com
Phone : + 33 1 30 77 27 37
Fax : + 33 1 30 77 61 75

The WWRF is a non-profit organisation registered in Switzerland

Chairman of the Forum:
Dr. Nigel Jefferies

Editor-in-Chief: Mr Sudhir Dixit

The WWRF Outlook Visions and research directions for the Wireless World

ISSN 1662-615X is published non-periodically by the Wireless World Research Forum http://www.wwrf.ch