Chapter 4: HCI Design

4.1 The Overall Iterative Process

In the first three chapters, we have studied important principles, guidelines and theories for "HCI design." In this book, "HCI design" is an integral part of a larger software design (and its architectural development) and is defined as the process of establishing the basic "framework" for interaction which includes the following iterative steps and activities. HCI design is the all preparatory activities for developing an interactive software to ensure high usability and good user experience up to before actual implementation. We will illustrate these four iterative steps using a concrete example after a short explanation of the respective steps.

- **Requirements analysis** – Any software design starts with a careful analysis of the functional requirements. For an interactive software with a focus on the user experience, we take a particular look at **functions that are to be activated directly by the user through interaction (functional-task requirements)**, and **functions that are important in realizing certain aspects of the user experience (functional-UI requirements),** even though it may not be directly activated by the user. An automatic functional feature of adjusting the display resolution of a streamed video based on the network traffic is one such example. Thus, it is not always possible to computationally separate major functions from their user interface. That is, certain functions actually have direct UI objectives. Finally, we identify **non-functional UI requirements which are UI features (rather than explicit computational**

function) that are not directly related to accomplishing the main application task. For instance, requiring certain font size or type according to a corporate guideline may not be a critical functional requirement but a purely HCI requirement feature.

- **User Analysis** – As we have emphasized numerously, a user analysis is a must step in HCI design. The results of the user analysis will be reflected back to the requirements, and could identify additional UI requirements (functional or non-functional). It is simply a process to reinforce the original requirements analysis to further accommodate the potential users in a better way. For instance, a particular age group might necessitate certain interaction features such as a large font size and high contrast, or a functional UI feature to adjust the speed of scroll.

- **Scenario and Task Modeling –** Equally important to user analysis is the task analysis and modeling. This is the crux of interaction modeling, identifying the application task structure and their sequential relationships. With a crude task model, we can also start to draw a more detailed scenario or storyboard to envision how the system would be used and assess both the appropriateness of the task model and feasibility of the given requirements. Again, one can regard this simply as an iterative process to refine the original rough requirements. By the nature of storyboarding, a rough visual look of the interface can be sketched and furthermore the storyboard will serve as another helpful medium in selecting the actual software or hardware interface. It will also serve as a starting point for drawing the object-

class diagram, message diagrams and use cases for preliminary implementation and programming.

- **Interface Selection and Consolidation** – For each subtasks and scenes in the storyboard, particular software interface components (e.g. widgets), interaction technique (e.g. voice recognition), and hardware (sensors, actuators, buttons, display, etc.) choices will be made. The chosen individual interface components need to be consolidated into a practical package, because not all of these interface components may be available on a working platform (e.g. Android based smart phone, desktop PC, mp3 player). Certain choices will have to be retracted in the interest of employing a particular interaction platform. For instance, for a particular subtask and application context, the designer might have chosen voice recognition to be the most fitting interaction technique. However, if the required platform does not support the voice sensor or network access to the remote recognition server, an alternative will have to be devised. Such conceding choices can be made out of many reasons aside from platform requirements, such as due to budget, time, personnel, etc.

Before we go through a concrete example of a HCI design, we first review, in the next section, possible and representative interfaces (hardware and software) to choose from.

[Figure 4.1] The overall iterative HCI design process (as a precursor to implementation).

4.2 Interface Selection Options

4.2.1 Hardware Platforms

While different interactions and subtasks may require various individual devices (sensors and displays), we take a look at the hardware options in terms of the larger "computing" platforms, which are composed of the usual devices. The choice or a design configuration of the hardware interaction platform is determined in large by the characteristics of the task/application which necessitates certain operating environment. The different platforms listed below are suited for and reflects various operating environments.

- **Desktop (Stationary) -** Monitor (typical size: 17 ~ 42 inch / resolution: 1280 x 1012 or higher), keyboard, mouse, speakers/headphones, (microphone).

  Suited for: Office related tasks, time consuming/serious tasks, multitasking.

[Figure 4.2] A basic desktop operating platform.

- **Smart phones / Hand-helds (Mobile) -** LCD screen (typical size: 3.5 ~ 5 inch / resolution: 720 x 1280 / weight: ~ 120g), buttons, touch screen, speaker/headphones, microphone, camera, sensors (acceleration, tilt, light, gyro, proximity, compass, barometer), vibrators, (mini

"qwerty" keyboard).

Suited for: Simple and short tasks, special purpose tasks.

- **Tablet/pads (Mobile)**: LCD screen (typical size: 7 ~ 10 inch / resolution: 720 x 1280 / weight: ~ 700g), buttons, touch screen, speaker/headphones, microphone, camera, vibrators, sensors (acceleration, tilt, light, gyro, proximity, compass, barometer).

  Suited for: Simple, mobile and short tasks but those that require relatively large screen (e.g. sales pitch).

[Figure 4.3] Making sales with pad-like devices leveraging on its mobility and relatively large screen size[1].

- **Embedded (Stationary/Mobile) -** LCD/LED screen (typical size: less than 3 ~ 5 inch / resolution: low), buttons, special sensors and output devices (touch screen, speaker, microphone, special sensors). Embedded devices may be mobile or stationary and offer only very limited interaction for few/simple functionalities.

  Suited for: Special tasks and situations where interaction and computations are needed on

[1] Toyota Korea, http://car.biz.chosun.com/site/data/html_dir/2010/12/16/2010121600551.html

the spot (e.g. printer, rice cooker, mp3 player, personal media player).

[Figure 4.4] Embedded interaction platform: (a) mobile and (b) stationary.

- **TV/consoles (Stationary) -** LCD/LED screen (typical size: > 42 inch / resolution: HD), button based remote control, speaker, microphone, game controller, special sensors and peripheries (camera, wireless keyboard, Wii-mote like device [8], depth sensor such as Kinect [3, 7]).

  Suited for: TV centric tasks, limited interaction and tasks that need privacy (e.g. wild gesture-based games in the living room).

[Figure 4.5] TV/Console interaction environment[2].

- **Kiosks/installations (Stationary) -** LCD screen (typical size: 10 ~ 13 inch / resolution: low ~ medium), buttons, speaker, touch screen, special sensors and peripheries (microphone, camera, RFID/credit card reader, heavy duty keyboard).

  Suited for: Public users and installations, limited interaction, short series of selection tasks,

[2] Xbox LIVE, http://www.xbox.com/ko-KR/Live/what-is-live?xr=shellnav

monitoring tasks.

[Figure 4.6] Various kiosk/installation types of interaction platform[3].

- **Virtual reality (Stationary) -** Large surround and high resolution projection screen / head-mounted display / stereoscopic display, 3D tracking sensors, 3D sound system, haptic/tactile display, other special sensors and peripheries (microphone, camera, depth sensors, glove).

  Suited for: Spatial training, tele-experience and tele-presence, immersive entertainment.

[Figure 4.7] Virtual reality interaction platform[4].

- **Free form (Stationary and Mobile) -** A special purpose hardware platform may be formed by a customized configuration of individual devices best suited for the given task (when cost is not the biggest factor). There are many such custom designed interfaces such as those shown in Figure 4.8.

[**Figure 4.8]: Examples of special purpose interfaces (from top left in counter clock-wise): (a)**

[3] PFU Systems Kiosk, http://www.pfusystems.com/kiosk-hardware/index.html

[4] Visbox, http://www.visbox.com/imgs/viscube-hd.html

pen tablet[5], (b) a glass type see-through head up display[6], (c) camera integrated scuba gear[7], (d) special military helmet for tactical command and control[8], (e) multi-touch table top platform for multiple users[9].

4.2.2 Software Interface Components

Most of the software components are quite well known and familiar to most of the readers and as such, we only highlight important issues to consider in the interface selection.

- **Windows/Layers** - Modern desktop computer interfaces are designed around "windows" which are visual output channels and abstractions for individual computational processes. For a single application, a number of subtasks may be needed concurrently and be interfaced through **multiple** windows. One window among the many (or task) would be "active" for whose window becomes "focused" by placing the mouse cursor over it or by an explicit click. For relatively large displays, overlapping windows may be used and however, as the display size decreases (e.g. mobile devices), **non-overlapping "layers"** (a screen full window) may be used in which individual layers are activated in turn by "flipping through" type of interactions (e.g. flickering on touch screens).

---

[5] Genius, KYE Systems G-Pen, http://www.geniusnet.com/wSite/ct?xItem=16835&ctNode=174

[6] Google glass, http://www.google.com/glass/start/

[7] Liquid Image Scuba, http://www.liquidimageco.com/collections/scuba

[8] Defensereview, http://www.defensereview.com/ (Photo credit: David Crane)

[9] Samsung MultiTouch Display, http://www.samsung.com/sec/news/presskit/hf

While multiple overlapped windows have been traditionally used for relatively large desktop platforms and layers for smaller devices, with the recent trend and requirement of "multiple device and single user experience," the Windows Metro-style interface has unified the two [5]. Even on the desktop, the Metro-style presents individual applications on the full screen without marked borders, but instead offers new convenient means for sharing data with other applications and switching between the applications or tasks. Other important detailed considerations for a window (for supporting interaction for a subtask) might be its size, interior layout, and management method (e.g. activation, deactivation, suspension).

[Figure 4.9] The hallmark of modern desktop user interfaces: multiple overlapping windows (left) and non-overlapping layers for smaller displays (right).

[Figure 4.10] The Microsoft Metro style interface that unifies the mobile and desktop interaction.

- **Icons** – Interact-able objects may be visually represented as a compact and small pictogram such as an icon (and similarly as **"earcon"** for the aural modality). "Clickable" icons are a simple and intuitive interface. As a compact representation designed for facilitated

interaction, icons must be designed to be as informative or distinctive as possible despite its size and compactness.

The recent Windows Metro-style interface has introduced a new type of icons called the "tiles" which can dynamically change its look with useful information associated with what the icon is to represent [2]. For instance, the e-mail application icon dynamically shows the number of new unread e-mails (Figure 4.10).

[Figure 4.11]: (1) Obscure Google Chrome browser[10] icon design (difficult to tell at a glance what the icon represents) vs. (2) informative icon design for the Angry Bird application[11] (the visual imagery of an "angry bird" is well captured).

- **Menus** – Menus allow activations of commands and tasks through selection (recognition) rather than recall. Typical menus are organized as one dimensional list or two dimensional array of items (represented in text or as icons/earcons).

  A menu item selection involves three subtasks: (1) activating the menu and laying out the items (if not already activated by default), (2) visually scanning and moving through the items

[10] Google chrome, https://www.google.com/intl/ko/chrome/

[11] Angry Birds, http://www.angrybirds.com/

(and scrolling if the display space is not sufficient to contain and show the whole menu items at once), and (3) choosing the wanted item. All these subtasks are realized by making discrete inputs by, e.g. mouse click, screen touch, button push, voice command, etc.

Menus (i.e. list of items) may be presented in a variety of styles and mechanisms: just to name some of the most popular ones, the pull-down, pop-up, tool-bars, tabs, scroll menu, 2D array of icons, buttons, check boxes, hot keys, etc. Table 4.1 shows how to best use these different types of menus.

[Table 4.1] Where to use different menu styles for.

Menu type	Usage
Pull down	Top level (main) categorical menu
Pop up	Object specific, context specific
Tool bar	Functional / operational tasks
Tabs	File folder metaphor (categorical menu)
Scroll menu	Long menu (many menu items)
2D array / Image maps	Identification of items by icons (vs. by long names) or pictures
Buttons / Hyperlinks	Short menu (few choices)

Check boxes / Radio buttons	Multiple choice / Exclusive choice
Hot Keys	For expert users
Aural menu	Telemarketing, For the disabled

[Figure 4.12] Different styles of menus 1: (a) pull down, (b) pop up, (c) 2D application bar, (d) 1D toolbar, and (5) tabs.

[Figure 4.13] Different styles of menus 2: (a) buttons, (b) check boxes and radio buttons, (c) slider menu, (d) image map.

The menu items are usually subtasks to be invoked or the target interaction objects for the certain tasks to be operated upon. In either case, it reflects that the menu must be organized, categorized and structured (typically hierarchically) according to the task and the associated objects). At each level of the menu, the number of items should be managed to be ideally below the "Magic Number 8" (the limit of our short term memory). However, it may not always possible to achieve such a design or modeling. If long menus are

inescapable, at least the items should be laid out in a systematic manner, e.g. in the order of their frequency, importance, alphabets, etc.

- **Direct Interaction** - The mouse/touch based interaction is strongly tied to the concept of "direct and visual interaction." Before the mouse era, the HCI was mostly in the form of keyboard inputting of text commands. The mouse realized the direct metaphoric "touch" upon the target objects (which are visually and metaphorically represented as concrete objects with the icons) rather than "commanding" the operating system to indirectly invoke the job. In addition to this virtual "touch" for simple enactment, the direct and visual interaction has further extended to direct manipulation, e.g. moving and gesturing with the cursor against the target interaction objects. The "dragging and dropping," "cutting and pasting," and "rubber banding" are such typical examples.

- **GUI components -** Software interaction objects are in effect mostly visual. We have already discussed the windows, icons, menus, and mouse/pointer based interaction which are the essential elements for the "graphical user interface (GUI) and also sometimes known as the "WIMP (Window, Icon, Mouse, and Pointer) [ref]." While the term is deliberately named negatively to contrast it to a newer up-coming generation of user interfaces (such as voice/language and gesture based), WIMP interfaces have greatly contributed to the proliferation of computer technologies to the mass. We will take a more systematic look in the next chapter at the GUI components as part of implementation knowledge. For now, in

considering interface options, it suffices to understand the following representative GUI components, aside from those for discrete selection (WIMP) as already explained above, for soliciting input from a user in a convenient way.

- Text box – for making short/medium alphanumeric input.
- Toolbar – A small group of frequently used icons/functions organized horizontally or vertically for a quick direct access.
- Forms – Mixture of menus, buttons and text boxes for long thematic input.
- Dialog/Combo boxes - Mixture of menus, buttons and text boxes for short mixed mode input.

[Figure 4.14] GUI interface components: (a) form, (b) toolbar, (c) dialog box, (d) combo box.

- **3D interface (in 2D interaction input space) –** Standard GUI elements are operated and presented in the 2D space. That is, they are controlled e.g. by a mouse or touch screen and laid out on a 2D canvas (screen). However, 2D control in 3D application is not sufficient (e.g. 3D games). The mismatch in the degrees of freedom brings about fatigue and inconvenience. For this reason, non-WIMP based interfaces are gaining more popularity.

Aside from a task such as 3D games and navigation, it is also possible to organize the 2D operated GUI elements in 3D virtual space. It is not clear whether such an interface brings about any particular advantages because despite the added dimension, the occlusion due to overlap will remain as the interface is viewed from only one direction (into the screen). In fact, the user can be burdened with the added control if one needs to place or manipulate GUI objects in three dimensions. However sometimes it is employed anyway for 3D games sometimes for aesthetic reasons and the "wow" factor.

[Figure 4.15]: 3D interface in 2D interaction input space (e.g. mouse) for (a) 3D task such as spatial navigation[12] and (b) for 2D GUI elements laid out in 3D space[13] (mainly for futuristic "wow" factor).

- **Others (non-WIMP interfaces) –** The WIMP interface is effectively synonymous to GUI. Despite its huge success, since its first commercialization in the early 80's, in making the computers easy to operate in general, thanks to continuing advances in interface technologies (e.g. voice recognition, language understanding, gesture recognition, 3D tracking) and changes in the computing environment (e.g. personal to ubiquitous, sensors everywhere),

[12] SecondLife, http://secondlife.wikia.com/wiki/User_Interface

[13] EpicGames Scaleform GFx, https://udn.epicgames.com/Three/Scaleform.html

new interfaces are starting to making its way into our everyday lives. In addition, cloud computing environment has enabled running computationally expensive interface algorithms, which non-WIMP interfaces often require, over mobile devices against large service populations. The last chapters in this book will take a look at some basic implementation issues for these new non-WIMP interfaces.

4.3 Wire-framing

The interaction modeling and interface options can be concretely put together using the "wire-framing" process. Wire-framing originated from making rough specifications for web site page design and resembles scenarios or storyboards. Usually wire-frames look like page schematics or screen blueprints, as a visual guide that represents the skeletal framework of a website or interface [1]. The wireframe depicts the page layout or arrangement of the UI objects, and how they respond to each other. The wireframe usually only focuses on what a screen does, not what it looks like. Wireframes can be pencil drawings or sketches on a whiteboard, or they can be produced by means of a broad array of free or commercial software applications. Figure 4.16 shows such a wire-framing tool. Wire-frames produced by these tools can be simulated to show interface behavior, and depending on the tools, the interface logic can be exported for actual code implementation (but usually not). Note there are tools that allow the user to visually specify UI elements and their configuration and automatically generate code. Regardless of which type of the tool used, it is important that the design and implementation stage be

separated. Through wire-framing, the developer can specify and flesh out in more concrete manner, the kinds of information displayed, the range of functions available, their priorities, alternatives, and interaction flow.

[Figure 4.16] An example of a wire-framing tool [14]. Designing the content of a screen (left) and overall interaction behavior, e.g. how screens switch upon interaction (right).

4.4 "Naïve" Design Example: "No Sheets 1.0"

4.4.1 Requirements Analysis

To illustrate the HCI design process more concretely, we will go through a design of a simple interactive smart phone (Android) application, called "No Sheets." The main purpose of this application is to use the smart phone to present sheet music[15] and replace handling of paper sheet music (Figure 4.17). An initial requirements list may look something like in Table 4.2. Note that again this would be part of any software development process. Here, we only focus more on the HCI related requirements for brevity.

---

[14] FluidUI, https://www.fluidui.com/

[15] Sheet music is a written recording of music which is transcribed in music notation.

[Figure 4.17] No Sheets: Replacing paper sheet music with the smart phone. No more flying pages, no more awkward flipping and bag searching.

Table 4.2: Initial requirements for "No Sheets."

1.	Use the smart phone as to present transcribed music like "sheet music. Transcription includes only those for basic accompaniment like the chord information (key and type such as C# dom7), beat information (e.g. 2nd beat in the measure).
2.	Eliminate the need to carry and manage physical sheet music. Store music transcription files using a simple file format.
3.	Help the user effectively accompany the music by timed and effective presentation of musical information (e.g. paced according to a preset tempo).
4.	Help the user effectively practice the accompaniment and sing-along through flexible control (e.g. forward, review, home buttons).
5.	Help user sing along by showing the lyrics and beats in a timed fashion.

4.4.2 User Analysis

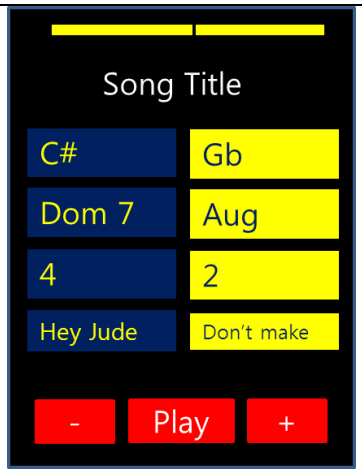
The typical user for "No Sheets" is a smart phone owner and novice/intermediate piano player (who wants to brag one's piano skill at a piano bar on the spot). Since a smart phone is used, we would have to expect a reasonable power of sight for a typical usage (e.g. a viewing distance

of about 50 cm subtending a letter of ±1 cm). There does not seem to be a particular consideration for a particular age group or gender. However, there may be a consensus on how the chord/music information should be displayed (e.g. portrait vs. landscape, information layout and locations of the control buttons, color coding method, up-down scrolling vs. left-right paging, etc.). A very minimal user analysis (that of the developer himself) resulted in (naïve first trial) interface requirements as shown in Table 4.3. Note that, for now, most of the requirements or choices are rather arbitrary without clear justifications. A revised design will based on a more careful user analysis/evaluation will be presented in Chapter 8.

[Figure 4.18] A typical usage situation for "No Sheets."

[Table 4.3] User interface requirements from a "very minimal" user analysis.

Display mode	Portrait.	
Layout	Top: Song title Middle: Chord – Beat – Lyrics Bottom: Control buttons	
Paging	Left to right	

	Current chord/music info in the left Next chord/music info. in the right	
Colors	Current chord: Yellow with blue background Next chord: Reversed Buttons: Red Background: Black	

4.4.3 Making a Scenario and Task Modeling

Based on the short requirements in Table 4.3, we derive a hierarchical simple task model as below and shown in Figure 4.19. Each task is to be activated directly by the user through an interface.

- Select song - Select the song to view
- Select tempo - Set the tempo of the paging
- Show Timed Music Information – Show the current/next chord/beat/lyric
  - Play/Pause – Activate/deactivate the paging
  - Fast Forward – Manually move forward to a particular point in the song
  - Review – Manually move backward to a particular point in the song
- Show instruction – Show the instruction as how to use the system

- Set Preferences – Set preferences for information display and others
- Show software information – Show version number and developer information

[Figure 4.19] A simple task model for "No Sheets." The top level application has six subtasks ("Select song", "Select tempo", etc.) and the third subtask, "Show music info" has yet another subtasks, "Play/Pause," "Fast Forward," and "Review."

The subtasks, as actions to be taken by the user, can be viewed computationally as event, or reversely as states that react and process according to the action events. Figure 4.20 shows a possible state transition diagram for No Sheets. Through such a perspective, one can identify the precedence relationship among the subtasks. From the top main menu (middle of the figure) the user is allowed to set/select/change/view the preference, tempo, song and software information. The user is also able to play and view the timed display of the musical information, but only after a song has been chosen (indicated by the dashed arrow). While the timed music information is displayed, the user can concurrently (the four states (or equivalently actions) in the transparent box in the right are concurrent) play/stop, move forward, and move backward. Such a model can start to serve as a rough starting point for defining the overall software architecture for No Sheets.

[Figure 4.20] A possible state transition diagram of No Sheets.

A storyboard is drawn based on the task model and for further envisioning its usage and possible interface choices. A storyboard is graphic illustrations organized in sequence and often used for pre-visualizing a motion picture, animation, and interactive experiences. There is no fixed format but each illustration usually includes depiction of important steps in the interaction augmented with textual description of important aspects (e.g. possible interface choice, operational constraints and any special consideration needed, usage contexts). Figures 4.21-25 show the initial storyboards for "No Sheets" illustrating the motivational context, a typical usage scenario and sequences, and rough mobile interface sketches.

[Figure 4.21] Motivational context for No Sheets.

[Figure 4.22]: A typical usage scene 1: The top level menu.

[Figure 4.23] A typical usage scene 2: Interface looks for three subtasks (e.g. song selection, tempo selection and showing instruction).

[Figure 4.24] A typical usage scene 3: Interface looks during "Play" and the three concurrently "activatable" subtasks (play/pause, move forward and move backward).

[Figure 4.25]: A typical usage scene 4: Moving between views/stages and quitting out of the application by using the standard Android menu button interface.

4.4.4. Interface Selection and Consolidation

Finally, we finalize the choice of particular interfaces for the individual subtasks. Table 4.4 shows the final decision and justifications. It is very important that we try to adhere to the HCI principles, guidelines and theories to justify and prioritize our decision. Note that we have started with the requirement that the application is to be deployed on a smart phone (the interface platform). Again, our initial choice will be rather not well thought out, just to illustrate that a naïve and hurried choice would expose the application to be at a great risk to eventually fail in terms of usability and user experience even if computationally satisfies the required functionalities. This will become more apparent as we evaluate the initial prototype and revise our requirements and design for "No Sheets 2.0" (to be presented later in Chapter 8). Figure 4.26 shows the final interface look and interaction flow using a commercial wire-framing tool.

[Table 4.4] Initial finalization of the interface design choice for "No Sheets."

Subtask	Interface Design Choice	Justification
Invoking main functions	• Touch Menu • Menu items in red	• Familiar interface • Catch attention
Selecting/changing song	• Scrolling menu • Return to main menu upon selection	• There may be many songs
Selecting/changing tempo	• Scrolling radio buttons • Return to main menu by ok button	• Only one tempo is chosen at a given time
Showing instruction	• Show a one page/screen image with condensed instructional content	• Present condensed content
Playing/Pause (view)	• Show progress bar on top • Control interface in the bottom • Provide sound beeps and vibration for 1st and	• Show status • Familiar interface • Use multimodal feedback for redundancy

	2nd beat • Color code different types of information	
Moving Forward (+)	• Forward button on the right	• Cultural consideration (moving from left to right) • Show status through progress bar
Moving Backward (-)	• Backward button on the left	• Cultural consideration (moving from left to right) • Show status through progress bar
Quitting	• Use platform button	• Use platform (e.g. Android) guideline

[Figure 4.26] Initial design wire-frame for No Sheets 1.0 using a wire-framing tool. Left: Icons and GUI elements in the menu in the left can be dragged on to the right to design the interface layer, Right: Navigation among the design layers can be defined as well (indicated with the blue arrows).

4.5 Summary

In this chapter, we described the design process of interactive application focusing on the interaction modeling and interface selection. It started with a requirements analysis and its continued refinement through user research and application task modeling. Then, we drew up a storyboard and carefully consider different options for particular interfaces by applying any relevant HCI principles, guidelines and theories. The overall process was illustrated with a specific example design process for a simple application. It roughly followed the aforementioned process but in a hurried and simplistic fashion leaving much potential for later improvement. Nevertheless this also reiterates that the design process is going to be unavoidably iterative, because it is not usually possible to have the provisions for all usage possibilities. This is why an

evaluation is another necessary step in a sound HCI design cycle, even if a significant effort is thought to have gone into the initial design and prototyping. In the next chapters, we first look at issues involved with taking the design into actual implementation. The implemented prototype (or final version) must be evaluated in real situations for future continued iterative improvement, extension and refinement.

Reference

[1] Brown, Dan M. *Communicating design: Developing Web site documentation for design and planning*. New Riders, (2010).

[2] Freeman, Adam. *Metro Revealed: Building Windows 8 Apps with HTML5 and JavaScript*. Apress, (2012).

[3] Freedman, Barak et al. "Depth mapping using projected patterns." U.S. Patent Application 12/522, 171. (2010).

[4] Miller, George A. "The magical number seven, plus or minus two: some limits on our capacity for processing information." *Psychological review* 63.2 (1956): 81.

[5] Petzold, Charles. *Microsoft XNA Framework Edition: Programming Windows Phone 7*. Microsoft press, (2010).

[6] Van Dam, Andries. "Post-WIMP user interfaces." *Communications of the ACM,* 40.2 (1997): 63-67.

[7] Microsoft "Kinect" http://www.xbox.com/en-us/kinect/, (2013)

[8] Nintendo "Wii" http://web.archive.org/web/20080212080618/http://wii.nintendo.com/controller.jsp, (2013)