Deploying Blaise to Tablet PCs for Mobile Use

David Hill, Senior Systems Analyst, Westat, USA

1. Introduction

In tablet PC computers, clipboard ergonomics and form factor may be returning to field interviewing in a smarter, more flexible device. There is no mistaking that a century of design and implementation development in the survey industry led to the preponderant use of hardcopy booklets on clipboards for field data collection. Using a pen to mark structured paper forms on the flat surface of the clipboard was preferred, an optimal combination of mobility, flexibility and effectiveness. Throughout the 20th century, the portable typewriter existed and could have been used for field interviewing but was not. From a usability perspective, portable typewriters strongly resemble laptop computers of today. Yet laptop computers are pervasive in current field computer assisted interviewing (CAI).

So far in the marriage of computer assistance and interviewing, keyboard devices have dominated due to their early development, consistent presence in available hardware platforms yielding software tuned to this as the input device and a history of user familiarity with keyboards. A long chain of evolving computing devices and uses relies on keyboards for input. First, keyboards predate computers on typewriters and this led to their adoption. Evidence lies in the continued presence of the QWERTY standard layout that originated on typewriters. For computers, the first major innovation from manually punched cards or operated punch devices was keyboard-based cardpunch machines. These machines strongly resembled typewriters (except they punched cards instead of impressed inked type on paper) so migrating the keyboard interface was obvious. When eliminating the cards evolved, computers took their input directly from keyboards, mostly in command form. Next multi-processing supporting many keyboard based terminals occurred. Then, CPUs became individualized in personal computers (PCs) while the "killer applications" of word processing and spreadsheets arose. Since both of these applications are text or digit based, keyboard input was emphasized in early PCs. Yet, as more diverse PC utilization, more applications, and advances in visual representations of data matured, the graphical user interface (GUI) won out. Fundamental to the GUI interface is interacting with the displayed graphics. To do so requires a pointing device. The mouse proved technologically practical and was embraced for this purpose. Thus, a keyboard and mouse became the standard PC interface. This determined that when PCs became small enough to be portable, a keyboard and pointing device was embraced as the main hardware interface. Portable computers quickly standardized on the clamshell laptop configuration fifteen or more years ago and have not since changed substantially, in this regard.

Laptops support general business portable computer usage well, no doubt determining the lack of continued configuration evolution. Plus, obvious interface alternatives, such as handwriting or voice recognition and non-manual pointing devices continue to be costly and suffer in reliability. However, while laptops support portable use, they do not support mobile use. That is, while laptops can be taken many places and set down--along with the user--to work, one cannot use them effectively while moving or standing. This is the essential difference between portable and mobile uses. Effectively using a keyboard-based laptop requires two hands. One cannot accomplish this while devoting one hand to hold the machine. A laptop must be set down on an appropriate working surface, since assorted attempted harnesses or stands for carrying laptop computers for mobile operation have proven unsatisfactory.

To date, the availability and cost points of laptop computers have determined that the field survey industry largely conducts CAI operations as portable implementations. Yet general field interviewing is not inherently a portable task. It is a mobile task. Or at least, if supported as a mobile task, it can also support portable implementations, while the converse is not true. This is the usability lesson we should inherit from the preference for the clipboard form factor of paper questionnaire administration.

Since many interviewing tasks involve moving, standing, or further set-up variations that do not accommodate laptop set-up, reliance on laptops for computer-assistance is a dead-end for mobility. Therefore, researchers designing survey projects tend to coerce the field environment to accommodate laptop use or abandon utilizing them. They must shoehorn their CAI use into portable (laptop) computer operations, without a practical mobile alternative. If portable set-up cannot be accomplished in the expected interview encounter environment, that choice eliminates the many advantages of CAI data collection. Field interviewing and operations software can fully support many mobile CAI tasks, yet a hardware platform to carry these applications has been absent.

Tablet PCs fill this prior mobile CAI device void and their flexibility, after overcoming other resistance points, may dominate most field applications. Just as the clipboard form factor dominated the survey world of paper form administration, the tablet PC is set to be the hardware platform of preference for field interviewers conducting non-trivial CAI.

The remainder of this paper contains the following sections. Section 2 discusses tablet PCs generally, focusing on distinctive software considerations and usage opportunities. Section 3 describes desirable elements of application interfaces for tablet PC operation. Section 4 discusses the utilization of Blaise on tablet PCs. This includes a review of strategies to deploy Blaise programs on tablet PCs and recommended modifications to Blaise software that would make it friendlier to tablet operation.

2. Tablet PCs: What are they? Why now?

Tablet PCs resemble hardcopy clipboards. They are slate style mobile computers. They use a flat display meant to be pointed at or written upon. The primary interface device is a stylus, essentially an electronic pen. As a pointing device, one points directly at the screen objects and touches the tip of the pen to the screen (taps) to select (the equivalent of mouse clicks). As a writing or drawing device, the tip is placed on the display surface and moved just like writing with an ink pen upon paper. Of course, electronic pens do not write tangible ink on the display surface but utilize virtual electronic inking that flows as a line on the graphics display. Attributes of electronic ink, such as color or thickness, are controlled by software setup.

While a number of previous generations of this form factor have been attempted (and generally called “pentop” computers), none of those attempts have achieved general commercial success other than spawning the next generation of devices. The difference in this device cycle is that the biggest organization in the PC industry is now embracing these devices. Microsoft has formed and proselytized a standard for this platform including hardware, performance and compatibility aspects that enable the machines to run the Windows XP operating system and achieve reliable, or at least predictable, operation of Windows programs.

Microsoft’s promulgation of a tablet standard and assurance that the Windows operating system functions on the devices made a new market for pen/slate computers. Numerous mainstream

hardware manufacturers (e.g. Compaq/HP, Toshiba, Gateway, plus others) now build devices[1] to this standard. This means slate style mobile PCs are no longer specialty hardware devices. Now they are commodity devices, another flavor of field-able PC computers suited to mobile tasks. A price premium[2] for tablet PCs exists over the more ubiquitous laptop models. This is a resistance point to general adoption.

For that cost difference, users buy practical mobile usability, in addition to portability. You can hold this slate form factor and use it while standing or moving. One hand supports the machine; the other operates it. You can easily set the pen down (or just drop it when the pen is tethered to the computer) and use the free hand for other purposes too. You can also set a tablet PC down on a surface, using it like a writing pad. Since the slate machine is flat, it does not have awkward balance and vantage point constraints as such laptop use. Also, when a tablet PC is set down for use, a now unencumbered second hand is free for some other action. Variously, one can prop a tablet PC display and use an auxiliary keyboard in a setup indistinguishable from a laptop computer (such dual use is the very design crux of the convertible form factor of some tablet PC models). In all, a tablet PC is more flexible than a laptop. It enables mobile use and supports a greater range of portable use too.

2.1 Software Considerations

Because tablet PCs operate a standard form of the Windows XP operating system, any application that runs on this operating system will run on tablet computers. This includes Blaise interviewing software. However, just running PC software is not the entire objective. The main reason to use a tablet PC is its mobile or more flexible ergonomic potential. Relying on keyboard input squanders the greatest strength and primary reason to use this platform. The pen interface is the obvious and practical mode of mobile and flexible input. This exerts presentation and interaction imperatives for applications intended for such mobile and pen/slate interface. These are alternatives to the historic keyboard input focus of traditional PC applications, including Blaise. Lack of such mobile tuning in software is another resistance point to general tablet PC adoption.

For pen focused operation, the crux software considerations are 1) the pen is most effective as a pointing device and the main desirable interface device for mobile or flexible machine use, and 2) what users point at is the direct rendering of the output on their screen. Neither of these is true

[1] The tablet market has split ostensibly into two, overlapping, machine configurations: pure slate and convertible form factors. Pure slate tablets look like electronic clipboards. They rely on pen input. You can plug auxiliary keyboards into them to achieve a laptop-like configuration. Convertible tablets are hybrid PCs that generally have a rotating hinge attaching a keyboard. In one hinge configuration they resemble laptops, although generally continue to rely on their pens as the built-in pointing device. Their rotating hinges allow the keyboard to flip around and lay flat against the screen portion of the machine. This yields a slate-like configuration (albeit with a thicker profile) where pen-only operation is the norm. The remainder of this discussion refers to slate-like use of the tablets, whether pure slate or convertible. This means pen interfacing, without the benefit, or variously the encumbrance, of a tangible keyboard.

[2] While a large price reduction in tablet PCs--over prior pentop PC models--occurred in this latest cycle, they remain more expensive than laptops. As relatively newly engineered devices, the full effects of mastering manufacturing specialization in tablet PCs, economies of scale and manufacturing cost-amortization among the producers are not yet fully expressed in unit prices. Plus, the presence of multiple producers making machines to the same standard creates market competition that is already exerting price reduction pressure and will continue to do so. This market effect did not exist previously for this class of machine, when each was a specialty device. Yet, the properties of tablet PCs determine that some price premium for such machines will continue since producing their pen-sensitive screens is simply more costly. In time, we expect this difference to settle under two hundred dollars, rather than the couple hundred dollars of the current market.

for keyboard operation and while a mouse is a pointing device, it does not directly point[3] at the target objects. In pen operation, the output surface for user interface is literally the input surface too. This triggers a number of immediate effects related to human skills. For example, since the pointing is direct, the accuracy of that pointing is limited by the ability of human users to accurately point at the target object. In practice, this proves less accurate than mouse pointing. Plus, there is no opportunity to adjust the sensitivity of motion, since there is no indirection in the portrayal of the movements. In a tuned pen interface, this suggests that target objects should be larger for pen pointing. This suggestion coincides with another factor on tablet PCs. Mobile users are typically less able to focus just on their computer and program interface. Mobile use is often required because the users are engaged in a further task(s) at the same time or are positioned in a less ideal, although more flexible, posture. This places more emphasis on clarity of layout and interaction. One element of that clarity can be more conspicuous objects, by being larger.

More detailed software considerations also appear in the tuning of application interfaces for tablet PCs. One area where keyboards excel is open text entry. Possibly the best means to address this in tablet operations is designing the instrument to avoid non-essential open text fields. This design objective is shared with web applications where processing and transmittal of open text data is less desirable in web browser/server architecture than immediately interpretable actions. Still, invariably in interview instruments, some need for open text entry arises and effective and well integrated alternatives are needed. A body of creative techniques is emerging as a richer set of pseudo-standard Windows objects that service open text entry without a tangible keyboard. Fully pen tuned applications should integrate these objects to advantage.

The screen of a tablet computer is its largest tangible hardware feature. The outside dimensions and shape of tablet PCs correspond to the screen display. Those dimensions now proportionally mirror conventional PC displays. That is, one dimension (typically horizontal) is longer than the second dimension (typically vertical). Unlike traditional PC monitors, tablet PCs can be rotated. In practice, holding tablet PCs by their narrower dimension is easier ergonomically. The Windows XP tablet compliant operating system supports this display rotation very well at a BIOS and OS level[4]. But this does not automatically solve application screen layout issues. For the application software, this hardware rotation converts the display screen from the traditional landscape orientation (horizontal wider than vertical) to a portrait orientation (the reverse). This is a familiar issue since printing output to a standard page of paper carries a corresponding dimension switch. Still, in tablet operation the issue is deeper because the rendering is not only an output format but must support input interaction too. Printed pages are only an output rendering, such that simple pagination of printed pages (both horizontally and more frequently vertically) often suffices for this dimension switch. The screen equivalent is scrolling and fully supportable in Windows programs, but generally not an optimal solution for sparse input

[3] Notwithstanding some early GUI advocates assertions so. I believe the assertion of mice as direct pointing devices was an effusive description to emphasize the utility of pointing at active objects introduced in GUI interfaces. Also, at the time, no other practical pointing device could make a more substantial claim to direct manipulation, so this elevated the status of mice. In reality, you roll a mouse on a secondary flat surface and that two dimensional movement is transformed to corresponding cursor movements on the screen. This is not direct pointing.

[4] Rotation of the display between landscape and portrait orientations was lacking or weakly supported in previous pentop OSs. If present, it was typically a video driver flip of the image performed as a last display manipulation. These drivers could be specific to video hardware and their precise dimensions. The OS had no comprehension of more than one screen display mode. Nor was there any generalized scheme for display dimensions across video devices. Therefore, software applications had no means to be sensitive to the mode of operation or a way to accommodate detailed dimension variation. Windows XP OS supports all of this.

interfaces. This dimension layout issue can be further aggravated by a desire for larger objects to facilitate pointing accuracy and clarity. It causes more expansion of the horizontal size needed for a full display of the application window. A sure sign of an un-tuned tablet PC application is when critical operation menu buttons disappear off the screen to the right, when presented in portrait screen orientation, and no means exists to scroll to the right to ever view or click on them. Even a number of Microsoft's purportedly tablet-friendly core applications suffer from this limitation. It is a pitfall of assuming that just because traditional Windows programs can run on tablet PCs, all usability aspects are satisfied. Or, it may determine that the program must always run in landscape mode, disregarding the ease of possibly holding the machine by another dimension.

2.2 Opportunities in Tablet PCs

While detailed points of software considerations above appear as a litany of negatives regarding pen/slate interfaces that must be overcome, one very persuasive positive should be remembered. Pointing is natural in human communication, perhaps innately so from pre-speech survival of the species. This is expressed in small children that naturally point before they can talk and at a developmental stage turn anything stick-like into a pointed toy. Later in development, one commonly must teach children not to point, as an avoidance of a socially unacceptable gesture, rather than to point. Typing on a keyboard or manipulating a mouse are learned skills that commonly are acquired later in age and require practice to sustain. Only expert typists ever make the leap to non-cognitive use of this skill. In contrast, often in fatigue, injury or other circumstance when verbal expression evades us, we retain the inclination to point as a communication skill. All of this is a practical clue of the natural human comfort with and effectiveness of pointing as a gesture. Capitalizing on this and applying it as the main action in any interface where greater flexibility is desired, is a self-evident advantage. For mobile PC computers, the challenge is overcoming years of bias toward keyboard operation built into applications due to the historic dominance of keyboard hardware.

A second opportunity delivered by tablet PCs is simply for CAI to "go where no one has gone before." Because tablet PCs confer viable mobile PC use, projects can realize CAI advantages in data collection in non-trivial scenarios previously unavailable (while trivial instrumentation migrates to palm or PDA devices). This can include such common interview scenarios as complex and rigorous doorstep screening of households in area probability samples or individuals circumstantially picked in encounter situations; interviewing respondents on the move or flexible situations where an obvious place to setup a laptop computer no longer determines session viability; supporting computer assisted data capture of complex observations where the user or observed event is mobile (such as moving around a hospital ward or a school wing); or encompassing field operation activities that are essentially mobile (like listing operations) combined with CAI on a single integrated machine platform. Also, because this cycle of tablet PCs operate a standard Windows XP operating system version, the full richness of the Windows environment becomes instantly accessible to these new mobile uses. No complications arise on standardization of machine resources, interface look-and-feel, core program operation and programming, external devices and ports to access them, or the basic inter-operability of instrumentation with other Windows computers (for example, a multi-modal operation where just some machines are tablets). The wealth of commercially available Windows-based programs or peripheral devices can be combined readily with tablet computers. Thus, such project functions as security, network connectivity (including wireless), data transmission, or case management applications that most organizations have solved already for the Windows environment do not require reimplementation for tablet PCs. Further, a new body of peripheral devices already integrated with the Windows OS is available for ready project integration, such as GPS units,

authentication tokens, scanners, printers, audio recording, digital cameras, external storage devices, etc.

Another opportunity in tablet PCs is the practicality of new data types. This has two major directions. First, through the use of assorted input devices (mostly external peripherals), new forms of data are viable and their integration is readily supported by the Windows OS (noted above). For example, a digital picture may be more effective than an observer description, or recording the sound of the interview may be more effective for validation and is certainly less expensive than in-person supervisory observation for technique coaching of interviewers. While not strictly a tablet advantage, many of these peripheral forms are more compelling because of the mobile character of tablet PCs. For example, the use of integrated and automated GPS recording is not compelling if the PC capturing the data does not move much or is only setup and operated inside buildings where the GPS signal cannot be recorded (as typical for laptops). Another great use of GPS technology in support of field staff is coordinated use with real-time mapping software to assist in location and route finding, or even location logging. This application of the technology can enhance staff effectiveness and reduce labor and travel costs. But, it only makes sense if the users have access to mobile, not just portable, computing. The second form of new data type utilizes the pen interface to draw, not just point. This enables user drawn diagrams, annotations added to other images, or just freeform notes (retained as an image, rather than coerced to recognized text). This data type is not practical via a keyboard or indirect pointing device (mouse) input interface. Specific extensions to Windows XP, in newer versions, support electronic ink as a new native data type to facilitate this form of data. Also, built-in objects for raw inking and application programs for rich note taking exist in the OS DLLs, available for application integration on tablet PCs.

The greatest tablet PC opportunity is the potential to implement user interfaces precisely developed and implemented to serve the task of interviewing (or other forms of CAI data capture). Going one step further, individual project implementations could be customized for greatest operational effectiveness. This can be done as a software specialization rather than a hardware modification. To date, almost all CAI implementations occur on hardware designed for general business use, not field data capture. The reason is the cost-effectiveness of buying the going PC models on the general market. But, PC manufactures inherently design their machines for their largest market sector, which remains general business users, or for portable machines, these same users on the go. No one designs PC hardware specifically for interviewing. Doing so is cost prohibitive. Leaving the survey industry to muddle through making the best of using hardware devices not designed for their specific use. For example, all laptop computers come with keyboards of *N* keys. But for the vast majority of enumerated interview questions, how many of these keys are actually used at any one time? Even being generous by including all navigation options this number is less than a dozen. Meanwhile, the basic keyboard interaction for an enumerated question is entering an alphanumeric coded response, the classic being 1=Yes and 2=No. So users must elicit the response, correlate it to a coded answer, locate that key(s) on the keyboard and press it to succeed in reliable data entry. While that numeric coding of responses may be second nature to researchers and programmers inured to abstracting data in coded form, there is nothing innate about it to interviewers. It actually takes a cognitive mental operation to transpose the response to the code and enter it (albeit one so often repeated that it may become ingrained). Why do we continue to rely on this as the main input interface: history, habit, and a lack of alternative via keyboard devices. Most interview software, including Blaise, already provides for pointing and clicking at the desired response object or label with a mouse as an alternative to key entry. This eliminates the cognitive encoding step. Still, the mouse is an indirect pointing device and is not well suited to portable set-up or portable built-in devices remain clumsy. This keyboard coding is so familiar that we rarely question it. Our best users are

so facile at it that changing the interface, thereby needing to convert the behavior of existing users, forms another resistance point. This is force of habit or resistance to change.

Tablet PCs, or more specifically their slate style input surfaces, offer a compelling alternative to the core key entry interaction described above. The critical differences are that the output and input surfaces are the same and what shows on the screen, that users then interact with, is entirely under software control. These differences are subtle, but important. They cause an evolution of the GUI principle of what-you-see-is-what-you-get (WYSIWYG) and indirect interaction with the graphics via a keyboard or mouse. On a tablet PC with a pen, now what-you-see-is-what-you-do and the interaction is direct. If the responses for a question are two enumerated choices and three navigation options, via software control we can put objects (rendered in any manner of our choosing) for those, and only those, interactions on a tablet display. The user selects one by pointing directly at and tapping the screen surface with the pen. This is a precise, direct interaction customized to a desired level of detail, under software control. Organizations can make full software development choices regarding the degree of desired customization. We do not have to convince a hardware manufacturer to build us a different keyboard or face static input mechanism arrangements of hardware ["Now let's see, where's that one key among the *N* pieces of plastic I must push to record this response?"] If we now desire to change it or scrap it, we do so by changing the software driving the output object display. If we want more copies, we load more software. If we want two variations simultaneously, we implement a programmed logic branch or make configuration options in one software driver. And for users, we can display any sort of label or image on our input objects and readily change them to suit each instance. ["If the answer is yes, tap the 'yes' button."] CAI processing is fully capable of recoding or deriving that on-screen button press into most any recorded data we desire. All of this enables us to make spectacularly capable and responsive interfaces for our users, with incredibly detailed variations when desired. More likely, we will rely on standardized software for most uses, as less development intensive and more user consistent, and therefore more cost-effective. Still, we have that choice and an untapped potential to do better for our users.

3. Desirable Elements of Tablet PC Interface

Before launching into details on Blaise implementation on tablet PCs, identifying the driving points of the desired interface is relevant. These can be reduced to a couple over-arching principles: visibility, clarity, size, easy interaction, screen arrangement, and open text alternatives. Since these may be principles of all user interfaces, rather than recapitulate the entire Windows GUI standards on these points, this discussion focuses on tablet PC variations with that standard or entirely novel angles. Each item appears individually below.

3.1 Visibility

Visibility of any computer screen and what appears on it are threshold issues for all hardware and applications. The hardware aspects become more critical in portable PCs since the environment of use is less controlled and the resources of the machine are more limited (e.g. size of display, battery supply to power luminescence, etc.). For a mobile PC, the environment gets even less controlled and often even hostile to visibility. Extremes like standing outdoors in bright sunlight or heavy shade can be commonplace for mobile household interviewers. Moreover, while what-you-see-is-what-you-do may be an interface usability advantage, it makes visibility issues more stark. One cannot do what one cannot see.

What can be done for tablet PC use to address visibility? First, get good hardware. Most standard tablet PC screens function decently in common indoor lighting. However, many do not

show well in outdoor light. A number of tablet PC manufacturers make outdoor visibility enhanced tablet displays. Of all the hardware upgrade options[5] this is likely the most valuable on tablet PCs for mobile use.

Some application behaviors in the rendering of screen output can contribute software pluses in visibility. The selection of colors, or lack thereof, can improve extreme lighting visibility. Tablet PC displays can render the full range of Windows supported colors, but not using them in mobile applications, or using them sparsely, can improve visibility. The reason is that the contrast of adjacent imaging is most significant in distinguishing it in very bright or dim lighting. The highest contrast achievable is returning to black and white display schemes. Removing color from the displayed imaging is literally returning to shading that is physically the greatest contrast. The next closest arrangement is juxtaposing exactly complimentary colors. However, this is not as good since the color itself can cause refraction of the desired high contrast for best visibility, can lead to very garish appearing screens, or more likely presents resolution problems for color blind users.

One more software visibility improvement is making objects bigger. Bigger images are inherently easier to see (especially for middle-aged and older users whose eyesight may be suffering from aging effects). Bigger objects are described in more detail below.

3.2 Clarity

Every interface design strives for clarity suited to its use. It must be clear what and where to accomplish some interaction. For tablet applications, especially in mobile use, this is particularly acute. The reason is that the utilization of mobile computing generally implies the user will be doing something else while working with the computer. This may be as mundane as just standing while working. It could be another distinct task. It likely means users cannot devote their full attention to what appears on the screen. They probably will glance at the screen to interact with it, but otherwise, their attention will be focused elsewhere[6]. Therefore, when users glance back at the screen, clarity of presentation will aid them in visually acquiring the point of interaction and the operations applicable to it.

This suggests that the sparsest practical screen presentation works best. Applications should only present those images that significantly contribute to the user performing the immediate task. Even the Windows GUI convention of graying-out disabled buttons is dubious here. First, in tough lighting, the distinction of graying-out can be lost. Second, it takes a moment of cognition to discern the graying-out and its meaning. The scheme of only showing what the user can do at any given time is more pragmatic. This reinforces the working principle of what-you-see-is-what-you-do. Conveniently, a sparse layout also saves screen space so the remaining, most important displayed objects can be enlarged to optimal extent. Also, response indirection or multiple places of representation should be avoided. Give users one place to interact and make it easy to visually locate. And big, so it is easy to access. All promotes clarity.

---

[5] Upgrade options for tablet PCs resemble other PC upgrades: faster CPUs, additional memory, assorted drives, software bundles, etc.

[6] This generally applies to all in-person interviewing. Establishing rapport with respondents, eliciting responses, probing and further interviewing techniques should be the focus of interviewer attention. More bluntly, interviewers should be interviewing; they are not dedicated computer operators. Computer-assistance should be a tool to conduct the questionnaire, not requiring the focus of the operator during live interviewing.

Other points of clarity are grouping and placement of items on the screen (covered below in screen arrangement) and making the necessary interaction obvious (covered below in easy interaction).

3.3 Size

On tablet PC screens, size matters. Regarding the hardware, tablet PCs exist due to a relationship with size and functionality of the human hand and an intrinsic, historically proven, general form factor. Tablet PC screens are relatively large, for handheld devices, which accomplishes better visibility, good manual interaction, and space for the composition of non-trivial applications. Current market conditions also appear to be coalescing around two size dimensions: larger for general users (approaching 8.5 x 11 inches); smaller for the most mobile, limited task oriented purposes (5 x 7 inches and smaller). Among those largest displays, dimensions are limited by the desirable size of the machine footprint. Two manufacturers have even enhanced their screen sizes slightly larger by expanding the screen to the very edge of the machine surface, but not expanding further.

In application software, large size objects enable a number of other benefits, already mentioned. Seeing large objects is easier. Clarity is promoted with fewer, larger objects. Because there are fewer, space becomes available for making them larger. Interacting with them is easier (more below). Further, the accuracy of pointing with the human hand does not correspond to the pixel accuracy of high density PC video. User lab testing indicates that the smallest area for reliable pen pointing/tapping is half a centimeter or larger[7]. In some Windows applications, this is approximately the size of some menu or scrolling buttons. Multiples of this size increases ease and accuracy of use by corresponding proportion.

3.4 Easy Interaction

Large size is one way to increase ease of interaction in tablet applications. A large hot spot area for handheld pens to land on, without undue attention to intricate accuracy, works better. Easy interaction also arises from visibility and clarity since the operative principle is what-you-see-is-what-you-do. Two further properties that apply here are the obvious functioning of object controls and suitability of a control for pen operation.

Alongside large objects that are clearly visible on the screen, easily recognizing what to do to make the object function is necessary to qualify as easy interaction. Naturally, a judgment may vary with the type of user, their level of training and experience with the application. What is obvious to one person may only be so because of their familiarity with the subject. Emergent Windows GUI and web browser standards define a decent set of objects for general users. With minimal training--if not known already from experience--most users can readily recognize click icons, radio buttons, check boxes, drop down arrows, scroll bars, spin controls, plus some others and hybrids. One caveat is that they may appear slightly unusual in large sizes (e.g. large radio buttons often look odd to background users when first encountered). The commonality of all these controls is their visual cues. Relatively consistent operation is also important. Not displaying a drop down arrow on a field controlled by a drop down box object can stymie a user. They get no cue to drop down the applicable selection list. They may have to tap around to discover this action or may conclude they have reached a terminal interface path without this discovery.

[7] Jarrett and Su, Building Tablet PC Applications. Microsoft Press, Redmond, WA, 2003. p. 31. See also Part I, chapter 2 for a general discussion of layout sizing of tablet PC applications.

Another easy interaction criteria for tablet PCs, is whether the control works well with a pen. Any object that requires multiple actions, like typing this sentence as text from a keyboard, is going to be less effective via pen input. Pens are good for pointing and simple selection. Avoiding intricacy and multiple trigger actions are good rules. For example, scrolling long lists, while possible with a pen, tends to 1) get too intricate for precise list positioning, 2) expanding the size of the function elements of a scroll bar can actually shrink the travel space of the scroll bar exacerbating the precision effect, and 3) scrolling and selecting takes multiple actions to control. In terms of simplicity, the set of controls of radio buttons, check boxes and drop down lists can accommodate much of plain interviewing interaction well. The reason for including drop down lists here relates to clarity. Moderate lists of radio buttons or check boxes can lead to visual clutter, especially if combined with a multi-field, form layout. A sparser layout, accepting that one tap pulls down the list of contextual responses while a second tap picks the desired one, can be more effective (even though it is literally two actions).

3.5 Screen Arrangement

The windows metaphor of the Windows OS is a screen arrangement protocol. GUI standards add another layer of conventions. However, due to the variation in tangible use of tablet PCs, some screen arrangement assumptions should be changed. The most significant screen arrangement item lies in response to users covering parts of the screen when reaching over it with their hand to interact with the surface. Unfortunately, the largest covering happens directly at the point of selection when the pen taps down on the surface. The hand is close to the selected item and closest to the screen surface, blocking more of it. The area of coverage comes from the lower corner of the screen, from where the user reaches their hand out, and for whichever hand the user is holding the pen. Since most of the population is right handed[8], this will be from the lower right. One advisable application feature is to make advancing to another screen a two tap action by placing a next button to move the user's hand away from covering the screen (i.e. lower right or left). While software is clearly capable of a tap-and-go-next action sequence, this is clearly not desirable regarding input recording accuracy. Getting the user's hand away from the screen, visually revealing their choice, leaves the opportunity to recognize and verify the correct item/response was recorded before moving to the next screen.

Grouping related items in presentation can aid the user in locating where the next interaction needs to occur. Crossing this with sparse presentation, this suggests presenting most single interview questions one at a time. Everything the interview requires to present the query and enter the response is one display unit, unless compounding with multiple fields that form one question unit. Some additional background operations are needed on most interview screens, such as just moving to the next item, backing-up, recording missing values, or calling more detailed help and should be readily accessible. Still, these are subsidiary to the question item of focus and should be treated visually so to promote clarity. This suggests that the main item of attention should be placed as the main item of the screen.
Assumptions about the prime space on a tablet screen are different for tablet PCs than keyboard and monitor based machines. In monitor/keyboard screens the prime area is center and then the closest area to the placement of the hands (that is downward). When holding a tablet this changes to the center and upwards, since the hand is held over the screen itself, in fact covers portions of it

[8] Organizations may want to make a choice whether to rearrange the screen for the roughly fifteen percent of the population that are left handed and therefore likely to reach across the tablet screen from the other direction. Note, most left handed people have pretty good coping mechanisms to deal with designs for the right handed majority.

toward the bottom, and most likely, users are looking upward to see any concurrent activity during mobile use (i.e. to make eye contact with respondents when interviewing). Placement assumptions change with this change in importance of screen areas. For example, placing general action menu bars across the top of the screen emphasizes them on a tablet display, since they occupy some prime screen area, not matching their functional use as subsidiary operations to the main action. Therefore, on tablet displays for mobile use, moving menu bars to the bottom of the display is advisable.

3.6 Handling Open Text and Alternatives

Many PC users take for granted that entering open text always exists. Until now, PCs have sported keyboards from their earliest commercial release and emphasized them. With tablet PCs you can either hinge the attached keyboard of convertible models into position or attach an external keyboard to a pure slate model (either by plugging or infrared port alignment) and conventional PC operation is achieved. However, with slate use of tablet PCs, eliminate the keyboard and how to handle open text? The machine does not limit keyboard use but the core purpose of utilizing a tablet (mobility or flexibility) makes it difficult.

The obvious open text solution when holding a pen device is to use it as a writing implement. Functionally, this is fully possible on tablet screens. In software enabled boxes, electronic inking works well, identical to tangible ink. But troubles arise when trying to recognize that freeform ink as text data. Handwriting recognition (HWR) is required. This is a topic, almost a full discipline, unto itself and beyond the scope of this paper. Suffice to say that applied computer scientists and experienced users commonly advise not relying solely upon text recognition. Some people can use it with great success. For most people the reliability is mixed. Some users can never get it to function. Therefore, the wisdom for tuned tablet applications is provide for HWR when relevant but leave another method for last recourse text entry.

Pop-up keyboards or keypads are the general alternative to open text entry on tablets without relying on HWR. These are pop-up windows with a series of button objects that commonly look like tangible keyboards and behave similarly. Tap a key on a pop-up keyboard and the corresponding letter or digit is entered in the active field. The Windows XP OS standard now includes a pop-up keyboard feature[9]. A secondary software market has also appeared providing alternative and specialized on-screen keyboards that mostly pop-up too. Many pop-up keyboard windows include multiple layout, function and presentation options. However, there are usability problems with pop-up keyboards. A simple one is that the pop-up keyboard takes some screen space and therefore covers something of the application window(s) beneath it. Making the pop-up keyboard drag-able solves this, remaining just an inconvenience when it must be moved out of the way. Another problem may be the activation of the pop-up keyboard. For easiest use, the keyboard should pop-up automatically when relevant or be a background utility that may be activated whenever needed to fill a text field. However, typically the application interface knows when the context may require open text entry, but a background utility is typically an operating system controlled process. Or, the program running on top must reach into or allow process invocation into a lower layer. This does not occur automatically. Applications must be programmed to do so. The Windows OS convention for this is placing a fixed icon on the desktop toolbar that activates the standard pop-up keyboard[10]. This is fine for general users

[9] This pop-up includes multiple input features, one of which is a keyboard. Other features include HWR boxes that then write the recognized text to the active field or voice recognition integration.

[10] Some vendors add a BIOS feature where the keyboard may be invoked through a small icon trailing certain field invocations (although presence or lack of this trailing icon can be problematic to precision

whom retain access to the toolbar. But, many specialty applications (Blaise included) seek to isolate the users from free access to the toolbar, to keep them focused on the special task or prevent inadvertent changes of window focus.

A more intrinsic problem is keyboards were never designed nor intended for this interaction. Organizing the many hot spots of a virtual keyboard by the metaphor of a tangible one is readily recognizable, both in look and function. A potential confusingly large set of distinct key actions are organized at once in a familiar form. Yet, that familiarity leads to another problem. All users hunt-and-peck on pop-up keyboards, even expert typists. While the appearance and function may be familiar, the action is not. The reason is that users interact with such a keyboard image via a pen. A pen is a singular pointing device, only having one selection point. On tangible keyboards, good typists use multiple parallel fingers, each with a distinct position and ability to press keys. The pop-up keyboard interaction is no different than using a tangible pen to type on a tangible keyboard. It is relatively slow and laborious. Keyboards simply are not ideal as a single point/select interface.

If an OS built-in solution for handling open text is not optimal, what are the alternatives? These involve two tactics (possibly in combination): design away the open text or finesse the application interface for tablet use. Both require some attention to tuning for this platform. Examples of designing away open text entry in interview applications include re-sequencing or perhaps extending question sets, into more precise querying or response categories, to eliminate open ended responses that necessitate open text entry. Another variation is decomposing responses into compound pieces that can become category or list selections. For example, a year entry field can be broken into a drop down with categories combining century and decade (the first three digits or a four digit year;) and then a second drop down of year within a decade (fourth digit of a four digit year). While an unorthodox division of a year date, this is effective by needing a small number of actions (2) against reasonably sized category lists (10 each to span 100 years). If drop down lists are situated on the actual entry fields, the interaction is cued in place and therefore easy to discern and operate.

While liberal and creative use of list selections or categorized responses, or careful crafting of questions and responses can do a lot in interview applications, invariably some required open text field still arises (such as names, addresses, other specify description, remarks, etc.). Pop-up keyboards or HWR—for those that can accomplish it—is tolerable in small doses. There is one more option. Taking the open text as freeform writing and retaining it as inked data, rather than converting it. This does not work for any entry that will be reused later in real-time, such as a name that is later displayed as question text fill. Without such additional reference, it does work well. For example, instead on insisting on real-time text recognition for an other specify field, take the response as ink and store it. The new Windows native "journal" data type (specifically for electronic inking, also supporting inking combined with other graphics, thus supporting annotations) is intended for this. In addition to the ink imaging, all the stroking and pace attributes of the electronic inking are saved in this native type so no degradation of later recognition occurs[11]. If the user can return to the inked text, at their convenience later, they can perform the recognition then. Or for an item such as other specify text, the inking can be passed

---

field sizing or placement) or by right-clicking (performed as a tap while depressing a button on the pen shaft with a stylus) a text field and selecting a pop-up keyboard item on a pop-up menu. These are techniques carried forward from prior pentop OSs, but are additions to the Windows XP standard.

[11] This was a weakness of earlier electronic ink/HWR implementations. The HWR had to be performed in real-time or the subsidiary ink attributes were lost because only the image of inking was stored permanently.

downstream in data flow to other human users for interpretation[12]. This method can be applied to other situations by analyzing when an open text entry really needs to be stored as text data. If not, retention as electronic ink in real-time is viable.

4. Blaise Use on Tablets

As mentioned, Blaise as a Windows program functions on tablet PCs. Still, while not abjectly unfriendly to tablet operations, Blaise is not tuned for it either. In its out-of-the-box configuration, the history of Blaise as a keyboard application is apparent[13]. For hardware and usability reasons regarding general field interviewing as a mobile use explained above, I suggest that ongoing Blaise development should be focusing on such software tuning factors for pen/slate operation. Doing so not only anticipates a probable eventuality of tablet-like field computer use, but by removing a resistance point for adoption, may speed that eventuality to fruition.

Westat currently has a number of field projects utilizing tablet PCs. The most developed are complex, custom implementations aimed at full utilization of the pen/slate interface. We also have implemented field data collection using Blaise instruments deployed on tablet computers. In doing both of these, we have gained insight on the potential of the pen interaction supporting interviewing and the pluses and minuses of more conventional Blaise software.

Blaise is clearly the leading COTS interviewing software. We are delighted that, as a Windows software product, it operates on tablet PCs out of the box. The programming, data preparation, instrument infrastructure, data storage aspects (all the non-presentation and interface) of Blaise function here, just as on other Windows PCs. However, having identified the output presentation and input interaction as key aspects of optimal tablet PC operations, these aspects work at a basic level for Blaise out of the box. Still, Blaise is a good example of application software that has inherited, for historic reasons, a user interface cultivated around keyboard input. As such, its operation with a pen interface shows some problems. Of these problems, existing presentation customizations, and therefore programmable variation, can address much of the output tuning. However, the input tuning is more dependent on supported system operations and therefore more limited.

4.1 Strategies for Blaise Implementation

The ideal for any software integrator is finding great applications perfectly suited to their particular operating environment, out-of-the-box or with very limited setting of prepared configuration options. However, expecting any one software package or system to deftly support all environments clearly is unrealistic, especially for relatively new platforms that may vary with the legacy ones for which the software was designed. If the ideal is unavailable, the alternative is finding software with configuration depth, modifiable or reconfigurable to satisfy the amount of precise variation desired. Blaise qualifies as this software for interview applications. We can even conceive of a number of strategies for how Blaise might be tuned for better pen/slate operation.

---

[12] In my experience, all other specify responses are passed to a backend data coding shop where a data clerk or subject expert reads them and back codes or specially codes the response or accumulates a list of extraordinary responses. If a human is interpreting the text and the data infrastructure is in place to pass the inking (as a blob field) to the point of interpretation, they can just as well read freeform inked writing as fully recognized text.

[13] If, for no other reason, there are some actions that only can be triggered by keyboard action (e.g. popping-up a table-based item search with a Backspace key press or mapping actions to function keys).

In picking strategies, we start with simpler ones and advance to the more complex (although a clear effort to reward ratio exists). The first strategy is the ideal: what is useful in Blaise ostensibly running in its default state, out-of-the-box. The next strategy is using the full depth of Blaise offered options, properties or programmable features to achieve a particular presentation. A new strategy is treating the instrumentation as a Blaise IS application (web instance). This may be an obvious approach in multi-modal instrument applications (where both CAPI and web-CASI administration of the same instrument is needed). Still, a local instance of a web server can be deployed on distributed PCs and the browser interacts with that mobile server. Regarding user interface details, this pushes implementation into the domain of web browser operation or what can be accomplished using web screen rendering (i.e. HTML, XML, XSL, Java Scripting). Another more intensive Blaise use on tablets strategy could arise from utilizing alien routing and replacing the I/O aspects of operation, at proportionally more customization effort. The last strategy takes this a step further. A new application can be programmed that only uses a Blaise backbone for instrumentation, via Open Blaise Architecture access, but sustains the whole session operation itself, including the user interaction. All of these amount to further strategies for Blaise implementation. We will try to put them in context.

4.1.1 Blaise, Out-of-the-Box (Default)

As noted, Blaise readily runs on Tablet PCs[14], directly out-of-the-box since it is a Windows program. However, as established in most of this paper, simple running on tablets in default configuration as a Windows program does not amount to easy use in pen/slate operation nor is advisable to do. With any project software implementation, much depends on the requirements, schedule, available resources and budget, and the culture of the implementing organization. For something fast, small in sample size or just trial, using out-of-the-box Blaise can be done.

Problems encountered here spread across usability criteria and generally can be labeled as a modest lack of pen operations tuning. The outset trouble may be visibility of the initial Blaise windows. While the blue, cream, grey and black writing color scheme is pleasing to the eye and a good branding of the software, it is not high in contrast. This means it washes out in bright lighting, (particularly the black on grey form pane of the screen) limiting visibility. The default sizes of the screen output verges on too small for effective visibility beyond a controlled PC use environment. For clarity default Blaise does well. The three pane organization of default Blaise screens conceptually separates operation areas, organizing output and input interaction. Consistency contributes to clarity here, even though intuitive visual cues are somewhat lacking. The redundancy of the info pane and form pane, with two expressions of an input field is not helpful to tablet/mobile users, or certainly occupies more screen space than worth any contribution to tablet operations. Regarding size of input objects, the defaults are plainly too small for easy pen interaction. Many objects are at or smaller than the half centimeter minimum sizing consideration for pen operation. Making everything larger to better utilize screen area would improve pen operability. Screen appearance of default Blaise operation roughly matches general business use software. The largest negative is placement of the menu bar which leads to more hand coverage of the screen display to get to regular application buttons (including the next button[15]), rather than less. For open text alternatives, default Blaise offers none beyond what Windows XP provides in the background. While features exist in Blaise 4.6+ to call a pop-up

[14] Referring here to the operation of the DEP on a programmed Blaise instrument.

[15] This is the only means to regularly advance to the next item that is display resident in default Blaise screen operations. The only other default method would be popping-up a keyboard and taping the enter key. This later method is much too burdensome for constant use between questions.

keyboard or HWR overlay, these amount to setting detailed field properties and therefore are more substantial than the default configuration. This leaves only standard Windows XP pop-up keyboard and HWR options. If the Windows desktop toolbar is not retained on top of the application, the activation icon for the Windows pop-up keyboard is unreachable. Also, some default Blaise operations require keyboard keystrokes to activate. Lastly, default Blaise is clearly screen oriented for standard PC monitors. It has landscape orientation dimensions with no easy configuration option or system states to vary the orientation. Further, even if the application were left as landscape orientation and the hardware rotated the screen presentation, significant screen components like the next button on the menu bar and later columns of multi-column answer fields start spilling off the right edge of the screen without any means to scroll in that direction to move them back onto the display for input interaction.

If the objective is running something very rapidly and there is recourse to using an external keyboard for the terminal issues, then perhaps one can show something. A summary judgment on the advisability of relying on this for production data collection is likely no. The system and application programs can run but users will quickly encounter multiple, and perhaps insurmountable, usability issues.

4.1.2 Blaise, Fully Tweaked Standard Environment

Another strategy for deploying Blaise for tablet PC slate operation is fully tweaking the Blaise DEP environment per provided configuration and programmable options. This is commensurately more development intensive than default configuration. Still, to a degree, one can manipulate the appearance as desired (although this can mean rigorous use of Blaise options or properties occurring in multiple places—some requiring coordination—and not all thoroughly documented). One note is that the continuum between default Blaise and fully tweaked Blaise in not as absolute as represented here (done as extremes to facilitate explanation and characterization of two strategies). Blaise configuration options largely stand-alone so they can be implemented individually yielding many incremental stages between default and full leveraging of options and configuration.

First, to achieve greatest color contrast to promote mobile visibility, the background and foreground colors of windows can be reset. To address the default small size of windows display content, new tags can be defined and added to the displayed text to make it bigger. Doing so comprehensively across a significant questionnaire requires moderate structuring of the tag scheme and organization and discipline in programming implementation. A problem arises when making large adjustment to the response labels displayed in the Blaise info pane. As the text gets bigger in font size, it also pushes horizontally to the right. When this causes wrapping onto the next row, the interactive objects (radio buttons and check boxes) can become misaligned with their matching large font labels. This confounds accurate data entry if interacting with these objects since it becomes hard to resolve which object to tap without scanning the object to label correspondence of most responses (for example, scanning down the rows of labels and buttons counting each occurrence to confirm the match-up).

Size of Blaise interface panes can be manipulated to eliminate, or at least minimize, one of the redundant panes (info or form). However, which to eliminate is somewhat problematic since desirable default behaviors reside in both. The default arrangement of the info pane for simple questions already fits desirable pen/slate objectives, so requires no change beyond object/text enlargement. In a typical instrument, the majority of responses are simple enumerated types. Thus, retaining the radio buttons or check boxes in the info pane is the easiest recourse. Yet, Blaise built-in open text and drop down entry fields (plus some further extended or callable entry

object types) are only available in the form pane. This difference means implementing one of two fully tweaked options. Either, vary the pane presentation per item context (info for most, form for some), which is specification and item detail intensive. Or, fully compose a form pane scheme to handle all item types (most likely done as type specific templates), which is more general implementation intensive. Since dimensions and placement of panes can be manipulated with some effort, making a portrait mode layout of Blaise screens is possible. However, a choice must be made whether landscape or portrait presentation is used; both cannot be accommodated.

Discussion above of the default menu bar sizing and placement apply in this strategy too since neither of these is reconfigurable in Blaise. A good pen friendly feature that can be enabled is a row of key-like buttons that appears at the bottom of the form pane for enumerated responses and also includes an on-screen Enter button/key. These are positioned well at the bottom of the screen and large enough for adequate threshold pen-capable sizing. Yet what appears on these buttons remains the numeric code representations of the enumerated responses so they are just screen proxies for the tangible keyboards keys. Consistent with the argument made earlier about the opportunity to interact directly with context sensitive on-screen answer objects as ideal for pen interaction, this still falls short of full pen tuning for Blaise applications.

The later Blaise releases include new object types that can be assigned to individual input fields in the form pane, such as drop down lists and date picking. Plus, an open user action calling mechanism to a specified external object is available. It appears to users as a button with ellipses at the end of the field. These are flexible and powerful extensions to the form pane operation and can be applied for pen/slate tuning. For example, drop down lists, while rather small in default sizing, still become configurable extensions of tweaked Blaise operation. The user action call button opens input operations further to supplemental or entire replacement objects. However, a glimmer of lack of pen tuning still remains here as these can be intensive extensions to specify per field or a better integrated behavior is desirable. For example, the user action can only call one object and requires the user to engage it. For ideal pen tuning, the relevant type of input object (pop-up keyboard or HWR box) per the context of the item should appear automatically. If the user must make a choice and action, it might be choosing between dual open text modes (pop-up keyboard or HWR) rather than whether to invoke something. Also, the consistent ellipse labeling of the user action button ends up masking any variation applied to this operation on subsequent fields. Allowing labeling (or better yet, an icon image) on this button could provide a useful field behavior cue.

For most survey organizations with existing Blaise experience, this is likely the optimal strategy. It continues to use the familiar Blaise development environment, with some configuration modifications, to construct, deploy and operate familiar Blaise DEP operations, just on a hardware variation, tablet PCs. Is there room for platform specific improvements? Yes. Perhaps this paper may identify and lead to some of those improvements.

4.1.3 Using Blaise IS Instrumentation

Another strategy for implementation of Blaise on a tablet PC, that surmounts issues noted above regarding the direct Blaise user interface, is using Blaise IS. This is the web extension system. This is an ideal solution when multi-modal administration of the instrument is desired and one of those modes is over the web. Even without the web requirement, a questionnaire instrument can still be developed and deployed for isolated local PC operation relying on a locally running web server. In doing, the user interface issues shift to what can web browser and server architecture support and then the details of the Blaise IS implementation that enable or limit the interface. Much of this subject is beyond the scope of this paper and still an emerging system and area of expertise. Generally, the web environment has two aspects that cross over well to pen/slate instrumentation. First, since most conventional web operations involve self-interacting users, web interfaces tend toward simpler methods. This corresponds well with pen/slate aims of clarity, easy interaction and screen appearance. Second, as a relatively late developing interface, web interaction supports newer conventional GUI item objects. Therefore, the available object set tends to have more potential breadth than straight Blaise DEP interface. Such objects as spin controls and rich forms can be supported within browser interfaces. Lastly, web interfaces emphasize point-and-click operations that matches the desirable point-and-tap interaction of pen/slate interaction.

The downside of this strategy is that it is more than just a presentation replacement. It involves a whole change in architecture, set-up and support to a web browser/server model and likely a new skill or tool set for the preparation of the interface. A metaphor for this change is that to get a new steering wheel and dashboard in an automobile, it comes connected to and only operates with a new engine under the hood and you have to go to new filling stations to get a new type of fuel.

4.1.5 Using Alien Router and Procedures Instrumentation

This strategy for implementing Blaise on a tablet PC is the first step in building a custom interface. One can get precise desired behavior at the impact of substantial programming to accomplish it. The implementation is replacing the presentation layer of Blaise operation with one custom developed and connected to DEP operation through alien router calls. The Blaise engine sustains the session operation but yields control to external modules to conduct the input interface. At Westat, we use this strategy for self-responding interview (CASI) instrumentation where a simpler or friendlier user interface is necessary. The same method could be applied to a targeted pen/slate interface.

Here the limit is the creativity, time and budget of the programming effort going into the external interface programming. The alien router to external module mechanism is pretty narrow. The external interface modules get limited information about any one screen instance through the calling mechanisms. The external modules can still read and write their own configuration and display aspects, to do more complex operations, yet this involves the burden of sustaining suitable external metadata and relevant hooks to the Blaise metadata to pull it all together for execution. A metaphor for this strategy is flying missions from an aircraft carrier. Those missions can do many things but one must supply the special airplanes that the vessel (Blaise) can carry and operate with the launching and landing system on board.

4.1.6 Using Open Blaise Architecture (OBA) Instrumentation

This strategy uses the Blaise backbone but drives the execution environment with a replacement program. Here an external process not only operates the interface but also sustains the program execution, only calling Blaise for specific operations. In Blaise system parlance, this is using the Open Blaise Architecture, the ability of external processes to reach into Blaise functionality through a program interface (primarily contained in the Blaise Component Pack (BCP)). This is a reversal of operation control in the alien router strategy described above. Operations that you might want Blaise to sustain include the data model and rules aspects as products of Blaise programming. From a response item (the first one must be primed), the external program returns the value to store to Blaise and invokes the Blaise rules to determine the next item to pose. Then the program queries Blaise for item metadata description and perhaps combines further external metadata, presents the question via its own interface and awaits further input. This input is returned to the Blaise engine and the cycle repeats.

Again the depth of the interface rests on the creativity, time and budget of the programming effort to build the external program. A metaphor for this strategy could be that you can fly to any destination, but first you must build the airplane, using construction modules for the crew/passenger compartment and avionics controls from Blaise, to do so.

4.2 Recommended Enhancements

The following list is reasonable enhancements that might be made to the user interface of the Blaise DEP, where system modifications could exert significant improvements in the pen/slate operation. Most are software system resolutions to usability problems identified in the first two strategies, mentioned above.

4.2.1 Misaligned and larger selection objects

The misalignment of selection objects, as font sizes of labels are increased and cause line wrapping, amounts to a bug in the DEP presentation layer. This bug should be addressed. Within reasonable sizing boundaries, the relatively simple controls (radio-buttons or check boxes) should line-up, displaying and operating as expected. Further, we suggest that the size of these controls should expand proportionally to their label text. In this usage, expanding the text is done to make it more visible, clearer, or easier to interface with the objects. Display objects that go along with these labels should adjust too, for the same reasons.

4.2.2 Richness in Placing Input Objects/Types

Resolving the duality of placing input objects between the info and form panes is desirable. The advantage in the info pane is easy implementation for the majority of simple enumerated items. The advantage in the form pane is necessary additional input object depth and greater display control, at considerable configuration burden. Perhaps it is time to drop the info pane and extend its easy default displays to the default form pane behavior while retaining the greater depth of field configuration in the present form pane. A further desirable enhancement is form-like combination of multiple fields for strongly related data fields (for example, collecting date of birth as distinct day, month, year fields). Layout specification through the ModeLib editor begins to support this, yet the row and column organization is klugey and imprecise. Refining this may amount to switching to a visual painter interface and undoubtedly represents significant effort to adopt a new scheme.

4.2.3 Expanding Keyboard-only Actions

Systemic actions that only can be invoked by tangible keyboard keystrokes should be expanded to on-screen buttons. A good example is the need to press a backspace key to launch a lookup list from its input field. Also, anything set to a function key is not accessible when operating without a keyboard. We suspect there may be further instances that we have not encountered. An additional level of enhancement here might be providing for application described action buttons in the Blaise GUI interface, although a workable scheme to relate a displayed object to any action may prove elusive. Some desired actions might be external calls, others might be proxies for existing Blaise operations.

4.2.4 Sizing or Placing Menu Bars, Sizing or Adding Menu Buttons

The standard Blaise menu bar should be larger for pen/slate operations. It now verges on too small for effective pen use. Further relocating it to another placement on the screen (specifically the bottom) is desirable. Also, adding application determined menu buttons is desirable. The menu bar is the regular place for general interaction invocations (such as backup or entering missing values). A good solution for placing a pop-up keyboard button is on the menu bar (more below).

4.2.5 Pop-up Keyboard Controls

Adding pop-up keyboards as a user action property of input fields (then appearing as an ellipse button) is a quick means to get this external call associated to individual fields. Yet this may still be one user action too many to simply start entering known open text for items or not consistent enough for a core object of pen interfaces. Mentioned tablet usability criteria (clarity and easy interaction) suggest that the keyboard should pop-up automatically when needed or background access to it should appear on the menu bar. The user action control then might remain available for supplemental HWR/ink boxes on such fields, while a menu button could remain the fallback method of entering open text. Or, if pop-up keyboard access was optionally configurable on the menu bar, it could become available for out-of-the-box Blaise use on tablets.

4.2.6 Dual Screen Orientation Support

Presently, only one screen set-up is provided in Blaise operations. To fully support dual screen orientation options, parallel set-up control for a second orientation is required. This may prove a large change. It may resemble the support relevant to multi-language support.

References

Binzer, G. and Hill, D. *NHANES mobile pentop interviewing environment*. (2002, August) Invited presentation, Joint Statistical Meetings, American Statistical Association, New York, NY. (2001, March) Originally presented, 2001 Federal CASIC Conference, Washington DC.

Cortez, J. *Exploring the use of Blaise in pen-based tablet computers*. (2004, March) 2004 Federal CASIC Conference, Washington DC.

Della Torre, K., Hill, D., O'Reilly, J., and Landerman, L., *Extending CAPI data quality to new settings with Tablet PCs*. (2003, December) Washington Statistical Society Symposium, Arlington, VA.

Hill, D. and Huey, R., *Technology enabling practical mobile data collection*. (2004, May) Invited presentation, Washington Statistical Society, Washington DC.

Hill, D. and Binzer, G. *Preliminary evaluation of Microsoft XP Tablets for household interviewing*. (2003, February) Invited presentation, 2003 Federal CASIC Conference, Washington DC.

Hill, D. and Montalvan, P. *Pen-based portable computers and handwriting recognition in large demographic household surveys*. (2002, February) 2002 Federal CASIC Conference, Washington DC.

Jarrett, R. and Su, P. Building Tablet PC Applications. Microsoft Press, Redmond, Washington, 2003.

Van West, J. Tablet PC Quick Reference. Microsoft Press, Redmond, Washington, 2003.