

Towards a unifying Approach to Mobile Computing

Carsten Magerkurth and Thorsten Prante
GMD – German National Research Center for Information Technology
IPSI – Integrated Publication and Information Systems Institute
Dolivostr. 15, D-64293 Darmstadt
{magerkur, prante}@darmstadt.gmd.de

Points of Departure

Today, the domain of *mobile computing* is as diverging as the domain of desktop computing was some twenty years ago. Just like in the early days of *home computing*, when computer manufacturers were enforcing incompatibility even within their own range of devices¹, there are numerous excluding alternatives emerging for the mobile user.

There is no widely accepted standard of what mobile computing should be like. Different *device classes* (such as palm-sized PC's, pocket-sized PC's, or WAP enabled mobile phones) with different *operating systems* (such as PalmOS, EPOC, or Windows CE) support *different paradigms* regarding the purpose and applicability of mobile computing (Greenberg et al., 1999, Myers et al., 1998).

From a developer's point of view, it is crucial, whether a mobile device is to be used as a mere data viewer or a remote control (as e.g. Rekimoto's "painter's palette" (1998)) or, on the other hand, as an autonomous system that can operate independently to some degree (Magerkurth & Prante, 2001). The former approach moves most of the application's functionality away from the mobile device requiring a tight connection to "fixed" computer systems, as it is the case with the mobile phone's WAP browser to the corresponding ISP. The latter approach enables more independent work, but puts higher demands on processing power and memory resources. Therefore, truly autonomous work may not be possible with all kinds of mobile devices due to resource constraints encountered, e.g., on mobile phones.

While this issue cannot be resolved analogous to the one of whether to use thin clients in client-server networks, there is a more immediate limitation to the role of mobile computing: Currently, there are numerous mobile technologies on the rise and most of them come along with very proprietary API's and little facilities to integrate with other technologies. This has led to the now highly cluttered market, where devices of different types are hardly able to communicate with each other.

Most of the proprietary platforms do not make up a market share worth investing to a duly degree. Even though, there are sufficient similarities amongst most kinds of mobile devices, applications are mostly available just for a single platform. In addition, the possibilities of cross-platform data exchange are still moderate, even if similar communications hardware and protocols (e.g. IrDA) are used. Even the largely device independent WAP technology has not yet gained enough attention to raise a sufficient software infrastructure. Many WAP internet offers are buggy or lack features (Bager & Bleich, 2000).

¹ The shift from Palm III to Palm V introduced essentially the same unintelligible serial port changes as from CBM64 to CBM+4 in the early eighties.

In our opinion, device independent application technologies like Java and to some extent WAP are not the only answer to these well known problems of the mobile domain. Relying on common features and standards inevitably both implies losing the flexibility to cater to specific features of the various devices as well as introducing a performance overhead (provided that the underlying hardware is indeed heterogeneous and the software performance is a crucial criterion).

Approach

Our approach is to make use of the similarities in mobile systems' user interfaces while refraining from any unworthy bytecode overhead that is especially fatal in the context of mobile devices with their limited resources. Regarding user interaction, most mobile devices have a lot in common: Text entry/ recognition mechanisms are too awkward to be fun, displays are too small to hold much information, and CPU's are too slow to allow for strategies to overcome the display shortcomings by, e.g., fast and accurate zooming or panning.

We opt for supporting various devices by introducing scaled implementations of user interface components that are applicable to more or less all mobile systems. For instance, a text button on a mobile phone might be made up of a simple sequence of characters surrounded by a black frame. To press it, users have to make use of the hardware buttons on the device, due to the lack of a touch-sensitive display. On an EPOC handheld PC, however, the button can be a more sophisticated control which can, e.g., easily be dragged about the display with a stylus. Nevertheless, taking the application's back-end view, a button is always just a button, i.e., a medium for communicating the user's request to have a corresponding action performed. Likewise, the application itself does not need to bother, if the user is picking his theatre tickets from a plain combobox or a highly polished, rotating, zoomable list-control.

In our opinion, it suffices to abstractly define only a very limited set of user interface controls applicable for mobile devices and implement them differently with regard to the specifications of the respective mobile platform. We found that we actually never needed more than a button, a list/ menu (of alternative choices), a text/ bitmap display, and an edit field. None of these controls are fed with exact positions or sizes from the corresponding application, since we are not dealing with user interfaces based on the windows metaphor, but mostly with modal interaction objects that have the exclusive attention of the user.

Experiences

We are developing a C++ application framework (*independent application framework, iAF*) that adheres to the principles described above. We use iAF not only for the development of our creativity support tool PalmBeach (Magerkurth & Prante, 2001), but also in the development of a number of commercial PDA entertainment software titles.

Taken that computer games may not be representative of all mobile applications, it can be shown that there is a surprising amount of similarities in user interfaces even between full blown desktop applications and mobile software (Magerkurth, 2001). By keeping a high level of abstraction regarding the user interaction code, the development cycle of a software title can be shortened significantly, especially when targeting multiple mobile systems.

Provided that the user interfaces are implemented using scaled, system-specific functionality it becomes trivial to serve different platforms. Applications based on iAF, e.g., are 100% source code compatible amongst all supported platforms. This is not only for the benefit of

unifying the mobile software infrastructure, but it addresses a further issue in software development in the context of mobile computing: Mobile devices hardly ever host their own development tools due to limited resources. Thus, cross-platform compilers running on well equipped desktop machines have to be used, instead. This implies that the development process is subject to a permanent overhead, for testing can only occur after an upload on the mobile device or inside an emulator. Therefore, a very positive side-effect of the use of our framework is a speed-up of the development. This is due to building desktop executables most of the time which permits test-drives of native applications instead of crawling through emulated code.

Conclusion

What we have discussed, is primarily geared towards user interaction on mobile devices and its implications on software development. We have proposed scalable interaction objects. Regarding the constraints of mobile computing, the next logical step would be augmenting scalable user interface implementations to scalable *content*, which also closely relates to WML and Web-Clipping in comparison to HTML.

References

- Bager, J., & Bleich, H. (2000): WAP-Galerie. In: c't Magazin 9/ 2000, S. 200 – 207.
- Greenberg, S., Boyle, M. & Laberge, J. (1999): PDAs and Shared Public Displays: Making Personal Information Public, and Public Information Personal, im Druck
- Magerkurth, C. (2001): Programming Windows To Create Palm Games. In: Proceedings of the Game Developers Conference 2001. San Francisco: CMP Media, 2001.
- Magerkurth, C. & Prante, T. (2001): Metaplan für die Westentasche - Mobile Computerunterstützung für Kreativitätssitzungen. In: Tagungsband der GI-Fachtagung Mensch & Computer 2001 (MC'01). Bad Honnef: Teubner Verlag, 2001.
- Myers, B. A., Stiehl, H. & Gargiulo R. (1998): Collaboration Using Multiple PDAs Connected to a PC. In: Proc. of the ACM Conference on Computer Supported Cooperative Work (CSCW'98), 1998, 285-294.
- Rekimoto, J. (1998): A Multiple Device Approach for Supporting Whiteboard-based Interactions. In: Proceedings, CHI'98: Human Factors in Computing Systems, 1998, 344-351.