Some thoughts about design of *MiniApps* : Software engineering and usability engineering

At the time being – year 2000 –, a lively discussion is going on in the SIGCHI forum (CHI-WEB@ACM.ORG) about the topic *Web design and business*. To begin with, there are three parties involved, the *business client* – who dares to object that this is the most important person? – may be a *freelancer*, the *software designer*, and, last but not least, the *interface designer*. Later, I shall identify two more parties.

It is difficult to define what *MiniApps* are. I hope that in the following the context of this term helps in an informal definition. The scope of the functionality of *MiniApps* certainly depends on the business environment considered, the person involved using one, her/his previous experience in computerized work, and – in my opinion – personal preferences. The term is widely used in the SAP terminology [1]; see e.g. for a different commercial definition in [2] – stupidly difficult to read as only written in lower case –.

Three decades ago one would have referenced the Kernigham/Plauger tools [3], easily combined with *filters* in a kind of serial processing steps by *piping* the pseudo files *standard input* and *standard output* , resp., [4]/[5] in an UNIX environment. Are these already MiniApps as discussed to-day? Probably not! It is a concept for programming. Now we have immediate interaction with the PC, no programming, no professional helpers for the laymen. MiniApps are a concept for business clients (and others), all users of interactive computer applications with their PCs.

There are a lot of *MiniApps* a business client may be interested in: Fixing a meeting; investigating price and possible time of delivery of a product; currency conversion, checking and/or updating a financial account – you name it –. And we all know it, the E-mail alerts shown in the browsers. In the office there is a workstation connected to the internet with a large screen, fully coloured, of course, where you can activate such a MiniApp by clicking an icon and working with the application kernel for a short time with a small number of interactions. ***Big is Beautiful !*** Sitting in a meeting you have your laptop computer in front of you, same icon, similar but

smaller window, WWW access may be. And, out of the office, you have your high-tech WAP mobile phone, with internet access, with 10up small keys which you hardly may operate with all your fingers, and with a mini-sized display screen, no more shiny coloured in sun light or dark night. Just the contrary: ***Small is Beautiful !*** The business client wants to, has to achieve the same tasks, of course, using the workstation, the laptop computer, the mobile phone for the MiniApps he is familiar with. What a scale of platforms and technical constraints!

The *software designer*, hopefully a well-educated software engineer, has to architecture and to implement an application system safely and robustly working behind the scene in the full scale of working situations; transparent to the business user. A challenging design assignment!

And the *interface designer*, hopefully well-trained in all aspects of usability engineering, has to imagine the broad spectrum of presentation and interaction needs of the application and user support component, needs again occurring in the full scale of working situations identified above; and has to map the needs to the really different platform capabilities, without touching semantics behind it. Again, a very challenging design assignment!

A side remark: Often, the roles of software designer and interface designer have to be filled up by the same person, doing such design works only occasionally and without intensive training. A problem to overcome, e.g., by appropriate tool support!

Coming back to the forum topic mentioned in the beginning. I think the discussion is necessary, just now. And it has to be broadened considering parties involved, not only viewing the circumstances of the scale of platforms and technical constraints, viz., in (at least) two more aspects: *Tool designers* and involved *project managers*.

Even educated software and/or interface designers should have available appropriate design tools, integrating the best of common knowledge of both fields. Are such tools available? *Tool design teams* – yes, interdisciplinary teams with

members coming from both fields – should (continue to/improve) research and develop powerful, attainable, usable design tools of this kind. These tools are not only *graphic interface builders*, much more functionality and quality checking methods have to be integrated.

All these efforts need time and money. Now, the *project managers* come in, to become open-minded to tool research and development, to allow for spending time and money in MiniApp design projects for realising high-end software and interface solutions. As this is true for traditional application development, it is much more true for MinApp development, due to the more demanding scale of working situations.

I want to bring up three more aspects which I think are important enough to be mentioned and followed on: Some kind of *interface standardization*, appropriate *training offerings for business clients*, and high-level *education* in an *interdisciplinary science of software and usability engineering* for software and interface designers involved.

Standardisation helps designers and end users, i.e., the business clients, to master the complexity of design work for usable MiniApps and to master the complexity of using different MinApps as they come up during a working day in their business. The problem with standardisation in the two fields involved is the level of granularity between *General User Interface Design Guidelines* (as, e.g., the resp. ISO or DIN standards) and *Detailed User Interface Design Guidelines* (as, e.g., in compendia with many hundreds of very specific rules), and/or *Company Style Guides* (oriented towards a specific company platform). In between I see what I call *Topical User Interface Design Guidelines*, a granularity level which may form an appropriate basis for a design-oriented standardisation (see [6]). The contents of this level is presently researched. Look to the rather successful rise of *patterns* [7] for many areas, just design, interaction, business processing, teaching, etc. However, according to my observation, the contents mentioned above do not cover the full scale of platforms in an unified way, do not respect all technical constraints of to-day and the future, and – not to forget it – characteristics of MiniApps, e.g., all forms of ad-hoc combinations, of different, often not obvious requirements and experience of

the intended users, the business clients. Tool designers are forcibly advised to follow (and to restrict themselves) to this level: ***Standardisation is Beautiful*!**

Where does a business client find training offerings with the broad background necessary to help him for a usage period of months/years? I don't see much of them, affordable and useful, especially for a freelancer.

In our tertiary education system the sciences *software engineering* and *usability engineering* are only rarely covered in one curriculum. The consequence is that – something seen in the forum discussion mentioned in the beginning – software engineers and usability engineers don't understand each other and don't work in design activities really side by side (often side against side). This situation has to be improved, for the best of all kinds of users, for the best of business clients using MiniApps. Many thanks to the established forum to bring such thoughts to an involved audience.

Please allow me – now, 2006 – to pick up again the hint to the Kernighan/Plauger tool approach [3] given above. The minimal concepts required there follow the ***Less-is-More*** principle advocated by many researchers and practitioners with a long experience in good design approaches. Do not forget this!

In a recent update work of this paper – year 2006 – it was interesting to find MiniApps integrated in Web applications as niceties for continuing on-line information access, e.g., for stock updates (http://www.maxmo.net/software/miniapps/stocks) and for weather state and forecast (http://www.maxmo.net/software/miniapps/weather) – both accessed March 31, 2006 –. Is the continuing *side-by-side screen presentation* the "modern form" of the "old" *piping concept* in UNIX work? Piping was and still is a concept of program serialisation of a computer processor, side-by-side presentation is a parallel human eye view process.

Summer 2000, updated March 2006

Hans-Jürgen Hoffmann
Prof. em., Dept. Computer Science, Darmstadt University of Technology

[1] NN (SAP Usabilitiy Engineering Center): SAP Interaction Design Guide for Internet Application Components;
http://www.sapdesignguild.org/resources/web_guidelines/index.htm,
– accessed March 31, 2006 –
[2] J. Smolka: sap miniapps; http://home.arcor.de/j.smolka/miniapps.html,
– accessed March 31, 2006 –
[3] B. W. Kernighan, D. J. Plauger: Software tools; Addison-Wesley, 1976
[4] D. M. Ritchie: The Evolution of the Unix Time-sharing System;
http://cm.bell-labs.com/cm/cs/who/dmr/hist.html, 1996
– accessed March 31, 2006 –
[5] N. Bezroukov: Unix Pipes -- powerful and elegant programming paradigm;
http://www.softpanorama.org/Scripting/pipes.shtm, 1996-2006
– accessed March 31, 2006 –
[6] H.-J. Hoffmann: Software enginering in user interface design with guidelines – from traditional applications to the Web sphere. In J. Vanderdonckt, C. Farenc (eds): Tools for working with guidelines; Springer, 2001, pp263 - 272
[7] E. Gamma et al.: Design Patterns: Elements of Reusable Object-Oriented Software; Addison-Wesley, 1995