I'd like to complain about this software…

David M. Nichols

Department of Computer Science
University of Waikato
Hamilton, New Zealand

dmn@cs.waikato.ac.nz

Software is often frustrating. You need only walk past a group of users to hear exclamations of annoyance as the users' expectations and the software interface clash head on.

Each one of these mini-incidents contains a small but useful piece of information – information that is almost always lost. Most user problems don't generate any sort of report that is passed back to the developers; they are short-lived episodes that are not recorded anywhere. A 'report' could be a simple as a one-click 'I've just experienced a problem' button, or semi-structured text allowing users to say 'how can I put a UK phone number into this form which only accepts US formats?'

Hartson at Virginia Tech has shown that users are actually quite good at critical incident usability detection but most software doesn't have any built-in facilities for reporting. This increases the cost to the would-be reporter who has to search through menus and linked web sites to find a way to complain (I've just tried this with Microsoft Office XP and failed to discover *any* channel to complain through. How do I complain about that?!). With the Mozilla web browser at least I can enter a bug at Bugzilla (http://bugzilla.mozilla.org), if I'm prepared to register and deal with a lot of technical vocabulary. This is better but still requires me to leave my application and fill out a web form. On the other hand, if Mozilla crashes then a talkback-enabled build allows anyone to easily send a crash report. Why isn't it as easy to submit a usability issue as a crash report?

Another reason users don't complain is that think they won't see any benefits. Complaints in many organizations (not just software companies) often seem to disappear into a black hole, never to be heard of again. Again the open source bug database, whilst not perfect, does at least provide a transparent mechanism for tracking progress. Although companies may have good commercial reasons for not opening up their entire development process that doesn't mean there isn't an intermediate position where users can participate without dealing with technical or confidential material.

The net effect of the lack of easy feedback channels is that the average user feels a sense of frustration and powerlessness. They get really irritated by their software *and* no-one is listening. At least when you enter a Bugzilla bug you feel if you have done something constructive. Maybe, just maybe, someone sometime in the future will experience a better interaction thanks to your report.

Nielsen suggests that the first rule of usability is not to listen to users (Alertbox, August 5 2001), mainly as self-reports are unreliable. However, if the reports immediately follow the problem and are supplemented by objective data from the program – then we probably *should* be listening to the users. Usability reporting has an advantage over Mozilla-style crash reporting in that more program information is available: e.g. the actions the user took just before the problem. Although there are privacy issues and the problem of dealing with potentially large amounts of data (what if every user sent a report whenever something went wrong!) decentralized, contextual, user-reported usability issues could (and should) be an valuable mechanism for software improvement.

Open source projects have provided mechanisms for coders around the world to participate in software development. So far, this has tended to be constrained to developers and technically literate users. Can we do for users what open source development has done for coders? We all know about participatory design — what about participative usage?