Nigel Ward

Programming as a Discipline?

Programming is not just an activity, it's a discipline, with its own value system—or so I believed. I'd like to describe an experience that shook my faith.

A few years ago I started building a system for understanding speech. I wanted to do so in part because speech understanding seemed to be an area where I, as a programmer, could contribute. Of course there are already hundreds of people working on speech recognition and understanding. But the predominant research style in speech is an engineering one [6], which seems to lack several qualities a real programmer could provide:

- One missing element was *ambition.* Rather than aiming to incrementally improve an existing system to achieve a slightly lower error rate, I aimed at an optimal model, that is, a prototype of an architecture powerful enough to support optional interpretation.
- The key to this seemed to be the full *integration* of different sources of knowledge. That is, to make sense of speech signals, noisy as they inevitably are, I aimed to exploit syntactic constraints, semantic preferences, the local context, and so forth, seamlessly and online as the input is received.
- The best way to arrive at an architecture for this seemed to be *introspection*. Since humans can understand speech, I aimed to build a system that mimicked that process. To do so, rather than undertake a full scientific investigation of how people understand, I took the simpler approach of just imagining what I would do, if someone handed me a speech signal and asked me to figure out its meaning.
- Finally, to produce an extensible system, I aimed for a clean, *esthetically pleasing design*, specifically, one that was modular, declarative, parallelizable, and so forth.

That, in a nutshell, was what I thought a programmer could contribute.

And many people agreed this was a good research strategy. I remember one computer science conference at whuch I outlined my goals and plans and a hundred people nodded with understanding and agreement at every point. However, at speech conferences my presentations were received coldly, although I didn't understand why.

I set to work, fleshing out and implementing my vision of the ideal speech understanding system. After three years learning and building and debugging, I had a system that worked, taking speech inputs from a microphone and outputting representations of their meaning [7].

I then evaluated its performance and discovered it was completely outclassed by the systems built by plain, old engineering approaches.

Re-reading the literature showed that my attempt was not novel, and neither was my failure. There have been many similar approaches to speech understanding, as surveyed in Klatt [3] and Ward [8]—after exciting starts, all have faded away without achieving much. The reasons for this include specific technical aspects of the speech understanding task, as discussed elsewhere [8]; but the root cause of the failure lies, I believe, in the value systems of the programmers.

The specific aspects of programming that led me (and others) astray were, in fact, the four

points I listed. To recapitulate, making the contrast with engineering practice clear:

1. Ambitiously wanting to build a radically new, high-performance, general-purpose system straight off, rather than accepting the need to build up know-how through a succession of limited but useful systems.
2. Prizing full integration, rather than just the amount of integration worthwhile [4].
3. Aiming at introspectively plausible processing, rather than identifying real issues.
4. Relying on aesthetic principles of design, rather than just designing a system to get the job done. These are discussed further in [9].

Of course, these tendencies are not invariably bad. They often pay off, and so have become some of the core values of the programming community. Ambition is important in a fast-moving field. Integration is an obvious good, as millions of people who work on systems to get information from where it is to where it's needed will testify. Introspection also is often useful, as seen by the multitudes of communications protocols, operating systems, and so forth, which exploit anthropomorphic metaphors. Principled design is another obvious good.

But these values and tendencies are not infallibly useful. They have led many researchers astray in speech understanding, and in other fields too. In human-computer interaction, in artificial intelligence, and in robotics, it can be argued that a programming mindset is often a liability [1, 2, 5].

Software, as a relatively new field of human endeavor, allows a more exciting methodology than plain, old engineering. Therefore, to the students in my software classes, I always try to instill a sense of programming as a discipline, complete with its own mind-set and values. But the experience with speech understanding has humbled me. Maybe programming should, after all, be considered not as a discipline, but just a skill, like writing or drawing. And maybe we should train students to be, say, engineers who can program, not programmers as such. C

NIGEL WARD (nigel@sanpo.t.utokyo. ac.jp) is an associate professor at the University of Tokyo.

REFERENCES

1. Brooks, R.A. Intelligence without reason. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 1991, pp. 569-595.
2. Cohen P.R. A survey of the eighth national conference on artificial intelligence. *AI Mag. 12* (1991), 16–41.
3. Klatt, D.H. Review of the ARPA Speech understanding project. *J. Acoust. Soc. Amer. 62* (1977), 1324–1366.
4. Moore, R.C. Integration of speech with natural language understanding. In D. B. Roe and J.G. Wilpon, eds., *Voice Communication Between Humans and Machines*. National Academy Press, Washington, DC, 1994, pp. 254–271.
5. Newman, W. A preliminary analysis of the products of HCI research, using pro forma abstracts. In *Proceedings of CHI '94*, pp. 278-284.
6. Roe, D.B. and Wilpon, J.G., eds. *Voice Communication between Humans and Machines*. National Academy Press, Washington, DC, 1994.
7. Ward, N. An approach to tightly-coupled syntactic/semantic processing for speech understanding. In *AAAI Workshop on the Integration of Natural Language and Speech Processing*, 1994, pp. 50–57.
8. Ward, N. Second thoughts on an artificial intelligence approach to speech understanding. In *Fourteenth Spoken Language and Discourse Workshop Notes (SIGSLUD-14)*, Japan Society for Artificial Intelligence, 1996, pp. 16–23.
9. Ward, N. Artificial intelligence and other approaches to speech understanding: A case study in methodology. *J. Exper. Theoret. Art. Int.* To appear.