COSC345 Week 26 Notes

Goals: what are standards, how to think about standards, how to find standards, how to criticise standards.

I sometimes start by pointing to a power socket in the wall and saying "*that* is governed by a standard. Why is that a good thing?"

Some standards are really useful. I have electronic copies of the C89, C99, and C11 standards, the C++ standards, both editions of the Pascal standard, several editions of the Fortran standard, the Prolog standard, the Smalltalk standard, three editions of the COBOL standard, the Common Lisp hypertext (basically a zero-price edition of the ANSI Lisp standard), the R[456]RS Scheme standards, the SQL 1992 and 2003 standards, the Ada 95, Ada '05, and Ada 2012 standards, the ECMAscript standards, and the Single Unix Specification 3 and 4 (basically, a zero-price edition of POSIX). I don't write Pascal any more, Pascal compilers having largely disappeared, but I wouldn't be without the others.

Except that some of them are, well, much less useful than others. The C++ standard does not seem to have been written for anyone to read. The C standard is astonishingly readable by comparison, and the Ada 95 standard is really good.

The Prolog standard is one that I was involved in. (The British Logic Programming Association invited me to be an editor of it, but the Australian university I was working for at the time wouldn't let me because there was no money to pay for my time on it.) Software Standards are developed by the same kind of people who develop software. Some of them are good at it. Some of them are concerned about the welfare of the paying customers. Some are not. There is a great deal of politics in the development of standards, a great deal of pride, and not a little malice.

The Smalltalk standard, for example, is (a) rather too small and (b) incredibly badly proof-read. It also manages to specify date-time calculations that are literally impossible to carry out: the DateAndTime class has to both use UTC and work for millennia into the future, but since it is literally impossible to predict when leap seconds will be added, it simply isn't possible to satisfy both requirements.

The Smalltalk standard is currently being revised. Close inspection of the standard reveals some anomalies, which leads me to suspect that nobody has actually been writing to it. For example, it has an equivalent of C's `atan()` function, which is useless, but no equivalent of C's `atan2()` function, which is essential. (Oddly enough, Squeak has it, but VisualWorks lacks it, yet both of them are based on the original Smalltalk-80.) In fairness, I must say that humans have exhibited so much perverse ingenuity in managing their clocks and calendars that trying to standardise them is genuinely hard, but for that reason, all the more essential.

In much the same way, I defy anyone to read the SQL 2003 standard and then implement date and time calculations conformably to that standard without checking with some other implementation to figure out what the standard really means. I wanted to make sure that my Smalltalk library worked with the ODMG (Object Data Management Group) specification, found that that was hopelessly vague, turned to SQL for clarification, and largely didn't get it.

Standards, like programs, have to be debugged. Some groups are quite good at that. The C standard has a problem reporting mechanism and there are

"technical corrigenda" (patches). For Ada, there is an "Ada Issue" reporting process and an Ada Rapporteur Group to deal with it, and you can find out how ambiguities and errors in the standard have been resolved.

Standards are important because we are humans living in a society of humans and we have to get on with each other. To work co-operatively, we have to come to agreement about what we mean and what we are going to do. This is where the Socratic "define your terms" comes from. You can't define all your terms, but imagine if two people agree that they are in favour of "the liberal immigration policy" and one of them means "the [restrictive] Liberal [Party] immigration policy" and the other means "the [generous] immigration policy [favoured by the Labour Party]"? In the same way, if you want to make a cake, it's no good if one person turns up with a 240V cake mixer and the other has a 110V kitchen. When I went to Britain to study, I arrived with a 240V electric razor, and found Britain used three different kinds of power socket, none of them NZ-compatible.

Microsoft Office [please excuse me while I wash my hands after defiling them by writing that] is a *de facto* standard. Except that they keep changing it. I still have painful memories of being given a colleague's Powerpoint slides and being unable to open them using Powerpoint. Let's see if I can remember:

between	**and**	**worked?**
PC Powerpoint 4	PC Powerpoint 5	OK
Mac Powerpoint 4	Mac Powerpoint 5	OK
PC Powerpoint 4	Mac Powerpoint 4	OK
PC Powerpoint 5	Mac Powerpoint 5	OK
PC Powerpoint 4	Mac Powerpoint 5	NO WAY!

Guess which conversion we needed? I have far worse, and more recent, memories: I had developed a number of examples using Visual Basic for Applications with Excel for a surveying class. Without my request, or even knowledge, a more recent version of Office was installed on my Mac. And VBA had completely disappeared, being replaced by AppleScript. Suddenly, *none of my examples worked*. If Microsoft can rip something as major as VBA out from under you, what kind of standard is it? What might they do to us next?

Another *de facto* standard is TeX (and the macro layer above it, LaTeX). I have files written in 1984 that I can still use unchanged in LaTeX. (There was a backwards incompatible change, but this is automatically recognised by LaTeX and a compatibility package is loaded.) TeX is a community standard, not a government standard or a standard supported by an "industry council" comprised of big companies and no users. It's open source, published originally as a quite readable book. I have the TeX source code; nobody can take it away from me. If anyone changes it incompatibly, I still have my old copy.

Standards can thus affect communication from the past to the future as well as communication between people at the same time.

Microsoft managed to get their Office formats accepted as an international standard by means that were not morally dubious at all. No doubt about it. It is not dubious that their methods were not moral. But the standard is not actually complete, and if it were complete, you still wouldn't be allowed to implement it. (Wouldn't it be wonderful if the acceptance of anything as an international standard automatically voided any and all relevant Intellectual Property claims?)

"Standards" don't have to be official. For example, American "English" is

the language of government and culture in the USA, yet there is no law that says so. In New Zealand, Māori and New Zealand Sign language are *official* languages; although English is the language of Government, the courts, and the general culture, it has no official standing that I know of.

Standards can be developed within a company, such as coding standards. In the lectures I refer to the Ada Quality and Style Guidelines (available on the web) which is a coding standard for Ada, and the Ellemtell style guide for C++. I type `'man perlstyle'` to show the students the Perl style recommendations. I took part in developing a Prolog style guide. It was supposed to be published in late 2009. It was eventually published in mid 2011. Boy, was it *hard* getting agreement on that, and in too many places it was agreement to disagree, just asking people to think about what they were doing.

There is a directory of free ISO standards at ∼ok/ANSI-standards.

Two handouts in 2010 are ISO 2382-4 (official vocabulary) and ISO 14977 (official EBNF syntax). Question for reflection: why are meta-standards like these important?

See http://xkcd.com/927/.