COSC345 Week 9

Standards

12 September 2017

Richard A. O'Keefe

What is a standard? I

1.1 A level of (acceptable) quality or achievement

1.2 Something used to measure quality or degree of something

1.3 A moral principle affecting people's behaviour

2. Something which is usual and normal

3. Spelling, grammar, *etc* which is acceptable

4. Best or widely used or recommended textbook

5. A flag associated with a person or group

What is a standard? II

A standard is

- an explicit document
- promulgated by an authority
- prescribing or constraining
- the form or quality of something
- or a process or behaviour
- for some reason.

What is a standard? III

There are personal, project, company, industry,

national, and international standards.

There are standards for programming languages,

code layout, commenting (JavaDoc), naming conventions,

testing, document structure, documentation (MILSTD 498),

quality, processes, you name it.

Standards tell you what to do

They are political as much as technical

Following or imposing : important for a business

Your contract may require following them

Content/conformance level may be negotiable

They must be periodically revised

ANSI, IEEE, ISO standards require some level of consent;

there is a public review period. Group stds do it too!

Why follow standards? I

"Many modern software engineering standards, like ISO/IEC 12207 (Information Technology—Software Life Cycle Processes), MIL-STD 498 (Software Development and Documentation), quality standards such as the ISO 9000 series, and process documents like the CMM, are just lists. Speaking as one of the designers of MILSTD498 and IEEE/EIA 12207, I can report that these standards were designed to be checklists of tasks to consider during software project planning. We instinctively use lists for survival. The motivation for using standards is not just good form—not just being the best. We need standards and other lists to avoid disaster (although they may be useful for more than that)."— Lewis Gray, Abelia Corp.

Why follow standards? II

Writing code "to the standard" increases portability

$\therefore$ code is of greater value

Process standards are "fossilised practical wisdom"

$\therefore$ don't make same mistakes again & again

Other people follow the same standards

$\therefore$ more to buy, less to build

$\therefore$ more sources of advice & tools

Standards must be enforced

It's not enough to *claim* to follow a standard

You have to *know* the standard is followed

Requirements that cannot be checked are not much use

Test suites help you check vendor end

Standards checkers are useful to check your end

Templates (for layout, documentation, processes) handy

When choosing standard, think about tools

Finding standards

Know what the relevant organisations are.

ISO, ANSI, IEEE, ECMA, ITU have web sites and catalogues.

ECMA (www.ecma-international.org)

Standards New Zealand (www.standards.co.nz)

Standards Australia (www.standards.com.au)

World Wide Web consortium (www.w3c.org)

Internet RFCs www.ietf.org (mirrors in lots of places)

www.cs.otago.ac.nz/cosc345/resources/stds (an old but useful list)

How standards fail

A standard may fail if it is

* too small; fails to address important things
* too large; too costly to implement (C++?)
* too late; ISO Pascal Extended
* too slow to adapt (OO Pascal)
* too quick to change (so implementors can't keep up)
* not supported by vendors & customers

Three ways standards don't fail

Some standards are very expensive, so people like us can't afford them. The intent of such standards may be to act as barriers to entry; keeping us out counts as success.

Some standards are very hard to understand, so that only the coterie of people and companies that developed them know what they say. The intent of such standards may be to act as barriers to entry; keeping us out counts as success.

Some standards may require the use of patented technology. The intent of such standards may be to create captive markets. Locking us in counts as success.

When a standard is too small

You may be unable to stay within a standard because it doesn't address your problems.

* C has open, close, read, write, rename, remove file operations, but no directory operations.

* Old Pascal could not open a file by name

* C and POSIX have no standard format for locale names

* ANSI Smalltalk has no graphics classes

When a standard is too big

Brian Kernighan said of C++ that

— “90% I don’t think I understand”.

A standard that is too big is hard to understand, costly to implement, very hard to get right, and leaves little time for optimisation.

Microsoft’s 6000 page standard is way too big!

When a standard is too late

Standard adoption is slow anyway; it was 7 years before
I could rely on having C89 compilers & libraries.

ISO Pascal Extended added modules but was years late;
by then UCSD Pascal modules had won, so little/no adoption.

ISO Prolog was 10 years late; still not universally followed.

$\therefore$ Sun wanted Java standard "fast-tracked"

When a standard is too slow to adapt

IPv6 is *just* in time

Support for wide characters (ISO 10646, Unicode)

Other internationalisation support needed

Most programming language stds in command line age

Too many or too fast

$XYZZY$ML "standards" based on XML are mushrooming;

they arrive too fast to keep up; cheaper to roll your own

they are not thoroughly revised; some are quite bad.

Microsoft *deliberately* keep releases coming out fast

to keep competitors off balance.

See http://www.noooxml.org; Anand Vaidya's slide are local at ODF-vs-OOXML-v1.2.pdf.

See also office-2007-compatibility-mode-confusion4793.pdf