Programming for Engineers

Department of Engineering, University of Waikato
Compiled by Professor Jonathan Scott

March, 2010

1 Sources and Methods

To compile the needs identified below, all academic staff in Electronic Engineering and a number in Mechanical and MaPE were polled. A number of practicing engineers in the Waikato, in Australia, in California and in the UK were polled. They represented a range of industries, some were managers others were on technical tracks. Published information from other engineering schools was checked. Academics in EE at two universities in Australia were also asked for their comments. A few respondents indicated that they had no significant opinion or no particular interest and their input has been deprecated. In soliciting the information we tried to avoid "leading questions", but when the responses were vague or irrelevant we followed up with subsequent emails with more specific requests. We sought answers in terms of abilities, such as "write a program to achieve such-and-such" but more often received responses along the lines of "need to know how to use such-and-such a tool". If a more profound analysis were to be undertaken, it might most usefully be carried out in terms of the Threshold Concepts that need to be covered.

2 Justification

Since we are required to "provide a world-class, relevant and sustainable programme of teaching and learning" including setting "appropriate learning outcomes" we need to define what these might be. Achieving this involves ensuring that "the links between graduate profiles and learning outcomes are developed through sound curricula" and in our case also "developing graduate profiles ... in consultation with stakeholders".

3 Summary of Responses

The most encompassing summary of the greatest need is probably a "working knowledge of a line-based, procedural language". Where the respondent cared about the exact language it was most often "C, *not* C++". Around 80% of respondents articulated this need, and about half went on to give examples of concepts that were expected such as "variables and data types", "addresses, pointers and storage allocation", "arrays", etc.

The most surprising outcome was the number of demands for what is best described as "scripting ability".[1] Most industry respondents thought this was either the most important or an important skill, with comments such as "I am more of a Scripter than a Programmer", and "the convenience of a script makes such tools essential". There was no consensus on the language, various shells, Perl, Python, VHDL, VerilogA, and others were mentioned. The single, most-cited scripting need

[1] I interpreted the respondents as meaning by the use of the term "scripting" something like "programming in an interpretive, high-level, command-line, environment".

was an “ability to program in MATLAB”,[2] though the majority of respondents agreed that once someone knew one language that adaption to others was straightforward.[3]

Following hard on the heels of “scripting” was “basic understanding of what’s going on under the hood”, including “the complete process flow of compilation and linking”, and an ability to recognise assembly code if not to actually use it. The majority of respondents thought object-oriented programming was unnecessary or a distraction (the author’s interpretation). Some wanted students to be able to recognise it.

Considerable attention was paid not to the programming nuts and bolts, but to the thought processes. The ideas of how to plan and architect a program, and especially how to document it, came up regularly. People wanted to see “tools that assist engineering quality software: E.g. Initial planning and specification, software version control, debugging tools, white box and black box testing”. Such things are presumably taught in modern CS courses, but may have been absent when respondents were graduates. Sydney University EEs have “EE programming” and “CS programming”.

I got the feeling that most respondents would describe ANSI C as a high-level language. Typical statements included the like of “object-oriented is probably overkill”. Without wishing to denigrate my colleagues’ abilities, the level of abstraction and code reuse that is familiar and required in the low-level, hardware-conscious, day-to-day activities of EEs is very elementary compared to modern application programming. One respondent cited IEC61131-3 and identified its various coding approaches with coding from “assembler” to “structured languages like C”, up to the most advanced level of “Function Block (FB)” programming that was “great for teaching modularisation of software and benefits of object based programming”.

4 What is Required

If Engineering is to allocate 25 units in COMP1xx and ENGG2xx to computer skills, these courses should cover what engineers need. Students should be able to design, write and execute a program in a procedural language. Students should understand problem decomposition and specification; program design; the design of algorithms; “good programming practices” such as commenting; variables and data types; operators, expressions, precedence and promotion; control structures and program flow; arrays and structures; storage allocation; functions and parameter passing mechanisms; pointers and pointer arithmetic; compilation and interpretation.[4] There are a number of contemporary textbooks available that address and reify these needs, for example “Introduction to Engineering Programming: In C, Matlab and Java”, by Mark Austin and David Chancogne.

[2]The language of MATLAB was mentioned more than any other single example in this category. It became apparent to me as I collated responses that this application is thought of by many engineers as providing a very rich “scripting language”.

[3]I speculate that this attitude might be bolstered by the fact that many languages encountered in engineering are fundamentally “C-like” and virtually all are procedural; in other words the view is narrow in computer science terms.

[4]ENGG287 (10pts) allocates about 5 units towards these goals, the remaining 5 units to the use of applications such as MATLAB and LaTeX. Staff seem to expect first-year computer science to have provided the “program design” and “practice of programming” parts, but on a procedural platform.