Introductory Programming Languages at Australian Universities at the Beginning of the Twenty First Century

M. de Raadt and R. Watson

Department of Mathematics and Computing
University of Southern Queensland
Toowoomba 4350, Queensland
{deraadt,rwatson}@usq.edu.au

M. Toleman

Department of Information Systems
University of Southern Queensland
Toowoomba 4350, Queensland
markt@usq.edu.au

Introductory programming instructors in Australian universities are choosing the programming language they teach primarily based on a perception of industry demand. This paper examines if this perception is justifiable, and offers instructors of all programming languages an insight into the issue of academic and industry balance, now and into the future.

Keywords: programming languages, introductory programming, industry demand
ACM Computing Classification System: D.3.0

1. INTRODUCTION

A census was conducted during the first half of 2001. All Australian universities were contacted and a unique and complete picture of tertiary level introductory programming instruction was captured.

The census revealed that perceived industry demand was the major factor in choice of introductory programming language. In order to determine whether the languages taught were in fact required by industry, a survey of industry demand was performed by examining advertisements for programming positions.

2. THE CENSUS

Census participants were instructors responsible for each introductory programming course taught in all Australian universities. Courses taught inside and outside computer science schools/ departments were included. Most courses covered were accredited by the Australian Computer Society (Australian Computer Society, 2000).

This census covered language choice, paradigm choice, tools used to support teaching and reasons given by academics for making these choices. The results of this census are reported in full in de Raadt, Watson and Toleman (2002).

Manuscript received: 7 March 2003
Communicating Editor: Sidney Morris

Thirty-seven of the thirty-nine Australian universities offer introductory programming courses. Eleven of these offer more than one course (one institution offers as many as six), so in total fifty-seven introductory programming courses were covered. The average student cohort for each course was just fewer than 350 students.

Nine different languages were taught in Australian universities during the first semester of 2001. The number of courses using each of the nine languages and the proportion of students taught each language is shown in Table 1. Java is most taught, followed by Visual Basic, although if C and C++ are combined this would rank second.

Language	Courses	Weighted by Students
Java	23	43.9%
VB	14	18.9%
C++	8	15.2%
Haskell	3	8.8%
C	4	5.5%
Eiffel	2	3.3%
Delphi	1	2.0%
Ada	1	1.7%
jBase	1	0.8%

Table 1: Languages taught

Instructors were asked to indicate the reasons for their language choice. They gave up to three reasons which are shown in Table 2. The most common reason (as indicated by 56% of participants) was the industry relevance of the language and its potential to attract students. The second most common reason was the teaching benefits of the language.

Used in industry / Marketable	56.1%
Pedagogical benefits of language	33.3%
Structure of degree/dept politics	26.3%
OO language	26.3%
GUI interface	10.5%
Availability/Cost to students	8.8%
Easy to find appropriate texts	3.5%
OS/Machine limitations of dept	1.8%

Table 2: Reasons for choosing language

Since most instructors believe they teach industry demanded languages it is reasonable to ask: are instructors actually choosing industry-relevant languages?

3. INDUSTRY DEMAND SURVEY

Each week *The Australian* newspaper publishes an IT section containing employment advertisements for IT professionals. From this, programming jobs were counted together with the languages indicated as required for each position advertised. This survey was conducted from January to June 2002. Only programming positions were considered, so, for example, while SQL was counted as part of programming positions, advertisements for database administrators were not considered. The survey examined 424 advertisements for programmers.

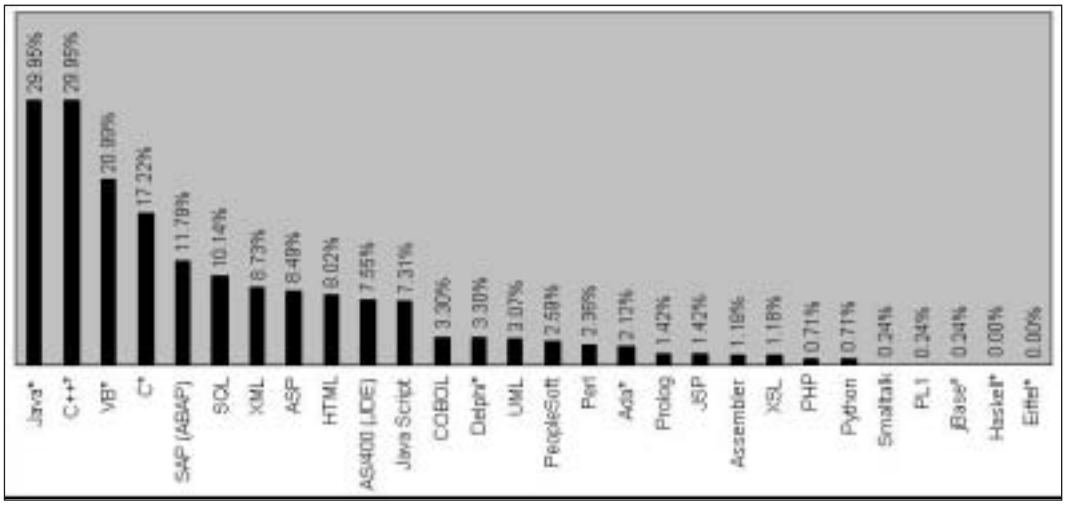

Figure 1: Advertisements by language

Figure 1 shows the percentage of all advertisements that required each language. The average advertisement required 1.84 languages and 48% of jobs required more than one language. The most popular languages were C++ and Java. Languages listed with an asterisk are taught in introductory programming courses in Australian universities. Complete results and analysis of this survey are given in de Raadt, Watson and Toleman (2003).

4. DISCUSSION

The census of introductory programming courses emphasised the importance of employability as a consideration for instructors. This outcome is also one that can be judged by the student, based on university advertising material, before they begin their study. Also, students are more enthusiastic when studying a language they feel will make them more employable. Clearly the choice of language is a key decision. The following is a set of hypothetical questions from instructors seeking to choose an introductory language, together with responses suggested by the current investigation.

What language is most demanded by industry? The languages most demanded by industry are C++ and Java, followed by Visual Basic and C. This demand correlates well with the language choices made by instructors of introductory programming courses as revealed by the census. Java is taught more widely than industry demand would require. C++ is equally the most industry demanded language, but is less represented in introductory programming courses at universities.

Should we switch from C++ to Java? Switching from C++ to Java or vice versa for reasons of perceived industry demand is not justified by the demand measured in this study. Choosing between

these two, must be made on other related factors such as pedagogical benefits, compiler and textbook availability, and the skills of teaching staff.

What about academic languages such as Haskell and Eiffel? Haskell and Eiffel are not demanded by industry. These languages are taught almost exclusively in sandstone universities (universities established prior to 1950) which are identified by Ashenden and Milligan (1999) as having the highest student demand for courses. Instructors at these universities considered the pedagogical benefits of their chosen language as most important, so the use of Haskell, at least, is likely to continue in academic settings.

What other languages should we teach? An introductory programming course cannot be considered in isolation from the remainder of a degree course. Graduate employability can conceiveably be improved by ensuring several languages are known. A progression from C to C++ to Java will qualify a graduate for 58% of advertised positions. Teaching Visual Basic qualifies a graduate for 21% of advertised positions. A course consisting of the related languages HTML and XML followed by a course teaching JavaScript and how it can be applied in ASP would qualify a graduate for 19% of advertised positions. Some universities include a course involving an industrial partnership with, for example, SAP (Hawking, McCarthy and Foster, 2002) and teaching ABAP programming would qualify a graduate for 12% of advertised positions. A degree program including all these languages would qualify a graduate for 86% of positions.

What is the future of programming languages? It is likely that the number of languages taught in universities will fall from the current nine to perhaps as few as five (Java, VB, C++, C and Haskell). Other influences may include the introduction of C# and Microsoft's .NET Suite. Microsoft's withdrawal of support for the Java Virtual Machine may impact on Java's industry popularity. As Internet growth continues, demand for skills in scripting and markup languages is also anticipated to grow. Compilable languages such as C and C++, associated with the high-speed delivery of information required for server-side web applications, would likewise increase in demand. A further run of the census will determine the affects of these influences on introductory programming courses and establish reliable trends.

REFERENCES

ASHENDEN, D. and MILLIGAN, S. (1999): *The good universities guide: Universities, TAFE and private colleges in 2000*. Australia, Hobsons.

AUSTRALIAN COMPUTER SOCIETY (2000): Accredited courses. 2001(August 29).

DE RAADT, M., WATSON, R. and TOLEMAN, M. (2002): Language trends in introductory programming courses. *Proc. The Proceedings of Informing Science and IT Education Conference*, Cork, Ireland, COHEN, E. and BOYD, E. (eds). InformingScience.org.

DE RAADT, M., WATSON, R. and TOLEMAN, M. (2003): Language tug-of-war: Industry demand and academic choice. *Proc. Fifth Australasian Computing Education Conference (ACE2003)*, Adelaide, Australia, 20:137–142, GREENING, T. and LISTER, R. (eds). Conferences in Research and Practice in Information Technology.

HAWKING, P., MCCARTHY, B. and FOSTER, S. (2002): Teaching E-Business concepts using SAP's OnLine store. *Proc. The Proceedings of Informing Science and IT Education Conference*, Cork, Ireland, COHEN, E. and BOYD, E. (eds). InformingScience.org.

BIOGRAPHICAL NOTES


Michael de Raadt

Michael de Raadt is a PhD student and instructor of programming at the University of Southern Queensland. Michael undertook undergraduate study at the University of Western Sydney and achieved his Bachelor of Applied Science Degree with Distinction in 1998, and achieved First Class Honours and was awarded the UWS University Medal in 1999. Michael is also a recipient of the ACS prize for Highest Achievement.


Richard Watson

Dr Richard Watson is a lecturer in the Department of Mathematics and Computing at the University of Southern Queensland. He has taught programming to undergraduates at all levels for the past 12 years. He conducts research into functional programming languages.


Mark Toleman

Dr Mark Toleman is an Associate Professor of Information Systems at the University of Southern Queensland where he has taught computing subjects to engineers, scientists and business students for 15 years. He has a PhD in computer science from the University of Queensland and is an Associated Academic of the Software Verification Research Centre there.