Introductory Programming: What's Happening Today and Will There Be Any Students to Teach Tomorrow?

Michael de Raadt   **Richard Watson**

Department of Mathematics and Computing
University of Southern Queensland
Toowoomba 4350, Queensland

{deraadt,rwatson}@usq.edu.au

Mark Toleman

Department of Information Systems
University of Southern Queensland
Toowoomba 4350, Queensland

markt@usq.edu.au

Abstract

This paper reports the findings of a census of introductory programming courses. Eighty five courses from Australian and New Zealand universities are included. The census aims to discover languages and paradigms taught, tools used, texts employed, method of delivery to on-campus students, instructor experience and how problem solving strategies are taught.

Of note in the 2003 census is the reduction in student enrolments in introductory programming courses since 2001, the differences in teaching between Australian and New Zealand courses, and trends relating to language, tools and paradigms.

Keywords: introductory programming, programming languages, problem solving strategies, census.

1   Introduction

Instruction of introductory programming as an area of teaching is young and still developing. It is not of benefit to any instructor in this area to work in isolation. An awareness of how other instructors are conducting their teaching permits well informed decision making and also encourages community building among instructors.

In the first semester of 2001 a census was undertaken which created a picture of the languages, tools and paradigms used in introductory programming courses in Australian universities, and why instructors chose to use them.

This current census has been conducted which attempts to discover longitudinal trends concerning languages, paradigms and tools. Additional data has been gathered to discover which texts are being used, what contact hours are employed for on-campus students, experience of instructors, and methodologies employed for the instruction of problem solving strategies.

A related study (Robins 1998) surveyed six universities within New Zealand and twenty within Australia. This survey covered language choice and some qualitative information. The 2003 census has been expanded to include participants from New Zealand universities in response to requests from instructors from New Zealand and to allow comparison of Australian and New Zealand systems.

This paper is organised into sections as follows. Section 2 briefly reviews the main findings of the 2001 census. Section 3 presents the main findings of the 2003 census as they relate to Australia and New Zealand. Section 4 presents notable trends between the 2001 census and the 2003 census and distinctions between Australian and New Zealand teaching that have appeared in the 2003 census. Section 5 presents some concluding remarks and suggestions for future studies.

The paper refers to courses presented over a single semester instructional period. They typically form part of a larger degree program. In some universities, this may be equivalent to a subject, unit or paper.

2   2001 Census and Related Work

The initial census (de Raadt, Watson and Toleman 2002) was conducted in the first half of 2001 and involved universities within Australia. This census explored language choice, paradigm choice, tools used to support teaching and reasons given by academics for making these choices. Brief statistics from the 2001 census are shown in Table 1.

Universities Teaching Programming	37
Courses	57
Total Students (Approx)	19,900
Average Students per Course	349

Table 1: Brief statistics from the 2001 census

Participants were contacted by telephone. The 2001 census included the following questions.

1. What programming language is being used?
2. How many students are currently undertaking this course?
3. Which languages were taught previously in the course and when did use of the current language start?
4. Why was this language chosen?
5. Are there plans to change the language?
6. What type(s) of student is your first programming course designed for?
7. What paradigm is being taught using the language (regardless of what is traditionally thought to apply to this language)?
8. Are environments and/or tools beyond a simple editor and command line compiler used to support teaching of the language in practical sessions?

Nine different languages were being taught in Australian universities during the first semester of 2001. The number of courses teaching each of the nine languages and the proportion of the student population taught each language is shown in Table 2.

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 2: Languages taught

Instructors were also asked to indicate the language taught prior to these 2001 languages. The results showed a reduction in language diversity from 18 languages taught in 1996, 17 in 1997, 16 in 1998, 14 in 1999, 11 in 2000, to 9 in 2001. This trend indicated that choice of language was tending towards a smaller group of languages.

Instructors were asked to indicate (potentially multiple) reasons for their language choice. 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 perceived pedagogical benefits of the chosen language.

Only five instructors indicated that they had definite plans to change the language they were teaching. There were no languages that were prominent among these changes, nor was there a pattern indicating that people using a language would change to another particular 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 3: Reasons for choosing language

Instructors indicated the types of students towards which their teaching was directed, for instance computer science, business, engineering or other. Most instructors indicated that they taught a broad range of students rather than a particular type.

Instructors were asked to classify their approach to teaching by paradigm. As can be seen in Table 4, over half of all introductory programming students were taught using a procedural paradigm, even though 81% were taught using an object-oriented language.

Paradigm	By Language	Taught By
Procedural	10%	51%
Object-Oriented	81%	40%
Functional	9%	9%

Table 4: Paradigm used in teaching

Participants were asked to indicate if any tools beyond a simple text editor and command line compiler were used to assist in teaching in practical lessons. When not forced to use an environment or tool by their choice of language, the majority of instructors avoided additional tools.

No Tool	45%
VB IDE	19%
Other IDE	13%
Other Tool	10%
Functional Environment	9%
BlueJ	4%

Table 5: Environments and/or tools used.

'Sandstone' universities (Australian universities established before 1950) offered four of the six courses teaching 'non-commercial' languages (Ada, Eiffel and Haskell). Haskell was only taught in Sandstone universities. Within Sandstone universities the ordering of the first two reasons for choosing a particular language

(see Table 3) was reversed, indicating that the pedagogical benefits of a language were more highly valued that the perceived marketability of a language.

The results of the initial census were sent to all participants and have been published. The results stimulated discussion among introductory programming instructors. The authors have consulted to a number of instructors from university and high school settings regarding the information gathered by the census.

Because many instructors perceived they were teaching industry relevant languages, a survey was subsequently performed in order to gauge industry demand by language during the period when the census was conducted. Information was gathered through advertisements in *The Australian* newspaper. The survey showed a correlation between language demand and language being taught in universities. Most demanded languages were equally C++ and Java, followed by Visual Basic then C. Full results of this survey are available (de Raadt, Watson and Toleman 2003a, 2003b).

3 Current Teaching in Australia and New Zealand

Another census was conducted in order to gather data relating to longitudinal trends and to capture new information relating to text books used, hours spent in lectures, tutorials and practicals by on-campus students, the number of years that each instructor has been involved in the teaching of introductory programming, and how problem solving strategies are presented to students. As with the 2001 census, participants were contacted by telephone. Telephone interviews lasted about five minutes. This method was used to ensure a high response rate. All contacted instructors willingly participated in the census. The 2003 census included all questions from the 2001 census with the following exceptions.

- The question "What type of student is your first programming course designed for?" was removed as it was felt that data gathered from this question was not reliable or important.
- A history of languages taught and the reasons for choosing the language were not sought unless the course was new to the census. Where a course previously covered had changed the language used in teaching, a history of the last two years was gathered.

In addition, new questions were included as follows.

1. What textbook(s) do you use in your course?
2. How many hours per week do on-campus students spend in lectures, tutorials and practicals?
3. How many years have you been involved in the teaching of introductory programming?

Also, a series of questions relating to how problem solving strategies are taught was posed. The questions included the following.

4. What percentage of your lecture time throughout the semester do you spend teaching problem solving strategies?

5. What percentage of your tutorial time throughout the semester do you spend teaching problem solving strategies?

Courses included in the 2001 census displayed a 28% reduction in student enrolments in the two year period prior to the 2003 census.

The number of Australian universities encompassed by the census includes the Australian Defence Force Acadamy, which runs an independent course from its `parent' institution UNSW. Only one university in Australia does not offer an introductory programming course.

The number of Australian courses increased from 57 in the 2001 census to 71 in the 2003 census. Part of the reason for this increase was some new courses came into existence in the two years since the 2001 census. Also greater success was experienced in searching for courses in non-Computer Science areas like Business and Engineering. Many courses can be found through Australian Computer Society accreditation, but many are not accredited and finding these courses relies on participants' knowledge of other courses at their universities.

A brief summary of the universities and courses involved in the 2003 census is shown in Table 6.

	Australia	N.Z.
Universities	40	8
Universities Teaching Programming	39	8
Courses	71	14
Total Students (Approx)	16300	3000
Average Students per Course	229	214

Table 6: Brief statistics

3.1 Languages in Australia

Table 7 shows the languages taught within Australian universities. This includes the number of courses using each language and the proportion of all students being taught each language.

Languages taught have not changed greatly since the 2001 census. Absent since the 2001 census are Delphi, Ada and jBase which were taught in one course each. One course now teaches Fortran. Another course teaches Matlab. The instructor of this Matlab course stated the course was an introductory programming course and not a mathematics course using Matlab.

Figure 1 shows the popularity of each language comparing the 2001 census and the 2003 census results. There have been changes in the ordering of languages when weighted by student numbers. C++ moved from third most taught language to second, not because of an increase in the teaching of this language, but because it did not lose as many students as Visual Basic. C passed

Haskell to become the fourth most taught language due to the discovery of a number of new courses teaching C.

Language	Courses	Weighted by Students
Java	29	44.4%
C++	8	18.7%
VB	19	16.4%
C	9	10.6%
Haskell	3	6.0%
Eiffel	1	2.1%
Matlab	1	1.0%
Fortran	1	0.7%

Table 7: Languages taught in Australia

3.2 Languages in New Zealand

In the 2003 census, participants included instructors of introductory programming languages at universities within New Zealand. Languages reported as used in the Robins (1998) study are presented in Table 8 alongside information gathered in the 2003 census. In New Zealand C is absent as is Haskell. JavaScript is taught in one course. Compared to Australia, Java is used more in New Zealand and C++ is used less.

Language	Courses 1998	Courses 2003	Weighted by Students 2003
Java	3	5	60.35%
VB	1	4	17.31%
Delphi	0	2	8.34%
JavaScript	0	1	7.34%
C++	1	2	6.67%
Pascal	2	0	
Haskell	1	0	

Table 8: Languages taught in New Zealand

3.3 Paradigms

In Australia, there is a mismatch between the paradigms commonly associated with the languages taught and the actual paradigms used to teach them. Although over 80% of instructors are choosing to use an object-oriented language, more than half of Australian instructors are choosing to teach using a procedural paradigm. One Australian instructor reported teaching the language C using a functional paradigm. Instructors in New Zealand display a closer paradigm-language match. Paradigm use by language and teaching method for both countries is described in Table 9.

	Australia		New Zealand	
	By Lang.	Taught	By Lang.	Taught
Procedural	11.7%	53.0%	8.3%	34.0%
Object-Oriented	82.2%	36.6%	91.7%	66.0%
Functional	6.1%	10.3%	0%	0%

Table 9: Paradigm used in teaching

	Java	C++	VB	C	Haskell	Eiffel	Matlab	Fortran	JBase	Ada	Delphi
■ 2001	43.9%	15.2%	18.9%	5.5%	8.8%	3.3%	0.0%	0.0%	0.8%	1.7%	2.0%
□ 2003	44.4%	18.7%	16.4%	10.6%	6.0%	2.1%	1.0%	0.7%	0.0%	0.0%	0.0%
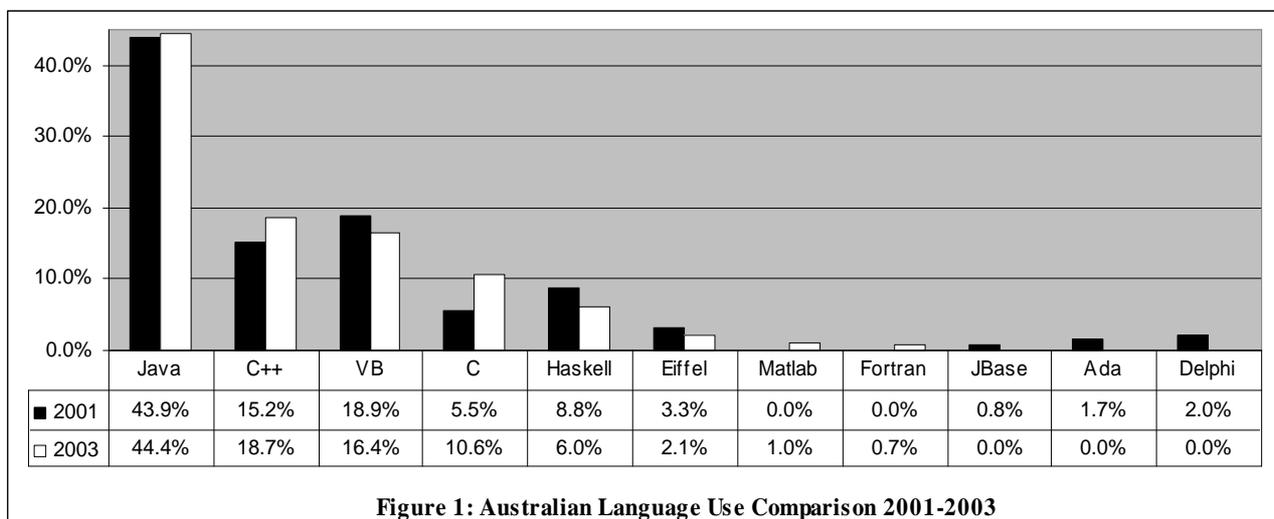
Figure 1: Australian Language Use Comparison 2001-2003

3.4 Tools

Tools used in teaching during practical work again show similar results to the 2001 census. Most instructors are choosing to use a simple editor and command line compiler when not forced to use an environment by the choice of language taught. There has been an increase in the number of courses using BlueJ. Use of tools is shown in Table 10.

	Australia		New Zealand	
	Courses	Students	Courses	Students
None	26	45.4%	5	55.7%
Other IDE	13	21.7%	2	12.7%
VB IDE	19	16.4%	4	18.9%
BlueJ	10	11.2%	1	12.7%
Functional	2	5.0%	0	0%
Other Tool	1	0.3	0	0%
Delphi IDE	0	0%	2	9.1%

Table 10: Tools used other than simple editor and command line compiler

3.5 Text Books

Instructors were asked for details of the text or texts they used, if any. Most instructors use one text, while some use none or two. Instructors tended to use texts that involved the language they were teaching. There was only one text that appeared to be widely used: "Simple Program Design" (Robertson 2000) is used by several courses in Australia and New Zealand.

3.6 Teaching to On-Campus Students

Table 11 shows time spent each week in lectures, tutorials and practical classes by on-campus students. The definition of a lecture was common to all participants, but tutorials and practicals are defined differently from institution to institution. In New Zealand, only four of fourteen courses had classroom tutorials and practicals; the remainder had practicals only. Many New Zealand instructors stated that there was an instructor conducting face to face teaching within practical classes.

	Australia	N.Z.
Lecture	2.2	2.4
Tutorial	0.6	0.4
Practical	1.8	2.4
Total	4.6	5.3

Table 11: Average hours spent in lectures, tutorials and practical classes per week.

3.7 Instructor Experience

Instructors were asked how many years they had been involved in teaching introductory programming. Table 12 shows that New Zealand instructors are, on average, more experienced.

	Australia	N.Z.
Minimum	0.5	3
Average	8.6	10.5
Std Dev	7.2	9.4
Maximum	30	40

Table 12: Instructors' experience in years

3.8 Problem Solving Strategy Instruction

Participants were asked to estimate what percentage of time in lectures and tutorials is spent on the teaching of problem solving strategies. Where a participant indicated that there was no tutorial or there was a combined tutorial/practical class, the amount in practicals was used. Average and standard deviations for percentage of instruction of problem solving strategies in lectures and tutorials is shown in Table 13.

	Australia	N.Z.
Lecture Average	29%	21%
Lecture Std. Dev.	22%	20%
Tutorial Average	46%	28%
Tutorial Std. Dev.	36%	31%

Table 13: Amount of problem solving strategy instruction in different class types

4 Discussion

The number of students enrolled in introductory programming courses has fallen by over a quarter in two years. Instructors were not reminded of their last response for class size and were not told that other institutions were showing shrinking class sizes, yet all but three repeat participants reported a reduction in enrolments.

Languages taught in Australia in 2003 are much the same as those in the 2001 census. The number of languages has fallen from nine to eight with five languages used in more than one course each. This follows the trend of reduction in diversity of languages that was predicted by the 2001 census.

As a measure of the stability of languages taught, instructors were asked to indicate if they had definite plans to change the language they were teaching. Nine Australian participants indicated they had plans to

change, which is a small rise from the five participants indicating an intention to change in 2001. As well as this, four Australian and two New Zealand participants planned a change from VB6 to VB.Net.

Some differences are immediately apparent when comparing courses taught in Australia with those taught in New Zealand. The languages C and Haskell are not taught in introductory programming courses within New Zealand. Delphi and JavaScript are taught in New Zealand but not in Australia at an introductory level. Pascal has not been taught in Australia since 1997, although one participant in the 2003 census indicated they were planning to start teaching Pascal in a coming semester. Pascal was taught in New Zealand up to 2002.

In New Zealand, paradigms used in teaching more closely reflect those commonly associated with the languages being taught. This could partly be due to the absence of C++, which in Australia is widely taught using a procedural paradigm.

Total teaching time for on-campus students differs slightly between Australia and New Zealand. The average New Zealand on-campus student receives 42 more minutes of instruction per week. There is a difference in delivery to students between the two countries. In Australia, 60% of courses offer a one hour tutorial per week. In New Zealand in most cases this time is spent in a laboratory setting instead.

Instruction of problem solving strategies varied greatly in the courses covered by the census. Estimates of the proportion of lecture time devoted to the instruction varied greatly. Some participants indicated that teaching problem solving strategies was not a part of their course. Several of these instructors felt problems used in their teaching were not of a large enough scale to warrant teaching problem solving strategies explicitly. Others said that their entire lecture time focussed on teaching of problem solving strategies. These instructors did not distinguish explicit teaching of problem solving strategies from other parts of their teaching. This variation may be due to instructors not having a common definition of what is involved in the explicit teaching of problem solving strategies.

Instructors in New Zealand are more experienced on average. It appeared to be more common for a single course (or paper) in New Zealand to be taught by a rotating group of instructors who taught for one semester at a time.

The two languages most highly demanded by industry are C++ and Java. From 1994 to 2001, Java was adopted by many instructors as they perceived it to be a highly marketable language and preferred over C++ because of this aspect. Between 2001 and 2003, the growth of Java in introductory programming courses seems to have plateaued.

5 Conclusions and Future Work

The reduction in numbers of student enrolments in introductory programming courses is a trend that will be of great importance to a future running of the census. A future census may indicate if this trend reverses and, if not, an investigation may be required to determine why the trend is occurring.

Having collected information about the texts being used in courses, an exploration of these texts could be performed to reveal the following aspects of interest.

- Amount of content dedicated to problem solving strategy instruction
- Target language (if any)
- Cost
- Additional resources (for instance, a language reference)

With this data instructors could make informed choices of textbook, and authors of future texts could see what is currently contained in introductory programming texts.

Answers relating to how problem solving strategies are taught are not reported in great detail here. Further analysis of these answers would create a picture of how instructors define problem solving strategy instruction and perceptions and importance of implicit versus explicit teaching of problem solving strategies. This may allow a better definition of what is meant by problem solving strategy instruction and lead to a more focussed effort for the improvement of teaching in this area.

As well as providing data about continuing trends in introductory programming courses, the census has also proven its potential for capturing real data on topical issues concerning introductory programming instructors. Another census is planned for 2005.

The authors would like to thank the census participants for their involvement.

6 References

de Raadt, M., Watson, R. and Toleman, M. (2002). Language Trends in Introductory Programming Courses. The Proceedings of Informing Science and IT Education Conference, Cork, Ireland, InformingScience.org.

de Raadt, M., Watson, R. and Toleman, M. (2003a). Introductory programming languages at Australian universities at the beginning of the twenty first century. Journal of Research and Practice in Information Technology 35(3): 163-167.

de Raadt, M., Watson, R. and Toleman, M. (2003b). Language Tug-Of-War: Industry Demand and Academic Choice. Proceedings of the Fifth Australasian Computing Education Conference (ACE2003), Adelaide, Australia, Australian Computer Society.

Robertson, L. A. (2000). Simple Program Design, Nelson Australia.

Robins, A. (1998). First language survey. Last accessed August 18 Access, 1998. Available online http://www.cs.otago.ac.nz/survey/surveyhome.html.