identify the most common problem: "many of the solutions would not compile due to syntax errors" (p132).

A major question in teaching IT is the choice of first programming language, this is part of an ongoing debate about which language is "better than another" (Jacobs 2002). The usual answer is that 'it shouldn't matter' and to justify a particular choice: 'language x is easily transferable to other languages'. Most languages are syntactically similar 'If you can write in x you can write in y, just miss out the z words' (except PERL which even its fans describe as obfuscatory).

In order to do this transfer, students would benefit from skills in comparing languages. In Finkel's (1996) comparison of programming languages the questions are matters of language description and use – terms of literacy: "what is the structure (syntax) and meaning (semantics) of the programming constructs?… (and) Is the programming language good for the programmer (is it easy to use, expressive, readable, elegant)?". The terminology of such considerations is that of language. Finkel, for example describes content free grammars such as Backus-Normal Form Pascal syntax which look much like definitions of sentence structures.

Jacobs (2002) argues one way to way to simplify application is to use a language with a simple syntax, but "just because a program is easy to write does not mean it is easy to read, there is nothing more irritating than inheriting cryptic or poorly designed code to maintain or fix". Such issues of maintainability through producing code that is readable and well commented seem much closer to issues of language than of mathematics. Perhaps it there is a link between the lack of linguistic understanding in computing and a problem Jacobs identifies as a lack attention to readability in code. Finkel also describes goals of a language: simplicity, uniformity, abstraction and clarity. Again the discussion is based around language. Jacobs (2002) for example argues for readability but against "noise words" like "do" and "then". He argues "these extra words are of no real use and therefore should not be included". Students would require tools of language to enter into such discussions.

In terms of clarity Finkel argues mechanisms should be well defined, people should be able to read them, knowing their function. C, for example, for the common confusion between the assignment operator (=) and the equality test operator (==). `Thing == Fred` is very different to `Thing = Fred`. One might argue that this is closer to maths than language. But we already are highly skilled at the importance of this in our everyday language: we know very clearly that "Sue hit John" is not the same as "John hit Sue" and that subtle differences in syntax can give quite different meaning: "Visiting aunts ARE boring" vs "Visiting aunts IS boring".

At the most advanced end of programming are the attempts to allow interaction with the computer via normal language. While this might benefit the user, it will require a high degree of language knowledge for the IT specialist. Barker and Szpakowicz (1995) for example describes clause level relationship analysis, and Inman (1997) discussed the importance of syntax for NLP.

3.2 Structured English

Structured English is used during the analysis stage of a project to identify business processes eg "If hours greater than 40 pay fixed rate plus actual – 40 times rate". It is described in class as 'in English but with half the words missed out'. Its cousin pseudocode is performed closer to actually writing the program and is written in a form that can be easily converted into programming statements. Pseudocode enables the programmer to concentrate on the algorithm, without worrying about the peculiarities of the programming language.

While being characterised as formally-styled natural language, neither Structured English or Pseudocode are defined in terms of notation. Few texts have much to offer beyond "miss out words" or advice such as the vocabulary used in the pseudocode should be the vocabulary of the problem domain, not of the implementation domain. There is an underlying structure, the use of six specific structured programming constructs: SEQUENCE, WHILE, IF-THEN-ELSE, REPEAT-UNTIL, FOR, and CASE.

Many students struggle with pseudocode and Structured English. This maybe because we do not do justice to the underlying language components, instead relying on "geekiness" to permeate students: "You know you're a geek when you write a poem that looks like pseudocode" (Walsh 2003).

3.3 HCI

A high level understanding of language is fundamental to work in human computer interaction as shown by the following examples:

Interactivity is defined in terms of encoding in language, conversation is used as basis for interactivity (Mann and McGregor 2002) and "individuals' interactions with computers, television, and new media are fundamentally social and natural, just like interactions in real life" (Reeves and Nass p5).

Koepsell and Rapaport (1995) describe how software is seen as both written work (like a book) and like a machine…in that it can perform certain differing functions". This dual nature from an ontological basis, they argue, is at the root of our difficulties in legal treatment of software: utilitarian objects can be patented, ideas cannot.

Smith (2002) argued that computer interactive fiction is limited by the conflict between narrative and interactivity. The solution, Smith proposes, is a framework for seeing the "user as a resource" to enable them to tell stories, or in other words, a language.

Seely Brown (1999) gives an example of an application based on conversation. He describes tech reps fixing photocopiers where most of their learning came from "just together weaving together a narrative". The created a distributed network to encourage these conversations then because fragments were lost to the ether created a web-based system and a process of transforming opinions and stories into warranted beliefs.

Cicognani (1998) describes how language and communication underlie models of cyberspace and promotes consideration of "cyberspace (as) a space for metaphors" (p18) and as a "linguistic construction" (p19). They argue that "the matter of cyberspace is language: it is written by it, and is navigable by it; the navigation tools are nothing else but pieces of software, id est: language" (p20). To study cyberspace, Cicognani concludes, we must study language: "the characteristics of cyberspace can be found in the characteristics of the language(s) on which it is based" (p23).

3.4 Data models

Data models are used to describe a static model of a system. Of particular importance are the relationships between entities in the system. These relationships can be expressed in words: <a plane flight may have many passenger>, <a team must have at least two players>, but are more usually expressed in diagrams, Entity relationship diagrams and the like. The process of developing a data model is essentially one of language (Sewell and Mann 2003). We describe the system, identify nouns, verbs, remove synonyms etc. It is very difficult to teach this when the class doesn't know what is meant when the book states that relationships should be prepositional phrases. Exercises in the structure of language: "Flying planes ARE dangerous" vs "Flying planes IS dangerous" would, we believe help in this area.

4. DISCUSSION

There are many cases where a study of metadata of languages would facilitate discussions about computing Eg IMDI metadata schema (ISLE 2001) eg could improve user task descriptions in logical design if included metadata information such as that described by Johnson and Dwyer (2002). They describe information use in the collection of examples of endangered languages. Such an approach may also aid in teaching students to document and understand interactions involving the 'new language' of computing.

It is apparent that students may benefit from both improved language skills an ability to understand and use linguistic terminology. It is not clear how this should be integrated into an IT curriculum. From a constructive perspective, such literacy skills should be taught in the context of IT (rather than as a prerequisite), but many questions remain. Who might benefit from a course? Opt-in or compulsory?

Further research is needed as to whether the similarities between specific technical methods and language skills are anything more than coincidence. Questions should be focused to answer specific questions, does, for instance, pretraining in sentence structure improve learning in data modelling?

Future research should also use a framework for analysis using the following components. We should treat the ability to perform the component (eg write grammatically correct sentences) separately to the ability to explicitly describe the component and talk about it in an informed way (eg rules for a what makes a correct sentence):

- the structure of texts, or discourse structure;
- grammar and syntax, or the way words and phrases are formed and combined (eg constituent rules such as sentence contains noun phrase and verb phrase)
- the conventions of written forms, including spelling and punctuation;
- semantics, or word meanings, and the relationships among these meanings;
- jargon and specific registers
- editing, processes of proofing, recrafting and revision
- vocabulary
- knowledge and argument expression

5. CONCLUSION

This review has identified four roles for literacy in an IT curriculum.

1) ability to write generally
2) ability to write in specific registers
3) strengthening language skills for imposition of a new lexicon
4) providing a platform of skills and knowledge about language itself to aid in learning of ideas that are to some extent rooted in language concepts.

Literacy does have an active role in tertiary IT teaching: it is a dual role in both meeting general and specific needs and as such requires considerable thought about intent and method.

REFERENCES

Barker, K. and Szpakowicz, S. (1995) Interactive semantic analysis of clause-level relationship analysis, In Proceedings of the Second Conference of the Pacific Association for Computational Linguistics (PACLING--95), pages 22--30, Brisbane, Australia, 1995. http://citeseer.nj.nec.com/barker95interactive.html

Bruce, C. S. (1997). Seven faces of information literacy. Adelaide, AUSLIB Press

Byrne, P. and G. Lyons (2001). The effect of student attributes on success in programming. ITiCSE, Canterbury UK.49-52

Cicognani, A. (1998). "On the linguistic nature of cyberspace and virtual communities." Virtual Reality 3: 16-24.

Evans, G. E. and M. G. Simkin (1989). "What best predicts computer proficiency?" Communications of the ACM 32(11): 1322-1327.

Finkel, R.A. (1996) Advanced Programming Language Design, Addison Wesley, Menlo Park, CA

Goodwin, B. (2002, 15 August). UK employers welcome combined degrees in finance and computing. Computer Weekly, p.4.

Gray, W. D., N. C. Goldberg, and Byrnes, S.A. (1993). "Novices and programming: merely a difficult subject (why?) or a means to mastering metacognitive skills." Journal of Educational Research on Computers 9(1): 131-140.

Hartman, J. D. (1989). Writing to learn and communicate in a data structures course. ACM, 2, 32-36.

Hay, D. C. (1998). "Making data models readable." Information Systems Management 15(1): 21-33.

Hughes, H. (2001). Opportunities and challenges await computer science graduates. Black Collegian, 31, 2, 68-71.

Inman, D. (1997) Syntax (in NLP) http://www.scism.sbu.ac.uk/inmandw/tutorials/nlp/syntax/syntax.htm accessed 15/5/03

ISLE Group (2001) Metadata elements for session descriptors, ISME Metadata initiative, Draft Proposal 2.4 7, http://www.mpi.nl/ISLE/documents/ accessed 1/6/03

Jacobs, B. (2002) How to design a programming language, a survey of scripting programming language feature options, http://www.geocities.com/tablizer/langopts.htm

Johnson, H., Dwyer, A. (2002) Customising the IMDI metadata schema for endangered languages, International Workshop on Resources and Tools in Field Linguistics, http://www.mpi.nl/lrec/

Koepsell, D. R. and W. J. Rapaport (1995). The ontology of cyberspace. Buffalo, SUNY Buffalo Dept Computer Science

Kress, G. and T. van Leeuwen (1996). Reading images: the grammar of visual design. London, Routledge.288

McCraken, M. (and 10 others). (2002). "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students." ACM SIGCSE Bulletin, 33(4):125-140

McPherson, B. (1998). Student perceptions about business communication in

their careers. Business Communication Quarterly, 61, 2, 68-79.

Mann, S. and McGregor, G. (2002) Conversation as a Basis for Interactivity . Proceedings of the 15th Annual Conference of the National Advisory Committee on Computing Qualifications, 2nd - 5th July 2002 Hamilton, New Zealand 281-288

Mayer, R. E., J. L. Dyck, et al. (1986). "Learning to program and learning to think: what's the connection?" Communications of the ACM 29(7): 605-610.

Miller, B. (1999). "An integrated taxonomy of student reading and learning development." Journal of Further and Higher Education 23(3): 309-316.

Ministry of Education (1994). English in the New Zealand Curriculum. Wellington

Reeves, B. and C. Nass (1996). The media equation: how people treat computers, television and new media like real people and places. Cambridge, Mass, MIT press

Seersmith (2003) Studying Poetry http://www.searsmith.net/classe101studpoetry.html accessed 15/5/03

Sefton-Green, J. and V. Reiss (2001). Developing the creative uses of new technology with young people. Young people, creativity and new technologies: the challenge of digital arts. J. Sefton-Green. London, Routledge.

Smith, J. H. (2002). The dragon in the attic - on the limits of interactive fiction. 2002.http://www.game-research.com/art_dragon_in_the_attic.asp

Sewell, A. and Mann, S. (2003) Data modelling approaches: divergence in practice. This volume

Tench, R. (2001). Perceptions of competence in public relations students' writing. Education & Training, 43,2/3, 94-104.

Unsworth, Len (Ed.) 1993, Literacy learning and teaching : language as social practice in the primary school, Macmillan Education, South Melbourne

Walsh, E. (2003) "Execution", http://liz.aura.dhs.org/poetry.html

APPENDIX A CASE STUDIES ON ECM² (CONTINUED FROM PAGE 348)

	NZ_Inst1	1995/6	1997/8	1999	00	2001	2002
People	Staff Development (2-4)	L2					L3
	Staff and student support (2-4)	L2			L3		
	Reward systems (3-4)						
	Specialisation (3-4)	L2					L3
	Opportunities for sharing (3)						
Processes	Course Planning (2-4)	L2					L3
	Quality Assurance (2-4)	L2					
	e-learning project management (2-4)	L2					L3
	Instructional Design (2-4)	L2					
	Funding (2-4)	L2					L3
	Standards (2-3)			L2			L3
	Reuse (2-3)			L2			L3
	Continuous improvement (5)						
	Knowledge management (5)						
Technology	Network Infrastructure (2)	L2					
	E-learning Infrastructure (2-3)	L2					
	Integrated Infrastructure (4)						L4
	Change management (4-5)						L4
	Technology diffusion (5)						

	NZ_Inst2	1995/6	1997/8	1999	2000	2001	2002
People	Staff Development (2-4)				L2		
	Staff and student support (2-4)				L2		
	Reward systems (3-4)						
	Specialisation (3-4)		L2				
	Opportunities for sharing (3)						
Processes	Course Planning (2-4)						
	Quality Assurance (2-4)						
	e-learning project management (2-4)						
	Instructional Design (2-4)						
	Funding (2-4)				L2		
	Standards (2-3)						
	Reuse (2-3)						
	Continuous improvement (5)						
	Knowledge management (5)						
Technology	Network Infrastructure (2)		L2				
	E-learning Infrastructure (2-3)			L3			
	Integrated Infrastructure (4)						
	Change management (4-5)						
	Technology diffusion (5)						