Language evolution: a natural phenomena

MARK J O'BRIEN[*], ADAM L BRIDGEN[*], ELIZABETH FERN[*], MARIA BAGIOKOU[†]
* Faculty of Computer Science,
University of Akureyri,
Akureyri, 601
ICELAND
† School of Computer Science and Information Technology,
University of Nottingham,
Nottingham, NG7 2RD
UNITED KINGDOM

Abstract: - The use of biological analogies is common in computing. But these analogies are questionable. This paper explores the use of genetic and evolutionary analogies by restating our basic understanding of evolution. Rather than act as a critique for validating existing genetic computing techniques this paper explores an alternative application. Now whereas it is common to think of systems as evolving the same cannot be true of programming languages. However, by drawing together ideas from biological evolution and linguistics the paper proposes a new study of programming language evolution. Early results have shown this to be feasible and an adaptive compiler has been constructed capable of dealing with syntactic changes in a programming language.

Key-Words: - languages, programming, evolution, adaptive, compiler

1 Introduction

Evolution appears in computing in a number of guises. Thus evolutionary computing in the shape of genetic algorithms or neural networks is now well established. Yet one of the more curious sidelights of those two techniques is that the biological analogies they were based on have proved to be wrong. On the one hand genetics is not a simple two-dimensional process where the linear structure of the DNA maps directly onto physical characteristics, and thus the massive diversity of a population is not due to massive variations in the DNA but rather to minor and subtle inflections of the genetic words. While on the other hand real neurons do not work in the same manner as neural nets.

More obvious evolution exists in computing in other areas which do not define themselves as such. Thus systems quite clearly evolve over time. And here it is quite clear that the evolution is more characteristic of biological evolution, since every system at some point become extinct. Of more immediate importance to this paper is the evolution of programming languages. And the terminology is quite explicit: we talk of first, second, third, fourth and fifth generation languages. Even indirectly the languages of one generation are in some sense derived from an earlier generation.

This paper introduces the idea that computer languages can evolve themselves in some way due to changes in their environment. To set the scene of this research the paper initially highlights the phenomenon of adaptation and evolution in nature. Then it describes certain aspects of software which suggest the need for software adaptation. The primary aims of software adaptation and evolution are the maintenance of software, the extension of the working life of software and an improved interface for human-program interaction. These objectives inspire and trigger the concept of software adaptation and evolution. To write software applications, programmers communicate with computers through the medium of programming languages. Therefore, to support software adaptation and evolution at their most fundamental level, the programming languages themselves must change.

Analogous to the use of nature as a model for software adaptation, natural languages are used as a prototype for the development of programming languages. The linguistic study of natural languages has shown that natural languages adapt and evolve through time as a result of use. The research described in this paper applies the idea of adaptation and evolutionary processes so that they become features of programming languages. Thus the investigation of the development of natural languages can provide a guide for the development of programming languages. Also, this investigation will be directed by the identification of common patterns between natural and programming languages. Once the ideas of adaptation and evolution of programming languages have been established then a mechanism to support them is needed. For this purpose an adaptive compiler has been built.

2 Some background

When Charles Darwin published his landmark book *On the Origin of Species by Means of Natural Selection* in 1859 [1] he set in motion a revolution that has radically altered the understanding of the living world. Darwinian evolutionary theory is fundamental to the understanding of the philosophy of this research. Thus this section commences with the definitions of evolution and adaptation that are used in this thesis; they are the essential concepts in this research. When these two concepts have been explained in terms of nature then they are analysed in terms of natural languages. Since the intention of this research is to apply adaptation and evolution to programming languages the study of these two concepts in terms of their effect and employment in natural languages is necessary. The attempt to utilise adaptation and evolution in programming languages is followed by a consideration of the broad effect of adaptation and evolution on software development. The following sections introduce the concepts of adaptation and evolution in the four areas of nature, software, natural languages and programming languages.

3 Adaptation and evolution in nature

Darwin did not invent the theory of evolution. In fact, biologists had come to accept the idea of evolution long before Darwin published his seminal work. The first attempt to provide a general theory of evolution was made in 1809 by the French biologist Chevalier de Lamarck. Lamarck's views were based on the Aristotelian *scala natura*, the 'scale of nature', also known as the 'great chain of being' [2]. Lamarck and his contemporaries adapted this idea into their theories of evolution, supposing that each species begins life on the lowest rungs of the ladder and over, long periods of time, gradually progresses up through the hierarchy of life in response to a natural unfolding of some inner force. Species is the term given to a group of organisms of the same type that can breed together successfully. The second half of the eighteenth century witnessed the realisation that the diversity of life on the planet could be easily explained as a consequence of evolution.

Darwin changed Lamarck's theory by insisting that there was no natural progression up an evolutionary ladder. He contended that the fate of all species is eventually either to become extinct or to be transformed into new species. But in either case, it is natural selection - reflected in the individual organism's ability to survive and, more importantly, to reproduce - which drives these changes, not some internal biological principle or 'life force' as Lamarck had assumed. Darwin's contribution to this debate was not to prove the theory of evolution but to provide a mechanism - natural selection - that could explain why evolution took place [2].

Evolution is the theory of the gradual development of characteristics in species over many generations, especially the development of more compatible forms, from earlier simpler forms [3]. Darwin considered how one species evolves into a different species in terms of individual organisms. Through the process of reproduction, organisms create new individuals. Darwin put forward the view that in any environment, via the reproduction process, species will accumulate over time the variations best fitting it to its surroundings. As the environment changes, new variants will become advantageous, and will tend to supplant variants that have become less well adapted. He appreciated that the effectiveness of this process of natural selection depends upon the variations being inherited. Moreover, Darwin valued the available variations during environmental changes as another factor that influences the process of natural selection.

The preservation of favourable variations in the struggle of life and the rejection of the unfavourable ones reflect Darwin's definition of natural selection as the mechanism which drives evolution and leads to change without the need for a designer to supervise every step. Natural selection produces an indefinite amount of change through time [4]. Change is inevitable in any organism and variations, owing to the struggle of life, will result in the preservation of that individual and will be inherited by its offspring. Therefore Darwin's mechanism, by sorting out the best of what mutation supplies, gives a direction to evolution and allows living organisms to escape the inevitability of extinction [5]. Natural selection cannot plan ahead. As Richard Dawkins has said: "natural selection is a blind watchmaker, achieving a remarkable end through a simple and inefficient means" [6].

The interaction between organism and environment is central to understanding the nature of evolution. Unless species can continuously change to accommodate the constantly altering physical and biotic environments to which they are exposed, extinction will follow. Thus, Darwin's theory has an important lesson: that evolutionary change is driven by a species' need to adapt to changing circumstances. For example, variations in the earth's climate have caused dramatic changes in vegetation and fauna [2]. Darwin's theory can account for changes of this kind because it assumes that the individual is the basic unit of evolution. It is the individual that reproduces or doesn't reproduce, and the individual that passes on its particular traits. Where earlier biologists viewed species as ideal types, Darwin and his colleagues began to see species as simply a collection of sometimes quite variable individuals who shared a number of key traits [2]. This variation between individuals is the potential that allows species to evolve, though evolution can only occur if natural selection makes change advantageous. The capability of a species to undergo evolutionary changes in response to natural selection, so that it adjusts to new or altered environmental conditions, is called adaptation. Hence a change produced by natural selection typically constitutes adaptation.

To Darwin, an adaptation is any feature of an organism that arose as a consequence of natural selection and hence enhances the fitness of the individual to survive in a particular environmental context. The observation that organisms seemed so well suited, that is to say adapted, to their lifestyle was an old one and was a popular subject of discussion for the natural theologians of Darwin's time. As Reznick and Travis [7] note, Darwin's contribution was to provide a naturalistic explanation and mechanism for the origin of the phenomenon. Amundson [8] states that the phenomenon of adaptation is at the core of modern evolutionary biology, and natural selection is first and foremost an explanation of adaptation. Whether conceived as a process, a generic state, or an individual trait, adaptation is a relational concept. The process of adaptation is the 'fitting' of one thing to another. The generic state is a relation between an organism and the environment to which it is adapted. An individual trait is a modified part of an organism which performs a biological function for the organism and thus contributes to the organism's state of adaptation. Although a universally satisfactory definition of adaptation is probably impossible, a rather broad and simple definition is useful for addressing many issues: an adaptation is a characteristic that helps its bearer to survive and reproduce [9]. Usually, and in most cases of interest, this implies that an adaptation is a characteristic that is maintained in the population by the direct action of natural selection. The concepts of evolution, natural selection and adaptation are studied in more detail in Chapter Two. However, the reader is advised to refer to Curtis [10], Green *et al* [11], Gould and Keeton [12] and Brandon [13] for more details about evolution and adaptation in biological terms since genetics is beyond the scope of this research.

So far the phenomena of adaptation and evolution have been defined in terms of nature. However, to understand how programming languages could adapt, a closer look at natural languages is necessary. The investigation of the way that natural languages evolve will help to: a) detect common patterns between natural and programming languages, b) draw, whenever possible, parallels between the two different types of language and c) highlight concepts and ideas useful for the evolution of programming languages. After all, from the early development of the Turing machine through to the computers of today, programmers have attempted to mimic human capabilities. It is necessary to ask why this could not also apply to natural languages. Therefore, it is time to consider adaptation and evolution in terms of natural languages. Once the concepts of adaptation and evolution have been explained in the domain of natural languages then they are discussed in terms of programming languages.

4 Adaptation and evolution in natural and programming languages

The Oxford English Dictionary [14] defines *language* to be a human system of communication which uses structured vocal sounds and can be embodied in other media such as writing, print, and physical signs. The primary function of natural languages is to serve as an instrument by which people communicate and record information. Over the last few decades language researchers seem to have reached a consensus that language is an innate ability, and that only a significant contribution from innate knowledge can explain the human ability to learn such a complex communication system [15]. Pinker [16] argues that language is a complex, specialised skill, which develops in a child spontaneously without conscious effort or formal instruction. Darwin in his book *The Descent of Man* [17] first articulates the concept of language as a kind of instinct. In this century, the most famous argument that language is like an instinct comes from the Massachusetts Institute of Technology (MIT) linguist Noam Chomsky [18] who first argued the idea that an innate universal grammar is the only way to account for language abilities. Therefore, children enter the world predisposed to learn natural languages. All normal children, raised in normal social environments, inevitably learn their natural language or mother tongue, whereas other species do not. This demonstrates that human brains are specially equipped for this function. For more details about the human anatomy and the brain mechanisms involved in the learning process of language, the reader should refer to [15], [16], [19], [20].

The study of natural languages relies upon the understanding of the structure of language, and the establishment of rules to explain and validate that structure. The basic element of any natural language is the sentence. The sentence is constructed with words and the meaning of words is provided in the dictionaries of languages. Grammar is the set of principles that specify how words can be combined to form sentences [21]. Consequently, grammar specifies the structure of the sentences and words are the independent units of the grammar [14]. Although the set of words of a natural language is finite, the number of possible sentences in a language is unlimited. There are just so many words, and it is possible to construct a list of them; but there is no limit to the number of sentences or to their lengths. Therefore, a set of principles that specifies how words can be combined to form sentences, is necessary. This set of principles is called *syntax* and is the most important part of the grammar. Syntax deals with the relations into which words enter with each other within larger structures, called phrases, and then with the shape of these larger units themselves, called sentences [22]. In contrast to syntax is the semantics of a language. *Semantics* is the set of rules and guidelines that specify the meaning of words, phrases and sentences in a natural language.

The study of languages is called *linguistics*. The purpose of linguistics is to describe natural languages and to characterise them as a general phenomenon. Linguists through their studies have come to the conclusion that natural languages evolve. The evolution of natural languages can be seen and understood from: a) the common roots of natural languages, b) the similarities and differences between them and c) the way in which every natural language changes due to environmental pressures such as mixing and separation among groups of different language speakers. As a result of the evolution of natural language, many languages are now extinct, several new ones have been developed and languages in current use are variably adapted. The evolution of natural languages is supported even more by Pinker who takes a step further than Chomsky's argument that the world's languages are all governed by the same universal grammar and each baby is born with a knowledge of that grammar [23]. Pinker in his book *The Language Instinct* supports and explains how language ability evolved in our ancestors in the same way as a biological instinct; it was "a beneficial adaptation favoured by natural selection" [16].

In contrast, programming languages have been constructed by humans to communicate instructions to computers. More specifically, programming languages have been created for intensive use with a specific, precisely defined set of purposes in view. Computers are restricted and require absolute precision in all their elements. As a result of this, programming languages are necessarily simpler and easier to understand than natural languages. Programming languages have similarities with and differences from the natural languages. The most basic similarities are the existence of grammar, syntax and semantics. Without them it would be impossible for the programmers to communicate with computers. As in the case of natural languages, the syntax rules describe the form of the sentences in the programming language and the semantic rules define the meaning of syntactically correct sentences in the programming language. Sentences in programming languages are usually called statements. Because computers are not able to tolerate any syntactic or semantic ambiguity, the syntactic and semantic rules of the programming languages are strict and well defined to avoid any confusion.

Although programming languages have been in existence for the last fifty years they have been through some major developments. Moreover, the progress of programming languages has been supported by their compilers which have bridged the gap between the high level abstraction at which humans work and the low level at which the computer functions. From the development of programming languages it can be seen that they have: a) become more structured and sophisticated, b) increased in number, c) become less machine

dependent, thanks to the establishment of international standards, and d) have increasingly become more intelligent. The development of programming languages reflects their efforts to resemble natural languages. The above mentioned changes have been beneficial to the programmers, primarily by making the languages easier to use.
The development in programming languages shows that they can, and do, change as a result of conscious action. Since the phenomena of adaptation and evolution are applicable to natural languages and programming languages have been developed using natural languages as a model, a transfer of ideas, however imperfect, from one domain to another would seem to be beneficial.
So far, this paper has introduced the concepts of adaptation and evolution in the domains of nature, natural languages and programming languages. The following section focuses on the application of adaptation and evolution in software.

5 Adaptation and evolution in software

Alexandridis [24] identified a fundamental problem with software applications in the eighties. The problem was that any new development of software systems made very little use of previous software systems development knowledge. The short time usually allocated for the development of software resulted in poor performance and sometimes complete failure. Such dedicated software was restrained from reuse not only in different application domains but also in successive versions of the same system. Moreover, software components that had been previously generated and tested were very rarely used in a new design, mainly because they were not built for reuse in the first place. As a result, the two issues of software reusability and adaptability became important issues for software systems or applications.

The trend towards software reusability has been driven by the tremendous costs involved throughout the lifecycle of a system. The cost is caused primarily by the rapid growth in the cost of software development, maintenance and enhancement. Better life-cycle models and the use of more abstract, reusable resources and patterns are needed to improve the productivity of the processes required to develop software. Examples of software reusability are: a) the reuse of a domain analysis in the form of a simple and easy to understand formalism (e.g. the BNF specification language for compiler writing) and related automation tools (e.g. a compiler-compiler) and b) the existence and use of software component libraries.

The second issue, adaptability, arises from the continual changes that occur during the long life of software systems, as well as the fact that such systems must accommodate large dynamic variations of incoming data and meet different computational requirements at different stages of the system. These requirements suggest the introduction of adaptable software that can: a) meet continuously changing mission needs, b) incrementally grow and evolve as environments change with minimal software rework, and c) adapt to changing specifics that appear as a problem solution progresses. Because of these forces, software developers need to deal with change. Thus, new software development tools and methods have been, and continue to be, introduced.

Fayad and Cline [25] state that the need for adaptable software systems is driving the move toward object-oriented technology. Certainly one of the promises of the object-oriented movement has been its ability to make software more adaptable. However, using object-oriented technology does not guarantee that the resulting software will be adaptable. Adaptability must be explicitly engineered into the software, even with object-oriented technology. Furthermore, adaptability is not a generic quality of a software system when taken as a whole; software systems are adaptable in specific, designated ways, if at all. Therefore, the adaptability must not only be explicitly engineered into the software, it must be engineered into the software in places where it will do the most good to the application. The principal advantages of software adaptability are in the areas of reliability, performance and software enhancement and maintenance [26].
In today's rapidly changing software application environments, it is no longer acceptable for a software system to be merely correct and to solve the problem for which it was designed. Ideally, a system must be able to grow and adapt to solve different problems over time. Nevertheless, software adaptation implies the evolution of software in a similar way as in nature where adaptation leads to evolution. As Fayad and Cline point out, the adaptation corresponds to the three stages of the evolution of software development: build the right thing (requirements' validation), build the thing right (software verification and correctness) and support the next thing (reusable and adaptable parts of software). The initial purpose of software

adaptation was the survival of software systems. Later on, software adaptation focused on another objective, that of the users. More specifically, the introduction and development of computers forced humans to alter their behaviour. In their first days, computers could only be handled by ‘adapted’, i.e. skilled, specialists with an appropriate education for their job. But the widespread adoption of the new technology increased the usage of computers and caused many people to become computer users. However, during the design process of software systems the user was often not taken into account. Realising the disadvantages caused by neglecting the needs of users, the designers’ attitude changed. They then started giving emphasis to user performance as well as to system performance. The effort to adapt users’ behaviour to computers and, in parallel, the increased sophistication of the computers, set in motion the integration of computers into the human world. Thus, the growing awareness of user problems led to research in the area of human computer interaction. Moreover, the differences between the user population and the idea that users are able to influence the environment of the software systems since users are themselves in a state of continual change as they modify because of experience and personal objectives, provide a good reason for adaptation [27]. Therefore, the study of users’ needs and routines became important and motivated the development of adaptive software systems and tools based on the requirements, habits and characteristics of the users. The study of such adaptive software systems is beyond the scope of this research but the reader can refer to [28], [29], [30] for more details.

The existing adaptive software systems support the rapidly changing requirements of their applications and have improved the effectiveness of communication between the computer and the user. The programmers of the software systems have experienced and supported all the software changes, from the first simple programs to the most sophisticated adaptive ones. The idea that adaptation has benefited both the software’s life and users raises the question that adaptation might be applied successfully to the programmers, too. Programming languages are the only tools that computer programmers have in order to communicate with the computer itself and to develop the software applications. Therefore, this thesis argues that the phenomenon of adaptation should be employed in programming languages by utilising programmers as the environmental forces that will trigger the adaptation. Nevertheless, to apply this phenomenon to programming languages there is a need for a tool that will be able to accommodate the language’s adaptation based on the needs of the programmer. Between the programmer and the computer stands the compiler as the translator of the programming language. Thus, this research proposes changes into the internal structure of compilers to accommodate the adaptive behaviour of the programming languages.

6 The need for an adaptive compiler

The application of adaptation and evolution as processes in programming languages is the main objective of this research. Thus the investigation of the natural languages where adaptation and evolution are an essential part of them can provide a guide for the evolution of programming languages. However, nobody has yet drawn an explicit parallel between natural languages and programming languages to analyse the adaptation and evolution of programming languages. To be able to study the adaptation of programming languages there is a need to search for common patterns in the evolution of both natural and programming languages. For instance, the ability of natural languages to change their syntax and semantics through their evolutionary processes establishes the two main themes for the evolution of programming languages; syntactic evolution and semantic evolution.

As mentioned above, the development of programming languages has been supported by the development of their compilers. Also, based on the fact that each programming language has a clearly defined set of syntactic and semantic rules then a compiler of that programming language also has its own precise rules in order to translate statements of the language into computer instructions. In the last two decades there has been much work in the process of developing faster and better compilers. Most of this has been to improve their performance and their support in the development of programming languages. Also much emphasis has been given towards semantic and syntactic parsing of natural languages, and interpretation [31], [32],[33], [34], [35].

Thus, to accomplish the evolution of programming languages this research proposes the development of a compiler that is able to adapt its internal structure in a way that will allow the support of adaptation and evolution of a programming language based on the behaviour of the programmer. More specifically, this research focuses on the syntactic evolution of programming languages because any syntactic change will eventually lead to a semantic change. However, the study of the semantic evolution of programming languages belongs to the future development of the adaptive compiler.

The work has so far produced a compiler capable of adapting the syntax of a computer language. The workings of this compiler will be described elsewhere. The purpose of this paper has to been to highlight the use of analogies in creating new systems.

References:

[1] Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life,* London.

[2] Dunbar, R. (1996). *Grooming, Gossip and the Evolution of Language*, Faber and Faber, London.

[3] Cowie, A.P. (1990). *Oxford Advanced Learner's Dictionary,* 4th ed., Oxford University Press, Oxford.

[4] Rose, M.R. and Lauder, G.V. (1996). *Adaptation,* Academic Press, San Diego, California.

[5] Jones, S. (1994). *The Language of the Genes - Biology, History and Evolutionary Future,* Flamingo Harper Collins, Glasgow.

[6] Dawkins, R. (1987). *The Blind Watchmaker,* W.W. Norton, New York.

[7] Reznick, D. and Travis, J. (1996). *The Empirical Study of Adaptation in Natural Populations,* Adaptation edited by Rose, M.R. and Lauder, G.V., Academic Press, San Diego, California.

[8] Amundson, R. (1996). *Historical Development of the Concepts of Adaptation,* in Adaptation, Edited by M. R. Rose and G.V. Lauder, Academic Press, California.

[9] Hudson, R.R. (1996). *Molecular Population genetics of Adaptation,* Adaptation edited by Rose, M.R. and Lauder, G.V., Academic Press, San Diego, California.

[10] Curtis, H. (1983). *Biology,* 4th ed., Worth Publishers Inc., New York.

[11] Green, N., Stout, G., and Taylor, D. (1990) *Biological Science – Systems, Maintenance and Change,* 2nd ed., Cambridge University press, London.

[12] Gould, J, and Keeton, W. (1996). *Biological Science,* 6th ed., W.W. Norton and Company, New York.

[13] Brandon, R. (1996). *Concepts and Methods in Evolutionary Biology,* Cambridge University Press, New York.

[14] Oxford English Dictionary, (1989). *The Oxford English Dictionary,* 2nd ed., prepared by Simpson J.A. and Weiner E.S.C., Clarendon Press, Oxford.

[15] Deacon, T. (1997). The Symbolic Species - The Co-evolution of Languages and the Human Brain, The Penguin Press, London.

[16] Pinker, S. (1994*). The Language Instinct, The New Science of Language and Mind*, The Penguin Press, London.

[17] Darwin, C. (1871). *The Descent of Man, and Selection in Relation to Sex,*2nd edition, Appleton, New York.

[18] Chomsky, N. (1972). *Language and Mind,* 2nd. Ed., Harcourt Brace Jovanovich, New York.

[19] Lieberman, P.(1992). On the Evolution of Human Languages, *Proceedings of the Workshop on the Evolution of Human Languages,* edited by Hawkins J.A. and Gell-Mann M., Vol. X, Aug. 1989 in Santa Fe, New Mexico, Addison Wersley 1992, Reading, Massachusetts.

[20] Deacon, T. (1992). Brain-Language Co-evolution, *Proceedings of the Workshop on the Evolution of Human Languages,* edited by Hawkins J.A. and Gell-Mann M., Vol. X, Aug. 1989 in Santa Fe, New Mexico, Addison Wersley 1992, Reading, Massachusetts.

[21] Dale, P.S. (1976). *Language Development, Structure and Function,* 2nd ed., Holt Rinehart and Winston, New York.

[22] Robins, R.H. (1989). *General Linguistics: an Introductory Survey*, 4th ed., Longman, London.

[23] New Scientist. (1994). An instinct for Language, *New Scientist*, No. 1931, June 1994.

[24] Alexandridis, N. A. (1986) Adaptable Software and Hardware: Problems and Solutions, *Computer,* Vol. 19, No. 2, February 1986.

[25] Fayad, M. and Cline, M.P. (1996). Aspects of Software Adaptation, *ACM Communications,* Vol. 39, No. 10, Oct. 1996.

[26] Bhargava, B. and Riedl, J. (1989). A model of Adaptable Systems for Transaction Processing, *IEEE Transaction of knowledge and data engineering,* Vol. 1, No. 4, Dec. 1989.

[27] Browne, D., Norman, M. and Riches, D. (1990b). *Why Build Adaptive Systems?,* Adaptive User Interfaces, Academic Press, London.

[28] Browne, D., Totterdell, P. and Norman, M. (1990a). *Adaptive User Interfaces,* Academic Press, London.
[29] Benyon, D. and Murray, D. (1993). Developing Adaptive Systems to Fit Individual Aptitudes, *Proceedings of the international Workshop on Intelligent User Interfaces*, ACM, pp. 115-121.
[30] Lazzaro, J. (1994). Adaptive Computing - Adapting GUI Software for the Blind is not so Easy, *BYTE,* Vol. 19, No. 5, May 1994.
[31] Ra, D. and Stockman, G.C. (1990). Use of Knowledge in a Semantic network for Guiding Natural Language Parsing, *Proceedings of the 3rd International Symposium on Artificial Intelligence*, pp.280-285
[32] Uehara, Y., Watanabe, T., Yoshida, Y. and Fukumura, T. (1990) Interpretation of Natural Language Queries on the Basis of a Semantic Data Model, *Proceedings of the 3rd International Symposium on Artificial Intelligence*, pp. 316-321.
[33] Hauptmann, A.G. (1991). From Syntax to Meaning in Natural Language Processing, *Proceedings of the 9th national Conference on Artificial Intelligence*, Vol. 1, pp. 125-130, July 14-19
[34] Dain, J.A. (1994). A Practical Minimum Distance Method for Syntax Error Handling, *Computer Languages,* Vol. 20, No. 4, pp. 239-252.
[35] Griswold, W.G., Atkinson, D.C. and McCurby, C. (1996). Fast, Flexible Syntacti pattern Matching and Processing, *IEEE 4th Workshop on Program Comprehension,* March 29-31. Berlin.