‘Evolution’ in Perspective

Ian Robertson
ir@cs.man.ac.uk

International Workshop on the Principles of Software Evolution at 20th. International Conference on Software Engineering, Kyoto, 1998. pp 172-176.

'Evolution' in Perspective

Ian Robertson
Informatics Process Group,
Department of Computer Science,
University of Manchester,
Manchester M13 9PL U.K.
ir@cs.man.ac.uk

1. Introduction

This work sets out to review the nature of classical natural evolution, and to see if we, in the field of software development, can reflect on these concepts to assist in developing evolvable software systems.

2. Biological Evolution

Evolution theory is the unifying theory of biology. It seeks to explain the continuous natural development of organic systems. It provides the basis of our understanding of how the functions of biological entities at all levels (population, organisms, genes) are the product of a non-random factor operating in conjunction with random factors within a framework of historical constraint

It provides an answer when biologists ask 'How did we get here', but not 'Why'.

Before Darwin, evolution was thought of as the 'unfolding of immanent potentialities within existing species', i.e. the manifestation of essences which lay dormant in earlier species. People were aware of the changes taking place in organisms over time, but did not understand the mechanism. One early popular theory, for example, had organisms progressing gradually from simple to complex, a misconception which remains today.

The principal thesis published in 'On the Origin of Species' [1] was that all organisms are descended from common ancestors, and that the principal mechanism of modification is the action of natural selection on individual variation.

From this, the currently accepted biological definition of evolution is: 'change in the properties of the population of organisms that transcend the lifetime of an individual'. Its principal features are:

- A population of individual organisms of a given species exists, and in this population the individuals are exhibited a variety of traits. Each individual exhibits a unique set of traits.
- In a specific environment, some traits will favour strength and longevity and thus favour reproduction of organisms exhibiting these traits.
- The traits of these offspring will substantially (but not wholly) reflect the traits of the parents.
- There is thus a gradual adaptation of the species to suit the current environment.

Thus only in its very broadest sense is evolution merely 'change'.

The properties exhibited by a living organism (phenotype) is determined by A) its genes and B) its environment. Changes to the phenotype directly caused by the environment are *not* evolution, because the changes are not inherited. The changes in populations that are considered evolutionary are *only* those that are heritable via the genetic material from one generation to the next. Successive generations of organisms exist in a succession of slightly different forms.The organism does not evolve to fit a ‘niche’ in the environment. In fact many aspects of the environment (e.g. other organisms) are themselves evolving.

There is one major mechanism for increasing the size of the gene pool (but not the only one), and that is mutation, and one for reducing (but also not the only one), and that is natural selection.

Natural evolution is neutral. It is not ‘progress’ from an inferior state to a superior one, neither can it be adjudged ‘good’: it is simply a matter of populations adapting to current surroundings. A trait which is considered to be ‘successful’ during one period may be ‘unsuccessful’ at another. The rate at which the species evolves compared with the rate of change of the environment is important. One of the reasons why a species can become extinct is that they do not evolve fast enough. e.g. Dinosaur extinction may have occurred at the end of Cretaceous and Tertiary Period of geological time when a large impact to this planet seems to have caused relatively rapid environmental change.

Evolution theory has undergone many changes in the 130 years during which it has been discussed since publication, but its central tenet, ‘adaptation benefiting the possessor’ remains unchanged [2].

3. The perspective from Software Engineering

For many years now, the only consistent overview of software system evolution has been provided by Lehman who contends [3] that programs evolve in a manner that is reminiscent of the evolution of biological organisms and of social groupings. Workers in other areas have been aware of the significance of evolution of systems and the problems manifested by it, however they have been unwilling or unable to relate the immediate concerns of their area with those identified by Lehman. Much of the concern of software engineers has focused on the problems arising from the problems of changing complex systems, in particular where the rate of change can be high. Manual systems of configuration management were inadequate, and the fact that software systems were accessible to software predisposed them to software configuration management. For many years the problems of evolution were synonymous with the problems of configuration management.

The situation tends to be perpetuated by the fact that information system methodologists are concerned about the evolutionary nature of system design (Dasgupta’s ‘ontogenic’ evolution [4]), rather than the evolution of the system itself (‘phylogenic’), and software maintenance is more concerned about the huge problems of how to best to adapt instances of systems that had been designed with little thought as to their future evolution.

In the context of software development, software engineers take two roles in the evolution of the system. Considering the software as a component of a socio-technical system, the software engineers generate ‘solutions’ to the problems posed as requirements, and, by some rational design process, eradicate non-favoured solutions. The solution preferred by the software engineers is then implemented. This is the case where the concepts upon which the system is based evolve. Once implemented, testing largely shifts from the domain of the software engineer to that of the user and owner in a real-world environment which identifies inconsistencies between what the users think the system ought to do, and what it actually does. These inconsistencies then become a new set of requirements for the developers for which they generate a number of solutions and select the most favoured. And so the cycle of evolution continues.

Software maintainers are immersed in the problems of how to absorb change into systems, how to deal with defect injection, and how to constrain 'de-evolution' of software. For them, software evolution consists of 'activities required to keep a software system operational and responsive after it is accepted and placed into production, including correcting defect, enhancing software, improving quality of existing software' [5], and their focus is on techniques and strategies for re-use, re-engineering, reverse engineering, correction, improvement and enhancement.

Evolution of database systems is more focused and determinate. It is essentially one of adapting schemas (or meta-data) from one definition to another without losing real data. The problem has been addressed, insofar as it is possible to perform incremental update to schemas, and the associated configuration management issue is under control. However the issue of migration of existing data into a new schema is far from being solved.

The need for adaptation has long been recognized in the software process community as the original view of the software process as linear, taylorist encapsulations of work have been discarded. Madhavji and Penedo [6], in a special issue of IEEE Software, stated that they considered evolution as 'A process of change which, from conception, the software process undergoes growth as well as adaptation', and its purpose is to 'fulfil the needs of people who perform the process (and) the developmental and organisational goals to be achieved'. However proposed mechanisms (referred to as 'meta-processes') concentrate on the problems of changing enacting models and again tend not to relate to Lehman's concerns. Recent thinking about the meta-process [7] provide a way of linking the micro- concerns of the software process technologist with the macro- concerns described by Lehman.

There are now signs of a shift in emphasis, with Perry[8] pleading for the adoption of a holistic view, stressing the nature and significance of the context of a system, the social and technical environment in which it operates.

According to M. M. Lehman [3], evolution is very natural to computer usage and the associated programs [9] and he observes that evolution first appears during the development process, the human activity that transforms a computer application into an operational system. His current work is focusing on the architecture of the multiple feedback loops that are said to drive system evolution.

Some computer scientists are concerned about systems that use genetic algorithms to optimize configurations but this work is still largely experimental although progress has been made towards an initial application - the design of printed circuits. In real world situations, the problem of optimization is made very difficult by complexity, uncertainty and the absence of rigorous specifications.

4. Features of evolving software systems

Our technical system consists of a multitude of software components: they are modules, procedures, statements which traditionally does not include the developers. The environment is the machines on which the software runs, the machines by means of which users interact, the users themselves, the owners of the 'problem'. There is no ‘variety generation’ from the system. The variety generation arises from the intellect of the developers as they solve the problem posed by the feedback from the environment. So, part of the environment is specially important, because it is that part which responds to the system and which generates feedback, which ‘sits in judgement’ of the system., i.e. the users, and system ‘owner’.

The following relevant features are expected or implied from biological theory in mechanisms to support evolution in dynamic artificial software systems:

- System and environment - what is included and what is excluded.
- Purpose - the reason for a system’s existence.
- Change - the nature of change.
- Testing - ascertaining how well purpose is being addressed.
- Feedback - how the results of testing are acted upon.
- Success - how to measure the suitability of the system.
- Timeous Change - the speed at which responsive change can be effected.
- Maintenance of structural integrity - change without increasing complexity.
- Learning/History - experience-based change.
- Analysis and decision about change - is the change feasible?
- Cycle of evolution - the algorithmic process.
- External/internal evolution - driven or spontaneous change.

There are thus many facets that can be considered under the umbrella of ‘evolution’. Of the term itself, we can say that it is ‘the adaptation of earlier form, structure, or behaviour of a system in response to environmental influences’. The changes are selected by rational analysis of generated alternative plans for the system, and the consequence of the change is that inconsistency between desired effects of the system and perceived effects are reduced or eliminated.

However any view of the evolution process is relative. For example, if a new version of software is released in which the only change is that it has been written in another language, the software objects have not evolved. From the language point of view, they have not evolved: they are utterly different. From a behaviour point of view, they have not evolved if the behaviour is the same. From a structural point of view, we can say that it has evolved if the second is say less complex and thus more understandable, than the earlier form.

5. Conclusions

A reflection on the concepts and issues associated with natural evolution indicates features that might be worthy of consideration in the software engineering domain. The previous section has indicated those which can be associated with evolving artificial systems. When considering evolution of such systems, to simply discuss their model as an object unrelated to its environment is of limited value and leads to isolated solutions.

Evolution of a system as a whole always takes place in an environment, or a context, and it is the appreciation of the difference between the system performance and what is expected of it in a particular environment that generates the drive for evolutionary change. This drive is made manifest by means of feedback between the environment and the system itself. The existence of such feedback would be expected from considerations of control theory related to self-regulating systems.

For any discussion on system evolution to be meaningful, it should always be made clear what the system under consideration is, the context within which the system is considered to be evolving, and the significant channels of feedback.

Acknowledgements

The genesis of this work was a discussion about concerns and issues with Badara Ali Kaba, recently of Centre de Recherche en Informatique de Nancy, Nancy, France. This was made possible initially through funding from the European Commission's PROMOTER BRA Working Group, and later through the Esprit Process Instance Evolution LTR Project (24840).

References

[1] C. Darwin. *On the Origin of Species*. John Murray, London, 1860.

[2] M. Ruse. *Taking Darwin Seriously*. Blackwell, Oxford, 1986.

[3] M. Lehman and L. Belady. *Program Evolution*. APIC Studies in Data Processing. Academic Press, 1985.

[4] A. Dasgupta. *Design Theory and Computer Science*. Cambridge Tracts in Theoretical Compter Science. Cambridge University Press, Cambridge, 1991.

[5] L. Arthur. *Software Evolution*. John Wiley and Sons, 1988.

[6] "Special issue on Process Evolution." *IEEE Transactions on Software Engineering*, 19, December 1993.

[7] J. Derniame, A. Kaba, and B. Warboys, editors. *Software Process: Principles, Methodology, Technology*. In preparation.

[8] D. Perry. "Dimensions of Software Evolution." In H. Müller and M. Georges, editors, *Proceedings International Conference on Software Maintenance*. IEE Computer Society Press, 1994.

[9] M. Lehman. "Programs, Life Cycles, and the Laws of Software Evolution." *Proceedings of the IEEE*, 68(9), September 1980.