# Software Self-Adaptability by Means of Artificial Evolution

Mariusz Nowostawski[1], Martin Purvis[1], and Andrzej Gecow[2]

[1] Information Science Department, University of Otago
PO Box 56, Dunedin, New Zealand
{MNowostawski,MPurvis}@infoscience.otago.ac.nz
[2] *temporary* Institute of Paleobiology Polish Academy of Science
ul. Twarda 51/55, 00-818 Warszawa, Poland
gecow@polbox.com

**Abstract.** Contemporary distributed software systems are reaching extremely high complexity levels which exceeds complexities of known engineering problems to date. Especially open heterogeneous multi-agent systems which may potentially be spread all around the globe, interacting with different changing web-services and web-technologies are exposed to demanding, dynamic and highly unpredictable environments. Traditional control-based handling of adaptability may not be suitable anymore, therefore there is a tendency for exploring different adaptability models inspired by natural/biological phenomena. In this article we review overall design of an adaptive software system based on a simple model of artificial evolution. We propose a new paradigm for handling complexity in dynamic environments based on a theory of self-producing self-adaptive software systems. We have substantial evidence to believe that a bottom-up approach based on self-production and self-maintenance may help to build more robust and more flexible self-adapting software systems. This paper introduces the new framework, provides analysis of some results, implications and future research directions toward a complete and self-contained theory of evolvable and self-adaptable software systems.

## 1 Introduction

In plain English *adaptation* is the act of changing something to make it suitable for a new purpose or situation. In software systems, the term *adaptation* is being used mostly, if not exclusively, with the second semantic meaning. What is usually meant by software adaptation, is that the system will continue to fulfil its original and the same purpose in a different circumstances, situation or environment. The adaptability in such software systems may be achieved by a set of feedback loops between the system, the controller monitoring and changing and *adapting* the system, and the environment itself. The system purpose is pre-defined in advance as a set of specifications, which are kept within the controller. The behaviour of the system is automatically altered if the expected outputs are outside of these pre-defined specifications. Such models are built analogously to

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 552–559, 2004.

a way automatic control systems work[1]. Most of them are based on top-down design and work well in limited environments, where changes in environment can be predicted and constrained in advance[2]. Such adaptive systems are tuned to particular kinds and specific levels of change in the environment.

Most of the adaptability in software systems is achieved via control mechanism like in automatics. There is a central system, with set of sensors and actuators, a controller, and an environment. Sensors sense an environment, system and controller are tied via a set of feedback loops and the controller tries to keep the system within a pre-defined boundaries. This model is very easily implementable, however it is extremely static and can be applied in situations where we can predict in advance all the changes and variations in the environment.

To make things more robust and flexible, we could implement into the controller an ability to learn, so the rules of changing the system become more dynamic, therefore the whole ensemble can follow changes in more dynamic environments. Yet, it still suffers some of the drawbacks of the simple model. Although in a different scale, there is a limit of environmental change the system can cope with, which is predefined within the learning mechanism itself.

## 2 Adaptability and Biological Inspirations

To fully benefit from life-like adaptability in artificial software systems, which (at least in theory) might match the levels of complexity of biological organism, we need a formal mathematical model of all the fundamental concepts like: life, organism, evolvability and adaptation. In this work we will use a formal deductive model of process of life described in details in [3,4]. Due to limited scope of this article we will only briefly highlight the main aspects of the theory.

The major step in understanding the process of evolution in natural life was done by Darwin[5], who proposed mechanisms by which purposeful adaptive changes take place via processes of random mutation and natural selection. Darwinian mechanisms postulate reproduction, statistical character of change processes, and the process of elimination. After elimination the object ceases to exist (is not alive anymore). The formal deductive model we are going to use is just based on these rudimentary darwinian mechanisms, and adaptability in software is inspired by the mechanisms which handle purposefulness in natural life.

There has been earlier attempts to formally or mathematically define *life*, *complexity*, *organism*, *organism boundary* and information content. In this work we use a theory of evolvable ensembles. Some of these ideas have been developed over the last three decades [6,4,3] with the roots of the proposed model can be traced back to the work of John von Neumann [7,8]. Von Neumann submitted that a precise mathematical definition must be given to a basic biological theories. The work of von Neumann has been, most noticeably, pursued and extended by Gregory Chaitin [9,10]. Slightly different approach in formalising process of life has been pursued by others (e.g. [11]).

Similarly to von Neumann and Chaitin, our model is based on the discrete model universe, an automata space, with a finite number of states. Note however,

that the formal definition of information, which is being used throughout this article, is defined in a context of static collection of bits (as it was originally proposed in [12]) rather than an algorithmic settings (as in [10]).

## 3 Theoretical Foundations

The model we discuss here can be applied to different software architectures. It is suited for object-oriented web technologies or multi-agent systems. It is not constrained however to these paradigms, and it can be easily implemented in any computing paradigm, for example the presented results were obtained on a simple computing model based on finite-state automata without memory.

For sake of uniformity we will use the term *object* to denote a coarse-grained unit of processing within a given computing framework. It can be the actual object like in the object-oriented paradigm, it can be an individual agent from agent-oriented paradigm, etc. The important part is that the individual *object* is an ensemble of lower-level structures, which can be manipulated at runtime. That is, the *object* can be disassembled into its individual components, and re-assembled again within the system during the actual operation of the system. In other words a certain level of reflection is needed within a computing paradigm for the proposed model to be implementable.

We will use here the basic notion of information as introduced in [12]. *Information* is a selection from a set of available choices. Each *object* contains in its structure the information of how to react with the external stimuli. Any given relation between two sets of events or even conditional probability which maps two sets of events, so that selection in one set causes the selection in another, represents the casual relationship between respective events. Such a relationship can be represented as *code*, and we will use the term *code* to refer to this kind of relationships. The process of mapping one set into another we will call: an *encoding*.

The mapping between the conditions and the expected reactions to these conditions of our software system can be understood as if it was a specification of the system. The specification traditionally describes the general behaviour and the relationships of all objects in a given system, therefore describes the overall behaviour, or expected behaviour of the system. Specification is based on some pre-defined properties and characteristics of the interactions of the software system with its environment. In our case however, the specification (or as we said earlier, the mapping between the condition and the expected reactions) is constructed dynamically together with the process of adaptation to the changing environment.

A *purpose* is a pre-defined effect, which can be obtained by a particular set of stimuli. We use here the term *stimuli* as if they were events observed and anticipated by the object. Therefore, to keep the software system within the expected trajectory the object tries to find the mapping between specific events (causes) which lead to particular effects. In other words the system tries to find such a configurations which lead to an expected results. The object responsible for finding such configurations is called *controlling unit*, or *controller* for short.

The actual recording of the selection from a set of causes to the given effect is called *purposeful information*, and this *purposeful information* is been stored in the structure of the objects themselves.

The relations between events in the environment and effects is, in most cases, not known in advance. This is why the basic (but not the only) activity of a controller is to test many hypotheses. The testing requires controller to setup hypothetical causes which are encoded via the interactions of the object with its environment into effects, and then record the obtained results within the structure of the object (for future reference).

## 4 Experimental Model

The aim of this work is to propose such a mechanism, which would follow in open-ended fashion the changes in the environment. We require the process of adaptation to be uniform and long. For this we use the theory of living systems which exhibits similar (if not the same) characteristics as those required for self-adaptive software systems.

Given an ensemble of objects we want it to automatically maintain itself within a presupposed limits, and to compensate any changes in the environment which may interfere with the operation of the ensemble. We require the process to be possibly most effective, i.e. such that will lead fastest to higher values of aptness. This leads to increased amount of redundancy which allows the controller to test as many hypotheses at the same period of time as possible. (However, this may be limited to one hypothesis at a time in some application.)

We expect an ensemble to continuously improve its aptness, that is to increase its effective information content within a given environment, and to follow any changes in the environment. This can be achieved by different alterations to the ensemble itself. There are some very characteristics structural tendencies which one can observe during the development (growth, adaptation and evolution) of the ensemble, and we discuss them in the next section in more detail.

We do not want objects to exist and maintain themselves just for the sake of exiting. We want all the software components which occupy our resources to be in some way *useful*, therefore there is we require all the objects not only to maintain their existence, but also to maintain their *usefulness*. All objects not *useful* anymore should be automatically removed from the system (in a similar way to garbage collection).

*Usefulness* is very subjective term, and it must be specified in advance. It is not similar to specification of the system, because it has to be expressed on different, higher level. It is a meta-purpose. It can be specified by means of a CPU cycles, e.g. processes not utilising CPU should be garbage collected, by user interactions, etc. Of course, all objects referenced by *useful* objects are useful and shall not be removed.

To facilitate experiments we have used a simple model of finite-state automata. Such a model has many advantages: it is simple, easy for analysis

and implementation, provides easy means to aggregate objects (automata) into higher-level structures (ensembles), re-arrange connectivity and relations, etc.

Consider a complex system, an object, transforming one set of signal into another one. The former set of signals represents the environment $x$ of the object, the latter set $y$ is the object's answer to the environment.

The system consists of deterministic finite-state automata without memory. Each automaton receives two or more input signals and it transforms them into two or more output signals. Signals are several bits long, in the simulation from 1 to 4. Each automaton represents a simple function operating on input signals. The number of input signals received and output signals sent may vary among automata, however during simulations we used 2-input and 2-output automata. Each output may provide input signals for only a single input, and the same for output signals. The input and outputs of automata are somehow interconnected to form an aggregate of automata. Free inputs of the whole aggregate receive signals form the environment $x$, while free outputs represent the answer $y$ of the aggregate. In our model, due to the symmetry, both: an object or the environment could represent an active process of encoding of a given cause into the effect, or the input signal into the output one. Note however that this time the active part is of the situation is the object, thus the environment plays the passive role. In the stable (passive) environment $x$ the signals $y$ are changing because object is changing. The outcome $y$, is compared with the assumed ideal goal $y^*$ and the similarity is the basis for calculated aptness of a given object.

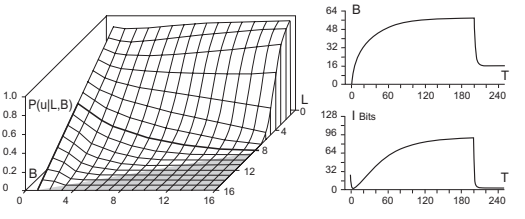


**Fig. 1.** Probability of accepting signal change

Figure 1: The probability $P(u|L, B)$ of accepting $u$ of the change of $L$ signals (2-bit each) from 16 signals of the result $y$, where $B$ signals match the ideal $y*$ (on the left). We are interested in the area of higher aptness B, e.g. $B > 8$. Area shown in grey represents all the non-acceptable changes, with $P(u) = 0$. One can see the clear tendency of accepting the small changes. On the right hand side there is plotted the example of the history of growing aptness and amount of purposeful information for $y$ with 64 2-bit signals. At the time $T = 200$ the condition of non-decreasing aptness has been cancelled and both characteristics rapidly dropped down to the point of the highest entropy.

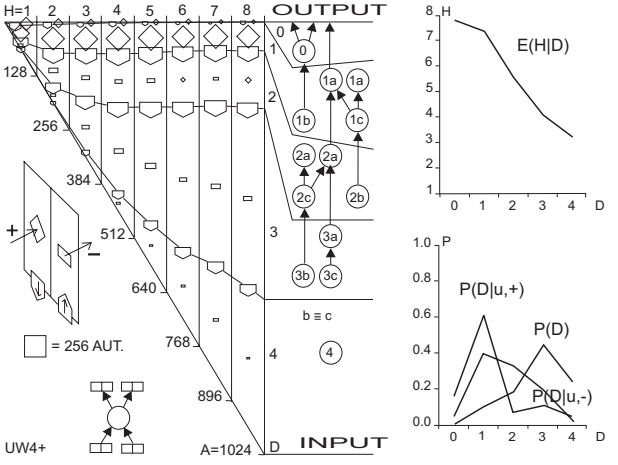


**Fig. 2.** Basic results of the structural tendencies simulations

## 5 Structural Tendencies

To investigate the structural tendencies for an evolving object we will use the model of automata aggregate is being proposed above.

Figure 2 shows, on the left diagram, an average growth of an aggregate with 2-bit signals. This is with strict non-decreasing aptness condition when adding and no constraints when removing automata. Aggregates were allow to grow up to 1024 automata. The growth is divided into 8 stages 128 automata each. Addition and removal process have had uniform probability distributions. Nevertheless, one can observe that addition was dominating, leading to increased size of the aggregate. In the middle we have shown the *depth* of the automata, in other words the measure of the functional order $D$. The depth of automaton depends on the receivers of its output signals. On the left hand side the aggregate is divided on the layers with the stated balance of addition and removal of automata. In between, we have shown the flow balance. It is clear that automata are pushed downward by the strong terminal addition, i.e. frequent addition of new automata at the terminal positions. The acceptance of changes ($u$) is contained only within very shallow regions, deeper it ceases (right bottom corner). It provides consistency of the historic order and functional order (the right upper diagram). The right diagrams cover all the higher stages from stage 4, where aggregate has already 384 automata. These diagrams are averaged over different runs and different setups, e.g. with 2-,3-, and 4-bits signals.

The changes of the object are constrained by the improvement condition. Before we investigate the influence of the object's changes on the changes of the signals $y$, we calculated the acceptance probability distribution of change of $y$ in function of the change's size and the object's aptness. We have noted the "small change tendency" [13,3] (see also Figure 1 on the left). This in itself is a base for all further investigated structural tendencies.

The object consists of many randomly (but subject to improvement condition) aggregated automata. Signal vectors $x$, $y$ and $y^*$ are up to a couple of dozens dimensions big: 64 in our simulations. Such an aggregate is subjected to random changes of the structure, i.e. the individual automata are added and removed from the aggregate at random (see Figure 2). We have investigated also the influence of changing environment and the ideal signal $y^*$. The obtained tendencies are in close correlation to these observed in common day life, in computer science or in biology. In particular, in software engineering one can observe this phenomenon at work. Some software systems reached the complexity levels that it became very difficult to make any changes. The only alternative is to add new features rather than modify the existing system. In biology such phenomena are observed and referred as de Beer's *terminal changes* or Wiessman's *terminal additions*. It is somehow similar to controversial Haeckel formulation of ontogeny recapitulates phylogeny [3]. The intuitive model proposed by Haeckel seems to be very adequate to the observed phenomena [6,14]. The aggregate was growing, the addition and removal of individual automata was concentrated at the area near output terminals, which are the youngest evolutionary regions. This process is also referred as *terminal changes*. In case there is positive balance of additions to removals in this region, it is called *terminal addition*.

When the environment changes, the aggregate has only one way to continue operating properly: it must reproduce within itself the disappearing signals from the environment. In our model it is achieved by addition of new automata. Some of the automata added as terminals reproduce the important changed signal from the environment and thus maintain the aptness of the aggregate. This is a very strong tendency which we call *covering tendency* [4,3].

There is another characteristics of our proposed self-adaptable software model. The controller will inherently pick these hypothesis, which are re-enforcing discussed here structural tendencies. In other words these characterised by the highest probability of increasing aptness. This happens because the actual probability distributions are part of the information recorded in the object structure (parameters of the controller). This is in a way self-regulating mechanism of adjusting the mechanisms of selecting the hypothesis to be tested - a mechanism (tendency) of re-enforcing structural tendencies.

## 6 Conclusions

Interestingly, coming independently from two different set of basic definitions and assumptions, both models (Chaitin [10] and Gecow [3]) achieved same conclusions. The process of improvement and the object growth are being accomplished by carrying along all the previously developed structure, as a new pieces of the structure is being added [15]. The simulations and statistical analyses together replicate similar conclusions of Chaitin. The experimental proof of this is that ontogeny recapitulates phylogeny, i.e. each embryo to a certain extend recapitulates in the course of its development the evolutionary sequence that led to it [10]. The preliminary results based on the finite-state automata model

discussed in the previous sections present very promising tendencies and robustness. In the discussed scenario the model exhibited self-adaptability and could be successfully used in some applications with binary input-output signals.

Future work will include a) formal definitions together with analysis of aggregation of aggregations, and tendencies to improve tendencies needs to be further explored and investigated; b) more experimental data needs to be collected, and bigger real-life problems must be tested and evaluated. Better understanding of the necessary reflective capabilities is also required.

## References


1. Kokar, M., Baclawski, K., Eracar, A.: Control theory-based foundations of self-controlling software. IEEE Intelligent Systems (1999) 37–45
2. Meng, A.C.: On evaluating self-adaptive software. In Robertson, P., Shrobe, H., Laddaga, R., eds.: Self-Adaptive Software. Number 1936 in LNCS. Springer-Verlag, Oxford, UK (2000) 65–74 IWSAS 2000, Revised Papers.
3. Gecow, A.: Statistical analysis of structural tendencies in complex systems vs. ontogeny. PhD thesis, Instytut badań Systemowych PAN, Warsaw, Poland (1986)
4. Gecow, A., Hoffman, A.: Self-improvement in a complex cybernetic system and its implication for biology. Acta Biotheoretica **32** (1983) 61–71
5. Darwin, C.: On the Origin of Species by Means of Natural Selection. John Murray (1859)
6. Gecow, A.: A cybernetical model of improving and its application to the evolution and ontogenesis description. In: Proceedings of Fifth International Congress of Biomathematics. Paris (1975)
7. von Neumann, J.L.: The general and logical theory of automata. In Taub, A.H., ed.: John von Neumann – Collected Works. Volume V. Macmillan, New York (1963) 288–328
8. von Neumann, J.L., Burks, A.W.: Theory of self-reproducing automata (1966)
9. Chaitin, G.J.: To a mathematical definition of 'life'. ACM SICACT News 4 (1970) 12–18
10. Chaitin, G.J.: Toward a mathematical definition of "life". In Levine, R.D., Tribus, M., eds.: The Maximum Entropy Formalism. MIT Press (1979) 477–498
11. Eigen, M., Schuster, P.: The Hypercycle: A Principle of Natural Self-Organization. Springer-Verlag (1979)
12. Shannon, C.E., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press (1949)
13. Gecow, A.: Obiekt żywy jako stan odchylony od statystycznego stanu równowagi trwałej [living object as a state displaced from the stable statistical equilibrium]. In: Proceedings of I Sympozjum Krajowe CYBERNETYKA-83. PTC, Warsaw, Poland (1983)
14. Gecow, A.: Strukturalne tendencje w procesie udoskonalania [structural tendencies in a process of improvement]. In: Proceedings of I Sympozjum Krajowe CYBERNETYKA-83. PTC, Warsaw, Poland (1983)
15. Simon, H.A.: The sciences of the artificial. MIT Press (1968)