

# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index  
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?  
Contact [book.department@intechopen.com](mailto:book.department@intechopen.com)

Numbers displayed above are based on latest data collected.  
For more information visit [www.intechopen.com](http://www.intechopen.com)



# From Evo to EvoDevo: Mapping and Adaptation in Artificial Development

Gunnar Tufte

Norwegian University of Science and Technology  
Norway

## 1. Introduction

Artificial development takes its inspiration from the processes of biological development (Wolpert; 2002). The process of mapping genetic information in the genotype to a phenotype that expresses structure, behaviour and functions (Hall; 2003). In nature this is the process of making a zygote (fertilised egg) develop to a multicellular organism. As all living systems and their subsystems are evolved, developmental processes are also a product of evolution. In biological development, an initial unit – a cell, holds the complete building plan (DNA) for the organism. It is important to note that this plan is generative – it describes how to build the system, not what the system will look like. Units have internal state, can communicate locally, can move, spawn other units or die. Groups of units may also exhibit group-wise behaviour i.e. a group state. The developmental stages from the zygote to the multicellular organism, although interdependent and not strictly sequential, may be categorized as *pattern formation; morphogenesis; cell differentiation and growth* (Wolpert; 2002). Information carried in the ever evolving genome is the information that passes from generation to generation and the genetic information is contained within a cell that serves as both the constructor and construct of the phenotype.

An important feature of natural development is that the developing organism develops and operates within an environment. The environment is not only the arena in which the behaviour and function of the organism unfolds, it is also an important source of information for the outcome of the developmental process. As stated, the genome may be considered information exploitable by the developmental process to “build” the organism. In biology the environment also serves as a source of information for the development of the organism. As such, the developmental process can include the environmental information as an input information source enabling adaptation through the developmental process. This implies that natural organisms include developmental plasticity, i.e. phenotypic plasticity (West-Eberhard; 2003). As such, the evolved organism is a product of the information in the genome, the present environment and the cellular developmental processes that is initialised in the zygote.

Recently developmental approaches have once again come to the front in the area of bioinspired systems. The motivation for moving towards developmental systems is multitudinous ranging from early work on self-reproduction (von Neumann; 1966), to more recent attempts in overcoming limitations in Evolutionary Computation (EC), e.g., scaling

Source: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos,  
ISBN 978-953-307-008-7, pp. 572, October 2009, I-Tech, Vienna, Austria

and complexity (Kitano; 1990) or towards a desire to include properties inherent in living organisms that may be favourable for artificial systems, e.g. robustness (Miller; 2004) or adaptation (Tufte; 2008).

Artificial developmental mapping include various techniques and methods ranging from rewriting systems, e.g. L-Systems (Lindenmayer; 1971), to cellular approaches, where the genome is present and processed autonomously in every cell (Laing & Arbib; 1971). Further, the information available to the mapping process can be varied from systems depending only on the genotype and an axiom (Bentley & Kumar; 1999), to systems including environmental information where exploitation of this information form the phenotypic structure and behaviour together with information carried in the genome (Tufte & Haddow; 2007b).

In a system including a developmental mapping the role of the genome is radically changed from a system where the genome is considered a description, e.g. a blueprint, to a system where the genome may be viewed as information on how to build the system. That is, the genome contains information that is exploited in the generative process of development. The processing of the genome may be based on gene regulation Lantin and Fracchia (1995). Each development step, or stage in the mapping, produces a candidate phenotype which emerges during development. Gene regulation implies that different parts of the genome are expressed in different cells at different times cued in context with the environment and the emerging phenotype. As such, the genome size may not reflect the size or complexity of the phenotype and opens for systems that can generate very large scale repetitive structures (Kitano; 1998), or even structure of arbitrary size (Sekanina & Bidlo; 2005).

The generative process of development is not limited to the process of development of an adult organism, or in the artificial domain a finalized phenotype. A developmental mapping is not a process that is “turned off” when the finalized adult stage is reached. The process is working on the organism throughout it's lifetime. This ever lasting process is important to keep the organism healthy and functional. Damage, e.g. at the cellular level, can be repaired by regenerating the damage tissue and cells may be replaced if they fail to fulfil their purpose. In artificial systems an ever ongoing developmental process can, as in it's natural counterpart, obtain robustness. Such robustness is the product of a system that can regenerate itself if it is disturbed or damage (Miller; 2004). In the artificial domain, if included, the plasticity property of the artificial organisms can be exploited to target systems that based on the present environment can adapt their structure (Tufte; 2008) to obtain a target functionality despite changes in the environment in witch they operate and/or artificial organisms that can change their computational function based on the present environment (Tufte & Haddow; 2007b).

In this chapter the main topics are robustness and adaptation in evolved artificial developmental systems. The chapter includes a discussion on principles for artificial development. Possible environmental influence is described before a developmental model is covered for use in following examples. There are three examples covering robustness, plasticity and adaptation in an environmental context. Further a discussion on scalability, complexity and evolvability in developmental systems is included.

## 2. Artificial development

Whatever the motivation for including a developmental mapping in a system is that there exists some target purpose for the system. As we are dealing with artefacts there is no

common purpose as reproduction is for living systems. Instead the purpose of the artificial evolved and developed organisms is connected to a specific problem domain. This implies that properties of artefacts are defined by the targeted problem.

In the domains of Artificial Life and Evolutionary Computation the target involves some sort of computation. Even though it is not trivial to distinguish what is computation and what is not (Toffoli; 2004), the phenotype or the developmental process itself is computational. To get a better way of defining what the purpose of a developmental system is the outcome, or developed phenotype, may be categorised in two: a structure or a structure of computational units. Constructing such categories can not be said to be an exact classification. The different classes are overlapping and for many cases there exists no clear classification category. However, to be able to clearly know, or decide, what the purpose of the artificial system is, it is crucial to know what kind of system that is targeted. That is, to be able to reach a system that fulfil its purpose we must know what function, i.e. computation, is to be the emergent result of evolution.

If systems targeting structures are considered the purpose of the system is the evolution and development of the structure itself. Here the process of development is used to create a structure from an initial condition, which may be some initial structural element(s). The structure may grow and/or reshape under some underlying developmental processes.

In Figure 1 an example taken from the work of Steiner et al. (2009) illustrates development of a structural purpose. In this figure the cellular growth can be seen as the output of the generative process growing structure by cell division. The initial structure is shown at top left and the finalized structure is shown at the two last panels at the bottom right. In the last panel a top plate is added to the developed structure. The purpose of the structure is to be a light-weight mechanical structure that can support mechanical stress from a weight placed on the added top plate.

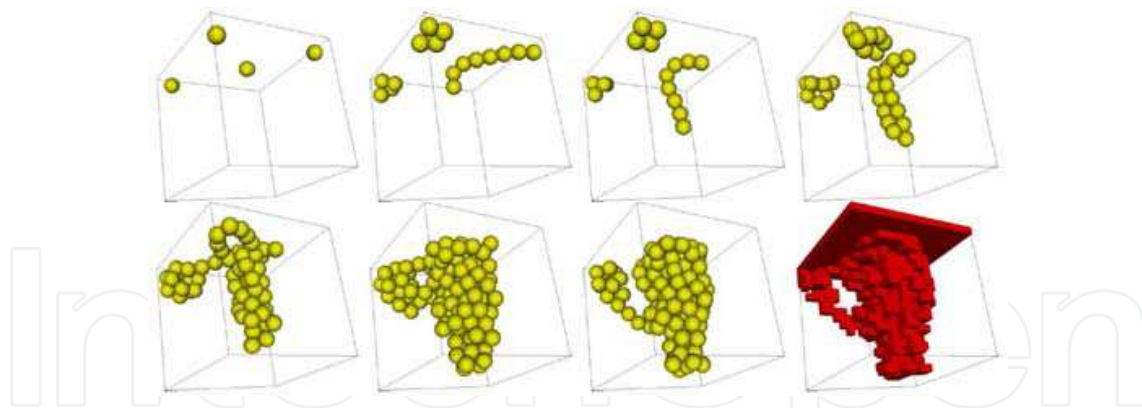


Fig. 1. Development of a mechanical structure. The developmental process starts at the top left corner. The finalized developed structure is at the bottom right corner. (Courtesy of Steiner et al. (2009))

The developmental phenotypes in Figure 2 show two developmental steps in the process of developing a structure with a computational function. Here the structure is not the target; the target is a computational function. As such, the developmental process generates structures of computational elements, here addition (+), subtraction (−) and multiplication (\*). The developing structure includes input nodes, at the left, and an output node, the last block at the right. In the figure a block is marked *DUP* this is a special operator that can copy and insert parts of the current function into itself, i.e. development by self-modification.

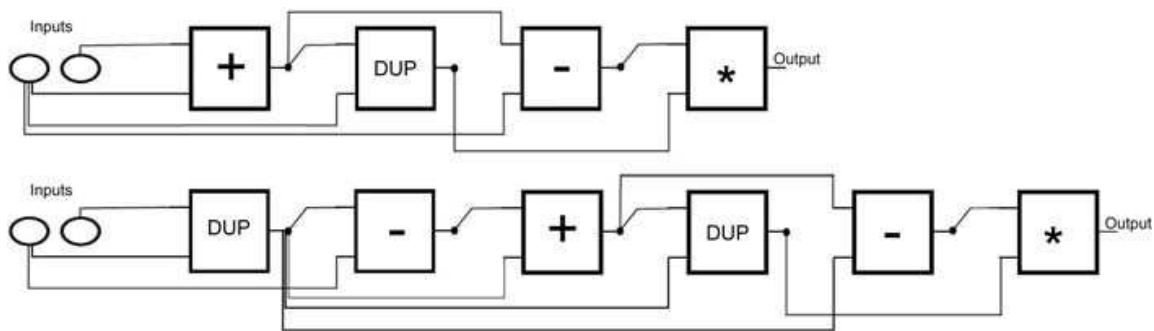


Fig. 2. Development of a structure consisting of computational and self-modification elements. Snapshot of two structures created by a developmental process. (Courtesy of Harding et al. (2007))

The two given examples of development in Figure 1 and 2 share and show key elements of developmental mappings. The evolution of a genome that carry information that is exploited by a developmental iterative process and the exploitation of information in intermediate phenotypes for the creation of the next phenotype on the developmental path toward the finalized phenotype.

In Figure 1 the genome, evolved by an Evolutionary Strategy (ES), is a short description of conditions and actions based on local cellular neighbourhood. Cell division is an action increasing the number of cells in the structure. The input information to the developmental process in a cell is local information in the cellular neighbourhood. This implies that the developmental process uses information from the emergent developing structure as information to decide the next cellular developmental action. As such, the structure emerges out of described conditions and actions in the genome regulated by information from the intermediate phenotypic structures.

The second example presented in Figure 2 also used an ES for evolving the genome. However, here the genome is an initial structure in the form of a graph consisting of connected computational nodes and some special nodes for self-modification. The iterative process can copy parts of the graph into another position or delete parts of the graph. As such, the initial size of the graph-genome can be a fraction of the developed graph-phenotype at end of development. The process of self-modification always operates on a current version of the graph generating the next intermediate graph-phenotype. This gives a system that can grow from a small graph to a large more complex graph based on self-modification.

Even though the two examples share basic principles that make them developmental they also clearly show the difference in evolutionary development of structures and computational structures. In the system of Steiner et al. (2009) the purpose was a lightweight structure that could resist mechanical stress. The structure of the phenotype itself is here directly evaluated. In the work of Harding et al. (2007) the actual structural composition is not measured. The result of an evaluation is based on the computation performed between the input nodes and the output node. As such, the developmental process works on a structure that may not reflect computational complexity of the phenotype.

This section illustrates what artificial development can be and the principles of large systems that emerges as a result of a generative process exploiting information from an evolved genome and intermediate phenotypic properties. The two examples illustrates the large diversity of developmental models and target purposes. In the following section the



bio-inspired approach of artificial development will be further investigated and expanded toward a synthesis of evolution, development and environment, i.e. EvoDevo (Robert; 2004; Hall et al.; 2004).

3. Evolution, development and environment

In the previous section the mapping process itself was discussed and exemplified. However, the mapping process was taken out of context. In an evolutionary developmental system the evolutionary process include an evaluation function together with the developmental phenotype and the environment. Further, if adaptation and robustness are issues, the question of robustness and adaptation to what must be answered. The obvious answer, if looking to biology, is the environment. For an artificial system this answer may not be obvious. However, here we stick to the same answer for the artificial systems considered.

3.1 Environmental influence

Let us consider artificial developmental systems in relation to environmental information. In Figure 3 three different ways of treating an environment in artificial developmental systems are presented. Figure 3(a) shows a system that does not include an environment. In this approach an Evolutionary Algorithm (EA) is used to evolve genomes that are mapped from genotype to phenotype taking the information in the genome and possible intermediate phenotypic properties into the mapping process. The phenotype that is produced by the mapping process is evaluated by a fitness function. In the illustration the double arrow between the mapping process and the emerging organism (phenotype) shows that there is a possibility for the mapping process to exploit information from the emerging phenotype. In this system the phenotype is given by the information in the genome and the mechanisms in the mapping processes, a totally deterministic system. If the system in Figure 3(b) is considered. An environment is explicitly included. The organism develops within an environment and the fitness evaluation is based on the organism's performance in the environment present. As such, the feedback of fitness depends on the environment. In this system the fitness is connected to the organism and the environment. As such, the system depends not only on the genome and the developmental mechanisms, the environment influence on the evolutionary path of the system.

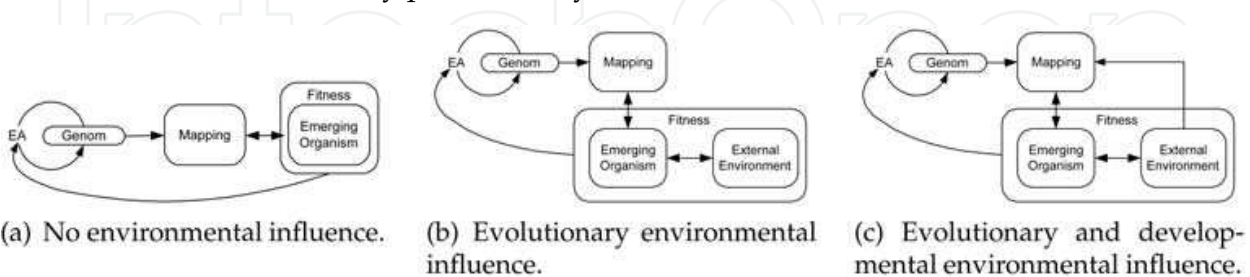


Fig. 3. Possible environmental influence on artificial EvoDevo systems.

In the last system illustrated in Figure 3(c) the environment is, as illustrated by the extra arrow from environment to the mapping, available and can influence on the outcome of the mapping process. This implies that the emergent phenotype is a product of the genome, the emergent phenotype and the environment. This implies that the fitness feedback is based on a phenotype that depends on specific environmental conditions. As such, the phenotype

here can change structure and/or function depending on environmental conditions. Further, the double arrow between organism and environment in Figure 3(b) and 3(c) indicate that there can exist a mutual dependency between an organism and its environment, i.e. mutual perturbatory channels (Quick et al.; 1999).

Now let's return to the question "robustness and adaptation to what"? As stated, the answer is the environment. In the presented systems the environment in all systems can be considered as information. The environment can serve as information enforced into the operating conditions of the artificial organism (Figure 3(b)). It is also possible to exploit the environmental information in the mapping process influencing on the phenotypic outcome (Figure 3(c)). Further, the environmental information is included in the evolutionary process by the possible influence of the fitness feedback to the EA. In Figure 3 the environment is termed external. External implies that environmental information is independent of the EvoDevo system and can change or fluctuate beyond the control of the EvoDevo system.

### 3.2 Robustness

If robustness is targeted in this context the system presented in Figure 3(b) featuring an environmental influence on the evolutionary outcome, systems with robustness to environmental fluctuations can be obtained. A system can be evolved to operate in different environmental conditions. However, this ability is based on a system operation that is robust in itself, i.e. not influenced by fluctuations in the environment. It is important to note that the view on environment as information include enforced disturbance or changes to the organism itself. As an example of this is shown by Miller (2004) where a developmental system evolved to generate a flag structure that was externally disturbed by changing parts of the structure. The system was able to regenerate itself through development.

Robustness through regeneration in developmental system is highly connected to selforganization. A system is in a state of stability and even if it is disturbed it will return to this state by regeneration.

### 3.3 Adaptation

Developmental adaptation is highly connected to plasticity, the ability for the individual in a species to respond to environmental conditions. Such response may materialise in form of individuals that can develop different structural and/or behaviour properties based on environmental conditions, i.e. plasticity. In an artificial developmental system plasticity gives a single genome several possible developmental paths. Which path an organism follows is given by the specific environmental conditions present during development. Further, due to the possibility to have an active developmental process throughout the lifetime of an individual fluctuations in the environment can cue reorganisation of structure and behaviour through gene activation as to be able to operate (Tufte; 2008).

Adaptation by plasticity may also be viewed in a context of self-organizing systems. In contrast to robustness making a system capable of returning to the same state after perturbations plasticity in a system allow the system to leave its state and there after move to a different state whilst maintaining its operation.

Robustness and adaptation on the evolutionary and developmental level is further explained and investigated in Section 5.

### 3.4 Levels of environmental information

In this section environment has been seen as an external information source that can be exploited by evolution and possible environmental sensitive mechanisms in the developmental mapping. However, environment is present in developmental systems at different levels. At the cellular level environment exists as an intra-cell environment that the artificial DNA resides in, also referred to as the *cell's metabolism* (Federici; 2004; Gordon & Bentley; 2005). The next level of environment, found in most development models, is the neighbour environment referring to the *inter-cell environment*, enabling communication between neighbouring cells (Bongard & Pfeifer; 2003; Tufte & Haddow; 2003; Miller; 2004; Federici; 2004). Further, the external environment influence discussed affecting the phenotype may be part of the available information for a development process.

## 4. Example: a cellular approach

In this section a developmental model is presented for later use in examples and to relate the topics in the previous sections to developmental mechanisms and features. The system as a whole is close to an Evolutionary Developmental (EvoDevo) approach. This implies a developmental system with a possibility to include information from the environment, intermediate structures and behaviour in addition to the genetic information carried in the genome. The artificial organisms the model target are developing structures capable of computation. The computational architecture is based on Cellular Automata (CA) originating from von Neumann (1966). von Neumann's Self-Reproducing Automata is in itself close to artificial development (Sipper et al.; 1998), cells with a finite number of states that can selfreplicate, i.e. an expanding cellular structure. The developmental phenotypes are based on the cellular computational machine paradigm (Sipper; 1997). A non-uniform CA is developed, i.e. the structure/form, emerges out of the set of developmental rules capable of cellular growth, differentiation and cell death. The details of the system are only discussed in brief. For a complete description of the system – see Tufte (2008) (developmental model) and Tufte (2006) (evolutionary algorithm).

### 4.1 Developmental model

The development model is based on cellular development. This implies that the genome is present and processed autonomously in every cell. In the model, the cell also contains the functional building blocks. For the experiments herein the application sought is that of a digital circuit (phenotype). Figure 4(a) illustrates the developmental system – the cell. The cell is divided into three parts: the genome (the building plan); the development process (mechanisms for cell growth and differentiation) and the functional component of the cell. The information in the functional components represents the type of the cell and the cell's state is described by the outputs of the functional components.

The genome consists of a set of rules. Rules are restricted to expressions consisting of the type and state of the target cell and the types and state of the cells in its von Neumann neighbourhood. There are two types of rules i.e. change and growth rules. Cell growth is a mechanism to expand the organism. A growth rule result provides the direction of growth: grow from north **Gn**; east **Ge**; south **Gs** or west **Gw**. It is important to note that these rules are expressed in terms of where the source of the cell growing into the target cell is. Describing where a cell is growing from enables a fully parallel implementation of the system to be created whilst retaining the possibility that cells in effect may grow in all four



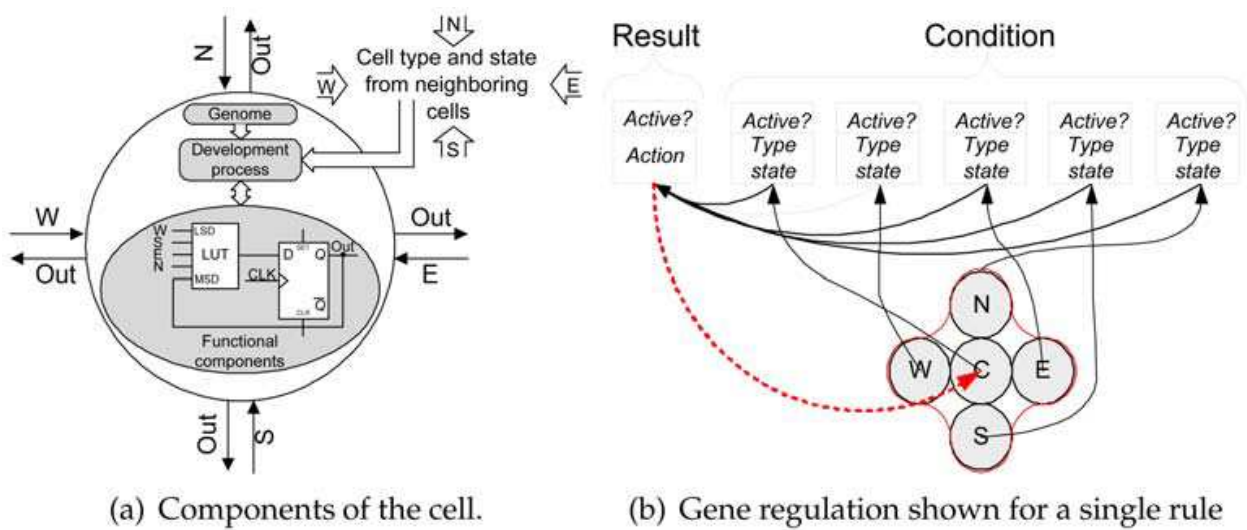


Fig. 4. The basic cell and a rule showing the gene regulation in the cellular development model.

directions simultaneously. Growth rules have two restrictions. First, the target cell must be empty – this is to prevent growing over an existing cell and thus specialising the cell with a new cell type. Secondly, the cell to be copied into the target can not be empty.

Differentiation changes a cell's type i.e. its functionality. The result part of a change rule states the type of cell the target is going to be changed into. Cells have the following types: valid cell types, don't care (DC) or empty. However, the empty cell is not a valid target cell type.

Each rule consists of a result and a condition. The conditional part provides information about the cell itself and each of the neighbouring cells. In the development model presented in (Tufté & Haddow; 2005), the type of the cell was applied to describe these cells. However, to introduce external environment, state information is also needed. State information provides a way to include information relating to the functionality of the organism at a given point in time as well as information about the external environment – the empty cells in the environment also have state information. As such, a cell is represented in the condition of a rule by two genes representing its type and its state. However, a target cell is only represented by one gene: it's type for change rules or growth direction for growth rules. The state of cell may be 0, 1 or DC. DC is introduced to provide the possibility to turn on or off this environmental influence. The development model is applied with and without the information from the external environment and functional organism.

Firing of a rule can cause the target cell to change type, die (implemented as a change of type) or cause another cell to grow into it. Figure 4(b) illustrates the process of evaluating a rule. For each cell condition, the cell type and state are compared and if the conditions are true then that part of the rule is active. If all conditions are active then the result will become active and the rule will fire. Activation of the result gene is expressed in the emerging phenotype according to the action specified.

In a development genome multiple rules are present. Multiple rules imply that more than one rule of a given cell may be activated at the same time if their conditions hold. To ensure unambiguous rule firing, rule regulation is part of the development process. If the first rule is activated, the second rule can not be activated. Activation of the second rule prevents activation of the third rule, etc.

The functional components of the cell is an Sblock (Haddow & Tufte; 2000). The content of the look-up table (LUT) defines functionality and is, herein, also used to define the cell type. The LUT is the combinatorial component and the flip-flop is the memory element – capable of storing the cell state. The output value of an Sblock is synchronously updated and sent to all its four neighbours and as a feedback to itself. The LUT definition for the cell types used herein can be seen in Table 1.














Cell type	LUT hex	Function name	Graphical representation
0	0xFFFF0000	<i>no change Emty</i>	
1	0x66666666	$XOR_d W \oplus S$	
2	0x3D3D3D3D	$XOR_c E \oplus S$	
3	0x0FF0FF0	$XOR_b N \oplus E$	
4	0x55AA55AA	$XOR_a W \oplus N$	
5	0x55FF55FF	$NAND W \bullet \overline{N}$	
6	0xFF00FF00	$\downarrow SouthPropagation$	
7	0xCCCCCCCC	$\uparrow NorthPropagation$	
8	0xF0F0F0F0	$\leftarrow EastPropagation$	
9	0xAAAAAAAA	$\rightarrow WestPropagation$	
10	0xE8808000	$T \geq 4$	
11	0xFEE8E880	$T \geq 3$	
12	0xFFFEFEE8	$T \geq 2$	

Table 1. Definition of cell types and their functionality

One update of the cell's type under the execution of the development process is termed a development step (DS). A development step is thus a synchronous update of all cells in the cellular array. The update of the cell's functional components i.e. one clock pulse on the flip-flop, is termed a state step (SS). A development step is thus made up of a number of state steps.

The initial condition is applied before development starts. This means that all empty cells are set or reset depending on the given initial condition. To avoid empty cells updating their output values from their von Neumann neighbourhood, all cells of type Empty are set to update their outputs based on only their own output value at the previous clock pulse. A empty cell will retain its initial state – environmental information, until the emerging organism grows into it.

4.2 Development of cellular computation machines

Figure 5 show an example of development of a cellular machine and its behaviour. The phenotype is an emerging non-uniform Cellular Automata (CA) (top). Development of the structure goes through steps, Development Steps, where the structure is formed by growth (expanding the number of cells) and differentiation (changing the rule of a given cell). The different colours in the emerging phenotype represent what CA rule the cell contains. White cells are considered empty. The dashed lines indicate that there exists events that are not shown in the figure, e.g. the phenotypic structure between DS 8 and DS 98 are not shown. The behaviour of the system in Figure 5 (bottom) is the state space produced from an initial state executed by the developing non-uniform CA. The space time plots for the behaviour consists of 100 State Steps for each development step. This implies that there exist 10 000

space time plots describing the behaviour of the system. It is important to note that in this system behaviour exists from the first cell throughout the life-time of the organism.

5. Examples: plasticity, robustness and adaptation

In order to highlight the working and gain an increased understanding of developmental mappings the developmental model presented in Section 4 is applied to some examples. The examples are all targeting robustness and adaptivity. The model presented targets organisms that structurally can be seen as a non-uniform cellular automaton that emerges as a product of development. Similar, the functionality can also be seen as the result of running the developing cellular machine. As such, the developing organism is the phenotypic cellular structure, shown at the top of Figure 5, and the functionality of the system is the dynamic behaviour of the organism, shown at the bottom of Figure 5.

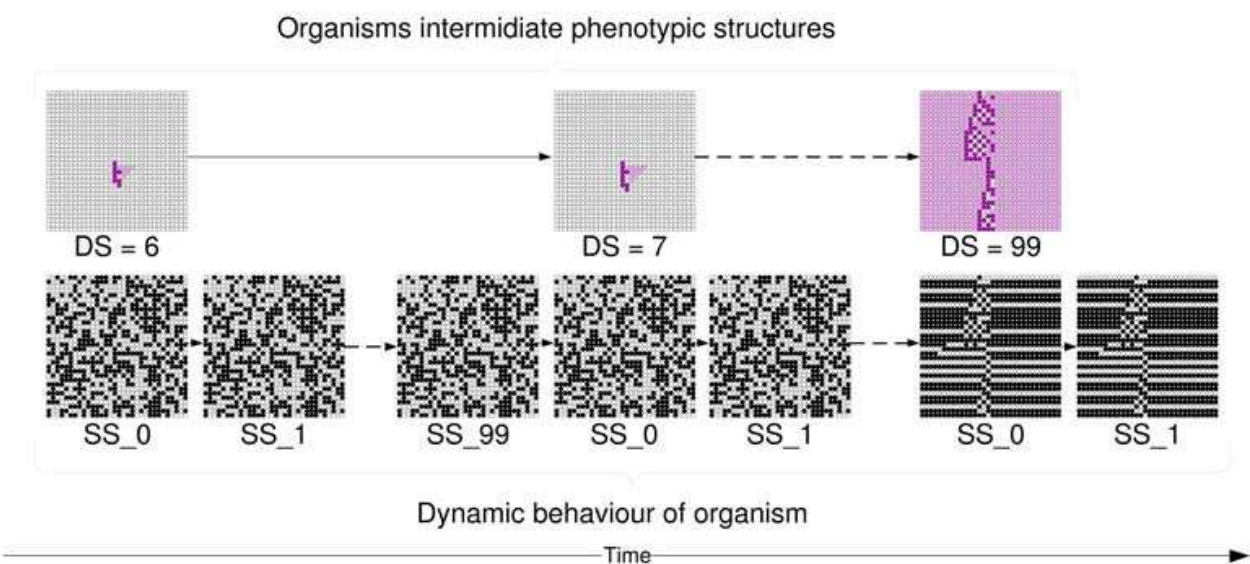


Fig. 5. Snapshot of the development of phenotypic structure (top) and the corresponding emergent behaviour (bottom) shown as space-time pattern.

The application chosen in these examples is a sequential counter where counting is based on the state information of the entire cellular space and the sequential operation of the functional components of the cells. The application thus places a requirement on the tuning of the development genome (by evolution) and the emerging phenotype (by development) for such sequential digital circuit behaviour. A counting sequence is defined in the cellular array as the number of logical “1”s in the cellular array increasing by one for each state step. As such, the functionality only concern the global count of cells outputting a logical “1”, i.e. an emergent global property out of local interactions of the functional components of the cells.

The fitness function for achieving this counter behaviour is based on the best counter sequence obtained at each developmental step, i.e. a lifetime evaluation. As such, the total fitness is the sum of the longest counter sequence found on each developmental step. As the main topic here is robustness and adaptation the main common property is development in fluctuating environments.

In all the examples presented the developmental process is allocated 100 development steps. At each development step 100 state steps is executed. As such the counter sequence is a

product of the dynamic behaviour caused by the state steps at each development step. The size of the genotype in all examples was set to 32 rules. Here the specific example results are selected from larger collections of experimental results. For all details on the results and setups see (Tufte & Haddow; 2007a,b; Tufte; 2008).

5.1 Robust phenotypes

As a first example the evolution of phenotypes that is robust to environmental fluctuations is presented. However to emphasis the principle of robust phenotypes an example with no evolved robustness is also presented together with a setup targeting evolved robustness. In these examples the environmental regulation, i.e. cell state information, is set to don't care in the developmental model. That is, evolutionary robustness, as illustrated previously in Figure 3(b).

The functional requirement in the two examples is the counter behaviour described. In Figure 6 the setup for showing the influence of a fluctuating environment is presented. A setup that is not exposed to environmental fluctuation during evolution is shown in Figure 6(a). In this experiment a single environment is used throughout evolutionary time (phylogeny axis). All organisms develop in this common environment (ontogeny axis). The evolved result is genomes that are specialized to cope by producing a functional phenotype in this environment. The setup in Figure 6(b) is changed to targeting genomes that can develop to functional phenotypes in a set of different environments. Here each genome is developed in ten different random generated environments. As such, the EvoDevo system targets organisms that can cope with different environments. The difference in fitness evaluation is that the system in Figure 6(a) only bases its evaluation on the functionality in a single environment while the setup in Figure 6(b) gets its fitness measurement by the average functionality in a set of ten different environments.

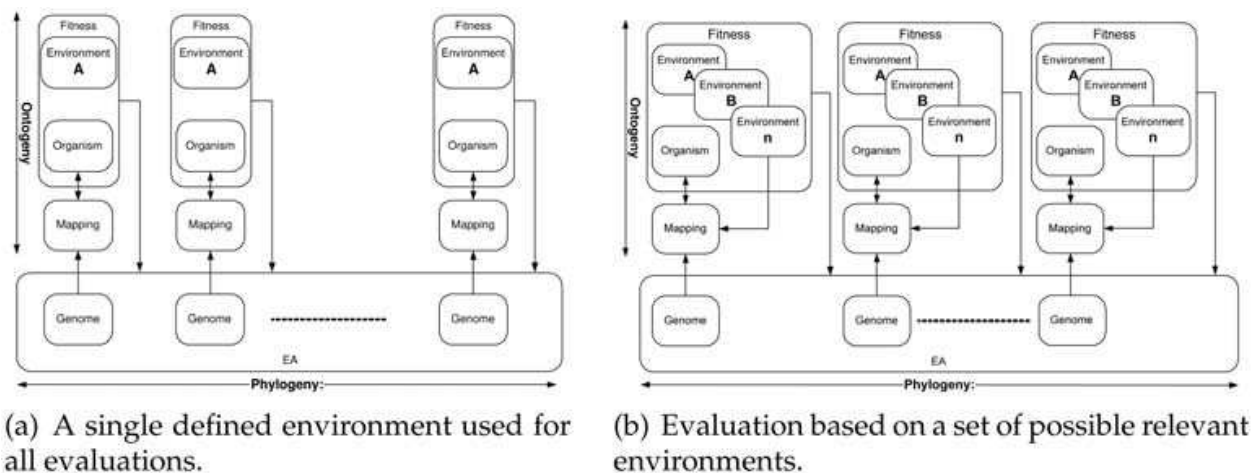


Fig. 6. Evolution and developmental setup including environment for the different experimental examples presented.

To actually test and expose the difference between the two approaches shown, the genomes evolved was exposed and re-developed in ten new random generated environments. The result of this exposure is shown in Figure 7. In the figure the fitness obtained from the evolutionary process is given by the black bar termed 0 on the X-axis. The resulting fitness value is given by the Y-axis in the plot. The measured fitness value for the re-evaluated and



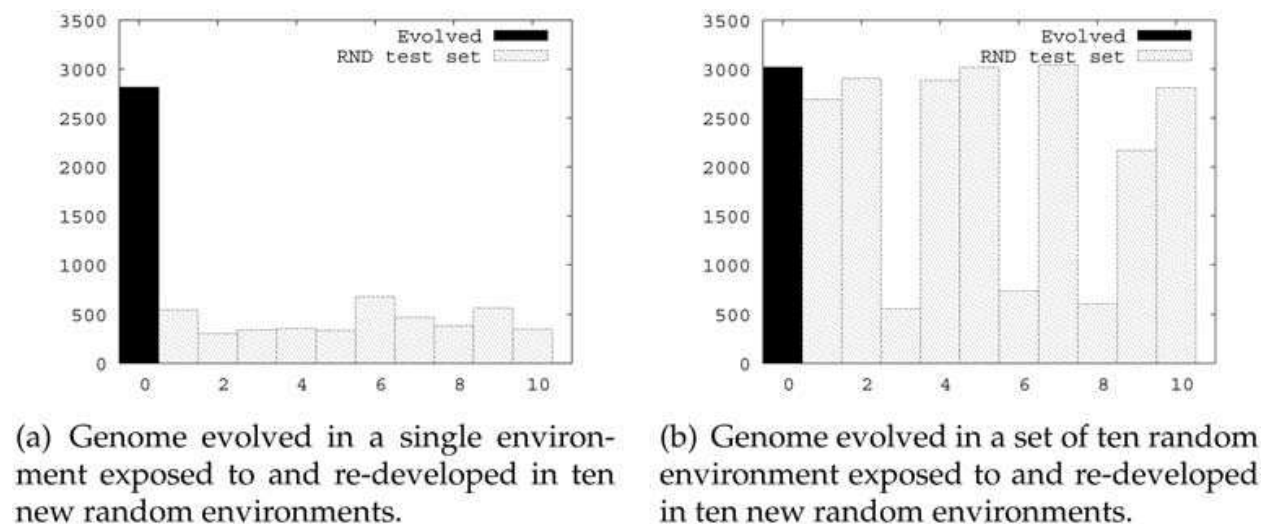


Fig. 7. Result of evolutionary robustness to fluctuating environments.

re-developed organisms in the ten new random generated environments is given by the grey bars, numbered 1 to 10. The plot in Figure 7(a) shows how the evolved genome was tuned to obtain a high fitness score in the presence of a known environment. However, when exposed to other environments, the organism totally fails to achieve its goal functionality. This is caused by the strong evolved adaptation to the environment present during evolution. The strong evolved dependency on a single environment to achieve functionality is decreased if the result from the EvoDevo setup in Figure 6(b) is re-developed and re-evaluated. In Figure 7(b), the result of a similar re-development and re-evaluation to ten new random environments is shown. The result shows how the same genome can develop and achieve the targeted functionality despite environmental fluctuations. However, the result shows that some environments, i.e. 3, 6 and 8, do not provide an environment that the evolved genome can develop within whilst retaining functionality.

## 5.2 Plasticity

In this example, the main goal is to demonstrate plasticity. Plasticity can be seen as the property of phenotypic plasticity in biological organisms. To achieve such plasticity, the evolutionary algorithm is set up to target genotypes that can develop to different structural organisms in two different environments, whilst the functional requirement of counting is maintained. To enable environmental influence, hence phenotypic plasticity, inclusion of a possibility to exploit cell state information in the genotype is put under evolutionary control. This move can be seen as moving from the EvoDevo system in Figure 3(b) to the system presented in Figure 3(c).

Figure 8 illustrates the setup for an EvoDevo system that can show the presence of phenotypic plasticity in artificial organisms. The EvoDevo system used to evolve organisms with the preferred functionality is shown in Figure 8(a). The evolutionary algorithm presents candidate solutions to the developmental mapping process. Each genome is developed in two different environments, A and B, shown in Figure 8(b) and 8(c). The initial cell (zygote) is set to be a cell of type 5 (NAND). The initial zygote is shown in Figure 8(d). This example implies that each genome is developed and evaluated twice in two different environments. As such, the difference in environmental information can be exploited to develop different structural organisms by influencing the rule regulation.



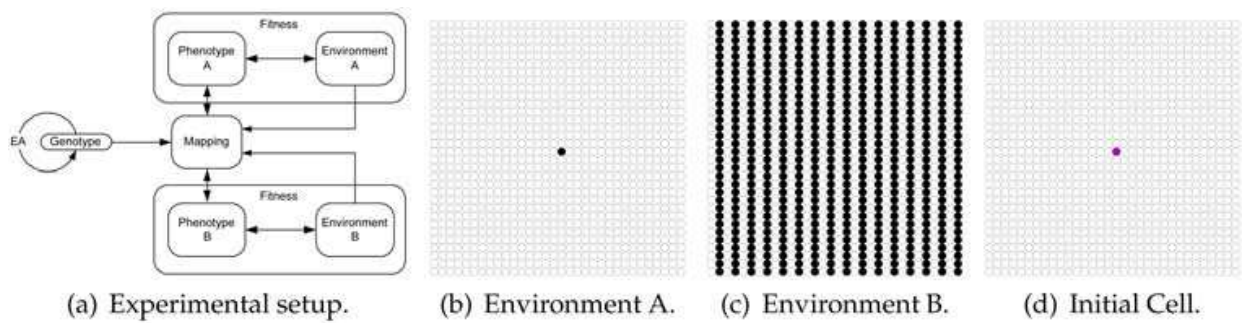


Fig. 8. Experimental setup for the achievement of demonstrating phenotypic plasticity cued by different environments.

The result of evolving genomes that can exploit information in the environment is presented in Figure 9. The difference in structural composition of the phenotypes resulting from the two different environment can clearly be seen in Figure 9(a) and 9(d). The environmental information has, as stated, in Section 4 influence on the regulation of the rules, hence a possibility to control what rules that are expressed and eventually the timing of expressing rules. Figure 9(b) and 9(e) show the activation pattern for rules in the given genome. The plotted star (\*) indicating that that specific rule was expressed at that given developmental step. The plotted line indicates the total number of cells in the organism expressing an active rule. As such, the cause of the achieved structural difference can be traced to the difference in expressing rules over the developmental time. By examining the plots it can be seen that there exist common rules expressed at different times, further there are rules that are only expressed for one of the two environments. Note also that only parts of the genome are actually expressed during development. There exist parts of the genome that is not

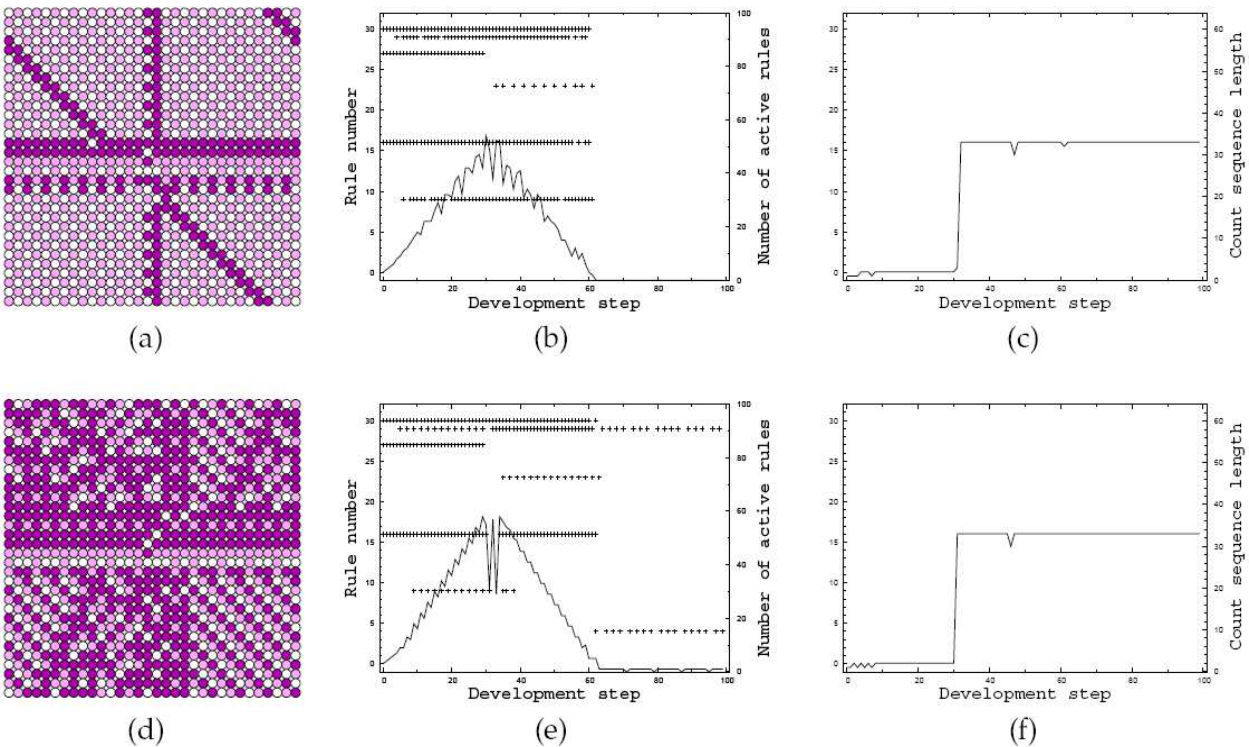


Fig. 9. Environmental influence exploited to retain functionality. Result of a genome developed in the two different environments.

expressed but may be genetic information that previously in evolutionary time was exploited. It may also be parts of the genome that has never been expressed and as such may be neutral information (Shipman et al.; 2000).

The purpose of the evolved organisms was not actually the structural composition but the functional counting behaviour of the developing organism. In Figure 9(c) and 9(f) the measured functionality, i.e. counter length, for the two developed organisms is plotted for each development step. Here the preservation of functionality despite the different environmental conditions can be observed. The two results show how the two different structural phenotypes are able to retain a common functional property by phenotypic plasticity.

### 5.3 Adaptive phenotypes

Toward organisms that are truly adaptive the EvoDevo example in Figure 7(b) is executed within a setting that opens for phenotypic plasticity. That is, the EvoDevo setup of Figure 3(c) is applied to the evolutionary example of evolving robust phenotypes in Figure 7(b). Now the organisms are not only robust to fluctuations in the environment, the organisms can adapt their structural composition to the environment present during development. As such, there is a possibility for evolution to exploit specific combinations of environmental traits to achieve organisms that can develop to phenotypic structures that depends on the environment which it develops within whilst retaining the functional goal.

The result of the approach can be seen in Figure 10. Here a single genome is capable of developing to functional phenotypes in all but one of the new random environments it is exposed to. If the phenotypic structure is examined, the presented result is obtained by a process that develops different phenotypic structures depending on environmental traits. However, even here the EvoDevo system does not achieve a result that is equally fit in all environments. This is caused by an evolved dependency on specific combinations of environmental traits. Such dependency can cause poor performing phenotype structures, e.g. by competing counters structures in a single organism cancelling each other out.

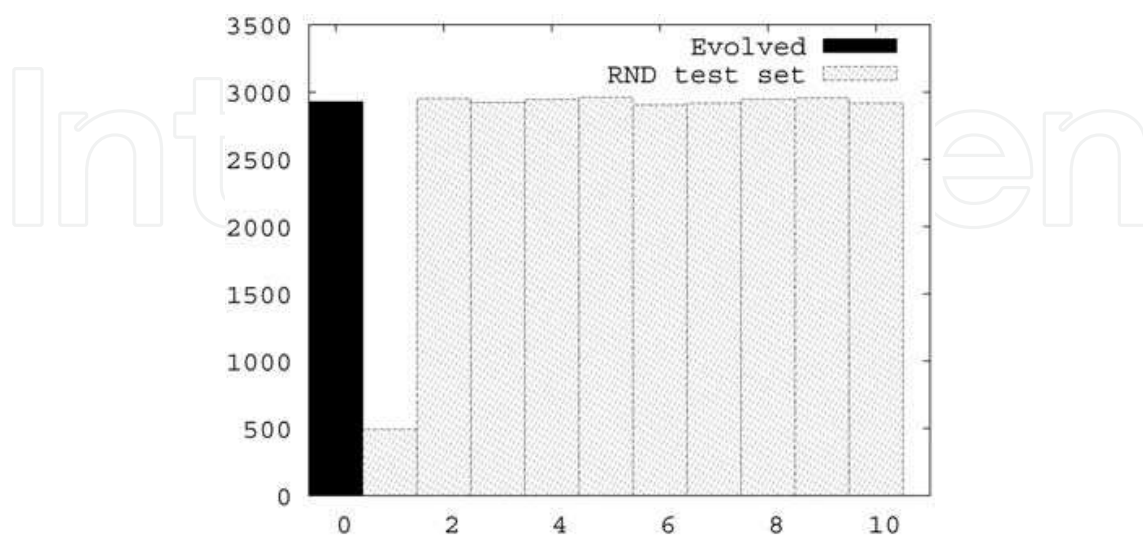


Fig. 10. Genome evolved in a set of ten random environment exposed to and re-developed in ten new random environments. Phenotypic plasticity exploitable.

### 5.4 Discussion

In this section robustness and adaptation was seen and presented as two different properties. However, the view on what robustness and adaptation is and imply, may be seen at different levels. If the examples given are viewed at a higher level of abstraction or in a more biological relation, the way an ability to obtain a given function in fluctuating environments is achieved only serves as two different approaches to reach the purpose of the system. Here the purpose is organisms that can produce a counting function despite fluctuations in the environment. Even though the two presented systems here serve an identical purpose the inclusion of possible phenotypic plasticity by environmental influence give two quite different systems. The difference in system attributes is the amount of information available to the developmental mapping.

In Figure 11 the examples of Section 5.1 and 5.3 is presented as trajectories (Tufte; 2009) showing the emergence of the phenotypic structures for the two examples. Figure 11(a) show the developmental paths, i.e. trajectories, from the ten environments (top) to the final stable phenotype structure (bottom). In this representation of the developmental history it is clear that this system will always develop to the same final phenotype structure, i.e. the structure of the phenotype is not influenced by external information in the environment. The trajectory for the system opening for phenotypic plasticity in Figure 11(b) on the other hand clearly show influence by external information. The ten initial conditions (in the middle) can take different paths to different phenotypic structures. In addition the developmental trajectory contain loops at several point indicating that the structure is unchanged for more then one developmental step. Such loops are broken if the state information, i.e. behaviour, changes to express a change in the phenotype by development.

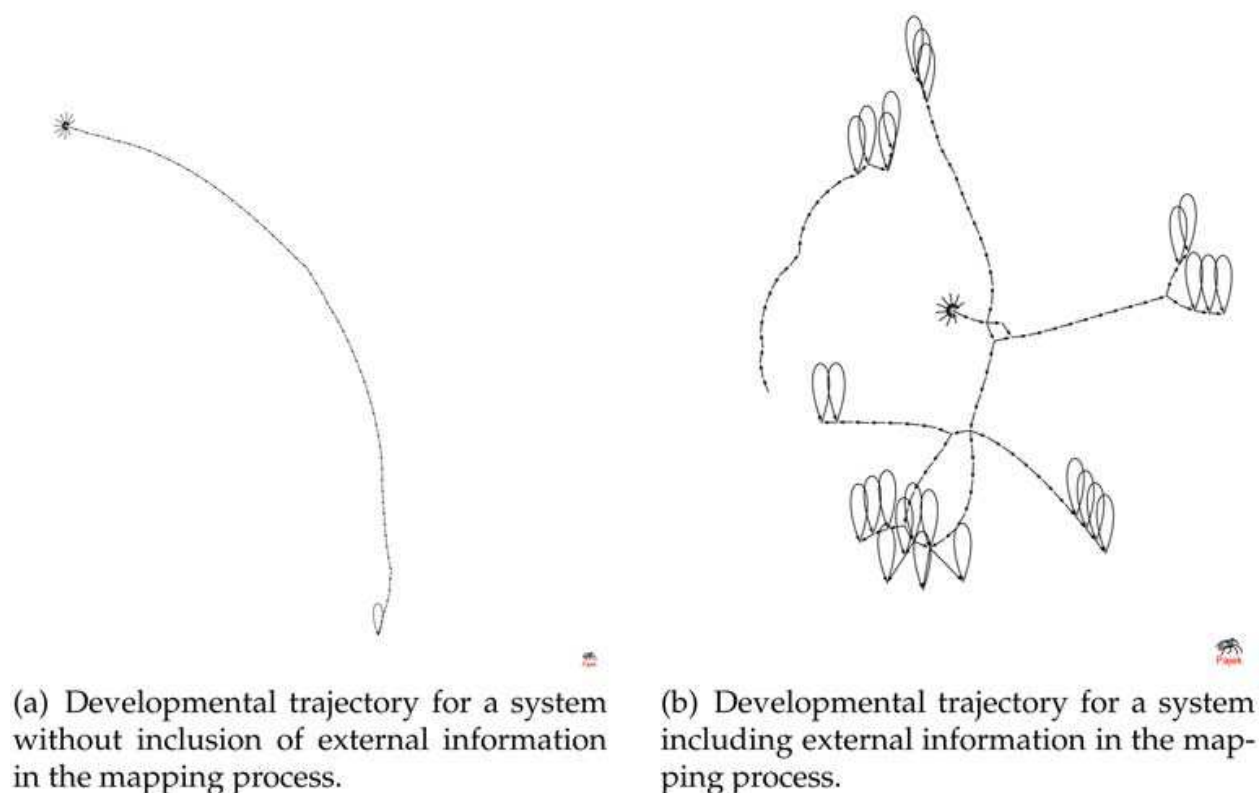


Fig. 11. The developmental trajectories for the emergent phenotype structures in a fluctuating environment. The plots are generated by PAJEK (Batagelj & Mrvar; 1991).

The highlighted difference between the two examples show the importance on how information is actually incorporated in artificial developmental models. Information amount and the included mechanisms for intercommunication between cells are the factors leading to what forms that can emerge out of a given developmental mapping (Laing; 1972). As such, the construction of developmental model should be based on an awareness on what information amount and exchange that is required for the process of developmental self-organising for the sought form, function and bio-inspired properties.

## 6. Notes on scalability, complexity and evolvability

The main topic here is developmental mappings in the context of robustness and adaptation. However, artificial development show promising features in other domain of EC.

The issue of scalability is one of the main topics of interest in introducing mapping processes inspired by development. Bentley and Kumar (1999) addressed the scalability by investigating the performance of direct encoding versus developmental encodings to generate phenotypes of different structural size, e.g. number of cells in the final phenotype. Other similar approaches dealing with phenotypic size as the scaling factor have looked into the affect on performance of the development process regarding scaling of phenotypic size and genome size (Tufté & Thomassen; 2006). Another way of considering scaling is to look at the functional property of the developed phenotype (Kitano; 1998). In such approaches work have been done on keeping-up functional performance when the size, i.e. number of neurons, of the phenotype increases (Gauci & Stanley; 2007). Examples of general solutions to scaling in developmental systems have been found. Sekanina and Bidlo (2005) demonstrated development of arbitrarily large sorting networks. Harding et al. (2007) attacked the problem of finding a general solution that generates sequences of squares Spector and Stoffel (1996). However, these examples show scaling within a specific problem, i.e. not systems that scales toward more complex problems. The diverse approaches and views described on scaling in developmental systems calls for an understanding of what the goal of scaling is, what changes to be made to scale, e.g. change in resources, and what results of scaling that can be expected for the problem at hand.

Complexity, or rather being able to evolve and possibly develop complex machines that are capable of complex computation, or evolutionary growth of complexity (McMullin; 2001) in such systems are areas of much interest. The motivation range from an urge to create complex artificial organisms (Eggenberger; 1997; Kitano; 1998) to a more theoretical interest (McMullin; 2001; Lehre & Haddow; 2003; Kowaliw; 2008) in the working of mapping processes and artificial organisms emerged from such processes. As such, understanding of the emergence of complexity in artificial systems is important to be able to exploit artificial development toward the creation of evolved large systems capable of complex computation. The difference in the way information in the genome is treated in developmental mappings compared to direct mapping approaches influence on the evolvability (Kirschner & Gerhart; 1998) of such systems. The fact that the information in the genome is relatively small and the information content in the phenotype is large implies that the genotype space is much smaller then the phenotype space. Even though, as shown in Section 5, a genome can be the origin of several phenotypes along the development path and that external information can be included in the gene regulation, the phenotypic space that a given genome size and mapping model can explore is usually given to larger then the genotype space. Further the intermediate phenotypes along the developmental path together with the property of



having a system with an ever working developmental process influence on how and when evaluation is to be carried out, i.e. influence on the fitness landscape. The process of regulation, e.g. herein gene regulation, imply neutrality. Parts of the genome is not expressed in the phenotype hence there exists more then one genotype describing a given phenotype. As such, introducing artificial developmental mappings strongly influence the evolvability of the system (Harding & Banzhaf; 2008) by changing the focus of evolution from the content of the genotype to an interplay between genome, environment and the emerging phenotype (Taylor; 1999).

## 7. Conclusion

Despite the fact that ideas of artificial development have been around for almost half a century (von Neumann; 1966; Codd; 1968; Laing & Arbib; 1971) the field is still young, as shown by the renewed interest. However as most youngsters it struggle whit what to make of itself. There are promising works in areas as robustness and adaptation, as described in this chapter. Furthermore the areas of complexity and scalability toward large complex systems have shown to be areas where artificial development is a most prominent player. Further there is an increased interest often motivated by an urge toward new mediums changing today's view on a machine's components, architecture and computation (Miller & Downing; 2002). As such new machines may require a drastic reconsideration of how we as human designers can design and construct such machines. A change toward design by EvoDevo systems seems a promising direction. It is ironic that von Neumann, one of the pioneers for computers as we know them today, worked on ideas close to artificial development and at the same time showed an interest for machines made up of unreliable parts that should be capable of reliable computation (von Neumann; 1956), a scenario of high interest toward e.g. nanoscale and biochemical machines.

## 8. References

- Batagelj, V. and Mrvar, A. (1991). Pajek program for large network analysis., <http://vlado.fmf.uni-lj.si/pub/networks/pajek/>.
- Bentley, P. J. and Kumar, S. (1999). Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem, *Genetic and Evolutionary Computation Conference (GECCO '99)*, pp. 35-43.
- Bongard, J. C. and Pfeifer, R. (2003). *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, Springer-Verlag, chapter Evolving complete agents using artificial ontogeny, pp. 237-258.
- Codd, E. F. (1968). *Cellular Automata*, Association for computing machinery, Inc. Monograph series, Academic Press, New York.
- Eggenberger, P. (1997). Evolving morphologies of simulated 3d organisms based on differential gene expression, *Fourth European Conference on Artificial Life*, MIT press.
- Federici, D. (2004). Evolving a neurocontroller through a process of embryogeny, *Simulation of Adaptive Behavior (SAB 2004)*, LNCS, Springer, pp. 373-384.
- Gauci, J. and Stanley, K. (2007). Generating large-scale neural networks through discovering geometric regularities, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 997-1004.



- Gordon, T. G. W. and Bentley, P. J. (2005). Development brings scalability to hardware evolution, *the 2005 NASA/DOD Conference on Evolvable Hardware (EH05)*, IEEE, pp. 272 -279.
- Haddow, P. C. and Tufte, G. (2000). An evolvable hardware FPGA for adaptive hardware, *Congress on Evolutionary Computation(CEC00)*, IEEE, pp. 553-560.
- Hall, B. K. (2003). Unlocking the black box between genotype and phenotype: Cell condensations as morphogenetic (modular) units, *Biology and Philosophy* 18(2): 219-247.
- Hall, B. K., Pearson, R. D. and Müller, G. B. (2004). *Environment, development, and Evolution Toward a Synthesis*, The Vienna Series in Theoretical Biology, MIT-Press.
- Harding, S. and Banzhaf, W. (2008). *Organic Computing*, Springer Verlag, chapter Artificial Development, pp. 201 - 220.
- Harding, S. L., Miller, J. F. and Banzhaf, W. (2007). Self-modifying cartesian genetic programming, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 1021-1028.
- Kirschner, M. and Gerhart, J. (1998). Evolvability, *Proceedings of the National Academy of Sciences of the United States of America* 95(15): 8420-8427.
- Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation systems, *Complex Systems* 4(4): 461-476.
- Kitano, H. (1998). Building complex systems using development process: An engineering approach, *Evolvable Systems: from Biology to Hardware*, ICES, Lecture Notes in Computer Science, Springer, pp. 218-229.
- Kowaliw, T. (2008). Measures of complexity for artificial embryogeny, *GECCO '08: Proceedings of the 10th annual conference on Genetic and Evolutionary Computation*, ACM.
- Laing, R. (1972). Artificial organisms and autonomous-cell rules, *Technical report*, The University of Michigan.
- Laing, R. and Arbib, M. (1971). Automata theory and development ii, *Engineering, college of - technical reports*, The University of Michigan.
- Lantin, M. and Fracchia, F. (1995). Generalized context-sensitive cell systems, *Proceedings of Information Processing in Cells and Tissues*, University of Liverpool, pp. 42-54.
- Lehre, P. K. and Haddow, P. C. (2003). Developmental mappings and phenotypic complexity, *Congress on Evolutionary Computation(CEC2003)*, IEEE, pp. 62-68.
- Lindenmayer, A. (1971). Developmental Systems without Cellular Interactions, their Languages and Grammars, *Journal of Theoretical Biology*.
- McMullin, B. (2001). John von neumann and the evolutionary growth of complexity: Looking backward, looking forward..., *Artificial Life* 6(4): 347-361.
- Miller, J. and Downing, K. (2002). Evolution in materio: Looking beyond the silicon box, *2002 NASA/DOD Conference on Evolvable Hardware*, IEEE Computer Society Press, pp. 167- 176.
- Miller, J. F. (2004). Evolving a self-repairing, self-regulating, french flag organism, *Genetic and Evolutionary Computation (GECCO 2004)*, Lecture Notes in Computer Science, Springer, pp. 129-139.
- Quick, T., Dautenhahn, K., Nehaniv, C. L. and Roberts, G. (1999). On bots and bacteria: Ontology independent embodiment, *ECAL '99: Proceedings of the 5th European Conference on Advances in Artificial Life*, Springer-Verlag, London, UK, pp. 339-343.

- Robert, J. S. (2004). *Embryology, Epigenesis and Evolution: Taking Development Seriously*, Cambridge Studies in Philosophy and Biology, Cambridge University Press.
- Sekanina, L. and Bidlo, M. (2005). Evolutionary design of arbitrarily large sorting networks using development, *Genetic Programming and Evolvable Machines* 6(3): 319-347.
- Shipman, R., Shackleton, M., Ebner, M. and Watson, R. (2000). Neutral search spaces for artificial evolution: a lesson from life, *Artificial Life VII*, 2000. URL: [citeseer.ist.psu.edu/shipman00neutral.html](http://citeseer.ist.psu.edu/shipman00neutral.html)
- Sipper, M. (1997). *Evolution of Parallel Cellular Machines The Cellular Programming Approach*, Springer-Verlag.
- Sipper, M., Tempesti, G., Mange, D. and Sanchez, E. (1998). Von neumann's legacy: Special issue on self-replication, *Artificial Life* 4(3).
- Spector, L. and Stoffel, K. (1996). Ontogenetic programming, in J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (eds), *Genetic Programming 1996: Proceedings of the First Annual Conference*, MIT Press, Stanford University, CA, USA, pp. 394-399. URL: [citeseer.ist.psu.edu/spector96ontogenetic.html](http://citeseer.ist.psu.edu/spector96ontogenetic.html)
- Steiner, T., Trommler, J., Brenn, M., Jin, Y. and Sendhoff, B. (2009). Global shape with morphogen gradients and motile polarized cells, *Congress on Evolutionary Computation (CEC2009)*, IEEE, pp. 2225-2232.
- Taylor, T. (1999). On self-reproduction and evolvability, *European Conference on Artificial Life*, Springer-Verlag, Berlin, pp. 94-103.
- Toffoli, T. (2004). Nothing makes sense in computing except in the light of evolution, *Int. Journ. of Unconventional Computing* 1(1): 3-29.
- Tufte, G. (2006). Cellular development: A search for functionality, *Congress on Evolutionary Computation (CEC2006)*, IEEE, pp. 2669-2676.
- Tufte, G. (2008). Evolution, development and environment toward adaptation through phenotypic plasticity and exploitation of external information, in S. Bullock, J. Noble, R. Watson and M. A. Bedau (eds), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, MIT Press, Cambridge, MA, pp. 624-631.
- Tufte, G. (2009). The discrete dynamics of developmental systems, *Proc. of 2009 International Conference on Evolutionary Computation (CEC 2009)*, IEEE, pp. 2716-2723.
- Tufte, G. and Haddow, P. C. (2003). Identification of functionality during development on a virtual sblock fpga, *Congress on Evolutionary Computation (CEC2003)*, IEEE, pp. 731-738.
- Tufte, G. and Haddow, P. C. (2005). Towards development on a silicon-based cellular computation machine, *Natural Computation* 4(4): 387-416.
- Tufte, G. and Haddow, P. C. (2007a). Achieving environmental tolerance through the initiation and exploitation of external information, *Congress on Evolutionary Computation (CEC2007)*, IEEE.
- Tufte, G. and Haddow, P. C. (2007b). Extending artificial development: Exploiting environmental information for the achievement of phenotypic plasticity, *7th International Conference on Evolvable Systems (ICES07)*, Lecture Notes in Computer Science, Springer, pp. 297-308.
- Tufte, G. and Thomassen, J. (2006). Size matters: Scaling of organism and genomes for development of emergent structures, *CODESOAR at Genetic and Evolutionary Computation (GECCO 2006)*, ACM.

- von Neumann, J. (1956). Probabilistic logics and the synthesis of reliable organisms from unreliable components, in C. Shannon (ed.), *Automata Studies*, pp. 43-98.
- von Neumann, J. (1966). *Theory of Self-Reproducing Automata*, University of Illinois Press, Urbana, IL, USA, 1966.
- West-Eberhard, M. J. (2003). *Developmental Plasticity and Evolution*, Oxford University Press.
- Wolpert, L. (2002). *Principles of Development, Second edition*, Oxford University Press.

IntechOpen

IntechOpen



## **Evolutionary Computation**

Edited by Wellington Pinheiro dos Santos

ISBN 978-953-307-008-7

Hard cover, 572 pages

**Publisher** InTech

**Published online** 01, October, 2009

**Published in print edition** October, 2009

This book presents several recent advances on Evolutionary Computation, specially evolution-based optimization methods and hybrid algorithms for several applications, from optimization and learning to pattern recognition and bioinformatics. This book also presents new algorithms based on several analogies and metafores, where one of them is based on philosophy, specifically on the philosophy of praxis and dialectics. In this book it is also presented interesting applications on bioinformatics, specially the use of particle swarms to discover gene expression patterns in DNA microarrays. Therefore, this book features representative work on the field of evolutionary computation and applied sciences. The intended audience is graduate, undergraduate, researchers, and anyone who wishes to become familiar with the latest research work on this field.

### **How to reference**

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Gunnar Tufte (2009). From Evo to EvoDevo: Mapping and Adaptation in Artificial Development, Evolutionary Computation, Wellington Pinheiro dos Santos (Ed.), ISBN: 978-953-307-008-7, InTech, Available from: <http://www.intechopen.com/books/evolutionary-computation/from-evo-to-evodevo-mapping-and-adaptation-in-artificial-development>

**INTECH**  
open science | open minds

### **InTech Europe**

University Campus STeP Ri  
Slavka Krautzeka 83/A  
51000 Rijeka, Croatia  
Phone: +385 (51) 770 447  
Fax: +385 (51) 686 166  
[www.intechopen.com](http://www.intechopen.com)

### **InTech China**

Unit 405, Office Block, Hotel Equatorial Shanghai  
No.65, Yan An Road (West), Shanghai, 200040, China  
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元  
Phone: +86-21-62489820  
Fax: +86-21-62489821

© 2009 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen