An Architecture for Hybrid Creative Reasoning *unformatted version

Amílcar Cardoso, Ernesto Costa, Penousal Machado, Francisco C. Pereira and Paulo Gomes

Centro de Informática e Sistemas da Universidade de Coimbra (CISUC), Pinhal de Marrocos, 3030 Coimbra, Portugal {amilcar, ernesto, machado, camara, pgomes}@dei.uc.pt

Abstract

Creativity is one of the most remarkable characteristics of the human mind. It is thus natural that Artificial Intelligence's research groups have been working towards the study and proposal of adequate computational models to creativity. Artificial creative systems are potentially effective in a wide range of artistic, architectural and engineering domains where detailed problem specification is virtually impossible and, therefore, conventional problem solving is unlikely to produce useful solutions. Moreover their study may contribute to the overall understanding of the mechanisms behind human creativity.

In this text, we propose a *computational hybrid architecture for creative reasoning* aimed at empowering cross contributions from Case-Based Reasoning (CBR) and Evolutionary Computation (EC). The first will provide us a long-term memory, while the later will complement with its adaptive ability. The background knowledge provided by the memory mechanism can be exploited to solve problems inside the same domain or problems that imply inter-domain transfer of expertise.

The architecture is the result of a synthesis work motivated by the observation that the strong similarities between the computational mechanisms used in systems developed so far could be explored. Moreover, we also propose that those mechanisms may be supported by a common knowledge representation formalism, which appears to be adequate to a considerable range of domains. Furthermore, we consider that this architecture may be explored as a unifying model for the creative process, contributing to the deepening of the theoretical foundations of the area.

1 Introduction

Creativity is consensually viewed as one of the most remarkable characteristics of the human mind. Its study has been a challenge for many scientists and researchers, specially for those of areas such as Cognitive Science and Psychology. During the last years, a growing number of Artificial Intelligence groups have been working towards the study and proposal of adequate computational models to creativity. As a result of this endeavor, a new AI area is emerging, usually named Creative Reasoning (CR). Artificial creative systems are potentially effective in a wide range of artistic, architectural and engineering domains, where detailed problem specification is virtually impossible and, therefore, conventional problem solving is unlikely to produce useful solutions. Moreover their study may contribute to the overall understanding of the mechanisms behind human creativity.

Several explanation models, originated in the Psychology and Cognitive Science fields, have been proposed for the creative process, like the ones suggested by Dewey [1], Guilford [2], and Wallas [3]. Generally, they split the process into steps, which may be summarized (with more or less differences) into the following ones: problem formulation and knowledge assimilation; conscious or unconscious search for a solution, proposal of a solution, and verification of the proposed solution. These models may constitute an important source of inspiration to computational models of creativity. Also relevant are the studies of De Bono [4]around the concepts of lateral and vertical thinking and their relation with creativity.

We may also look for other sources of inspiration besides human creativity. When we look to nature we see that all living species struggle permanently for life. The Neo-Darwinist theory, revising Darwin first ideas at the light of modern genetics, give us a scientific framework that explains how life forms survive by adapting themselves to environmental changes. In the center of this process is a mechanism that selects the fittest individuals and recombines their genetic material. Putting together "good" parts of different individuals can give rise to a new and better one. This is clearly a way of producing innovative solutions [5]. But, is nature capable of producing creative solutions? This is a more difficult question to answer. In fact the biological adaptive processes have a limitation: they do not have a long-term memory *(although we can view multiploidy as a limited memory mechanism)*, and memory is an important part of the creative process. Thus, if we want to have a computational model of creativity, inspired in nature, we must introduce a memory element. This points out to hybrid solutions.

In the past some creative systems based either in Case Based Reasoning (CBR) or Evolutionary Computation (EC) techniques were proposed. CBR approaches can explore previous knowledge and have good explanatory capabilities. Nevertheless, they have difficulties exploring large search spaces. On the other hand, EC approaches are efficient in dealing with complex search spaces and explore parallelism in a natural way. However, they lack long-term memory and

the incorporation of problem-specific knowledge and the interpretation of results is problematic.

In this text, we propose a *computational hybrid architecture for creative reasoning* aimed at empowering cross contributions from CBR and EC. The first will provide us a long-term memory, while the later will complement with its adaptative ability. The background knowledge provided by the memory mechanism can be exploited to solve problems inside the same domain or problems that imply inter-domain transfer of expertise.

The architecture is the result of a synthesis work motivated by the observation that the strong similarities between the computational mechanisms used in systems developed so far could be explored. Moreover, we also propose that those mechanisms may be supported by a common knowledge representation formalism which appears to be adequate to a considerable range of domains. Furthermore, we consider that this architecture may be explored as a unifying model for the creative process, contributing to the deepening of the theoretical foundations of the area.

The remaining of this document is organized as follows. In Section 2 we give a synthetical presentation of the background concepts of CBR and EC. Those who are familiar with these two techniques may safely skip this section. In Section 3 we present a state of the art on creative systems, as well as on hybrid systems which resort on CBR and EC. In Section 4 we give an overview of some systems developed by our team using CBR and EC approaches. These systems motivate the presentation, in Section 5, of a common representation formalism and a unifying hybrid architecture. Section 6 is devoted to the presentation of an example which illustrates the application of the proposed architecture to creative reasoning. In Section 7 we present some improvements to the architecture which we are currently exploring. In Section 8 we draw some conclusions.

2 Background

This Section is intended to the readers who are not familiarized with CBR and/or EC. We will briefly present the basic concepts behind CBR and EC, focusing on the main steps of their basic cycles and on the main issues around representation.

2.1 Case Based Reasoning

Case-based reasoning (CBR) uses past experience to solve new problems [6][7]. In a CBR system, experience is stored by way of cases, which form a case library. *Cases* are episodic chunks of knowledge that are used in the resolution of a new situation. The problem solving process in CBR is based on the premise that identical problems have identical solutions. When a CBR system has a new problem to solve, it *retrieves* cases with similar problem descriptions. If the selected cases are different from the target problem, they must be modified in

order to fit it. This modification of a retrieved case is called *adaptation*. Finally, the new case created by adaptation is *evaluated* and may be stored in the case base. The reasoning steps are illustrated in Figure 1, where the iterative nature of the process is outlined.

Case representation is crucial in CBR, since the case representation delimits the CBR steps. For example, cases can only be efficiently retrieved if the important features of the problem description are represented. Cases can be represented by attribute/value pairs, hierarchies of concepts, objects (object oriented style), textual description, causal models, or combination of them.

In the case retrieval phase, the CBR systems must decide which cases are the best candidates to solve the target problem. There are several ways of doing it, but the most used ones are: using an indexing scheme, where there is a set of indexes associated to each case that are used to retrieve the case; using a metric function like K-nearest neighbor to assess the similarity of the target problem and the case problem's description. Both have their advantages and disadvantages.

Case adaptation is one of the most unexplored phases of CBR. This may be explained by the complex and domain dependent nature of the adaptation process. In general terms, during adaptation, the CBR system must first identify the differences between the target problem and the cases' problem's description, and then apply modification operators to change the case solution. These operators can be production rules, formulas, heuristics, or specific procedures.

The evaluation of the solutions generated by adaptation is important to provide feedback to the system. In this phase, the new solution is evaluated and, accordingly to the evaluation's result, it is stored in the case library and presented to the user, or it is rejected, which leads the system back to the adaptation phase. The evaluation can be done automatically or by a human.

The case learning step introduces flexibility and also the capability to adapt to new situations. In this phase, the feedback gathered from the evaluation phase can be stored, in order to be used again in similar situations. Several things can be learned, and the new case created to solve the target problem is the more basic and obvious of them. Other things can be learned, for instance, new adaptation operators, cases where the solution failed, and so on.

In summary, CBR provides a fast reasoning mechanism, specially suited for domains where there is no causal model, can be used for evaluation where no algorithmic methods exist, and can avoid previous failure situations. Cases are also useful for interpretation of ill-defined concepts. One of the main drawbacks of CBR is that it is difficult to make the right index selection. Also, the solution space is focused around the case points, thus constraining the possible solutions.

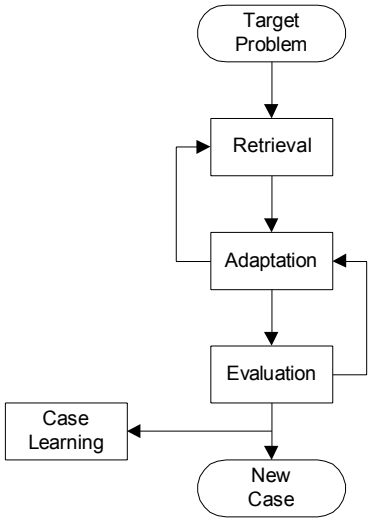

Figure 1. The CBR cycle.

2.2 Evolutionary Computation

In recent years, a new paradigm for problem solving, called Evolutionary Computation (EC), has emerged. EC can be viewed as a set of stochastic search procedures inspired by the biological principles of natural selection and genetics [8]. Historically, these sets can be divided into four families, namely, Evolution Strategies [9][10], Evolutionary Programming [11], Genetic Algorithms [12]and Genetic Programming [13]. In spite of their differences they are all instances of the following general algorithm:

```
Procedure EC
  t=0;
  Intialize P(t);
  Evaluate P(t);
  While stoping_criterium_false  do
        t = t+1;
        P'(t) = select_from P(t-1);
        P''(t) = use_op_modification P'(t);
        Evaluate P''(t);
        P(t) =  merge P''(t) , P(t-1)
End_do
```

We start with a set of candidate solutions, called a *population*, usually defined randomly. Each element of that initial population, called an *individual*, is then evaluated using a *fitness function* that gives a measure of the quality of that element. Each individual is in fact an aggregate of smaller elements or units, which are called *genes*. Each gene can have different values or *alleles*. The algorithm enters then a cycle in order to generate a new population. We start by probabilistically *selecting* the fittest individuals. Then they undergo a modification process, using genetic inspired operators like *crossover* or *mutation* that will eventually alter the alleles of some genes. Finally, the old and new populations are combined and the result becomes the next generation that will in turn be evaluated. The cycle stops when a certain condition is achieved (for instance, a pre-defined number of generations). The selection mechanism introduces the possibility of *exploiting* promising parts of the search space. The crossover operator works by exchanging genetic material between two individuals, while the mutation operator modifies the alleles of some individuals. That way they promote the *exploration* of different areas of the search space. A good balance between exploitation and exploration is essential for the success of the EC algorithm. Another important aspect is the question about what is manipulated. The algorithm just described work, generally, with a low-level representation of each individual called its *genotype*. Nevertheless, in complex problems, the fitness function acts upon a high-level representation of an individual, its *phenotype*. It is thus necessary to have *decoders* from genotypes to phenotypes.

It is outside the scope of this text to refer all the variants of an EC algorithm. The interested reader can have a deeper idea about the many EC algorithms that were proposed and their practical applications in Back et al. [8].

The success of EC algorithms is linked to their ability to solve difficult problems. Problems where the search space is large, multi-modal and when domain knowledge is scarce and/or difficult to obtain. When this is the case EC proves to be more efficient than traditional algorithms.

3 Creative Systems and Hybrid Systems: State of the Art

There are multiple approaches to Creative Reasoning, and a diversity of applications has been explored by researchers of the area. In this Section we will first give an overview of the present state of the art in computational creativity, focusing on the paradigms used, the domains in which they were applied and some successful experiences.

Afterwards, we will also present an overview of works that combine CBR and EC in a hybrid way.

3.1 Creative Systems

Several AI techniques have already been applied to tasks that are usually considered to require creativity, such as Design, Music Composition, Image Generation, Scientific Discovery, Architecture, and others. One of the paradigms that has been used with this purpose is Case-Based Reasoning. It has been thoroughly applied, for example, in Creative Design, which is generally defined as a cognitive task where some knowledge for the mapping process between the problem and solution spaces are missing [14][15]. The solutions generated in creative design define new classes of artifacts, thus expanding the space of known designs. In this exploration process, designers often use old solutions to solve new problems – which suggests the suitability of CBR to this problem. In creative design, the old solutions are changed in novel ways or used in novel situations. Several researchers have used the CBR paradigm as a framework for building systems to tackle this task. Some very interesting CBR-based works in this area are those of: Kolodner and Wills [16], which apply case indexing accordingly to various perspectives, in order to allow the search of the case memory for remindings that might be represented in a different way in the light of the current problem; Simina and Kolodner [17], which propose a computational model that accounts for opportunistic behavior, which is considered to be characteristic of creative behavior; Sycara and Navinchandra [18][19], which use a thematic abstraction hierarchy of influences as a retrieval method. In this framework, case organization provides the main mechanism for cross-contextual reminding, which is very important in creative design. They also stress the importance of composition of multiple cases and case parts.

Still in the CBR area, we can find works in Music Composition, such as Pereira et al [20], presented later in this chapter, which applies musical analysis structures to build new musical pieces. Each of these structures is considered a decomposable case. The work of Arcos et al [21] on expressive performance based on CBR is also an interesting one. Its cases consist on information extracted from spectral analysis of performances and the scores themselves. From this set of cases, the system infers a set of possible expressive transformations for a given new phrase.

Case Based Reasoning has also been applied in Architecture [22], and we believe it is a promising paradigm to other kinds of creativity demanding tasks.

As nature by itself is known to be creative, it is not surprising that Evolutionary Computation paradigms have also been used as a mean to implement computational creativity. The difficulty of creating an evaluation function in domains such as image or music generation has led, frequently, to the use of Interactive Evolution (IE). In these systems the user evaluates the individuals, thus guiding evolution. IE has great potential as the countless already developed applications show. In the field of music, it has been applied in the evolution of rhythmic patterns and melodies [23]; in jazz improvisations [24]; in composition systems [25]. The works of Dawkins [26], which uses IE to evolve artificial

creatures based on the aesthetic preferences of the user, Sims [27], Todd [28], Rooke [29] and Machado et al [30], which resort to IE to evolve images, and Baker [31], where IE is used to evolve human faces, are some examples of the application of IE in the field of image generation. IE has also been successfully applied in the fields of design [32][33] and animation [27] [34] [35].

As far as we know, in the field of image generation there has been only one attempt to automate fitness assignment, the work of Baluja et al [36]. However, the results produced by this system, which uses neural networks to evaluate images, were disappointing.

There have been several attempts to automate fitness assignment in the musical field. Some examples of this type of work are: Horner et al [37], which use GAs to evolve thematic connections between melodies; McIntyre [38] which uses GAs to generate musical harmonization; Spector [39][40] which resort to Genetic Programming to evolve programs that generate jazz melodies from an input jazz melody; Papadopoulos et al [41] use GAs to evolve jazz melodies based on a progression of chords. However, and in spite of the numerous applications, Wiggins et al [42], which has studied the performance of this type of systems, defends that these approaches are not ideal for the simulation of human musical thought.

CBR and EC are not the only approaches to the resolution of tasks demanding creativity. Other techniques currently used are Knowledge Based Systems, as in Harold Cohen's Aaron [43] and Ed Burton's ROSE [44], in Visual Arts, and the work of Pachet and Roy [45] in Music; Mathematical Models (e.g., the Markov Chains of Cambouropoulos [46] to assist on music composition); Grammars (e.g., the works of Cope [47] in Music, and of Stiny [48] in architecture).

3.2 Hybrid Systems Based on CBR and EC

There are a few systems that try to combine CBR and EC. Most of the work was done by Sushil Louis and his co-workers [49][50][51][52], which built the CIGAR system, and by Ramsey and Grefensttete [53]. The main idea of CIGAR was to use a base of cases to initialize the population, and then let the EC algorithm do its typical adaptation work. The first cases are former solutions of old problems obtained by running the GA alone (see Figure 2).

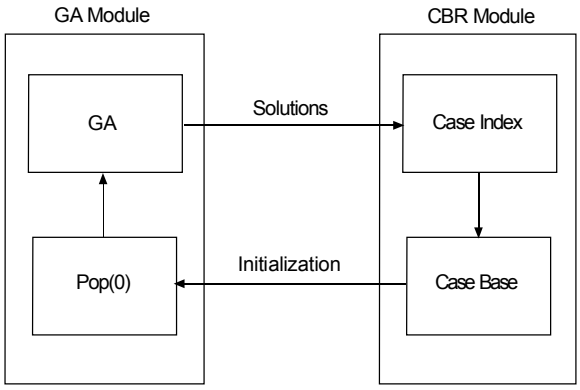

Figure 2. The CIGAR system

Louis used his system to solve a similar problem or a set of similar problems. He also studied how many cases to inject into the population, coming from the base of cases, and which ones should be chosen. Finally, he also studied the possibility of injecting cases not only into the initial population but also in intermediates ones. Some of the problem domains used to test these ideas were combinational circuit design, open shop scheduling and re-scheduling and function optimization. The results presented showed that with a judicious choice the combination CBR and EC gave better results. Other authors had also used the idea of injecting new, random generated, individuals into a population at certain times. For instance, Eshelman [54], in his algorithm named CHC, replaces the mutation operator, used in the standard genetic algorithm to insure population diversity, by a restarting process applied to the population. Once again, the definition of when to do it, how many new individuals should be generated randomly and which ones should be kept from the previous generation is an important issue and is most relevant when we deal with creative systems.

As far as we know, nobody has tried to use a hybrid system to produce creative solutions. Nevertheless, Goldberg [5] states that there are forms of adaptation which go beyond innovation. He suggests two ways about how this can be achieved: *remapping the primitives*, that is, changing the representation; and performing *metaphorical transfer*, that is, the transfer of a solution from a known problem domain to another.

4 Creative Reasoning

During the last years, the authors have centered most of their research work on the development of CBR-based and EC-based approaches to Creative Reasoning. In this Section we will present with some detail the main applications developed so

far. Some common characteristics of the works, particularly in which concerns to the adoption of a tree-like knowledge representation, will constitute the basis for the hybrid architecture that we will propose in subsequent Sections.

4.1 Creative Reasoning with CBR

The capability of reminding previous experiences to draw analogies with the current situation has made case-based reasoning a good framework to support creative design [16][55]. In the next subsections we present several approaches to creative reasoning with CBR which we have developed so far. There are three main systems: IM-RECIDE, a generic creative reasoning shell; CREATOR and CREATOR II, creative design systems in the area of digital circuit design; and SICOM/INSPIRER, a system that uses CBR for creating new musical pieces.

4.1.1 IM-RECIDE

IM-RECIDE [55] is a generic creative reasoning shell that uses CBR as the main reasoning mechanism. Its reasoning cycle comprises several steps: problem definition, space initialization, problem solving, verification and evaluation. The first step comprises problem specification, where the user states a new problem in terms of goals and constraints. In the initialization phase the system clusters cases in different sets, each one called a reasoning space. These spaces allow a gradual exploration of the case library enabling the system to generate new designs. In the problem-solving phase, reasoning operators are applied to old solutions for generation of new ones. If no solution is generated within a specific space, the system switches to the next space from the list that was created during the initialization phase. When a solution cannot be generated, and there are no more spaces to search for, the user is asked to give a solution for the problem. After a solution has been generated it has to be verified and evaluated. In a first step, it is internally validated by failure cases (verification). Failure cases represent constraints in the generation of new solutions. If a failure case is triggered by the new solution, then the solution is rejected. If a solution passes the internal validation, the user is asked to accept or reject the new case. If she/he rejects the new solution, then the user is asked to explain this rejection in terms of failure cases. After this, the process returns to the problem-solving step. This last phase is called evaluation because the user has to make a decision about the originality and validity of the case.

Case representation is very important for a CBR system, because it determines the capabilities of the system. Within IM-RECIDE, a case is represented by a triple <P, S, R> with P and S, respectively, a set of facts representing the problem and solution descriptions, and R a set of rules representing a causal justification (for a more detailed description see [56]). A fact is composed by a function name (functor), and *n* arguments, with *n* equal or greater than zero. The justification is a

causal tree, linking problem facts to solution facts through rules. The case library also comprises failure cases representing design constraints.

For case retrieval, we consider four spaces of knowledge: Space I, Space II, Space III and Space IV. Each space comprises the cases possessing a set of common properties concerning the target problem. Each space forms a cluster in the case library. The definition of each space is done in terms of the characteristics that the cases within this space share with the target problem. Creativity can be seen as the result of reasoning on spaces of cases increasingly further away from the target problem. As the system goes from space I to space IV, it drifts away from the problem, trying to find nontrivial solutions. The retrieval process starts with the cases in Space I, going from space to space, until it reaches Space IV.

Space I comprises the cases for which all functor/arguments pairs belonging to the problem description, match the new problem. Space I is considered the space normally associated with the current problem. Most of the current CBR systems use cases from this space. Cases for which all functors describing the problem component match the new problem, belong to Space II. This space is related with problems similar to the target problem. This space is often called as the innovation space, where parametric adaptations are usually done, sometimes resulting in using a novel value for a well-known functor. Space III contains the cases which have explanation rules with all functor/arguments pairs matching the target problem. Space III is defined using causal knowledge, which makes similarities between cases and the target problem more abstract, but also more important. This space is usually associated with creative solutions, but also with bizarre ones.

Space IV gathers all cases that contain at least one explanation rule with at least one functor/arguments pair matching the target problem. Space IV is like a speculation space where cases have remote similarities with the target problem, because constraints were relaxed. This relaxation allows the system to explore cases considered distant from the target problem. Once again, the knowledge used to do this is the causal knowledge comprised in the explanations.

We now describe the adaptation mechanisms used in IM-RECIDE, called adaptation operators. These operators modify cases in the current space in order to solve the new problem. The cases that are used for generation of the new case are called the source cases. The selection of the cases for adaptation is performed through a metric [57]. These cases are selected from the set of cases in the current working space.

Each space possesses a set of predefined adaptation operators. These operators are used accordingly to the type of cases that the space comprises. The operators in Space IV are more powerful than the operators in Space I. This is an obvious situation, because cases in Space IV have less similarities with the problem. In order to reach a valid solution, more difficult adaptation operations must be done. Associated with the operator capabilities is the complexity of the computational process originated by each operator, the cognitive risks involved, and the probability of generating a more creative solution.

A solution of one case belonging to Space I does not need to be modified in order to become a solution to the target problem. Therefore, a metric is used for selection of the best case, and its solution is the one for the target problem. Cases in Space II have only some different values regarding the problem description. In order to meet the target problem requirements, it is necessary to modify the old case solution. This solution is derived from the old case by the propagation of the differences in the old problem to the old solution. The causal knowledge is used to guide the propagation process. The selected case is chosen using a metric function, which measures the similarity of cases against the target problem. Space III generates new solutions by splitting and merging of case pieces. IM-RECIDE starts selecting the most similar case from the set of episodes comprising Space III. Then, it splits the case into pieces selecting the pieces that match part of the target problem. These pieces are then merged to form a new case. If the problem description in the new case has some missing parts in regard to the target problem, other cases are selected to contribute with case pieces to complete it. Pieces from these cases that are relevant for the new case are merged with it. In space IV, there are several adaptation operators, and they can be applied in sequence. Splitting and merging is one of the operators within this set. The other operators are elaboration, reformulation, substitution, and generalization. Elaboration comprises relaxing and/or strengthening of constraints described in a case problem, in order to match the target problem description. The case solution is suggested as the new solution. Reformulation involves changing the new problem description according to constraints imposed by failure cases. Substitution comprises replacing a functor/arguments pair in the past case in order to make it similar to the new problem. The solution that results from this substitution is given as the one for the new problem. Generalization involves considering values initially not considered in the problem description of a past case, and assuming the case solution remains unchanged.

After a solution is created, it is verified by the failure cases. If the solution matches one failure case, then it is rejected. Only solutions that the system assumes to be correct, by its current knowledge, are shown to the user.

4.1.2 CREATOR & CREATOR II

CREATOR is a case-based creative design system in the domain of digital circuit design. CREATOR comprises four different modules: reasoning, knowledge base, evaluation and meta-control. The system was developed having the SBF models [58][59] as the case representation formalism. The reasoning module is responsible for problem elaboration, retrieval of relevant cases and adaptation of cases. The knowledge base comprises the case base, general domain knowledge and memory structure. Hierarchies of functions, structures and substances are used as general domain knowledge. This is important for several purposes, one of which is the construction of the memory structure. The memory structure has two main goals: to index cases and to allow space exploration. The evaluation module

verifies and validates the generated solutions, while the meta-control module controls and co-ordinates all the other modules. The evaluation module and adaptation processes are being implemented in CREATOR II, which is the successor of CREATOR.

Within our framework, design cases are represented in the form of SBF models. These models are based on the component-substance ontology developed by Bylander and Chandrasekaran [60]. A case comprises three parts: (1) problem specification; (2) explanation; (3) design solution. The explanation is in the form of a causal chain, representing the design behavior. The case solution describes the design structures that accomplish the functionalities described in the target problem. So the problem specifications are related to the design function, the explanation to the design behavior, and the solution to the design structure.

The problem specification comprises a set of high level functionalities (HLFs) and a set of functional specifications (FSs) which must be held by the design. HLFs are abstract functionalities, used to help the user in specifying the design problem. While a HLF is a function that can be decomposed into several subfunctions, an FS is undecomposable. An FS is defined in detail in accordance to input and output substances. A design problem is represented by a tree of functionalities, where leaves are FSs and the high levels in the tree represent HLFs. Each leaf in the tree represents an FS in a schema comprising the initial behavior state, the final behavior state, behavioral constraints, external stimulus to the design, and structural constraints.

The design solution is in the form of a hierarchy of device structures. Each structure can be viewed as a set of device structures where substances can flow through. The structure schema comprises information such as: structure class, sub-structures, super structures, relations, properties and functions. Figure 3 shows the high level representation of an Arithmetic and Logic Unit (ALU), where each node in the tree represents a structure. Each of these structures has a corresponding structure schema.

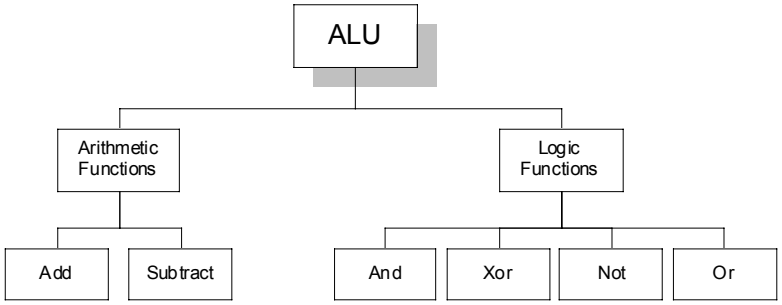

Figure 3 – A representation for an Arithmetic and Logic Unit (ALU) in CREATOR.

A case explanation describes the causal behavior of the design in terms of directed graphs (DGs). The nodes of a DG represent behavioral states and the edges represent state transitions. One or more substance schemas can compose a behavioral state. A substance schema characterizes the properties and the property values of a substance. A state transition represents the conditions under which the transition between behavioral states occurs.

The memory structure that supports case retrieval comprises two substructures: (1) a hierarchy of HLFs and FSs, and (2) a graph, whose nodes represent cases, and whose links describe functional differences between cases. The hierarchy of functions is used as an index structure for selection of the starting cases. The starting cases have at least one FSs in common with the target problem. These cases are then used as starting points for exploration of the case graph. The leaves of the hierarchy are nodes that describe a HLF instance. These nodes are called List of Functions (LF) and they comprise a set of HLFs and/or FSs. These nodes are extracted from cases and index the case they belong, thus connecting the hierarchy of functions to the graph of cases. Two cases can be connected by difference links, which represent the differences between the problem description of the cases linked. A difference link is created only when the cases it connects have at least one FS in common. A difference link connecting two cases comprises three parts:

- the set of FSs that belong to the first case but don't belong to the second one;
- the set of FSs that belong to the second case but don't belong to the first one;
- the set of FSs common to both cases.

The reminding of useful experiences in a case-based system is a critical issue. The accuracy of case retrieval in case-based creative reasoning is important, but even more important than that is the capability to explore several solutions. Within our framework, accuracy is achieved by the use of functional indexes, and space exploration takes place through the use of difference links in the graph of cases. The FSs defined in the target problem are used as probes to retrieve the set of starting cases. Then, a best starting case is selected as a starting point in the search space. The search space is represented by the graph of cases. Exploration is performed using the difference links necessary to go from one case to another.

An important feature of the exploration algorithm is the selection of cases accordingly to the adaptation strategy that will be used for generation of the new solution. The retrieval algorithm explores the case graph, searching for cases with features suitable for the adaptation method that will be applied. This makes retrieval an adaptation-guided process as defined by Smyth and Keane [61], although there are some differences to their process. Two of the adaptation strategies considered within our framework are thematic abstraction and composition. Thematic abstraction is an adaptation strategy that generates a new solution from a single case. It consists on the transfer of knowledge from a case to the target problem, in order to create a new design. The composition strategy deals with one or multiple cases. It splits and/or merges case pieces generating new solutions - it is a multi-case strategy.

4.1.3 SICOM/INSPIRER

Following some of the features from IM-RECIDE, we designed INSPIRER [62] for creative problem solving in domains in which knowledge can be represented by hierarchically structured cases. This framework was deeply tested in the Music domain, and its implementation, SICOM [20] generated some pieces of music from a small case-base with three compositions from a XVIIth century Portuguese baroque composer, Carlos Seixas. In SICOM, each case consists on a highly detailed analysis with several layers of abstraction, in which a piece of music is progressively subdivided according to thematic groupings (following harmonic, melodic and rhythmic principles from Music Theory). The result is a strictly hierarchical structure complemented by causal links that establish non-hierarchical relations (see Figure 4). This kind of structured organization is common within music and some other artistic domains like architecture, literature and visual arts.

Generally, each case in SICOM is a complete piece of music, represented by a set of interrelated nodes (case-pieces) extracted from music analysis.

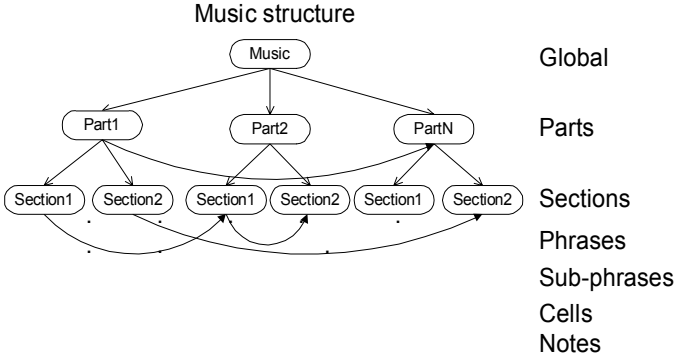

Figure 4 – Representation of a musical piece in SICOM

The generation of a composition consists on the creation of a new structure applying case-pieces from the case-base. This generation is taken in a top-to-bottom and left-to-right sequence, as illustrated in Figure 5 (i.e., it starts by choosing the more abstract and temporally preceding nodes).

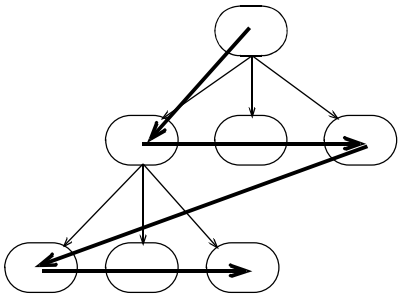

Figure 5 – Sequence of generation of a musical piece in SICOM

One key point of systems such as INSPIRER/SICOM is that of the similarity metric. It is according to this measurement that it selects which nodes to apply or to avoid in a new composition. Unfortunately (or fortunately), there are no formulas to evaluate a good solution in the Musical domain. Moreover, we don't have a clear definition of what is or is not a good choice for any specific situation during the composition of a new music. We do have the clear notion that context and structure can be determinant to the successfulness of a choice and we have a set of composition rules to help on evaluation and adaptation. The similarity metric that we apply in SICOM [63] takes two main aspects of a node into account: its intrinsic properties (e.g., its internal attributes, like the melodic contour it defines) and its context (e.g., the causal links connected to it, the attributes of its parents, its position in the whole structure). After applying this similarity metric, SICOM can use one of several orderings to select the node (e.g., choosing the most similar; the least similar; avoid the first 20% of the candidate list, etc.).

The output compositions of the project SICOM/INSPIRER, although being not comparable to those of a professional composer, are nevertheless very interesting, specially taking into account that it has a library with only three cases. A particular example of its performance is the introductory part of its compositions. Each of the three Carlos Seixas' pieces had a similar introductory part, but SICOM was able to generate several new structures with different and correct (according to the style) solutions.

4.2 Creative Reasoning with Genetic Programming

As stated before EC has great potential for creative reasoning. In this section we will make a brief yet comprehensive description of NEvAr (Neuro Evolutionary Art).

NEvAr is an evolutionary art tool, inspired in the works of Sims [27] and Dawkins [26], that allows the evolution of populations of images from an initial one using Iterative Evolution. The presentation of the underlying model will follow. We will finish this section by showing some experimental results and drawing some overall remarks.

4.2.1 Representation

Like in most GP applications, in NEvAr the individuals are represented by trees. Thus, the genotype of an individual is a symbolic expression, which can be represented by a tree.

The trees are constructed from a lexicon of functions and terminals. The internal nodes are functions and the leaves terminals. In NevAr, we use a function set composed mainly by simple functions such as arithmetic, trigonometric and

logic operations. The terminal set is composed by the variables *x* and *y*, and by constants which can be scalar values or 3d-vectors[1].

The interpretation of a genotype results on a phenotype, which in NEvAr is an image. The easiest way to explain how this is achieved is through an example. Lets consider the function *f(x,y)=(x+y)/2* with *x, y* ∈ [-1,1]. This function can be represented by the tree presented in Figure 6.

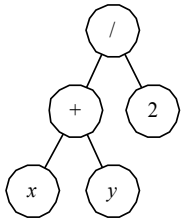

Figure 6. Representation of *f(x,y)=(x+y)/2* in the form of a tree.

How can we visualize this function? One hypothesis is making a tridimensional graphic such as the one presented in Figure 4a. Another option would be to view this graphic from the top and indicate the value of the function through a color. The value -1 could correspond to 0% luminance (black) and 1 to 100% luminance (white), the values in between −1 and 1 would be represented by intermediate luminance values. This approach yields an image similar to the one presented in Figure 7b. In Figure 8 we present some examples expressions and the images generated by them.

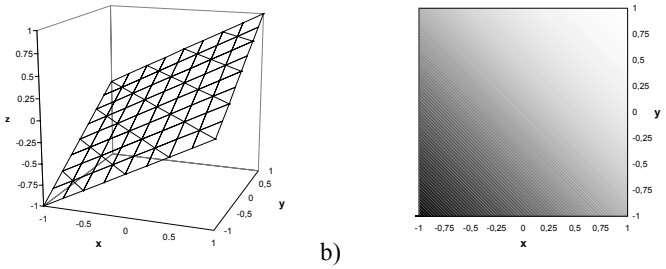

Figure 7. a) Tridimensional graphic of *f(x,y)=(x+y)/2*. b) A color graphic of the same function.

[1] The three dimensional vectors are necessary to produce color images: each dimension of the vector corresponds to a color channel (Red, Green and Blue).

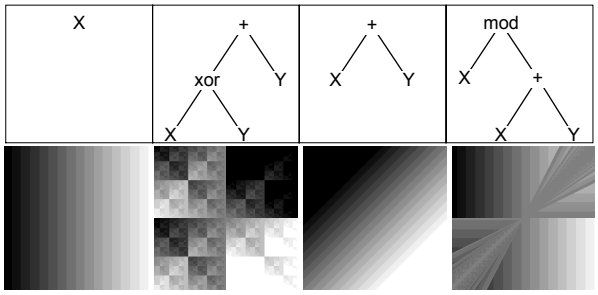

Figure 8. Some simple functions and the corresponding images.

4.2.2 Genetic Operators

In NEvAr we use two kinds of genetic operators: recombination and mutation. As recombination operator we use the "standard" GP crossover operator [13] which exchanges sub-trees between individuals (see Figure 9): given two individuals A and B, we select randomly two crossover points (one node of A and one of B) PA and PB; these nodes are the roots of two sub-trees; then we swap the sub-trees, thus obtaining two new individuals A' and B'.

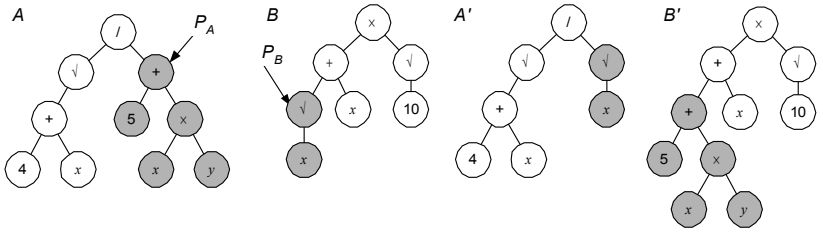

Figure 9. Example of crossover between the individuals A and B.

We use five mutation operators that are similar to the ones used in [27]:

- Sub-tree swap – randomly select two mutation points and exchange the corresponding sub-trees.
- Sub-tree replacement – randomly select a mutation point and replace de corresponding sub-tree by a randomly created one.
- Node insertion – randomly select an insertion point for a new, randomly chosen, node. If necessary, create the required arguments randomly.
- Node deletion – the dual of node insertion.
- Node mutation – randomly select a node and change its value.

These operators induce changes at the phenotype level. In Figure 10, we show examples of the application of the crossover operator. As can be seen, the crossover between two images can produce interesting and unexpected results.

Additionally, there are cases in which the images seem to be incompatible, i.e. images that, when combined, result in "bad" images.

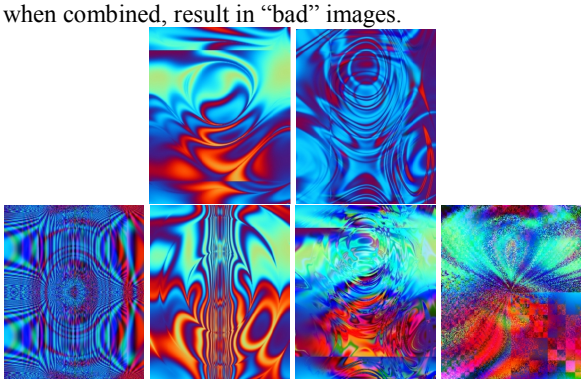

Figure 10. In the top row, the progenitor images. In the bottom row, some examples of the results generate by the crossover among them (color images available at http://www.dei.uc.pt/~amilcar/softcompbook/fig10.htm)

In Figure 11 we give examples of images generated through mutation. Once again, the results of this operation can give quite unexpected results.

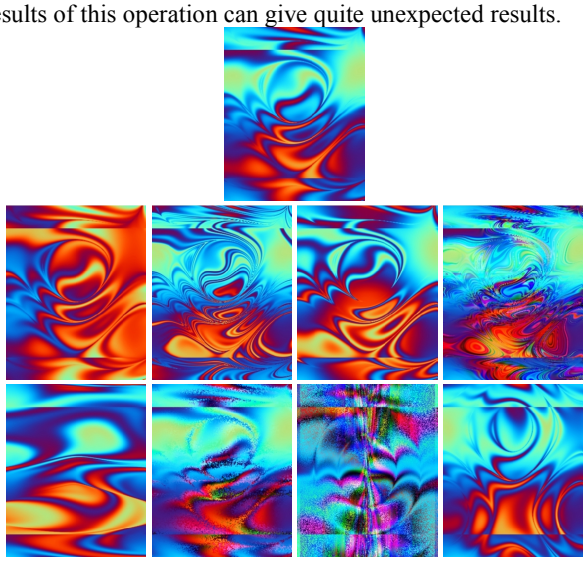

Figure 11. In the top row, the original image. In the bottom rows, several mutations of the original image. The images *a* and *b* where generated through node mutation, *c* and *d* through node insertion, *e* through node deletion, *f* and *g* through sub-tree swap, h by sub-tree replacement (color images available at http://www.dei.uc.pt/~amilcar/softcompbook/fig11.htm)

4.2.3 Model

In NevAr, the assignment of fitness is made by the user and, as such, she/he has a key role. The interaction of human and computer poses several problems. For instance, we cannot use populations with a large number of individuals, or make big runs. It would be unfeasible to expect that a human would be willing to evaluate one hundred individuals per population over a period of one thousand or more populations. Thus, to produce appealing images, NevAR must do it in few evolutionary steps and in a low number of individual's evaluations.
The fact that NEvAr is an interactive tool also has the advantage that a skilled user can guide the evolutionary process in an extremely efficient way. She/he can predict which images are compatible, detect when the evolutionary process is stuck in a local optimum, etc. In other words, the user can change its evaluation criteria according to the specific context in which the evaluation is taking place. In the design of NevAr, we took under consideration these aspects.

Figure 12 shows the model of NEvAr. In the following we will call *experiment* to the set of all populations, from the initial to the last, of a particular GP run.

NEvAr implements a parallel evolutionary algorithm, in the sense that we can have several different and independent experiments running at the same time. It is also asynchronous, meaning that we can have an experiment that is in population 0, for example, and another one that is in population 100. Additionally we can transfer individuals between experiments (migration) and can also transfer individuals from one population to another.

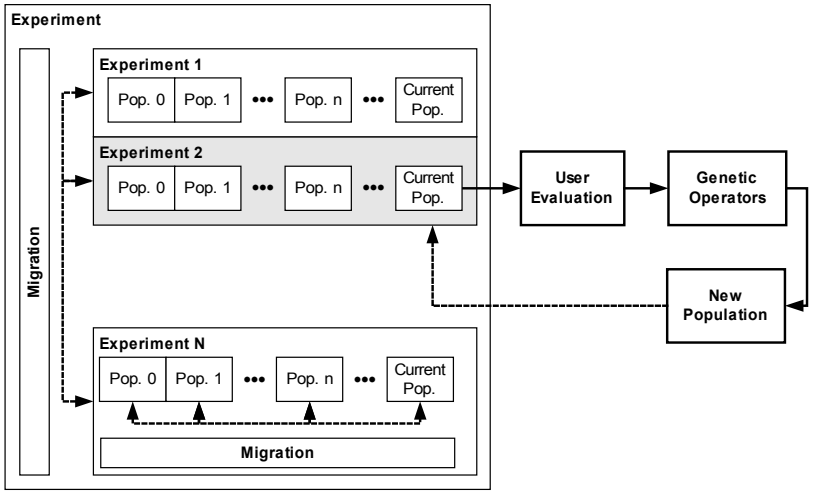

Figure 12. The evolutionary model of NEvAr. The active experiment is depicted in gray.

We will illustrate the utilization of this model through an example. Let's suppose that the user creates, to start, two different experiments, *a* and *b*, the initial

population of *a* is randomly generated and has size N, and the initial population of *b* has size 0. The user focuses his efforts in experiment *a* and evaluates the individuals of successive populations generated by NEvAr. When the user finds images that she/he likes, she/he adds these images to the current population (in this case the population 0) of experiment *b*. If at a given point the user feels that the evolutionary process would benefit if the next population was generated by the combination of the individuals of the current population with individuals previously transferred to population *b*, she/he adds those individuals to the current population and the evolutionary process continues.

If the user, at a certain point, chooses to focus on experiment *b*, NEvAr will generate a new population from the current one (population 0), which is composed, exclusively, by individuals transferred from *a*. Thus, the initial population of experiment b is not random, but exclusively composed by fit individuals that were originally generated in other experiments. In fact, experiment *b* can be seen as a database of images, which may be used to initialize future experiments. We may generalize this approach by organizing a gallery of images.

As stated before, NEvAr also allows the migration within experiments. This feature is important due to the limited size of each population, since it allows the revival of images from previous populations. It is also possible to go back to a previous population and change the evaluation of the individuals, which allows the exploration of different evolutionary paths.

In Figure 13 we give some examples of images generated by NEvAr). These images were presented on the exhibit "*Art and Aesthetics of Artificial Life*", Nicholas Gesseler (curator), which took place at the Center for the Digital Arts of the UCLA.

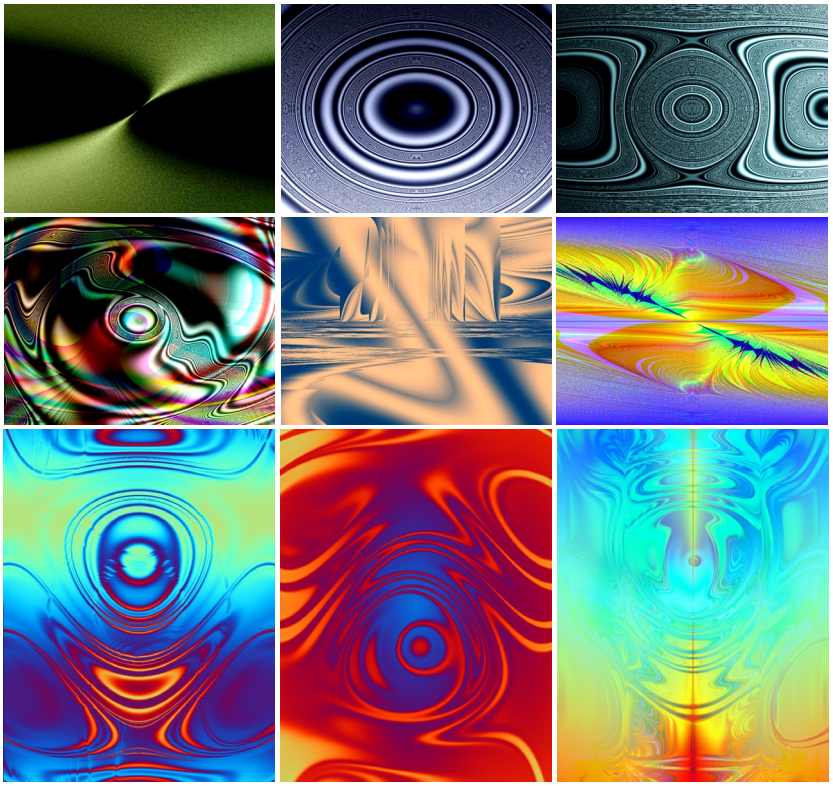

Figure 13. Some of the images presented the "*Art and Aesthetics of Artificial Life*" exhibit (color images available at http://www.dei.uc.pt/~amilcar/softcompbook/fig13.htm)

The results achieved with NEvAr clearly show the power and potential of interactive evolution techniques, and the advantages of the reuse of past solutions (images). The galleries of images play an increasingly important role in the generation of new images. The chosen representation allows the recombination of images in interesting ways yielding unexpected, yet fit, images.

The results also show that it isn't necessary to resort to complex primitives, such as fractals or other complex functions, to generate complex and interesting images. What is necessary is a set of simple functions that can be combined in complex ways.

5 The Unifying Architecture

The analysis of our previous work, described in the last section, took us to a point where we realized that it was possible to settle a common framework

(representation language) for the objects manipulated by the algorithms in different domains. The common representation language which emerged is a set of hierarchical structured objects (HSOs). It can be defined more precisely by means of a grammar that we partially show below using the traditional BNF formalism:

<structure> ::= <node> | <node>({<structure>}$^{+}$)
<node> ::= <name> | <name>({<attribute>}$^{+}$)
<attribute> ::= <type-attrib>(<name>)
<type-attrib> ::= link-in(<value>) | link-out(<value>)

It is possible to use the same common language to represent objects from different domains. The objects represented in Figures 3, 4 and 6 are, in fact, instantiations of the above described HSOs, and only the non-terminals <name> and <value> depend on the domain. The main advantages of the adoption of such a representation are that it permits a full integration of EC and CBR mechanisms in the same architecture, and also the inclusion of objects from different domains in the same knowledge base. This last characteristic makes the implementation of methaforical transfer mechanisms possible, which will be further explored in Section 7.

The proposed architecture (Figure 14) builds upon this representation and models the creative process as an iterative sequence of steps, which resembles some models proposed by psychologists like by Guilford [2], Dewey [1], Mansfield and Busse [64], Poincaré [65], Rossman [66], Wallas [3] and others. The main goal is to propose new ideas, which are transformations of hierarchical structured objects contained in a Knowledge Base (KB). The quality of a new idea depends on its novelty and on its suitability to solve a given problem. Two key modules play a central role in the process: the Selector and the Generator. The first one is intended to produce ideas (which are also hierarchical structured objects) by exploring the knowledge space. The second one manipulates the selected ideas and proposes new ones to the user (typically, a human user). To conveniently feed the Generator, the first module must be fluent (i.e., must have the capability to produce a wide variety of ideas), even if at the expenses of taking cognitive risks. The second module, which must contribute to control the overall quality of the proposed ideas, may adopt two strategies: it may try to increase the overall novelty of the proposed ideas (e.g., by recombining them) or act towards an increase of their overall appropriateness (e.g., by adapting them to the problem to solve).

The KB is initially set up through a Knowledge Filler, which may be controlled by a human user or act autonomously. It also may fill the KB with domain knowledge or randomly create the necessary structures. The KB may evolve during the process. This is done through a Feedback Controller, which may feed the KB with new ideas proposed by the Generator. Similarly to the Knowledge Filler, the Feedback Controller may act autonomously or not.

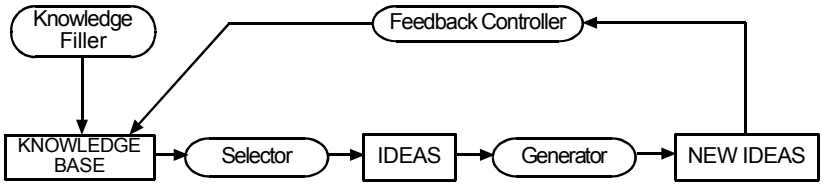

Figure 14. The proposed architecture

We can draw a parallelism between the proposed architecture and the way EC and CBR solve problems (see Table 1). In EC, a population (KB) is evaluated and individuals (ideas) are selected according to a fitness function. This may be seen as a *selection* process. Afterwards, genetic operators play the role of the Generator and proposed ideas are fed back to the KB. In CBR, cases (ideas) from a case base (KB) are *selected* according to a metric. Adaptation operators transform (generate) the selected cases. Proposed cases may be fed back to the KB.

Table 1. Similitude between the hybrid architecture, EC and CBR

Proposed Architecture	**Evolutionary Computation**	**Case Based Reasoning**
Knowledge Base	Population	Case Base
Selector	Evaluation/Selection (w./fitness function)	Selection (w./ metric)
Generator	Genetic Operators (crossover, mutation, ...)	Adaptation Operators

The architecture may be explored in many ways. The objects may be produced, used and manipulated either by CBR and EC mechanisms. We may use CBR in the Selector and apply genetic operators in the Generator to improve diversity. We may also use the fitness function to select ideas and adaptation operators to gain adequacy. We may even change the mechanisms for the Selector and the Generator in each cycle; the choice for each combination of them may depend on the evaluation of the intermediate results and/or on the specific goals in mind.

In the next section, we will show an example of how this architecture may deal with creative problem solving using the proposed common knowledge representation.

6 Example

The following example illustrates a possible way of combining EC with CBR in the framework of the proposed hybrid architecture, showing how a case library and a retrieval metric can be coupled with an EC system.

Consider that we have a case library of images and that the user chooses one of these images (see Figure 15). Using a similarity metric, the system compares the chosen image with the other images in the database. In this example, and for the

sake of simplicity, we decided to use the root mean square error as similarity metric.

It is worth noting that this measure isn't the most adequate for our goals. It would be probably best to use a metric that takes into account the similarity between the genotypes of the individuals. Considering that the individuals are represented by graphs (trees in this particular case), we could use, for instance, the Hamming distance or the maximal common sub-graph as metrics [67].

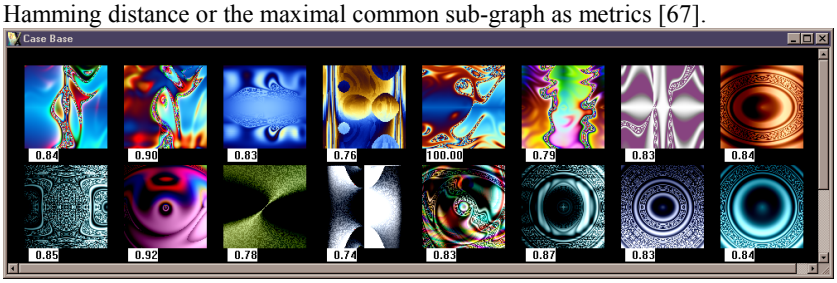

Figure 15. Some of the cases in the case library. The selected image has the score of 100 (color image available at http://www.dei.uc.pt/~amilcar/softcompbook/fig15.htm)

The most similar images will be used to initialize de EC algorithm. Thus, these images will be added to the initial population. In this example the number of images added was five, including the image selected by the user. The first population (see Figure 16) will therefore be composed by five images from the case library and eleven randomly created images (population size was set to sixteen).

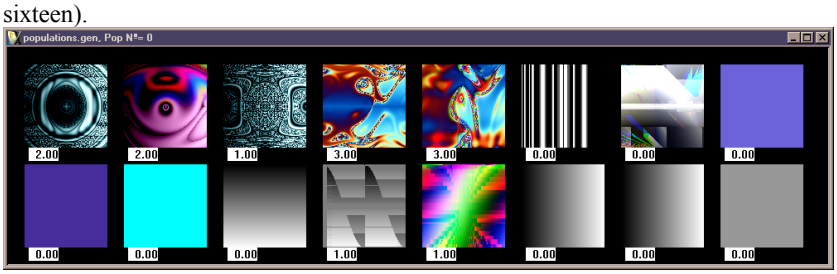

Figure 16. The initial population. The first five images were retrieved from the case library using the similarity metric; the other images were randomly created (color image available at http://www.dei.uc.pt/~amilcar/softcompbook/fig16.htm)

From this point on, the system uses the EC process to create new populations of images. The process is similar to the one described earlier: the user makes the assignment of fitness, and the genetic operators are the ones previously described.

In Figure 17, we can see the first generated population. The numbers bellow the images indicate the fitness score assigned to each image by the user. In Figure 18, we show the twentieth population, yet to be evaluated.

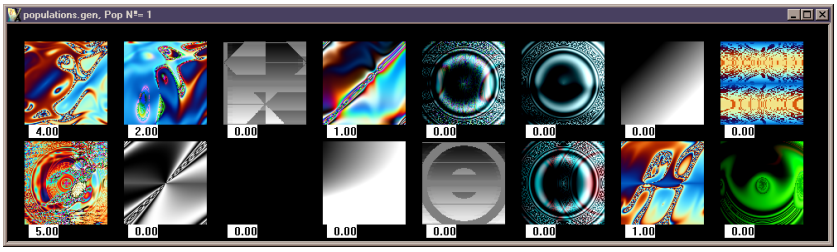

Figure 17. Population number 1 (color image available at http://www.dei.uc.pt/~amilcar/softcompbook/fig17.htm)

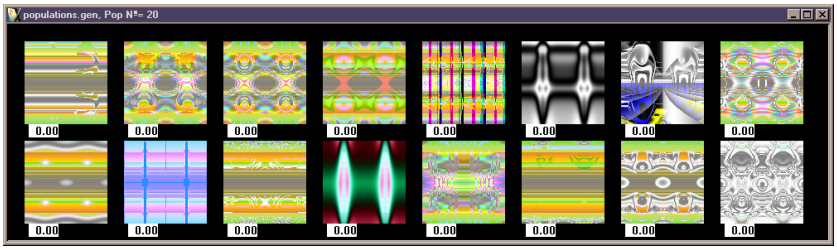

Figure 18. Population 20, not yet evaluated (color image available at http://www.dei.uc.pt/~amilcar/softcompbook/fig18.htm)

While the evolutionary process is taking place, images that have a fitness score above a given threshold value are added to the case library, thus feeding the case library with the best individuals found. In Figure 19 we show a partial snapshot of the case library after twenty populations. The first row of images comprises the images that were added to the case library from the generated populations.

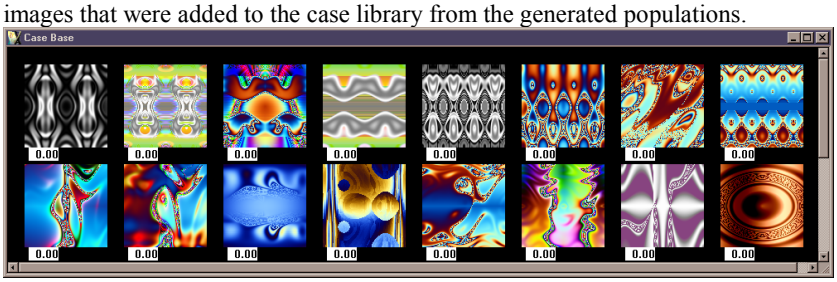

Figure 19. Snapshot of the case library after twenty populations (color image available at http://www.dei.uc.pt/~amilcar/softcompbook/fig19.htm)

We verify that by using the proposed model this way we speed up significantly the process of image generation.

7 Improvements

The proposed architecture opens an exciting range of research opportunities. We are currently exploring ways of taking the best of it to produce creative systems with improved capabilities.

The unifying characteristics of the architecture leads us to a situation in which there can be several different domains represented through the same principles and coexisting in the same environment. According to many creativity researchers (e.g., Guilford [68]) and to our own intuition, the ability to interrelate ideas from different domains can be determinant to a creativity outcome. In fact, the core of many creative products (be they artistic, scientific or others) lies exactly in the association of apparently unrelated ideas. Moreover, the Human Being inhabits a heterogeneous world and his own survival depends upon the understanding and processing of such a complex and widespread information. This takes us to the conclusion that our architecture can be much improved (i.e., be more creative) if we add a process to interrelate different domains that can coexist in the same knowledge base.

Metaphor theories [69][70] centre mainly on the understanding of metaphors, establishing correspondences between concepts of the domains involved (e.g., in the metaphor "Star Wars is the King Arthur Saga", Veale and Keane establish correspondences between the concepts that are present in both stories). In a project named Dr. Divago, Pereira and Cardoso [71] explore these Metaphor theories to search for cross-domain mappings that are used to make translations of concepts between domains. These translations are necessary to apply cases (Dr. Divago is also a CBR project) from one domain onto the other.

We think the ability to establish cross-domain transfer of knowledge is a vital future development for our work. With this feature, our systems will be able to explore more wide and varied spaces, and get ideas from apparently sterile grounds.

Another improvement that we are exploring comes from the observation that, in our framework, cases may be represented as partonomic hierarchies, whose leaf nodes can be represented as trees, thus making a two level case representation. This makes the representation more flexible to solve problems at different levels of granularity. It also gives the possibility to exploit the case representation from the evolutionary and adaptation viewpoints, allowing these different representation levels to be used in more complex reasoning processes. The more abstract level of the representation is associated with the functional description of a case, while the less abstract level is associated with the structural and/or behavioral aspect of the node it is associated with.

As an example of this kind of representation, we return to the case already presented in Section 4, describing an Arithmetic and Logic Unit (ALU). In Figure 3 we have shown the more abstract level representation that describes how the ALU is divided. The leaf nodes of this hierarchy can have attached a tree

representing the function implementation. For instance, Figure 20 describes the implementation of the Xor, that is used in the ALU. This dual representation enable us to do a two stage evolutionary process. We can first evolve the functional description of the case and then evolve the structure associated to each function node. In the ALU example, we can reach a different ALU, while at a finer level we can also evolve the Xor implementation.

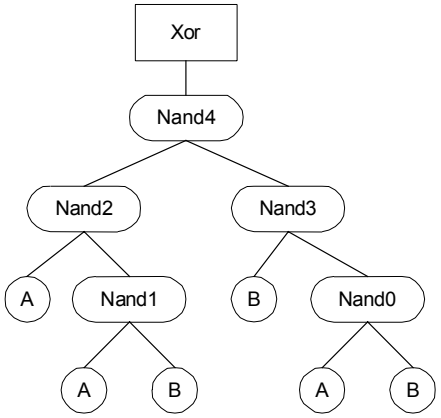

Figure 20. The representation of the Xor node in Figure 3.

8 Conclusions

Creative Reasoning is increasingly challenging research groups mainly from the area of Artificial Intelligence. Several computational models have been proposed, mostly inspired on cognitive and on biological models, and a wide range of artistic, architectural and engineering domains of application are being explored. There is a diversity of computational approaches to the problem, but Case Based Reasoning and Evolutionary Computation are the most common techniques and, in our opinion, the most promising ones. EC techniques offer diversity while adapting to environmental changes, are efficient in dealing with complex search spaces and explore parallelism in a natural way. CBR techniques can explore previous knowledge in versatile ways and have good explanatory capabilities. We argue that creative reasoning will benefit from the cross-contribution of these techniques.

During the last years we have separately explored CBR and EC-based computational approaches to creativity. In Section 4 we have briefly presented some of the results achieved, focusing on four developed applications in the domains of Design, Music Composition and Image Generation. From these efforts two key ideas emerged: first, the representation formalisms we were using could

be generalized into one common knowledge representation; second, there were strong similarities between the computational mechanisms we were using.

As a result of this synthesis work we proposed in Section 5 a hybrid architecture which empowers cross contributions from CBR and EC. The architecture builds on the above mentioned common representation language.

The proposed architecture, as was illustrated by the example in Section 6, fully integrates EC and CBR techniques: we may use the typical mechanisms of both paradigms in each of its core modules, the Selector and the Generator. The generic characteristics of the representation language and the computational mechanisms we use allow its application to a wide range of domains. Moreover, in spite of being a creativity-oriented architecture, its features leads us to believe that it may prove useful in other problem-solving tasks.

This framework also enables the coexistence of objects from multiple domains in the same knowledge base, providing the means to explore advanced creativity-related concepts like metaphoric transfer.

9 References

[1] Dewey, J. (1919). *How we think*. Boston: D. C. Heath. Boston, USA.

[2] Guilford, J. P. (1968). *Intelligence, creativity and their educational implications*. San Diego, CA: Robert Knapp.

[3] Wallas, G. (1926). *The art of thought*. Nova Iorque: Harcourt Brace.

[4] De Bono, E. (1986). *El pensamiento lateral. Manual de la creatividad*. Barcelona, Spain: Paidós. (in Spanish)

[5] Goldberg, D. (1998). The design of innovation: lessons from genetic algorithms, lessons for the real world, IlliGAL Report Nº 98004, February 1998, Department of General Engineering, University of Illinois at Urbana-Champain.

[6] Kolodner, J., (1993). *Case-Based Reasoning*. Morgan Kaufman Publ., San Mateo, CA.

[7] Aamodt, A. and Plaza, E., (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and Systems Approaches (39-59). *AICOM* Vol7 N.1

[8] Back, T., Fogel, D., Michalewicz, Z. (eds.) (1997). *Handbook of Evolutionary Computation*, Oxford University Press, New York.

[9] Rechenberg, I. (1965). Cybernetic solution path of an experimental problem, Lybrary Translation nº 1122, Royal Aircraft Establishement, Farnborough, UK.

[10] Schewefel, H-P. (1965). Kybernetishe evolution als strategie der experimentallen forschung in der stromungstechnik, Diplomarbeit, Tecnische Universitat Berlin, 11965.

[11] Fogel, L. (1962). *Autonomous automata*, Industrial Research 4**:** 14-19.

[12] Holland, J. (1975). *Adaptation in natural and artificial systems*, University of Michigan Press, Michigan..

[13] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection,* MIT Press, Cambridge, MA, USA.

[14] Gero, J. (1994b). Computational Models of Creative Design Processes. In Dartnall, T. (ed.), *Artificial Intelligence and Creativity*, pp. 269-281. Kluwer Academic Publishers, Dordrecht..

[15] Gero, J. (1994a). Introduction: Creativity and Design. In Dartnall, T. (ed.), *Artificial Intelligence and Creativity*, pp. 259-267. Kluwer Academic Publishers, Dordrecht..

[16] Kolodner, J., and Wills, L., (1993). Case-Based Creative Design. In *AAAI Spring Symposium on AI+Creativity*, Stanford, CA.

[17] Simina, M. and Kolodner, J. (1997). Creative Design: Reasoning and Understanding. In *Proceedings of the International Conference on Case-Based Reasoning* (ICCBR 97), Providence - Rhode Island, USA.

[18] Sycara, K., and Navinchandra, D., (1991). Influences: A Thematic Abstraction for Creative Use of Multiple Cases. In *Proceedings of the first European Workshop on Case-Based Reasoning*, Springer-Verlag, Washington, DC..

[19] Sycara, K., and Navinchandra, D., (1993). Case Representation and Indexing for Innovative Design Reuse. In *Proceedings of the Workshop of the 13th International Joint Conference on Artificial Intelligence*, Paris, France, Morgan-Kaufmann Publishers..

[20] Pereira, F. C., Grilo, C., Macedo, L. and Cardoso, A. (1997). Composing Music with Case-Based Reasoning. In *Proceedings of Computational Models of Creative Cognition (Mind II)*. Dublin, Ireland.

[21] Arcos, J.L., Mantaras, R. L. and Serra,.X. (1997) SAXEX: A Case-Based Reasoning system for generating expressive musical performances. Proceedings of the 1997 International Computer Music Conference. Thessaloniki, Greece.

[22] Do, E. and Gross, M. D. (1995) Supporting Creative Architectural Design with Visual References. In J. Gero et al (ed), 3rd International Conference on Computational Model of Creative Design (HI '95). Heron Island, Australia.

[23] Ralley, D. (1995). Genetic algorithm as a tool for melodic development. In *International Computer Music Conference*. Alberta, Canada.

[24] Biles, J. (1994) A genetic algorithm for generating jazz solos. In *International Computer Music Conference,* Aarhus, Denmark.

[25] Jacob, B. L. (1995). Composing with Genetic Algorithms. In *International Computer Music Conference*. Alberta, Canada.

[26] Dawkins, R. (1987). *The Blind Watchmaker,* W.W. Norton & Company, Inc., New York.

[27] Sims, K. (1991). Artificial Evolution for Computer Graphics. *ACM Computer Graphics,* 25**,** 319--328.

[28] Todd, S. and Latham, W. (1992). *Evolutionary Art and Computers,* Academic Press, Winchester, UK.

[29] Rooke, S. (1996). The Evolutionary Art of Steven Rooke, http://www.azstarnet.com/~srooke/.

[30] Machado, P., Cardoso, A. (1999). NEvAr. In *Evolutionary Design by Computers CD-ROM*, Bentley, P. (ed.). Morgan Kaufmann. San Francisco, California, USA.

[31] Baker, E. (1993). *Evolving Line Drawings*, Harvard University Center for Research in Computing Technology, Technical Report TR-21-93.

[32] Bentley, P. (Ed.) (1999). *Evolutionary Design by Computers,* Morgan Kaufman Publ., San Francisco, California, USA.

[33] Graf, J. and Banzhaf, W. (1996). Interactive Evolution for Simulated Natural Evolution. In *Artificial Evolution*, Alliot, J.-M., Lutton, E., Ronald, E., Schoenauer, M. and Snyers, D. (eds.), Vol. 1063, Springer Verlag, Nimes, France. pp. 259--272.

[34] Angeline, P. J. (1996). Evolving Fractal Movies. In *Genetic Programming 1996: Proceedings of the First Annual Conference* (Eds, Koza, J. R., Goldberg, D. E., Fogel, D. B. and Riolo, R. L.) MIT Press, Stanford University, CA, USA, pp. 503--511.

[35] Ventrella, J. (1999). Animated Artificial Life. In *Virtual Worlds - Synthetic Universes, Digital Life, and Complexity*, Heudin, J. C. (ed.) New England, Complex Systems Institute Series on Complexity. Perseus Books, pp. 67-94.

[36] Baluja, S., Pomerlau, D. and Todd, J. (1994). Towards Automated Artificial Evolution for Computer-Generated Images. *Connection Science,* **6,** 325-354.

[37] Horner, A. Goldberg, D. (1991). Genetic algorithms and computer-assisted composition. In *Genetic Algorithms and Their Applications: Proceedings of the Fourth International Conference on Genetic Algorithms*, Urbana-Champaign, Illinois, USA. pp. 427-441.

[38] McIntyre, R. A. (1994). Bach in a Box: The Evolution of Four-Part Barroque Harmony Using Genetic Algorithms. In *IEEE Conference on Evolutionary Computation*. Orlando, USA.

[39] Spector, L. and Alpern, A. (1994). Criticism, culture, and the automatic generation of artworks. In *Proceedings of Twelfth National Conference on Artificial Intelligence,* AAAI Press/MIT Press, Seattle, Washington, USA, pp. 3-8.

[40] Spector, L. and Alpern, A. (1995). Induction and Recapitulation of Deep Musical Structure. In *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI'95 Workshop on Music and AI,* Montreal, Quebec, Canada.

[41] Papadopoulos, G. and Wiggins, G. (1999). AI Methods for algorithmic composition: A Survey, a Critical View and Future Prospects. In *AISB Symposium on Musical Creativity*, Wiggins, G. (ed.) Edinburgh, UK.

[42] Wiggins, G., Papadoupoulos, G., Phon-Amnuaisuk, S. and Tuson, A. (1999). Evolutionary Methods for Musical Composition. In *International Journal of Computing Anticipatory Systems*.

[43] McCorduck, P. (1991) AARON's Code: Meta-Art, Artificial Intelligence and the Work of Harold Cohen. W.H. Freedman and Company. New York: Computer Science. USA.

[44] Burton, E. (1997). Representing Representation: Artificial Intelligence and Drawing. In Mealing, S. (Ed) *Computers & Art*. Intellect Books. Exeter, UK.

[45] Pachet, F., and Roy, P. (1998). Formulating Constraint Satisfaction Problems on Whole-Part relations: the Case of Automatic Harmonisation. In *Workshop at ECAI'98. Constraint Techniques for Artistic Applications*, Brighton, UK.

[46] Cambouropoulos, E. (1994). *Markov Chains as an Aid to Computer Assisted Composition*. *Musical Praxis*, 1 (1): 41-52.

[47] Cope, D. (1992). Computer Modeling of Musical Intelligence with EMI. *Computer Music Journal*.

[48] Stiny, G. (1976). Two exercises in formal composition, *Environment and Planning B: Planning* and *Design,*3 pp. 187-210.

[49] Louis, S., McGraw,G., Wyckoff, R. (1992). CBR assisted explanation of GA results, *Journal of Experimental and Theoretical Artificial Intelligence* 5 : 21-37.

[50] Louis, S., Xu, Z. (1996), Genetic algorithms for open shop scheduling. In *Proceedings of ISCA, 11th Conference on Computers and their Applications*, pp 99-102.

[51] Louis, S., Johnson, J. (1997). Solving similar problems using genetic algorithms and case-based memory. In: Back., T. (ed.) *Proceedings of the 7th International Conference on Genetic Algorithms*, ICGA'97, East Lansing, Michigan, USA. Morgan Kaufmann, pp 283-290.

[52] Louis,S., Zhang, Y. (1999). A sequentially metric for case injected genetic algorithms applied to TSPs. In Banzhaf et al. (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO'99, Orlando, Florida, USA. Morgan Kaufman Publishers, Inc., California, USA. pp 377-384.

[53] Ramsey, C., Grefensttete, J. (1993). Case-based initialisation of genetic algorithms. In Forrest, S. (ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, ICGA'93, Urbana-Champain, Illinois, USA. Morgan Kaufmann, San Mateo, California. pp 84-91.

[54] Eshelman, L. (1991). The CHC adaptive search algorithm: how to have a safe search when engaging a non-traditional genetic recombination. In: Rawlings, G. (ed.) *Foundations of Genetic Algorithms* (FOGA-1), Morgan Kaufman, San Francisco, USA. pp. 265-283.

[55] Gomes, P., Bento, C., Gago, P., and Costa, E, (1996). Towards a Case-Based Model for Creative Processes. In Wahlster, Wolfgang (ed.), *Proceedings of the 12th European Conference on Artificial Intelligence* (122-126). West Sussex: John Willey & Sons.

[56] Bento, C., Macedo, L., and Costa, E. (1994). Reasoning with Cases Imperfectly Described and Explained. In Wess,S.; Althoff, K.-D.; and Richter, M. M., eds*., Topics in Case-Based Reasoning - Selected Papers from the First European Workshop on Case-Based Reasoning*, Kaiserslautern, Springer Verlag, 1994.

[57] Bento, C., and Costa, E. (1994). A Similarity Metric for Retrieval of Cases Imperfectly Explained. In Wess,S.; Althoff, K.-D.; and Richter, M. M., eds*., Topics in Case-Based Reasoning - Selected Papers from the First European Workshop on Case-Based Reasoning*, Kaiserslautern, Springer Verlag, 1994.

[58] Goel, A., (1992). Representation of Design Functions in Experience-Based Design. In Brown, D., M. Waldron, & H. Yoshikawa (Eds.), *Intelligent Computer Aided Design*, pp. 283-308. Elsevier Science Publishers, Amsterdam..

[59] Stroulia, E., Shankar, M., Goel, A., and Penberthy, L., (1992). A Model-Based Approach to Blame Assignment in Design. In *Proceedings of the 2nd International Conference on AI in Design*, pp. 519-537, Kluwer, Dordrecht..

[60] Bylander, T., and Chandrasekaran, B., (1985). Understanding Behaviour Using Consolidation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (450-454), Los Angeles, CA, USA.. Morgan Kaufman.

[61] Smyth, B., and Keane, M., (1995). Experiments on Adaptation-Guided Retrieval in Case-Based *Design*. In Veloso, Manuela, & Agnar Aamodt (Eds), *Topics in Case-Based Reasoning Proceedings of the International Conference on Case-Based Reasoning* (313-324). Berlin: Springer Verlag.

[62] Macedo, L., Pereira, F. C., Grilo, C. and Cardoso, A. (1998). A Computational Model for Creative Planning. In: Schmid U, Krems JF, Wysotzki F (Eds.) *Mind Modelling: A Cognitive Science Approach to Reasoning, Learning and Discovery* pp.193-208. Pabst Science Publishers. Lengerich, Germany.

[63] Macedo, L., Pereira, F. C., Grilo, C. and Cardoso, A. (1997). Experimental Study of a Similarity Metric for Retrieving Pieces from Structured Plan Cases: its Role in the Originality of Plan Case Solutions In *Proceedings of the 2nd International Conference on Case-Based Reasoning, ICCBR-97*, Brown University, Providence, Rhode Island USA, Lecture Notes in Artificial Intelligence, LNAI-1266, Springer-Verlag.

[64] Mansfield, R., and Busse, T. (1981). *The psychology of creativity and discovery*. Chicago: Nelson-Hall.

[65] Poincaré, H. (1913). *The foundations of science*. Lancaster, PA: Science Press.

[66] Rossman, J. (1931). *The psychology of the inventor: A study of the patentee*. Washington, DC: Inventors Publishing Co.

[67] Macedo, L., and Cardoso, A. (1999). Labelled Adjacency Matrices for Labelled, Directed Multigraphs: Their Algebra and Hamming Distance, In *Procs. of the 2nd. IAPR-TC15 Workshop on Graph-based Representation*s, GbR'99, Castel of Haindorf, Austria,.

[68] Guilford, J. P. (1967). *The Nature of Human Intelligence*. McGraw-Hill, New York.

[69] Veale, T. and Keane, M. T. (1994). Metaphor and Memory: Symbolic and Connectionist. Issues in Metaphor Comprehension. In *Proceedings of the European Conference on Artificial Intelligence Workshop on Neural and Symbolic Integration*, Amsterdam.

[70] Indurkhya, B. (1992). *Metaphor and Cognition*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

[71] Pereira, F. C. and Cardoso, A. (1999). Dr. Divago: searching for new ideas in a multi-domain environment *Proceedings of the 8th Cognitive Science of Natural Language Processing (CSNLP-8),* Ireland.